Enhancing Performance of Recommender Systems

A Thesis Submitted to

Nirma University

In Partial Fulfillment of the Requirements for

The Degree of

Doctor of Philosophy

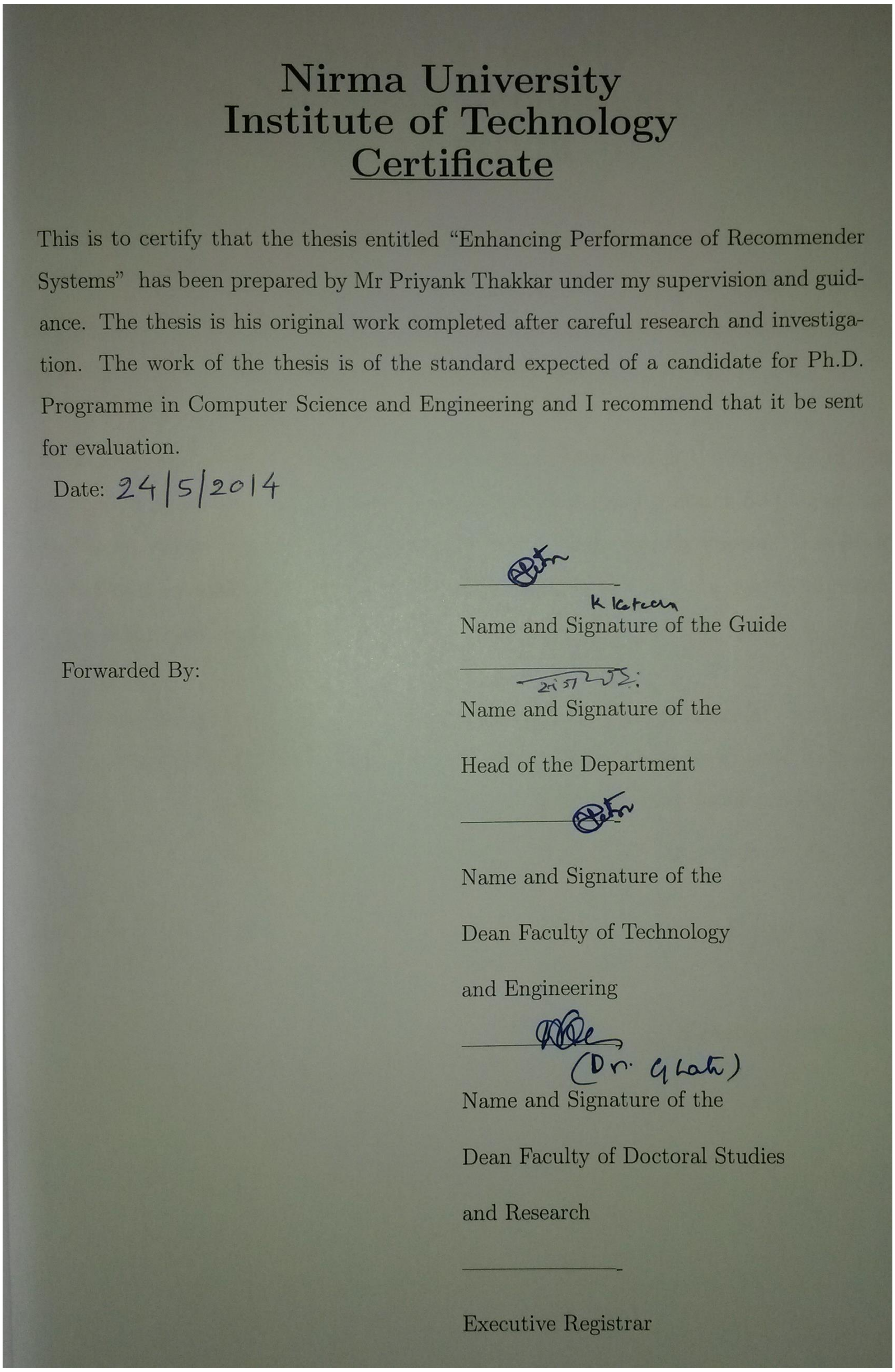
in

Technology and Engineering

by

Priyank Thakkar (09EXTPHDE25)

Institute of Technology
Nirma University
Ahmedabad-382481
Gujarat, India
May 2014



Nirma University
Institute of Technology
Certificate

This is to certify that the thesis entitled "Enhancing Performance of Recommender Systems" has been prepared by Mr Priyank Thakkar under my supervision and guidance. The thesis is his original work completed after careful research and investigation. The work of the thesis is of the standard expected of a candidate for Ph.D. Programme in Computer Science and Engineering and I recommend that it be sent for evaluation.

Date: 24/5/2014

[illegible]
K Kotecha
Name and Signature of the Guide

Forwarded By:

[illegible]
Name and Signature of the
Head of the Department

[illegible]
Name and Signature of the
Dean Faculty of Technology
and Engineering

[illegible]
(Dr. Ghate)
Name and Signature of the
Dean Faculty of Doctoral Studies
and Research

Executive Registrar

Abstract

Freedom of choice is our right. Coincidentally, the right to choose is the freedom. It may also be said that it is very difficult to make right choices when many options are available. In terms of Internetworking applications, this century can be dedicated to Social Networking. Information sharing across the globe, is rampant. Most of the options are also already available online. It is crucial to get information suiting ones requirements from the available data.

There is a need for practical applications that help users to deal with information overload and provide personalized recommendations, content and services to them. These recommendations are highlighted by vendors in terms of advertisements, short interviews during tele-marketing and through feedbacks. Every transaction also records the choice of an individual. Servers record these transactions and use them in commercial ratings. Still, there exist confusion among the users to choose the best fitting option within their defined behavioural constraints.

Recommender system is one of the applications to predict rating or preference for the items that have not been seen by a user. This system typically produces a list of recommendations. Recommending books, CDs, and other products at amazon.com, movies by MovieLens, and news at VERSIFI Technologies (formerly adaptiveInfo.com) are examples of such applications to name a few. However, despite these developments, the current generation of recommender systems still requires further improvements to make recommendation methods more accurate and applicable to an even broader range of real-life needs including recommending vacations, research articles, URLs, music artists, social tags and certain types of financial services to investors. These improvements also cater to mapping the behaviour of an individual with the corresponding choice of an item. Hence, advanced recommendation modelling methods, incorporation of various contextual information into the recommendation process, and measures to determine performance of recommender systems are considered.

Raw data used for processing is available under various categories. These categories are positive and unlabelled data, combination of labelled and unlabelled data and time-series data to name a few. Different applications generate different categories of data. Data generated through social bookmarking systems (Bibsonomy or Delicious) is an example of positive and unlabelled data. Labelled and unlabelled data is aggregated in feedback application for any item, where an item may be liked or disliked by individuals (E.g. movie feedback system). Continuously changing data (time-series) is available for example from stock market. Depending on type of data, processing in recommender system changes. Hence, this thesis focuses on optimizing performance of recommender system, catering to the requirements of these different categories of data.

Recommending on the basis of positive and unlabelled data is crucial, especially if the information is available through Social Networking sites. Social bookmarking sites such as Bibsonomy or Delicious allow user to bookmark URLs and submit the research articles. User bookmarking a resource (URL) or submitting a resource (research article) on this system, implicitly indicates his likings to this resource. These resources are considered as positive examples of the user preference. Other resources (URLs/research articles), however, do not imply negative preference of the user about them. This leads to the situation where we have positive examples but no negative examples for user preference. A recommender which is learnt from positive and unlabeled examples is devised to recommend these social resources to user.

The work also focuses on recommending tags for the resources being submitted first time to the social bookmarking site. The task is modeled as multi-label text classification. Naive-Bayes classifier is used as the base learner of the multi-label text classifier and multinomial distribution is fitted to the data for experimentation.

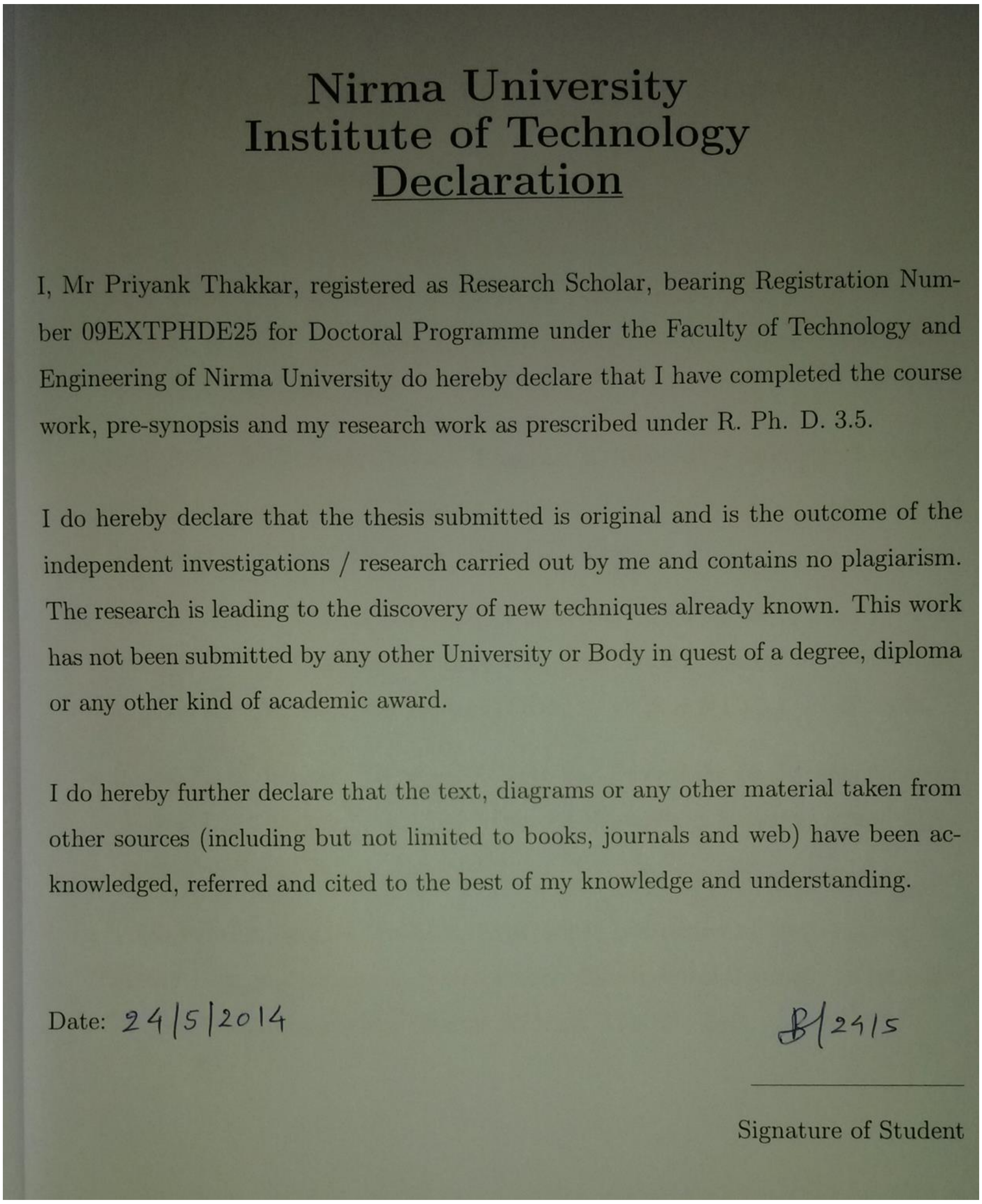
Recommendation system is then developed considering a variety of tags associated with an item. A movie recommendation application is one such application. A movie recommendation system is hence proposed, that combines ratings, tags, genres and star cast of movies.

In a recommender system, the items which are liked or disliked by the user are the labelled examples about the users' preference. However, there are many other items which are not rated by the user. These items form the set of unlabelled examples. One of the major problem with recommender system is that it does not have adequate number of labelled examples for the new user. Efficacy of co-training algorithm in learning a recommender by exploiting unlabelled examples in the presence of small number of labelled examples is critical.

Recommendation system is also one of the outcomes of prediction in several applications. The best application of such a recommendation system is in stock market where data keeps changing with time. The problem of predicting direction of movement of stock price and stock market index is handled by first converting the data into trend deterministic data and then learning through various machine learning techniques using this data.

To predict future values of stock market indices, hybrid techniques combining support vector regression with other machine learning techniques is developed. The techniques predict value of stock price indices for 1 to 10, 15 and 30 days in advance.

In a nutshell, these recommendations are not limited to a list of unseen items, but also include some predictions or forecasts to help users in making appropriate decisions.



Nirma University
Institute of Technology
Declaration

I, Mr Priyank Thakkar, registered as Research Scholar, bearing Registration Number 09EXTPHDE25 for Doctoral Programme under the Faculty of Technology and Engineering of Nirma University do hereby declare that I have completed the course work, pre-synopsis and my research work as prescribed under R. Ph. D. 3.5.

I do hereby declare that the thesis submitted is original and is the outcome of the independent investigations / research carried out by me and contains no plagiarism. The research is leading to the discovery of new techniques already known. This work has not been submitted by any other University or Body in quest of a degree, diploma or any other kind of academic award.

I do hereby further declare that the text, diagrams or any other material taken from other sources (including but not limited to books, journals and web) have been acknowledged, referred and cited to the best of my knowledge and understanding.

Date: 24/5/2014

[illegible] 24/5

Signature of Student

Acknowledgements

I take the pleasure to present this research work related to "Enhancing Performance of Recommender Systems" to the Almighty, for being present in all my endeavors.

I would like to thank Dr K Kotecha, Guide and the Director, Institute of Technology, Nirma University, Ahmedabad for motivating me and providing continuous support throughout my Doctoral studies.

My sincere thanks to the reviewers Dr N D Jotwani and Dr D P Ahalpara, who had given their valuable feedback during my Research Progress Committee meetings.

I would like to express my sincere regards to Dr Sanjay Garg, Head of Computer Science and Engineering Department for his never-ending support and cooperation.

I can not forget to thank Dr. Sharada Valiveti, who helped me a lot in shaping the draft of the thesis.

I am also thankful to all faculty members, staff members and students of our department for providing all types of resources and support in time.

There are no words to thank the Almighty for gifting me with a wonderful child, Pratham, who bore the wrath of my journey towards this research. My wife Falguni needs special accolade for her patience and constant support provided throughout this work. Finally, my sincere most thanks to the most important and special people of my life - my parents; who have been ever motivating, endearing and highly cooperative in all my endeavors.

I thank one and all, who have kept encouraging and motivating me. Without everyone's support, this thesis is beyond imagination.

Priyank Thakkar

09EXTPHDE25

Contents

Certificate i

Declaration ii

Publications related to Thesis iii

Abstract iii

Acknowledgements vi

List of Tables xi

List of Figures xiii

Abbreviations xiv

1 Introduction 1
- 1.1 Overview 1
- 1.2 Problem Statement 2
- 1.3 Organization of the Thesis 2

2 Background 5
- 2.1 Recommender Systems 5
- 2.2 Content-based Recommenders 6
- 2.3 Collaborative Recommenders 8
- 2.4 Hybrid Recommenders 9

3 Predicting Direction of Movement of Stock Price and Stock Market Index 11
- 3.1 Introduction 12
- 3.2 Related Work 12
- 3.3 Research Data 16
  - 3.3.1 Continuous Representation - The Actual Time Series 19
  - 3.3.2 Discrete Representation - Trend Deterministic Data 19
- 3.4 Prediction Models 21
  - 3.4.1 ANN Model 21
  - 3.4.2 SVM Model 23
  - 3.4.3 Random Forest 25
  - 3.4.4 Naive Bayes Classifier 26
- 3.5 Experimental Evaluation 28

3.5.1 Evaluation Measures 28
3.5.2 Experimental Methodology & Results 29
3.6 Discussions 30
3.7 Conclusions 34

4 Predicting Stock Market Index using Fusion of Machine Learning Techniques 39
4.1 Introduction and Literature Review 39
4.2 Single Stage Approach 42
4.2.1 Artificial Neural Network 43
4.2.2 Support Vector Regression 44
4.2.3 Random Forest 46
4.3 Two Stage Fusion Approach 46
4.4 Experimental Evaluation 49
4.4.1 Data for Experimentation 49
4.4.2 Evaluation Measures 49
4.4.3 Results and Discussions 50
4.5 Conclusions 65

5 Social Resource Recommendation using Learning from Positive and Unlabeled Examples 67
5.1 Introduction 68
5.2 Related Work 69
5.3 Collaborative Filtering for Resource Recommendation 70
5.4 Learning from Positive and Unlabeled Examples 71
5.4.1 Usefulness of Unlabeled Data 71
5.4.2 Theoretical Foundations of PU Learning 72
5.5 The Proposed Two Step Techniques 73
5.5.1 Finding Reliable Negative Resources 73
5.5.2 Building and using Recommender to make Predictions 74
5.5.3 Feature Selection 76
5.6 The Proposed Direct Method 77
5.7 Experimental Evaluation 79
5.7.1 Data set 79
5.7.2 Experimental Methodology 80
5.7.3 Evaluation Measures 80
5.7.4 Results and Discussions 81
5.8 Conclusions 86

6 Recommending Tags for New Resources in Social Bookmarking System 88
6.1 Introduction 89
6.2 Related Work 90
6.3 Proposed Approach 92
6.4 Data Preprocessing 93
6.5 Proposed Tag Recommendation Algorithm 99
6.6 Experimental Evaluation 102
6.7 Conclusions 104

7 Movie Recommender System - Hybrid Filtering Approach 106
7.1 Introduction 106
7.2 Related Work 108
7.3 Item-Based Collaborative Filtering 109
7.4 Proposed Approach 109
7.4.1 Movie's Tag Profile Generation 109
7.4.2 Preference Profile Generation 111
7.4.3 Neighborhood Formation 112
7.4.4 Rating Prediction Formula 113
7.5 Experimental Evaluation 113
7.5.1 Data Set 113
7.5.2 Experimental Methodology and Results 114
7.6 Conclusions 117

8 Evaluating a Recommender learnt from Labeled and Unlabeled Data 119
8.1 Introduction 119
8.2 Related Work 120
8.3 Co-Training Algorithm 121
8.3.1 Experimental Evaluation 122
8.3.2 Data set 122
8.3.3 Evaluation Measures 122
8.3.4 Experimental Methodology, Results and Discussions 123
8.4 Conclusions 125

9 Conclusions and Future Work 126

Works Cited 131

Index 143

List of Tables

3.1 Number of increase and decrease cases (days) percentage in each year in the entire data set of S&P BSE Sensex 17
3.2 Number of increase and decrease cases (days) in each year in the parameter setting data set of S&P BSE Sensex 18
3.3 Number of increase and decrease cases (days) in each year in the comparison data set of S&P BSE Sensex 19
3.4 Selected technical indicators & their formula (Kara, Acar Boyacioglu, and Baykan) 20
3.5 ANN parameters and their values tested in parameter setting experiments 23
3.6 SVM parameters and their values tested in parameter setting experiments 25
3.7 Best three parameter combinations of ANN model and their performance on continuous-valued parameter setting data set 30
3.8 Best two parameter combinations (one for each type of kernel) of SVM model and their performance on continuous-valued parameter setting data set 31
3.9 Best three parameter combinations of random forest model and their performance on continuous-valued parameter setting data set 32
3.10 Performance of prediction models on continuous-valued comparison data set 33
3.11 Best three parameter combinations of ANN model and their performance on discrete-valued parameter setting data set 35
3.12 Best two parameter combinations (one for each type of kernel) of SVM model and their performance on discrete-valued parameter setting data set 36
3.13 Best three parameter combinations of random forest model and their performance on discrete-valued parameter setting data set 37
3.14 Performance of prediction models on discrete-valued comparison data set 38

4.1 ANN parameters and their values tested 44
4.2 SVR parameters and their values tested 45
4.3 Best parameter combination reported by parameter tuning experiments for each of the SVRs in first stage of two stage fusion approach 48
4.4 Prediction performance of 1-day ahead of time (for CNX Nifty) . . . 51
4.5 Prediction performance of 2-days ahead of time (for CNX Nifty) . . . 51
4.6 Prediction performance of 3-days ahead of time (for CNX Nifty) . . . 52
4.7 Prediction performance of 4-days ahead of time (for CNX Nifty) . . . 52
4.8 Prediction performance of 5-days ahead of time (for CNX Nifty) . . . 53

4.9 Prediction performance of 6-days ahead of time (for CNX Nifty) . . . 53
4.10 Prediction performance of 7-days ahead of time (For CNX Nifty) . . . 54
4.11 Prediction performance of 8-days ahead of time (for CNX Nifty) . . . 54
4.12 Prediction performance of 9-days ahead of time (for CNX Nifty) . . . 55
4.13 Prediction performance of 10-days ahead of time (for CNX Nifty) . . 55
4.14 Prediction performance of 15-days ahead of time (for CNX Nifty) . . 56
4.15 Prediction performance of 30-days ahead of time (for CNX Nifty) . . 56
4.16 Prediction performance of 1-day ahead of time (for S&P BSE Sensex) 57
4.17 Prediction performance of 2-days ahead of time (for S&P BSE Sensex) 57
4.18 Prediction performance of 3-days ahead of time (for S&P BSE Sensex) 58
4.19 Prediction performance of 4-days ahead of time (for S&P BSE Sensex) 58
4.20 Prediction performance of 5-days ahead of time (for S&P BSE Sensex) 59
4.21 Prediction performance of 6-days ahead of time (for S&P BSE Sensex) 59
4.22 Prediction performance of 7-days ahead of time (for S&P BSE Sensex) 60
4.23 Prediction performance of 8-days ahead of time (for S&P BSE Sensex) 60
4.24 Prediction performance of 9-days ahead of time (for S&P BSE Sensex) 61
4.25 Prediction performance of 10-days ahead of time (for S&P BSE Sensex) 61
4.26 Prediction performance of 15-days ahead of time (for S&P BSE Sensex) 62
4.27 Prediction performance of 30-days ahead of time (for S&P BSE Sensex) 62
4.28 Average prediction performance for CNX Nifty 63
4.29 Performance improvement in (%) (Single stage models vs. Two stage fusion models) for CNX Nifty 63
4.30 Average prediction performance for S&P BSE Sensex 63
4.31 Performance improvement in (%) (Single stage models vs. Two stage fusion models) for S&P BSE Sensex 64

5.1 Results on *delicious* 81
5.2 Results on *lastfm* 82

6.1 Attributes of three files 94
6.2 Tag assignment to resource 95
6.3 Bookmarked web page in Bibsonomy 95
6.4 Bookmarked publication in Bibsonomy 96
6.5 Tag recommendation in social bookmarking system - results on Bookmark data set 102
6.6 Tag recommendation in social bookmarking system - results on BibTeX data set 103

7.1 Experimental results 117

8.1 Aggregated results 124

List of Figures

2.1 Content-based Recommender 7
2.2 Collaborative Recommender 8

3.1 Architecture of ANN model (Kara, Acar Boyacioglu, and Baykan) . . 22
3.2 Predicting with continuous-valued data 33
3.3 Predicting with trend deterministic data 34

4.1 General architecture of single stage approach for predicting n day ahead of time 42
4.2 General architecture of two stage fusion approach for predicting n day ahead of time 46
4.3 Details of two stage fusion approach for predicting n day ahead of time 47
4.4 Prediction performance comparison of ANN and SVR-ANN for predicting 5 day ahead of time for CNX Nifty 65

5.1 Learning from positive and unlabeled examples (Liu) 71
5.2 Collaborative filtering - *delicious* data set 83
5.3 Collaborative filtering - $lastfm$ data set 83
5.4 F-measure @ Top5 - *delicious* data set 84
5.5 F-measure @ Top10 - *delicious* data set 84
5.6 F-measure @ Top15 - *delicious* data set 85
5.7 Fmeasure @ Top5 - $lastfm$ data set 85
5.8 Fmeasure @ Top10 - $lastfm$ data set 86
5.9 Fmeasure @ Top15 - $lastfm$ data set 86

6.1 BibSonomy: social bookmark and publication sharing system 89
6.2 Tag recommendation in BibSonomy 90
6.3 Histogram for tag frequency distribution in Bookmark data set 97
6.4 Histogram for tag frequency distribution in BibTeX data set 98
6.5 Conceptual flow of preprocessing steps 99
6.6 Flowchart of proposed tag recommendation system 101
6.7 Impact of feature selection on Bookmark data set 103
6.8 Impact of feature selection on BibTex data set 104

7.1 Conceptual flow of the proposed recommender system 114

8.1 Impact of feature selection on accuracy 124
8.2 Impact of feature selection on f-measure 125

List of Abbreviations

AFSA	Artificial Fish Swarm Algorithm
AI	Artificial Intelligence
ANFIS	Adaptive Neuro Fuzzy Inference System
ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
BPN	Back-propagation Neural Network
BR	Binary Relevance
BSE	Bombay Stock Exchange
CART	Classification and Regression Tree
CBR	Case-Based Reasoning
CCI	Commodity Channel Index
CF	Collaborative Filtering
CGA	Chaotic Genetic Algorithm
EGARCH	Exponential Generalized Autoregressive Conditional Heteroskedasticity
EM	Expectation-Maximization
EMA	Exponential Moving Average
FA	Firefly Algorithm
FDP	Financial Distress Prediction
FN	False Negative
FP	False Positive
FPE	Final Prediction Error
GA	Genetic Algorithm
GARCH	Generalized Autoregressive Conditional Heteroskedasticity
GEP	Gene Expression Programming
GJR-GARCH	Glosten-Jagannathan-Runkle GARCH
GP	Genetic Programming
HMM	Hidden Markov Model
IBCO	Improved Bacterial Chemotaxis Optimization
IF	Information Filtering
IGARCH	Integrated GARCH
IR	Information Retrieval
KOSPI	Korea Composite Stock Price Index
LDA	Linear Discriminant Analysis
LM	Levenberg Marquardt
LS-SVM	Least Squares - Support Vector Machine
LU	Labeled and Unlabeled
MACD	Moving Average Convergence Divergence
MAE	Mean Absolute Error

MAPE	Mean Absolute Percentage Error
MEP	Multi Expression Programming
MLP	Multilayer Perceptron
MND	Multinomial Distribution
MRLF	Morphological Rank Linear Forecasting
MSE	Mean Square Error
MVBD	Multivariate Bernoulli Distribution
NB	Naive Bayes
ND	Normal Distribution
NN	Nearest Neighbor
NOF	Number of Features
NSE	National Stock Exchange
NYSE	New York Stock Exchange
PNN	Probabilistic Neural Network
PU	Positive and Unlabeled
QDA	Quadratic Discriminant Analysis
RBF	Radial Basis Function
RBFNN	Radial Basis Function Neural Network
RF	Random Forest
RN	Reliable Negative
rRMSE	Relative Root Mean Square Error
RS	Recommender System
RSI	Relative Strength Index
SMA	Simple Moving Average
SOM	Self-Organizing Map
SRM	Structural Risk Minimization
SSU	Set of Similar Users
SSVR	Seasonal Support Vector Regression
SVC	Support Vector Classification
SVM	Support Vector Machine
SVR	Support Vector Regression
TAEF	Time-delay Added Evolutionary Forecasting
TEPIX	Tehran Exchange Price Index
TF	Term Frequency
TFIDF	Term Frequency-Inverse Document Frequency
TN	True Negative
TP	True Positive
TS	Tabu Search
URL	Uniform Resource Locator
WTR	Weighted Tag Recommender
WTRR	Weighted Tag Rating Recommender
WWW	World Wide Web

Chapter 1

Introduction

1.1 Overview

For the past two decades, the rate at which World Wide Web has expanded is enormous. Instantaneous access to a large diversity of knowledge items became possible due to the advent of the first generation of the World Wide Web (WWW). The second generation of the WWW is typically denoted by Web 2.0. It has instigated a fundamental change in the way people interact with and through the World Wide Web. This is why Web 2.0 is also referred to as the participatory Web. It can be characterized as a paradigm that facilitates communication, interoperability, user-centered design, information sharing and collaboration on the Web (Sharma Eijlander and Bogers).

Moreover, in the transition to Web 2.0, a paradigm shift from local and solitary to global and collaborative is exhibited. The shift also accords with a shift from accessing and creating information to understanding information and the people who deal with this information. Information management and access has been moving to many distributed places on the Web, instead of, creating, storing, managing, and accessing it on only one specific computer or browser. Collaboratively created websites such as Wikipedia are accessed and edited by anyone from anywhere. Users have started documenting and sharing many aspect of their lives online using blogs, social networking sites, and video and photo sharing sites (Eijlander and Bogers).

A global information society with a growing number of users around the world has emerged due to these developments. This has resulted in to avalanche of information at our door step. The difficulty in finding what we want, when we need it, and in a manner which best meets our requirements is rapidly and continuously increasing. These days, for users, too many options to choose from is a common and constantly confronted situation. Users need help to explore and filter out their preferences from the myriad possibilities. Internet Search Engines, designed originally to be helpful, now commonly find many thousands of potentially relevant sites, thus losing their effectiveness (Montaner Rigall et al.).

A great amount of research on how Artificial Intelligence (AI) can help people to find out what they want on the Internet has been carried out by the AI community. As a result, users who require assistance in searching, sorting, classifying, filtering and sharing the vast amount of information now available on the Web have widely accepted the idea of recommender systems. The key task of a recommender system is to find items, information sources and people related to the interest and preferences of a single person or a group of people (Montaner Rigall et al.).

1.2 Problem Statement

“Enhancing Performance of Recommender Systems” deal with improving performance of recommender systems applicable to various domains. Goal of this work is to make recommendation methods more accurate and applicable to broader range of real-life needs.

Different applications generate different categories of data. Depending on nature of data, data processing varies. Suitable representation of data would also optimize performance of a recommender system.

Hence, need is to identify suitable data processing scheme along with the best suited data representation to optimize performance of recommender systems pertaining to different domains.

1.3 Organization of the Thesis

Rest of the thesis is organized as follows:

Chapter 2 (Background): This chapter presents an overview of the field of recommender systems. Three main recommendation approaches along with current generation recommender systems are discussed. However, literature survey specifically related to different recommender systems developed in this thesis, is discussed, whenever the corresponding chapter is elaborated.

Chapter 3 (Predicting Direction of Movement of Stock Price and Stock Market Index): This chapter deals with predicting movement of stock price and stock market index. Four prediction models, Artificial Neural Network (ANN), Support Vector Machine (SVM), Random Forest (RF) and naive Bayes (NB) with two approaches for input to these models are compared. Experiments are carried out on 10 years of historical data of two stocks and two stock market indices from Indian stock markets.

Chapter 4 (Predicting Stock Market Index using Fusion of Machine Learning Techniques): This chapter addresses the task of predicting future values of stock market index. A two stage fusion approach is proposed and three hybrid models are developed for the task. The chapter also discusses about experimental results and compares proposed fusion models with conventional single stage models.

Chapter 5 (Social Resource Recommendation using Learning from Positive and Unlabeled Examples): The problem of recommending social resources is focused in this chapter. Social resources are part of social resource sharing websites. Resources on these websites fall in to the category of positive and unlabeled data. Two step techniques and a direct method capable of learning from positive and unlabeled data is proposed in the chapter for the task of social resource recommendations.

Chapter 6 (Recommending Tags for new Resources in Social Bookmarking System): A tag recommendation algorithm based on the concept of multi-label classification is presented in the chapter. Impact of feature selection and representation of the resource on the performance of the recommender is also presented.

Chapter 7 (Movie Recommender System - Hybrid Filtering Approach): The focus of this chapter is movie recommendation task. Prediction task is modelled as classification task where the aim is to predict whether the movie will be liked or disliked by the user. An item based recommender which combines usage, tag and movie specific data such as genres, star cast and directors is proposed in this chapter.

Chapter 8 (Evaluating a Recommender learnt from Labeled and Unlabeled Data): The main objective of this chapter is to examine the influence on the accuracy of the recommender when it is built using unlabeled examples in addition to the labeled examples. Co-Training algorithm which allows to incorporate unlabeled examples while learning a classifier/recommender is discussed. Usefulness of this algorithm is investigated by means of experimental study using hetrec2011-movielens-2k data set.

Chapter 9 (Conclusions and Future Work): In this chapter, conclusions drawn from various implementations mentioned above are discussed. Future scope of work in this area is also mentioned in the chapter.

A separate section for the *Indexes* used in the thesis is covered towards the end.

The *Works Cited* section consists of related research work cited in the thesis work.

Chapter 2

Background

This chapter explores the general related work covering a few sub-areas of the work presented in this thesis. All related work precisely relevant to the work described in each of the following chapters is discussed in those respective chapters.

2.1 Recommender Systems

The explosive growth of the World Wide Web in the 1990s resulted in an equal growth of the amount of information available online, outgrowing the capacity of individual users to process all this information. A strong interest in the specific research fields and technology that could help manage this information overload was felt. Information Retrieval (IF) and Information Filtering (IF) are the highly related fields (Eijlander and Bogers).

Information Retrieval originated in the 1950s as a research field. It is concerned with automatically matching a users information need against a collection of documents. A change in focus from small document collections to the larger collections of realistic size was observed in 1990s. It was needed to cope with the ever-growing amount of information on the Web (Eijlander and Bogers). This change resulted to the advent of Information Filtering systems.

Users could make optimal use of their limited reading time by filtering out unwanted information in order to expose only the relevant information from a large flow of information, such as, news articles, with the help of IF systems (Hanani, Shapira, and Shoval). Typically, a model of a users interests is constructed by such systems. This model is then matched against the incoming information objects. Although, IR and IF are considered separate research fields, their focus is on analysing textual content (Belkin and Croft Eijlander and Bogers).

Recommender systems are third type of technology designed to combat information overload. A recommender system exploits a variety of information sources related to both the user and the content of items to identify set of items that are likely to be of interest to a certain user. Information filtering technology is aimed at removing items from the information stream (Hanani, Shapira, and Shoval). In contrast to this, recommender systems actively predict which items the user might be interested in and add them to the information flowing to the user (Eijlander and Bogers).

Recommender system can be viewed as one of the applications to predict rating or preference for the items that have not been seen by a user. This system can also produce a list of recommendations. The view is not just limited to this but can also include some predictions or forecasts that help users in making appropriate decisions (Adomavicius and Tuzhilin).

Based on how recommendations are made, recommender systems are usually classified into the following categories (Balabanović and Shoham Adomavicius and Tuzhilin):

- Content-based Recommender: The user is recommended items similar to the ones the user preferred in the past.
- Collaborative Recommender: The user is recommended items that people with similar tastes and preferences liked in the past.
- Hybrid Recommender: These recommenders combine collaborative and content-based methods. These type of recommenders can also be learnt based on combination of usage and content data.

2.2 Content-based Recommenders

The roots of content-based approach to recommendation are in information retrieval (Baeza-Yates, Ribeiro-Neto, et al. Salton) and information filtering (Belkin and Croft) research. Many current content-based systems focus on recommending items containing textual information, such as documents, web sites (URLs), and Usenet news messages. One of the reason behind this is the significant and early advancements made by the information retrieval and filtering communities. It is also because of the importance of several text-based applications.

Content-based filtering approaches rely on building some kind of representation of the content in a system. They also focus on learning a profile of a users interests. To find the items (products) that are most relevant to the user, the features extracted from the content representation of items are matched against users profile. This basic idea of content-based recommender is depicted in Figure 2.1.

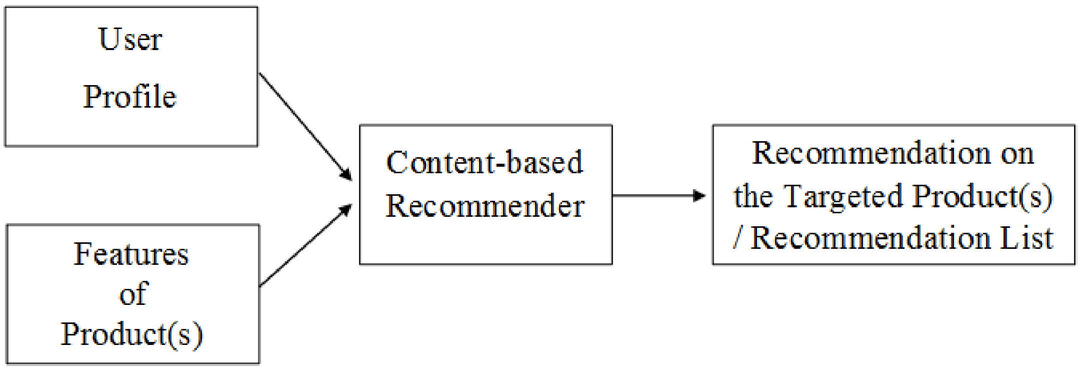

User Profile
Features of Product(s)
Content-based Recommender
Recommendation on the Targeted Product(s) / Recommendation List

Figure 2.1: Content-based Recommender

The representation of user profiles is long-term model. It is continuously updated as more preference information becomes available (Burke). Use of user profiles that contain information about users tastes, preferences, and needs allow content-based recommender systems to improve over the traditional information retrieval approaches. The profiling information can be further improved from users explicitly, e.g., through questionnaires, or implicitly - learnt from their transactional behavior over time (Adomavicius and Tuzhilin Eijlander and Bogers).

Generally, content-based filtering for recommendation is approached as either an IR problem or as a machine learning problem. When it is approached as an IR problem, document representations are matched to user representations on textual similarity. As a machine learning problem, the textual content of the representations are incorporated as feature vectors, which are then used to train a prediction algorithm (Eijlander and Bogers). (Whitman and Lawrence) and (Bogers and Van den Bosch) are couple of examples of IR approach while (Lang Pazzani and Billsus) and (Mooney and Roy) are examples of machine learning based approach.

Advantage of content-based filtering algorithms is that they do not require domain knowledge. Implicit feedback from the users about their item preferences is sufficient. This can make content-based filtering, the preferred algorithm in the domains where eliciting explicit ratings from users is difficult or cumbersome, and where domain knowledge is hard to come by. However, content-based algorithms are not suitable where content analysis is difficult or impractical. The user has to rate sufficient number of items before content-based recommender system can really understand the users preferences and present the user with reliable recommendations. Therefore, a new user, having very few ratings, would not be able to get accurate recommendations (Eijlander and Bogers), (Adomavicius and Tuzhilin).

2.3 Collaborative Recommenders

Collaborative recommenders predict the ratings of items for a particular user based on the items previously rated by other like-minded users. This basic idea of collaborative recommender is depicted in Figure 2.2.

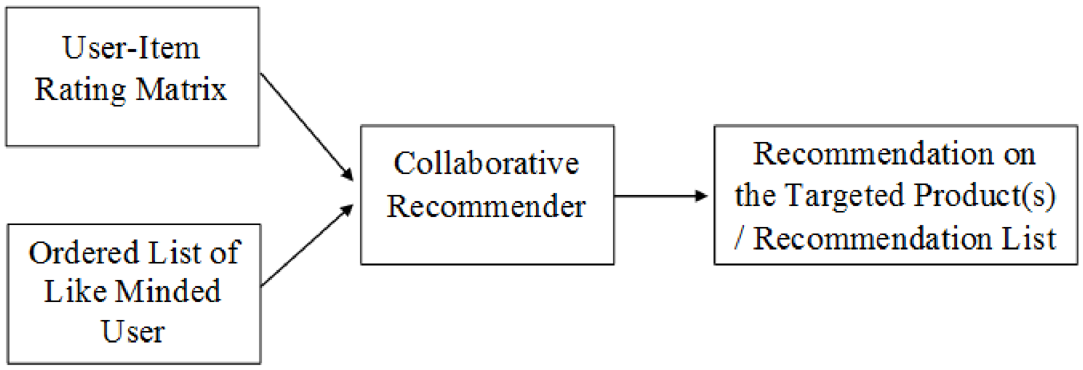

User-Item Rating Matrix
Ordered List of Like Minded User
Collaborative Recommender
Recommendation on the Targeted Product(s) / Recommendation List

Figure 2.2: Collaborative Recommender

Memory-based (or heuristic-based) and model-based are two general categories of collaborative recommenders (Breese, Heckerman, and Kadie).

Examples of memory-based collaborative recommenders are (Delgado and Ishii Nakamura and Abe Shardanand and Maes) and (Resnick et al.). These recommenders are essentially heuristics that make rating predictions based on the entire collection of previously rated items by the users. That is, the value of the unknown rating $r_{c,s}$ for user c and item s is usually computed as an aggregate of the ratings of some other (usually, the N most similar) users for the same item s (Adomavicius and Tuzhilin).

Approaches to model-based collaborative filtering are discussed in (Billsus and Pazzani, “Learning Collaborative Information Filters.” Getoor and Sahami Goldberg et al. Hofmann, “Collaborative filtering via gaussian probabilistic latent semantic analysis” Marlin Pavlov and Pennock) and (Ungar and Foster). These recommenders use the collection of ratings to learn a model, which is then used to make rating predictions (Adomavicius and Tuzhilin).

Domains where content analysis is difficult or costly such as movie and music recommendation, Collaborative Filtering (CF) algorithms are especially useful. New user, new item and sparsity are the problems associated with CF algorithms. New user problem is the same problem as with content-based recommenders. New items are added frequently to recommender systems. Collaborative recommender relies only on users preferences to make recommendations. Therefore, the recommender system would not be able to recommend the new item until it is rated by a substantial number of users. In any recommender system, the number of ratings already collected is usually very small compared to the number of ratings that need to be predicted. This leads to a situation where ratings are to be predicted only from the small number of available examples. The availability of a critical mass of users also influences the success of the collaborative recommender system. For example, in the movie recommendation system, there may be many movies that have been rated by only few people and these movies would be recommended very rarely, even if those few users gave high ratings to them. The other critical thing is that for the user whose taste is unusual compared to the rest of the population, there will not be many other like-minded users, leading to poor recommendations (Adomavicius and Tuzhilin).

2.4 Hybrid Recommenders

Certain limitations of content-based and collaborative systems can be overcome by using hybrid approach. Hybrid approach can combine collaborative and content-based methods. Different ways to combine collaborative and content-based methods into a hybrid recommender system can be classified as follows:

- Implementing collaborative and content-based methods separately and combining their predictions (Claypool et al. Pazzani Billsus and Pazzani, “User modeling for adaptive news access” Tran and Cohen).

- Incorporating some content-based characteristics into a collaborative approach (Balabanović and Shoham Good et al. Melville, Mooney, and Nagarajan).
- Incorporating some collaborative characteristics into a content-based approach (Soboroff and Nicholas).
- Constructing a general unifying model that incorporates both content-based and collaborative characteristics (Popescul, Pennock, and Lawrence Schein et al. Hofmann, "Probabilistic latent semantic indexing" Basu, Hirsh, Cohen, et al.).

All of the above approaches have been used by the researchers of recommender systems (Adomavicius and Tuzhilin).

However, despite these developments, the current generation of recommender systems still requires further improvements to make recommendation methods more accurate and applicable to an even broader range of real-life needs including recommending vacations, research articles, URLs, music artists, social tags and certain types of financial services to investors.

In the era of social networking and sharing, there is a distinct need of recommender systems to assist various activities on social networking sites. One possible example is to recommend set of appropriate tags to the scientific article or URL being bookmarked on social bookmarking site. There is also a need for personalized recommendations rather than recommending the same to the different mass of people. Possible direction for improvements may include better methods for representing user behaviour and the information about the items to be recommended. More advanced recommendation modelling methods, incorporation of various contextual information into the recommendation process and utilization of multi-criteria ratings are also worth exploring (Adomavicius and Tuzhilin).

Chapter 3

Predicting Direction of Movement of Stock Price and Stock Market Index

This study addresses problem of predicting direction of movement of stock price and stock market index for Indian stock markets. The study compares four prediction models, Artificial Neural Network (ANN), Support Vector Machine (SVM), Random Forest (RF) and Naive Bayes (NB) with two approaches for input to these models. The first approach for input data involves computation of ten technical parameters using stock trading data (open, high, low & close prices) while the second approach focuses on representing these technical parameters as trend deterministic data. Accuracy of each of the prediction models for each of the two input approaches is evaluated. Evaluation is carried out on 10 years of historical data from 2003 to 2012 of two stocks namely Reliance Industries and Infosys Ltd. and two stock market indices CNX Nifty and S&P Bombay Stock Exchange (BSE) Sensex. Experimental results suggest that for the first approach of input data where ten technical parameters are represented as continuous values, Random Forest outperforms other three prediction models on overall performance. Experimental results also show that the performance of all the prediction models improve when these technical parameters are represented as trend deterministic data.

3.1 Introduction

Predicting stock and stock price index is difficult due to uncertainties involved. There are two types of analysis which investors perform before investing in a stock. First is the fundamental analysis. In this, investors look at intrinsic value of stocks, performance of the industry and economy, political climate etc. to decide whether to invest or not. On the other hand, technical analysis is the evaluation of stocks by means of studying statistics generated by market activity, such as past prices and volumes. Technical analysts do not attempt to measure a security's intrinsic value but instead use stock charts to identify patterns and trends that may suggest how a stock will behave in the future. Efficient market hypothesis states that prices of stocks are informationally efficient; which means that it is possible to predict stock prices based on the trading data (Malkiel and Fama). This is quite logical as many uncertain factors like political scenario of country, public image of the company, etc. will start reflecting in the stock prices. So, if the information obtained from stock prices is pre-processed efficiently and appropriate algorithms are applied, trend of stock or stock price index may be predicted.

Since years, many techniques have been developed to predict stock trends. Initially, classical regression methods were used to predict stock trends. Since stock data can be categorized as non-stationary time series data, non-linear machine learning techniques have also been used. ANN (Mehrotra, Mohan, and Ranka) and SVM (Vapnik) are two machine learning algorithms which are most widely used for predicting stock and stock price index movement. Each algorithm has its own way to learn patterns. ANN emulates functioning of a human brain to learn by creating network of neurons while SVM uses the spirit of Structural Risk Minimization (SRM) principle.

3.2 Related Work

(Hassan, Nath, and Kirley) proposed and implemented a fusion model by combining the Hidden Markov Model (HMM), ANN and Genetic Algorithms (GA) to forecast financial market behaviour. Using ANN, the daily stock prices are transformed to independent sets of values that become input to HMM. (Wang and Leu) developed a prediction system useful in forecasting mid-term price trend in Taiwan stock market. Their system was based on a recurrent neural network trained by using features extracted from Autoregressive Integrated Moving Average (ARIMA) analysis. Empirical results showed that the network trained using 4-year weekly data was capable of predicting up to 6 weeks market trend with acceptable accuracy. Hybridized soft computing techniques for automated stock market forecasting and trend analysis was introduced in (Abraham, Nath, and Mahanti). They used Nasdaq-100 index of Nasdaq Stock Market with Neural Network for one day ahead stock forecasting and a neuro-fuzzy system for analysing the trend of the predicted stock values. The forecasting and trend prediction results using the proposed hybrid system were promising. (Chen, Leung, and Daouk) investigated the probabilistic neural network (PNN) to forecast the direction of index after it was trained by historical data. Empirical results showed that the PNN-based investment strategies obtained higher returns than other investment strategies. Other investment strategies that were examined include the buy-and-hold strategy as well as the investment strategies guided by forecasts estimated with the random walk model and the parametric GMM models.

A very well-known SVM algorithm developed by (Vapnik) searches for a hyper plane in higher dimension to separate classes. SVM is a very specific type of learning algorithm characterized by the capacity control of the decision function, the use of the kernel functions and the scarcity of the solution. (Huang, Nakamori, and Wang) investigated the predictability of SVM in forecasting the weekly movement direction of NIKKEI 225 index. They compared SVM with Linear Discriminant Analysis, Quadratic Discriminant Analysis and Elman Backpropagation Neural Networks. The experiment results showed that SVM outperformed the other classification methods. SVM was used in (Kim) to predict the direction of daily stock price change in the Korea Composite Stock Price Index (KOSPI). Twelve technical indicators were selected to make up the initial attributes. This study compared SVM with Back-propagation Neural Network (BPN) and Case-Based Reasoning (CBR). It was evident from the experimental results that SVM outperformed BPN and CBR.

Random Forest creates n classification trees using sample with replacement and predicts class based on what majority of trees predict. The trained ensemble, therefore, represents a single hypothesis. This hypothesis, however, is not necessarily contained within the hypothesis space of the models from which it is built. Thus, ensembles can be shown to have more flexibility in the functions they can represent. This flexibility can, in theory, enable them to over-fit the training data more than a single model would, but in practice, some ensemble techniques (especially bagging) tend to reduce problems related to over-fitting of the training data. (Tsai et al.) investigated the prediction performance of the classifier based on ensemble method to analyse stock returns. The hybrid methods of majority voting and bagging were considered. Moreover, performance using two types of classifier ensembles were compared with those using single baseline classifiers (i.e. Neural Networks, Decision Trees, and Logistic Regression). The results indicated that multiple classifiers outperform single classifiers in terms of prediction accuracy and returns on investment. (Sun and Li) proposed new financial distress prediction (FDP) method based on SVM ensemble. The algorithm for selecting SVM ensemble's base classifiers from candidate ones was designed by considering both individual performance and diversity analysis. Experimental results indicated that SVM ensemble was significantly superior to individual SVM classifier. (Ou and Wang) used total ten data mining techniques to predict price movement of Hang Seng index of Hong Kong stock market. The approaches included Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), K-Nearest Neighbor (KNN) Classification, Naive Bayes based on Kernel Estimation, Logit Model, Tree based Classification, Neural network, Bayesian Classification with Gaussian Process, SVM and Least Squares - Support Vector Machine (LS-SVM). Experimental results showed that the SVM and LS-SVM generated superior predictive performance among the other models.

It is evident from the above discussions that each of the algorithms in its own way can tackle this problem. It is also to be noticed that each of the algorithm has its own limitations. The final prediction outcome not only depends on the prediction algorithm used, but is also influenced by the representation of the input. Identifying important features and using only them as the input rather than all the features may improve the prediction accuracy of the prediction models. A two-stage architecture was developed in (Hsu et al.). They integrated Self-Organizing Map (SOM) and Support Vector Regression (SVR) for stock price prediction. They examined seven major stock market indices. Specifically, the self-organizing map was first used to decompose the whole input space into regions where data points with similar statistical distributions were grouped together, so as to contain and capture the non-stationary property of financial series. After decomposing heterogeneous data points into several homogenous regions, SVR was applied to forecast financial indices. The results suggested that the two stage architecture provided a promising alternative for stock price prediction. Genetic Programming (GP) and its variants have been extensively applied for modelling of the stock markets. To improve the generalization ability of the model, GP have been hybridized with its own variants (Gene Expression Programming (GEP), Multi Expression Programming (MEP)) or with the other methods such as Neural Networks and boosting.

The generalization ability of the GP model can also be improved by an appropriate choice of model selection criterion. (Garg, Sriram, and Tai) worked to analyse the effect of three model selection criteria across two data transformations on the performance of GP while modelling the stock indexed in the New York Stock Exchange (NYSE). Final Prediction Error (FPE) criteria showed a better fit for the GP model on both data transformations as compared to other model selection criteria. (Nair et al.) predicted the next day's closing value of five international stock indices using an adaptive ANN based system. The system adapted itself to the changing market dynamics with the help of Genetic Algorithm which tuned the parameters of the Neural Network at the end of each trading session.

The study in (Ahmed) investigated the nature of the causal relationships between stock prices and the key macro-economic variables representing real and financial sector of the Indian economy for the period March, 1995 to March, 2007 using quarterly data. The study revealed that the movement of stock prices was not solely dependent on behaviour of key macro-economic variables. (Mantri, Gahan, and Nayak) estimated the volatilities of Indian stock markets using Generalized Autoregressive Conditional Heteroskedasticity (GARCH), Exponential GARCH (EGARCH), Glosten-Jagannathan-Runkle GARCH (GJR-GARCH), Integrated GARCH (IGARCH) & ANN models. This study used fourteen years of data of BSE Sensex & NSE Nifty to estimate the volatilities. It was concluded that there was no difference in the volatilities of Sensex & Nifty estimated under the GARCH, EGARCH, GJR GARCH, IGARCH & ANN models.

(Mishra, Sehgal, and Bhanumurthy) tested for the presence of nonlinear dependence and deterministic chaos in the rate of returns series for six Indian stock market indices. The result of analysis suggested that the returns series did not follow a random walk process. Rather it appeared that the daily increments in stock returns were serially correlated and the estimated Hurst exponents were indicative of marginal persistence in equity returns. (Liu and Wang) investigated and forecast the price fluctuation by an improved Legendre Neural Network by assuming that the investors decided their investing positions by analysing the historical data on the stock market. They also introduced a random time strength function in the forecasting model. The Morphological Rank Linear Forecasting (MRLF) method was proposed by (Araújo and Ferreira). An experimental analysis was conducted and the results were compared to Multilayer Perceptron (MLP) networks and Time-delay Added Evolutionary Forecasting (TAEF) method.

This study focuses on comparing prediction performance of ANN, SVM, Random Forest and naive Bayes algorithms for the task of predicting stock and stock price index movement. Ten technical parameters are used as the inputs to these models. A Trend Deterministic Data Preparation Layer which converts continuous-valued inputs to discrete ones is proposed. Each input parameters in its discrete form indicates a possible up or down trend determined based on its inherent property. The focus is also to compare the performance of these prediction models, when the inputs are represented in the form of real values and trend deterministic data. All the experiments are carried out using 10 years of historical data of two stocks - Reliance Industries and Infosys Ltd. and two indices S&P BSE Sensex and CNX Nifty. Both stocks and indices are highly voluminous and vehemently traded in and so they reflect Indian Economy as a whole.

3.3 Research Data

Ten years of data of total two stock market indices (CNX Nifty and S&P BSE Sensex) and two stocks (Reliance Industries & Infosys Ltd.) from Jan 2003 to Dec 2012 is used in this study. All the data is obtained from http://www.nseindia.com/ and http://www.bseindia.com/ websites. These data form our entire data set. Percentage wise increase and decrease cases (days) of each year in the entire data set are shown in Table 3.1.

Table 3.1: Number of increase and decrease cases (days) percentage in each year in the entire data set of S&P BSE Sensex

Year	**Increase**	**Increase (%)**	**Decrease**	**Decrease (%)**	**Total**
2003	146	58.63	103	41.37	249
2004	136	54.18	115	45.82	251
2005	147	59.04	102	40.96	249
2006	148	59.92	99	40.08	247
2007	139	55.82	110	44.18	249
2008	114	46.72	130	53.28	244
2009	127	52.70	114	47.30	241
2010	134	53.39	117	46.61	251
2011	116	47.15	130	52.85	246
2012	128	51.82	119	48.18	247
Total	1335	53.94	1139	46.06	2474

This study uses 20% of the entire data as the parameter selection data. This data is used to determine design parameters of predictor models. Parameter selection data set is constructed by taking equal proportion of data from every year of the ten years. The proportion of percentage wise increase and decrease cases in each year is also maintained. This sampling method enables parameter setting data set to be better representative of the entire data set. The parameter selection data is further divided into training and hold-out set. Each of the set consists of 10% of the entire data. Table 3.2 depicts the number of increase and decrease cases (days) for parameter selection data set. These statistics are for S&P BSE Sensex. Similar data analysis is done for CNX Nifty, for Reliance Industries and Infosys Ltd.

Optimum parameters for predictor models are obtained by means of experiments on parameter selection data. After that, for comparing ANN, SVM, Random Forest and Naive Bayes, comparison data set is devised. This data set comprises of entire ten years of data. It is also divided in training (50% of the entire data) and hold-out (50% of the entire data) set. Details of this data set of S&P BSE Sensex is shown in Table 3.3. These experimental settings are same as in (Kara, Acar Boyacioglu, and Baykan).

Table 3.2: Number of increase and decrease cases (days) in each year in the parameter setting data set of S&P BSE Sensex

Year	**Training (Days)**			**Holdout (Days)**		
	Increase	**Decrease**	**Total**	**Increase**	**Decrease**	**Total**
2003	15	10	25	15	10	25
2004	14	11	25	14	11	25
2005	15	10	25	15	10	25
2006	15	10	25	15	10	25
2007	14	11	25	14	11	25
2008	11	13	24	11	13	24
2009	13	11	24	13	11	24
2010	13	12	25	13	12	25
2011	12	13	25	12	13	25
2012	13	12	25	13	12	25
Total	135	113	248	135	113	248

There are some technical indicators through which one can predict the future movement of stocks. Here in this study, total ten technical indicators as employed in (Kara, Acar Boyacioglu, and Baykan) are used. These indicators are shown in Table 3.4. Two approaches for the representation of the input data are employed in this study. The first approach uses continuous-valued representation, i.e., the actual time series, while the second one uses trend deterministic representation (which is discrete in nature) for the inputs. Both the representations are discussed here.

Table 3.3: Number of increase and decrease cases (days) in each year in the comparison data set of S&P BSE Sensex

Year	**Training (Days)**			**Holdout (Days)**		
	Increase	**Decrease**	**Total**	**Increase**	**Decrease**	**Total**
2003	73	52	125	72	52	124
2004	68	58	126	67	58	125
2005	74	51	125	73	51	124
2006	74	50	124	73	50	123
2007	70	55	125	69	55	124
2008	57	65	122	57	65	122
2009	64	57	121	63	57	120
2010	67	59	126	66	59	125
2011	58	65	123	58	65	123
2012	64	60	124	63	60	123
Total	669	572	1241	661	572	1233

3.3.1 Continuous Representation - The Actual Time Series

Ten technical indicators calculated based on the formula as shown in Table 3.4 are given as inputs to predictor models. It is evident that each of the technical indicators calculated based on the above mentioned formula is continuous-valued. The values of all technical indicators are normalized in the range between [-1, +1], so that larger value of one indicator do not overwhelm the smaller valued indicator. Performance of all the models under study is evaluated for this representation of inputs.

3.3.2 Discrete Representation - Trend Deterministic Data

A new layer of decision is employed, which converts continuous-valued technical parameters to discrete value, representing the trend. This layer, proposed in this study, is thereby named as the "Trend Deterministic Data Preparation Layer".

Table 3.4: Selected technical indicators & their formula (Kara, Acar Boyacioglu, and Baykan)

Name of Indicators	Formula
Simple n(10 *here*)-day Moving Average (SMA)	$\frac{C_t+C_{t-1}+\cdots+C_{t-9}}{n}$
n(10 *here*)-day Exponential Moving Average (EMA)	$EMA(k)_t = EMA(k)_{t-1} + \alpha \times (C_t - EMA(k)_{t-1})$
Momentum (MOM)	$\mathrm{C}_t - C_{t-9}$
Stochastic K% (STCK%)	$\frac{C_t-LL}{HH-LL} \times 100$
Stochastic D% (STCD%)	$\frac{\sum_{i=0}^{n-1} K_{t-i}}{10}\%$
Relative Strength Index (RSI)	$100 - \frac{100}{1+(\sum_{i=0}^{n-1} UP_{t-i}/n)/(\sum_{i=0}^{n-1} DW_{t-i}/n)}$
Moving Average Convergence Divergence (MACD)	$MACD(n)_{t-1} + \frac{2}{n+1} \times (DIFF_t - MACD(n)_{t-1})$
Larry William's R% (LWR)	$\frac{HH-C_t}{HH-LL} \times -100$
Accumulation/Distribution (A/D) Oscillator (ADO)	$\frac{[(H_t-O_t)+(C_t-L_t)]}{[2\times(H_t-L_t)]} \times 100$
Commodity Channel Index (CCI)	$\frac{M_t-SM_t}{0.015D_t}$

Here, C_t is the closing price, O_t is the opening price, L_t is the low price and H_t the high price of t^{th} day, $DIFF_t = EMA(12)_t - EMA(26)_t$, EMA is Exponential Moving Average, $EMA(k)_t = EMA(k)_{t-1} + \alpha \times (C_t - EMA(k)_{t-1})$, α is a smoothing factor which is equal to $\frac{2}{k+1}$, k is the time period of k-day Exponential Moving Average, $EMA(k)$ is the k-day moving average, LL and HH imply the lowest low and the highest high during the look back period (10 days here), respectively. $M_t = \frac{H_t+L_t+C_t}{3}$, $SM_t = \frac{(\sum_{i=1}^{n} M_{t-i+1})}{n}$, $D_t = \frac{\sum_{i=1}^{n} |M_{t-i+1} - SM_t|)}{n}$, UP_t means upward price change while DW_t is the downward price change at time t.

Each technical indicator has its own inherent property through which traders generally predict the stock's up or down movement. The job of this new layer is to convert this continuous values to '+1' or '-1' by considering this property during the discretization process. This way, the input data to each of the predictor models is converted to '+1' and '-1', where '+1' indicates up movement and '-1' shows down movement.

Simple Moving Average (SMA) and Exponential Moving Average (EMA) help smooth price action and filter out the noise. SMA and EMA of 10-days will hug prices quite closely and turn shortly after prices turn. So when SMA and EMA at time t is greater than at time $t-1$, it suggests up movement for stock i.e. '+1' and vice-a-versa for down movement i.e. '-1'.

Stochastic K% (STCK%), Stochastic D% (STCD%) and Larry Williams R% are stochastic oscillators. These oscillators are clear trend indicators for any stock. When stochastic oscillators are increasing, the stock prices are likely to go up and vice-a-versa (Kim). MACD, RSI, CCI and A/D oscillators also follow the stock trend. Using these indicator values, the trend deterministic input set is prepared and given to the predictor models. Performance of all the models under study is evaluated, for this representation of inputs also.

3.4 Prediction Models

3.4.1 ANN Model

Inspired by functioning of biological neural networks, ANN are a dense network of inter-connected neurons which get activated based on inputs. A three layer feed-forward neural network is employed in this study (Mehrotra, Mohan, and Ranka Han, Kamber, and Pei). Inputs for the network are ten technical indicators which are represented by ten neurons in the input layer. Output layer has a single neuron with log-sigmoid as the transfer function. This results in a continuous value output between 0 and 1. A threshold of 0.5 is used to determine the up or down movement prediction. For the output value greater than or equal to 0.5, prediction is considered to be the up movement, else the down movement. Each of the hidden layer's neurons employed tan-sigmoid as the transfer function. The architecture of the three-layered feed-forward ANN is illustrated in Figure 3.1.

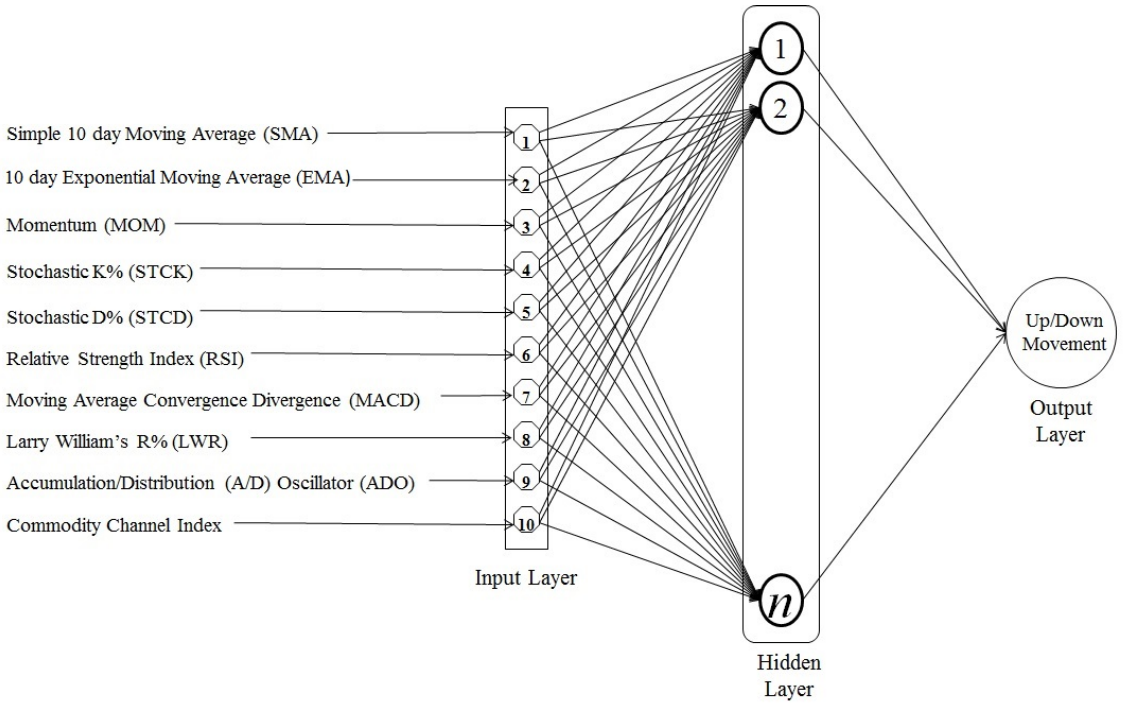

Simple 10 day Moving Average (SMA)
10 day Exponential Moving Average (EMA)
Momentum (MOM)
Stochastic K% (STCK)
Stochastic D% (STCD)
Relative Strength Index (RSI)
Moving Average Convergence Divergence (MACD)
Larry William's R% (LWR)
Accumulation/Distribution (A/D) Oscillator (ADO)
Commodity Channel Index
1
2
3
4
5
6
7
8
9
10
Input Layer
1
2
n
Hidden Layer
Up/Down Movement
Output Layer

Figure 3.1: Architecture of ANN model (Kara, Acar Boyacioglu, and Baykan)

Gradient descent with momentum is used to adjust the weights, in which, at each epochs, weights are adjusted, so that a global minimum can be reached. Comprehensive parameter setting experiments to determined parameters for each stocks and indices are performed in this study. The ANN model parameters are number of hidden layer neurons (n), value of learning rate (lr), momentum constant (mc) and number of epochs (ep). To determine them efficiently, ten levels of n, nine levels of mc and ten levels of ep are tested in the parameter setting experiments. Initially, value of lr is fixed to 0.1. These parameters and their values which are tested are summarized in Table 3.5.

These settings of parameters yield a total of $10 \times 10 \times 9 = 900$ treatments for ANN for one stock. Considering two indices and two stocks, total of 3600 treatments for ANN are carried out. The top three parameter combinations that resulted in the best average of training and holdout performance are selected as the top three ANN models for comparison experiments on comparison data set. For these top performing models, learning rate lr is varied in the interval of [0.1, 0.9].

Table 3.5: ANN parameters and their values tested in parameter setting experiments

Parameters	**Value(s)**
Number of Hidden Layer Neurons (n)	$10{,}20{,}\cdots,100$
Epochs (ep)	$1000, 2000, \cdots, 10000$
Momentum Constant (mc)	$0.1, 0.2, \cdots, 0.9$
Learning Rate (lr)	0.1

3.4.2 SVM Model

SVM was first introduced by (Vapnik). There are two main categories for SVMs: Support Vector Classification (SVC) and Support Vector Regression (SVR). SVM is a learning system using a high dimensional feature space. (Khemchandani, Chandra, et al.) stated that in SVM, points are classified by means of assigning them to one of the two disjoint half spaces, either in the pattern space or in a higher-dimensional feature space.

The main objective of SVM is to identify maximum margin hyper plane. The idea is that the margin of separation between positive and negative examples is maximized (Xu, Zhou, and Wang).

SVM finds maximum margin hyper plane as the final decision boundary. Assume that $x_i \in R^d, i = 1, 2, \cdots, N$ forms a set of input vectors with corresponding class labels $y_i \in \{+1, -1\}, i = 1, 2, \cdots, N$. SVM can map the input vectors $x_i \in R^d$ into a high dimensional feature space $\Phi(x_i) \in H$. A kernel function $K(x_i, x_j)$ performs the mapping $\phi(\cdot)$. The resulting decision boundary is defined in Equation 3.1.

$$f(x) = sgn(\sum_{i=1}^{N} y_i\alpha_i \cdot K(x, x_i) + b) \tag{3.1}$$

To get the values of α_i, involved in Equation 3.1, quadratic programming problem shown in Equation 3.2 is solved.

$$\begin{aligned} Maximize \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j \cdot y_iy_j \cdot K(x_i, x_j) \\ Subject\ \ to\ \ 0 \leq \alpha_i \leq c \\ \sum_{i=1}^{N} \alpha_i y_i = 0, i = 1, 2, \ldots, N \end{aligned} \tag{3.2}$$

The trade-off between margin and misclassification error is controlled by the regularization parameter c. The polynomial and radial basis kernel functions are used in this study and they are shown in Equations 3.3 and 3.4 respectively.

$$Polynomial \quad Function : K(x_i, x_j) = (x_i \cdot x_j + 1)^d \tag{3.3}$$

$$Radial \quad Basis \quad Function : K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2) \tag{3.4}$$

where d is the degree of polynomial function and γ is the constant of radial basis function.

Choice of kernel function, degree of kernel function (d) in case of polynomial kernel, gamma in kernel function (γ) in case of radial basis kernel and regularization constant (c) are considered as the parameters of SVM in this study. To determine them efficiently, four levels on d, ten levels of γ and four to five levels of c are tested in the parameter setting experiments. These parameters and their values which are tested are summarized in Table 3.6.

For one stock, these settings of parameters yield a total of 20 and 40 treatments for SVM employing polynomial and radial basis kernel functions respectively. Considering two indices and two stocks, total of 240 treatments for SVM are carried out. One parameter combination for each of the polynomial kernel SVM and radial basis kernel SVM, that resulted in the best average of training and holdout performance is selected as the top SVM model for comparison experiments.

Table 3.6: SVM parameters and their values tested in parameter setting experiments

Parameters	**Values (polynomial)**	**Values (radial basis)**
Degree of Kernel Function (d)	$1, 2, 3, 4$	-
Gamma in Kernel Function (γ)	-	$0.5, 1.0, 1.5, \cdots, 5.0, 10.0$
Regularization Parameter (c)	0.5, 1, 5, 10, 100	0.5, 1, 5, 10

3.4.3 Random Forest

Decision tree learning is one of the most popular techniques for classification. Its classification accuracy is comparable with other classification methods, and it is very efficient. The classification model learnt through these techniques is represented as a tree and is known as a decision tree. ID3 (Quinlan, "Induction of decision trees"), C4.5 (Quinlan, *C4. 5: programs for machine learning*) and CART (Breiman et al.) are decision tree learning algorithms. Details can be found in (Han, Kamber, and Pei).

Random Forest belongs to the category of ensemble learning algorithms (Breiman). It uses decision tree as the base learner of the ensemble. The idea of ensemble learning is that a single classifier is not sufficient for determining class of test data. Reason being, based on sample data, classifier is not able to distinguish between noise and pattern. So, it performs sampling with replacement, such that, given n trees that are to be learnt, are based on these data set samples. In the experiments performed in this study, each tree is learnt using 3 features selected randomly. After creation of n trees, when testing data is used, the decision which majority of trees come up with is considered as the final output. This also avoids problem of over-fitting. Implementation of random forest algorithm in this study is summarized in the Algorithm 1.

Number of trees ($ntrees$) in the ensemble is considered as the parameter of Random Forest. To determine it efficiently, it is varied from 10 to 200 with increment of 10 each time during the parameter setting experiments. For one stock, these settings of parameter yield a total of 20 treatments. Considering two indices and two stocks, total of 80 treatments are carried out. The top three parameter values that resulted in the best average of training and holdout performance are selected as the top three Random Forest models for the comparison experiments.

Algorithm 1 Random Forest

```
Input: training set $D$, number of trees in the ensemble $k$
Output: a composite model $M*$
1: for $i = 1$ to $k$ do
2:     Create bootstrap sample $D_i$ by sampling $D$ with replacement
3:     Select 3 features randomly
4:     Use $D_i$ and randomly selected three features to derive tree $M_i$
5: end for
6: return $M*$
```

3.4.4 Naive Bayes Classifier

Naive Bayes classifier assumes class conditional independence (Han, Kamber, and Pei Markov and Larose). Given test data, Bayesian classifier predicts the probability of data belonging to a particular class. To predict probability it uses concept of Bayes' theorem. Bayes' theorem is useful in calculating the posterior probability, $P(C|X)$, from $P(C)$, $P(X|C)$, and $P(X)$. Bayes' theorem states that

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)} \tag{3.5}$$

Here, $P(C|X)$ is the posterior probability which tells us the probability of hypothesis C being true given that event X has occurred. In this work, hypothesis C is the probability of belonging to class Up/Down and event X is our test data. $P(X|C)$ is a conditional probability of occurrence of event X, given hypothesis C is true. It can be estimated from the training data. The working of naive Bayesian classifier, or simple Bayesian classifier, is summarized as follows.

Assume that, m classes $C_1, C_2, \ldots, C_m$ and event of occurrence of test data, X, is given. Bayesian classifier classifies the test data into a class with the highest probability. By Bayes' theorem (Equation (3.5)),

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}, i = 1, 2, \cdots, m \tag{3.6}$$

Given data sets with many attributes $(A_1, A_2, ..., A_n)$, it would be extremely computationally expensive to compute $P(X|C_i)$. In order to reduce computation in evaluating $P(X|C_i)$, the naive assumption of class conditional independence is made. This presumes that the values of the attributes are conditionally independent of one another, given the class label of the tuple (i.e. that there are no dependence relationships among the attributes). Therefore,

$$\begin{aligned} P(X|C_i) &= \prod_{k=1}^{n} P(x_k|C_i) \\ &= P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i) \end{aligned} \tag{3.7}$$

Here, x_k denotes the value of attribute A_k for tuple X. Computation of $P(x_k|C_i)$ depends on whether it is categorical or continuous. If A_k is categorical, $P(x_k|C_i)$ is the number of observations of class C_i in training set having the value x_k for A_k divided by the number of observations of class C_i in the training set. If A_k is continuous-valued, Gaussian distribution is fitted to the data and the value of $P(x_k|C_i)$ is calculated based on Equation 3.8.

$$\begin{aligned} &f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \\ &so\ that, \\ &P(x_k|C_i) = f(x_k, \mu_{C_i}, \sigma_{C_i}) \end{aligned} \tag{3.8}$$

Here, μ_{C_i} and σ_{C_i} are the mean (i.e., average) and standard deviation, respectively, of the values of attribute A_k for training tuples of class C_i. These two quantities are then plugged into Equation 3.8 together with x_k, in order to estimate $P(x_k|C_i)$. $P(X|C_i)P(C_i)$ is evaluated for each class C_i in order to predict the class label of X. The class label of observation X is predicted as class C_i, if and only if

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \quad for\ 1 \le j \le m; j \ne i \tag{3.9}$$

Bayesian classifiers also serve as a theoretical justification for other classifiers that do not explicitly use Bayes' theorem. For example, under specific assumptions, it can be demonstrated that many neural networks and curve-fitting algorithms output the maximum posteriori hypothesis, as does the naive Bayesian classifier.

3.5 Experimental Evaluation

This section discusses about evaluation measures, experimental methodology and results of the experimentations.

3.5.1 Evaluation Measures

Accuracy and F-measure are used to evaluate the performance of proposed models. Computation of these evaluation measures require estimating Precision and Recall which are evaluated from True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN) (Han, Kamber, and Pei Markov and Larose). These parameters are defined in Equations 3.10, 3.11, 3.12 and 3.13.

$$Precision_{positive} = \frac{TP}{TP + FP} \tag{3.10}$$

$$Precision_{negative} = \frac{TN}{TN + FN} \tag{3.11}$$

$$Recall_{positive} = \frac{TP}{TP + FN} \tag{3.12}$$

$$Recall_{negative} = \frac{TN}{TN + FP} \tag{3.13}$$

Precision is the weighted average of precision positive and negative while Recall is the weighted average of recall positive and negative. Accuracy and F-measure are estimated using Equations 3.14 and 3.15 respectively. It is to be noticed that, the value range of these measures is between 0 and 1, where 0 indicates the worst performance while 1 indicates the best performance.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.14}$$

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.15}$$

Accuracy of a classifier on a given test set is the percentage of test set observations that are correctly classified by the classifier. It is also to be mentioned that F-measure used throughout the thesis is traditional/balanced F-measure or F_1-score.

3.5.2 Experimental Methodology & Results

Experiments are carried out in two phases.

First phase of experimentation considers input as the continuous-valued data. The best parameter combinations are identified by means of experiments on parameter setting data set for each of the prediction models. These parameter combinations with corresponding accuracy and f-measure during parameter setting experiments are reported in Tables 3.7, 3.8 and 3.9. It is to be noted that there are no parameters to be tuned for naive Bayes classifier.

Comparison data set (which is a whole data set) is used to compare the performance of various prediction models. 50% observations of these data set are used to learn various prediction models along with the parameters identified during parameter tuning experiments. Remaining 50% observations are used as holdout observations and accuracy reported in the chapter shows the accuracy on these holdout observations. Table 3.10 reports average accuracy and f-measure of each of the models during comparison experiment. Average accuracy and f-measure reported are averaged over the top performing models. It can be seen that naive Bayes with the Gaussian process is the least accurate while Random Forest is the most accurate with average accuracy of nearly 84%. Figure 3.2 depicts the prediction process when data is continuous-valued.

Second phase of experimentation is identical to the first one except that the input to the models is trend deterministic data. The idea is depicted in Figure 3.3. Tables 3.11, 3.12 and 3.13 show result of best performing combinations for ANN, SVM and Random Forest respectively during parameter setting experiments. It is to be noted that when data is represented as trend deterministic data, naive Bayes classifier is learnt by fitting multivariate Bernoulli distribution to the data. Results on comparison data set for all the proposed models are reported in Table 3.14. Final comparison shows that all the models perform well with discrete data input but SVM, random forest and naive Bayes performance better than ANN. The accuracy of SVM, Random Forest and naive Bayes is nearly 90%.

3.6 Discussions

Stock market data is an example of non-stationary data. At particular time there can be trends, cycles, random walks or combinations of the three. It is desired that if a particular year is part of a cycle, say a bullish one, the proposed models should follow this pattern for trend prediction. Same can be considered for a trending year. However, usually stock values of a particular year are not isolated and there are days with random walks. Stock values are also affected by external factors creating trends and state of the country's economy. Political scenarios are also the influencing factors which may result in cycles.

Table 3.7: Best three parameter combinations of ANN model and their performance on continuous-valued parameter setting data set

CNX Nifty			
	ep:*n*:*mc* & *lr*=0.1		
	10000:20:0.6	7000:10:0.7	7000:10:0.9
Accuracy	0.8434	0.8450	0.8558
F-measure	0.8614	0.8606	0.8686
S&P BSE Sensex			
	ep:*n*:*mc* & *lr*=0.1		
	1000:80:0.1	2000:40:0.2	10000:100:0.1
Accuracy	0.7968	0.7827	0.7723
F-measure	0.7743	0.7982	0.7862
Infosys Ltd.			
	ep:*n*:*mc* & *lr*=0.1		
	1000:70:0.7	8000:150:0.7	3000:10:0.3
Accuracy	0.7417	0.7023	0.6949
F-measure	0.7581	0.7098	0.7412
Reliance Industries			
	ep:*n*:*mc* & *lr*=0.1		
	8000:50:0.6	6000:40:0.4	9000:20:0.5
Accuracy	0.6356	0.6326	0.6898
F-measure	0.6505	0.6116	0.7067

It can be seen from the results that all the models perform well when they are learnt from continuous-valued inputs but the performance of each of the models is further improved when they are learnt using trend deterministic data. The reason behind the improved performance is justified in the remainder of this section.

Table 3.8: Best two parameter combinations (one for each type of kernel) of SVM model and their performance on continuous-valued parameter setting data set

CNX Nifty		
	Kernel:Polynomial	Kernel:RBF
	c=100,d=1	c=0.5,γ=5
Accuracy	0.8427	0.8057
F-measure	0.8600	0.8275
S&P BSE Sensex		
	Kernel:Polynomial	Kernel:RBF
	c=100,d=1	c=0.5,γ=5
Accuracy	0.8136	0.7823
F-measure	0.8321	0.8015
Infosys Ltd.		
	Kernel:Polynomial	Kernel:RBF
	c=0.5,d=1	c=0.5,γ=5
Accuracy	0.8139	0.7836
F-measure	0.8255	0.7983
Reliance Industries		
	Kernel:Polynomial	Kernel:RBF
	c=0.5,d=1	c=1,γ=5
Accuracy	0.7669	0.6881
F-measure	0.7761	0.7023

Trend deterministic data is prepared by discretizing the continuous-valued data. The idea is based on the intuition that each continuous-valued parameters when compared with its previous day's value indicates the future up or down trend. The data is discretized based on these heuristics. When this data is given as the input to the model, we are already inputting the trend based on each input parameters. It is actually the situation where each of the input parameters signifies about the probable future trend and we have the actual future trend to identify the transformation from probable trends to the correct trend. This is a step forward because original dataset is converted to trend deterministic data set. Now prediction models have to determine co-relation between the input trends and output trend. Though it is non-linear, it is easy to create a model which can transform input trends to the output trend.

Table 3.9: Best three parameter combinations of random forest model and their performance on continuous-valued parameter setting data set

CNX Nifty			
	ntrees		
	140	20	30
Accuracy	0.9148	0.9146	0.9099
F-measure	0.9186	0.9185	0.9162
S&P BSE Sensex			
	ntrees		
	80	50	70
Accuracy	0.8819	0.8719	0.8786
F-measure	0.8838	0.8742	0.8802
Infosys Ltd.			
	ntrees		
	50	110	200
Accuracy	0.8138	0.8059	0.8132
F-measure	0.8202	0.8135	0.8190
Reliance Industries			
	ntrees		
	160	60	150
Accuracy	0.7368	0.7441	0.7450
F-measure	0.7389	0.7474	0.7478

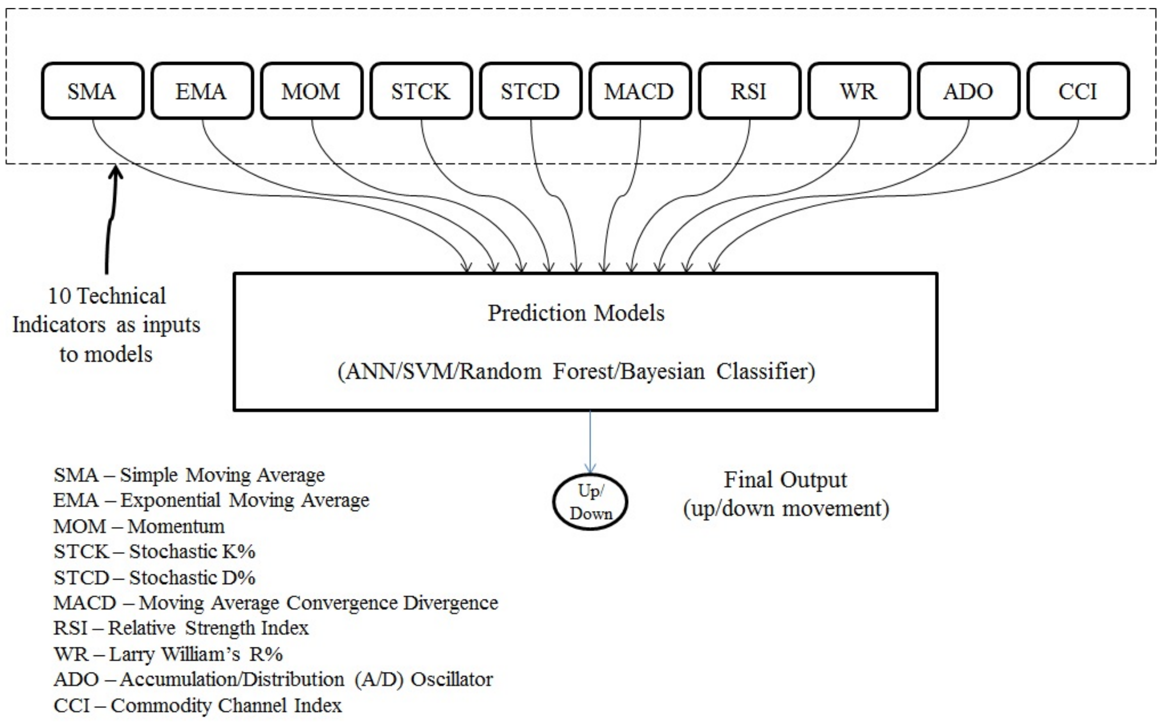

SMA
EMA
MOM
STCK
STCD
MACD
RSI
WR
ADO
CCI
10 Technical Indicators as inputs to models
Prediction Models
(ANN/SVM/Random Forest/Bayesian Classifier)
Up/Down
Final Output (up/down movement)
SMA – Simple Moving Average
EMA – Exponential Moving Average
MOM – Momentum
STCK – Stochastic K%
STCD – Stochastic D%
MACD – Moving Average Convergence Divergence
RSI – Relative Strength Index
WR – Larry William's R%
ADO – Accumulation/Distribution (A/D) Oscillator
CCI – Commodity Channel Index

Figure 3.2: Predicting with continuous-valued data

Table 3.10: Performance of prediction models on continuous-valued comparison data set

Stock/Index	**Prediction Models**			
	ANN (Kara, Acar Boyacioglu, and Baykan)		**SVM**	
	Accuracy	F-measure	Accuracy	F-measure
S&P BSE Sensex	0.7839	0.7849	0.7979	0.8168
CNX Nifty 50	0.8481	0.8635	0.8242	0.8438
Reliance Industries	0.6527	0.6786	0.7275	0.7392
Infosys Ltd.	0.7130	0.7364	0.7988	0.8119
Average	0.7494	0.7659	0.7871	0.8029
Stock/Index	**Random Forest**		**Naive Bayes(Gaussian)**	
	Accuracy	F-measure	Accuracy	F-measure
S&P BSE Sensex	0.8775	0.8794	0.7354	0.7547
CNX Nifty 50	0.9131	0.9178	0.8097	0.8193
Reliance Industries	0.7420	0.7447	0.6565	0.6658
Infosys Ltd.	0.8110	0.8176	0.7307	0.7446
Average	0.8359	0.8399	0.7331	0.7461

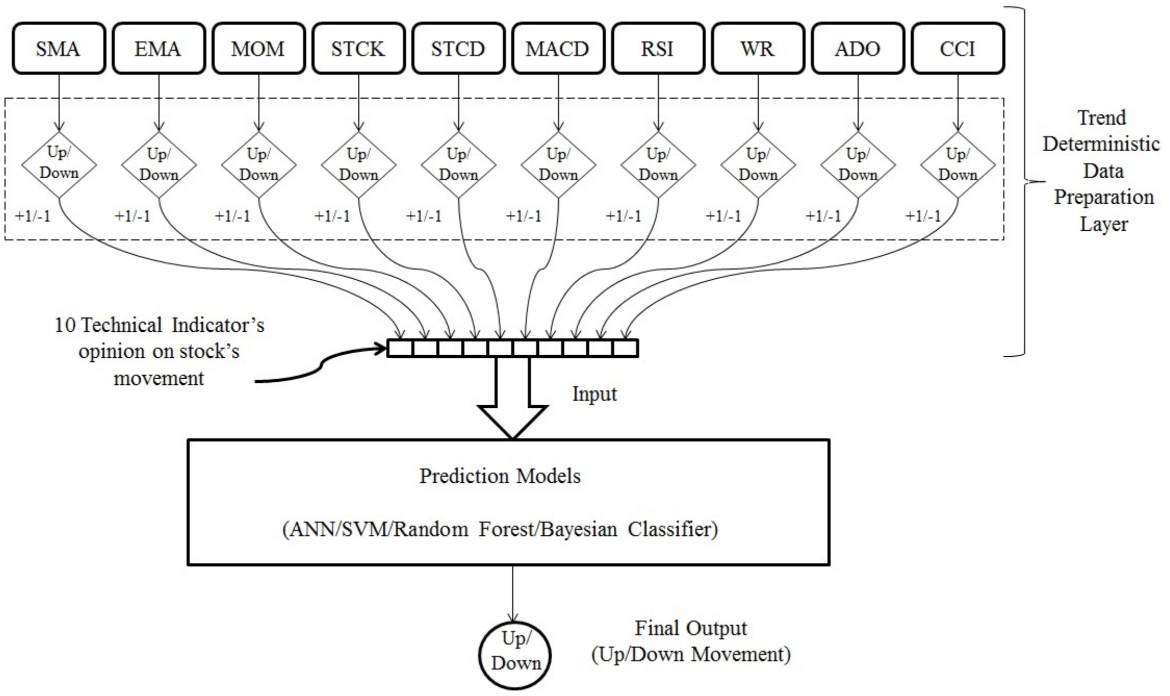

SMA
EMA
MOM
STCK
STCD
MACD
RSI
WR
ADO
CCI
Up/ Down
+1/-1
Trend Deterministic Data Preparation Layer
10 Technical Indicator's opinion on stock's movement
Input
Prediction Models
(ANN/SVM/Random Forest/Bayesian Classifier)
Final Output (Up/Down Movement)

Figure 3.3: Predicting with trend deterministic data

Further to notice is that for any stock or index, there are scenarios, when a stock or index is trading at some value, say 200, then due to some external factors, it may start trading at higher price, say 400, and then stabilize at that higher value. If prediction model is given direct continuous-valued input, it is possible that it tries to establish relation between the values in 200 and that in 400, which is not required as far as predicting future trend is considered.

Each parameter is relative while signifying future trend. It means that the important thing is how its value has changed with respect to previous days rather than the absolute value of change. Therefore, trend deterministic data which is discrete in nature is basically the statistical indication of whether the stocks are over-bought or over-sold and is value independent. Hence, these input parameters, when represented as probable future trends serve as a better measure of stocks condition rather than the scenario, when they are represented as continuous-valued.

3.7 Conclusions

The task focused in this study is to predict direction of movement for stocks and stock price indices. Prediction performance of four models namely ANN, SVM, Random Forest and Naive Bayes is compared based on ten years (2003-2012) of historical data of CNX Nifty, S&P BSE Sensex, Infosys Ltd. and Reliance Industries from Indian stock markets. Ten technical parameters reflecting the condition of stock and stock price index are used to learn each of these models.

Table 3.11: Best three parameter combinations of ANN model and their performance on discrete-valued parameter setting data set

CNX Nifty			
	ep:*n*:*mc* & *lr*=0.2		
	4000:50:0.8	1000:100:0.6	3000:70:0.3
Accuracy	0.8703	0.8740	0.8729
F-measure	0.8740	0.8768	0.8801
S&P BSE Sensex			
	ep:*n*:*mc* & *lr*=0.1		
	6000:100:0.4	2000:30:0.3	4000:90:0.1
Accuracy	0.8563	0.8728	0.8717
F-measure	0.8632	0.8771	0.8759
Infosys Ltd.			
	ep:*n*:*mc* & *lr*=0.1		
	6000:50:0.1	4000:70:0.2	9000:80:0.4
Accuracy	0.8531	0.8717	0.8468
F-measure	0.8600	0.8742	0.8503
Reliance Industries			
	ep:*n*:*mc* & *lr*=0.2		
	1000:100:0.1	4000:90:0.9	8000:100:0.5
Accuracy	0.8573	0.8747	0.8808
F-measure	0.8620	0.8799	0.8826

Table 3.12: Best two parameter combinations (one for each type of kernel) of SVM model and their performance on discrete-valued parameter setting data set

	CNX Nifty	
	Kernel:Polynomial	Kernel:RBF
	c=1,d=1	c=1,γ=4
Accuracy	0.9010	0.8808
F-measure	0.9033	0.8838
	S&P BSE Sensex	
	Kernel:Polynomial	Kernel:RBF
	c=1,d=1	c=5,γ=1.5
Accuracy	0.8959	0.8780
F-measure	0.8980	0.8810
	Infosys Ltd.	
	Kernel:Polynomial	Kernel:RBF
	c=0.5,d=1	cc=1,γ=3
Accuracy	0.8895	0.8865
F-measure	0.8916	0.8880
	Reliance Industries	
	Kernel:Polynomial	Kernel:RBF
	c=1,d=1	c=0.5,γ=4
Accuracy	0.9221	0.8923
F-measure	0.9229	0.8932

A Trend Deterministic Data Preparation Layer is proposed on the basis of the fact that each technical indicator has its own inherent property through which traders generally predict the stocks' up or down movement. Utilizing these heuristics, the trend deterministic data preparation layer converts each of the technical parameter's continuous value to +1 or −1, indicating probable future up or down movement respectively.

Table 3.13: Best three parameter combinations of random forest model and their performance on discrete-valued parameter setting data set

CNX Nifty			
	*ntrees*s		
	30	120	20
Accuracy	0.8913	0.8973	0.8969
F-measure	0.8934	0.8990	0.9005
S&P BSE Sensex			
	*ntrees*s		
	20	90	110
Accuracy	0.8886	0.8981	0.9011
F-measure	0.8914	0.9012	0.9028
Infosys Ltd.			
	*ntrees*s		
	50	60	70
Accuracy	0.9035	0.8964	0.9004
F-measure	0.9051	0.8980	0.9019
Reliance Industries			
	*ntrees*s		
	30	10	40
Accuracy	0.9079	0.9088	0.9070
F-measure	0.9085	0.9098	0.9078

Experiments with continuous-valued data show that naive Bayes (Gaussian Process) model exhibits least performance with 73.3% accuracy and random forest with highest performance of 83.56% accuracy. Performance of all these models is improved significantly when they are learnt through trend deterministic data. ANN is slightly less accurate in terms of prediction accuracy compare to other three models which perform almost identically. The accuracy of 86.69%, 89.33%, 89.98% and 90.19% is achieved by ANN, SVM, Random Forest and Naive Bayes (Multivariate Bernoulli Process) respectively.

Table 3.14: Performance of prediction models on discrete-valued comparison data set

Stock/Index	**Prediction Models**			
	ANN		**SVM**	
	Accuracy	F-measure	Accuracy	F-measure
S&P BSE Sensex	0.8669	0.8721	0.8869	0.8895
CNX Nifty 50	0.8724	0.8770	0.8909	0.8935
Reliance Industries	0.8709	0.8748	0.9072	0.9080
Infosys Ltd.	0.8572	0.8615	0.8880	0.8898
Average	0.8669	0.8714	0.8933	0.8952
Stock/Index	**Random Forest**		**Naive Bayes**	
	Accuracy	F-measure	Accuracy	F-measure
S&P BSE Sensex	0.8959	0.8985	0.8984	0.9026
CNX Nifty 50	0.8952	0.8977	0.8952	0.8990
Reliance Industries	0.9079	0.9087	0.9222	0.9234
Infosys Ltd.	0.9001	0.9017	0.8919	0.8950
Average	0.8998	0.9017	0.9019	0.9050

Trend Deterministic Data Preparation Layer proposed in this chapter exploits inherent opinion of each of the technical indicators about stock price movement. The layer exploits these opinions in the same way as the stock market's experts. In earlier researches, the technical indicators were used directly for prediction while this study first extracts trend related information from each of the technical indicators and then utilizes the same for prediction, resulting in significant improvement in accuracy. The proposal of this Trend Deterministic Data Preparation Layer is a distinct contribution to the research. Owing to the noteworthy improvement in the prediction accuracy, the proposed system can be deployed in real time for stocks' trend prediction, making investments more profitable and secure.

Chapter 4

Predicting Stock Market Index using Fusion of Machine Learning Techniques

The study focuses on the task of predicting future values of stock market index. Two indices namely CNX Nifty and S&P BSE Sensex from Indian stock markets are selected for experimental evaluation. Experiments are based on 10 years of historical data of these two indices. The predictions are made for 1 to 10, 15 and 30 days in advance. A two stage fusion approach is proposed in this study. First stage employs SVR for preparing data for the second stage. The second stage of the fusion approach uses ANN, Random Forest (RF) and SVR resulting in to SVR-ANN, SVR-RF and SVR-SVR fusion prediction models. The prediction performance of these hybrid models is compared with the single stage scenarios where ANN, RF and SVR are used single-handedly. Ten technical indicators are selected as the inputs to each of the prediction models.

4.1 Introduction and Literature Review

Prediction of stock prices is a classic problem. Efficient market hypothesis states that it is not possible to predict stock prices and that stocks behave in random walk manner. But technical analysts believe that most information about the stocks are reflected in recent prices, and so, if trends in the movements are observed, prices can be easily predicted. In addition, stock market's movements are affected by many macro-economical factors such as political events, firms' policies, general economic conditions, commodity price index, bank rate, bank exchange rate, investors' expectations, institutional investors' choices, movements of other stock market, psychology of investors, etc. (MIAO, CHEN, and ZHAO). Value of stock indices are calculated based on stocks with high market capitalization. Various technical parameters are used to gain statistical information from value of stocks prices. Stock indices are derived from prices of stocks with high market capitalization and so they give an overall picture of economy and depends on various factors.

There are several different approaches to time series modelling. Traditional statistical models including moving average, exponential smoothing, and ARIMA are linear in their predictions of the future values (Rao and Gabr Hsieh Bollerslev). Extensive research has resulted in numerous prediction applications using ANN, Fuzzy Logic, Genetic Algorithms (GA) and other techniques (Lee and Tong Hadavandi, Shavandi, and Ghanbari Zarandi, Hadavandi, and Turksen). ANN SVR are two machine learning algorithms which have been most widely used for predicting stock price and stock market index values. Each algorithm has its own way to learn patterns. (Zhang and Wu) incorporated the Backpropagation neural network with an Improved Bacterial Chemotaxis Optimization (IBCO). They demonstrated the ability of their proposed approach in predicting stock index for both short term (next day) and long term (15 days). Simulation results exhibited the superior performance of proposed approach. A combination of data preprocessing methods, GA and Levenberg Marquardt (LM) algorithm for learning feed forward neural networks was proposed in (Asadi et al.). They used data pre-processing methods such as data transformation and selection of input variables for improving the accuracy of the model. The results showed that the proposed approach was able to cope with the fluctuations of stock market values and also yielded good prediction accuracy. The Artificial Fish Swarm Algorithm (AFSA) was introduced in (Shen et al.) to train Radial Basis Function Neural Network (RBFNN). Their experiments on the stock indices of the Shanghai Stock Exchange indicated that RBFNN optimized by AFSA was an easy-to-use algorithm with considerable accuracy. (Hadavandi, Ghanbari, and Abbasian-Naghneh) proposed a hybrid artificial intelligence model for stock exchange index forecasting. The model was a combination of GA and feed forward Neural Network.

The Support Vector Machine (SVM) introduced by (Vapnik) has gained popularity and is regarded as a state-of-the-art technique for regression and classification applications. (Kazem et al.) proposed a forecasting model based on chaotic mapping, firefly algorithm, and SVR to predict stock market price. SVR-CFA model which was newly introduced in their study, was compared with SVR-GA , SVR-CGA (Chaotic GA), SVR-FA (Firefly Algorithm), ANN and ANFIS models and the results showed that SVR-CFA model performed better than other models. (Pai et al.) developed a Seasonal Support Vector Regression (SSVR) model to forecast seasonal time series data. Hybrid Genetic Algorithms and Tabu Search (GA/TS) algorithms were applied in order to select three parameters of SSVR models. They also applied two other forecasting models, ARIMA and SVR for forecasting on the same data sets. Empirical results indicated that the SSVR outperformed both SVR and ARIMA models in terms of forecasting accuracy. By integrating GA based optimal time-scale feature extractions with SVM, (Huang and Wu) developed a novel hybrid prediction model that operated for multiple time-scale resolutions and utilized a flexible nonparametric regressor to predict future evolutions of various stock indices. In comparison with Neural Networks, pure SVMs and traditional GARCH models, the proposed model performed the best. The reduction in root-mean-squared error was significant. Financial time series prediction using ensemble learning algorithms in (Cheng, Xu, and Wang) suggested that ensemble algorithms were powerful in improving the performances of base learners. The study by (Aldin, Dehnavr, and Entezari) evaluated the effectiveness of using technical indicators, such as Moving Average, RSI, CCI, MACD, etc. in predicting movements of Tehran Exchange Price Index (TEPIX).

This study focuses on the task of predicting future values of stock market indices. The predictions are made for 1 to 10, 15 and 30 days in advance. A two stage fusion approach involving (SVR in the first stage is proposed. The second stage of the fusion approach uses ANN, Random Forest and SVR resulting in SVR-ANN, SVR-RF and SVR-SVR prediction models. The prediction performance of these hybrid models is compared with the single stage scenarios where ANN, RF and SVR are used single-handedly.

4.2 Single Stage Approach

The basic idea of single stage approach is illustrated in Figure 4.1. It can be seen that for the prediction task of n-day ahead of time, inputs to prediction models are ten technical indicators describing t^{th}-day while the output is $(t+n)^{th}$-day's closing price. These technical indicators which are used as inputs are summarized in Table 3.4. The prediction models which are employed in this study are described in the following sub-sections.

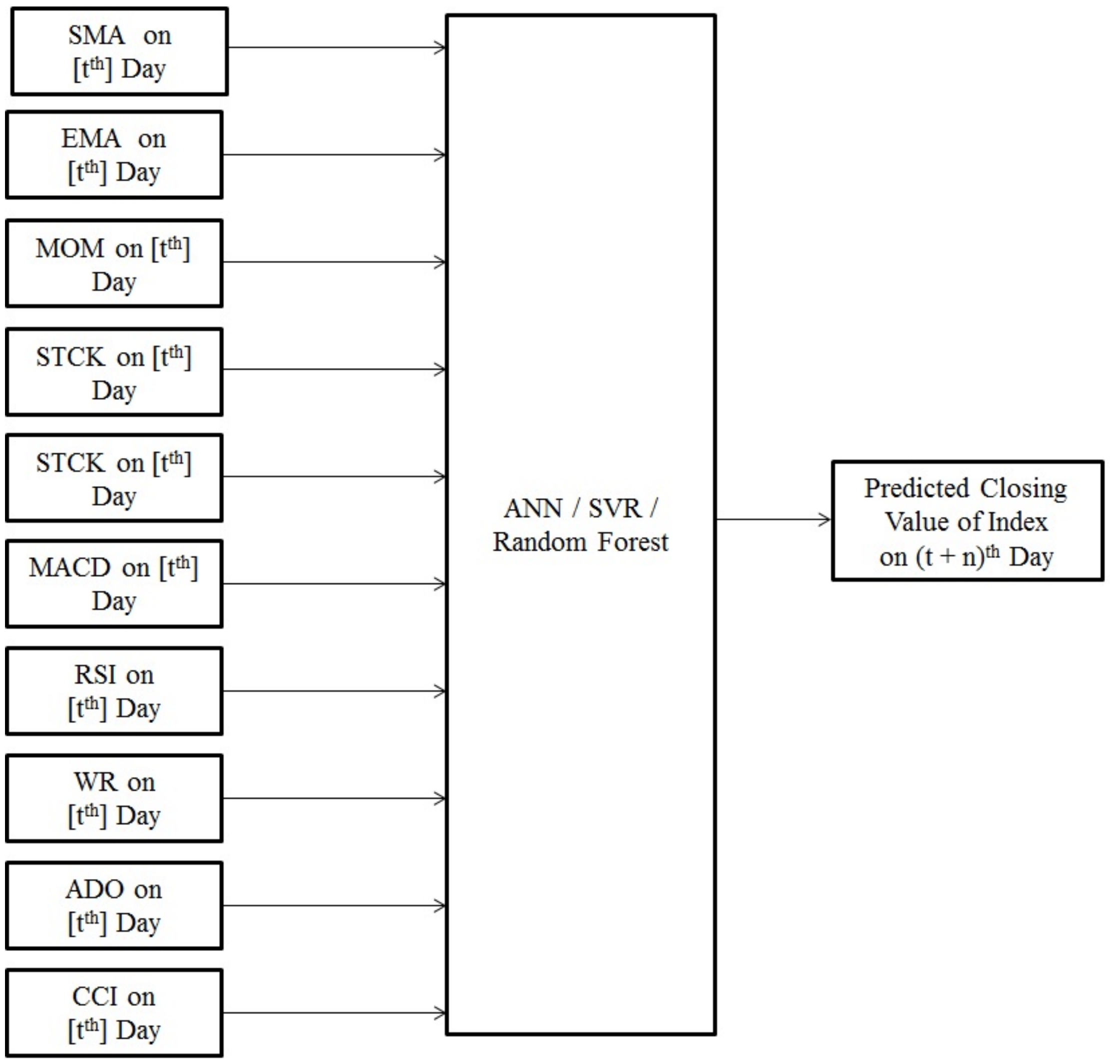


Figure 4.1: General architecture of single stage approach for predicting n day ahead of time

4.2.1 Artificial Neural Network

Three layer feed forward back propagation ANN similar to that shown in Figure 3.1 is employed in this study (Mehrotra, Mohan, and Ranka Han, Kamber, and Pei). The only difference is that, the transfer function of the neuron, in the output layer, is linear. This neuron in the output layer predicts closing price/value instead of the up/down movement as was the case in the previous chapter. Input layer has ten neurons, one for each of the selected technical parameters. The value of the index which is to be predicted is represented by the single neuron in the output layer. Adaptive gradient descent is used as the weight update algorithm. A tan-sigmoid is used as the transfer function of the neurons of the hidden layer. The output of the model is a continuous value, signifying the predicted value of the index. The reason behind using adaptive gradient descent is to allow learning rate to change during the training process. It may improve the performance of the gradient descent algorithm. In adaptive gradient descent, first, the initial network output and error are calculated. The current learning rate is used to calculate new weights and biases at each epoch. Based on these new weights and biases, new outputs and errors are calculated. If the new error exceeds the old error by more than a predefined ratio (1.04, in this study), the new weights and biases are discarded and the learning rate is decreased (to 70% of its current value, in this study). Otherwise, new weights and biases are kept and the learning rate is increased (by 5% of the current value, in the experiments reported in this thesis).

The procedure ensures that the learning rate is increased only to the extent that the network can learn without large increases in error. This allows to obtain near optimal learning rate for the local terrain. At the same time, as long as stable learning is assured, learning rate is increased. When it is too high to assure a decrease in error, it is decreased until stable learning resumes.

Number of neurons in the hidden layer and number of epochs are considered as the parameters of the model. Comprehensive number of experiments are carried out by varying the parameter values as shown in Table 4.1.

Table 4.1: ANN parameters and their values tested

Parameters	Values
Number of Hidden Layer Neurons (n)	10,20,$\cdots$,100
Epochs (ep)	1000, 2000, $\cdots$,10000

4.2.2 Support Vector Regression

The SVR uses the same principles as the SVM for classification, with only a few minor differences (Vapnik). First of all, because output is a real number, it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance ϵ is set in approximation to the SVM. Up until the threshold ϵ, the error is considered 0. However, the main idea is always the same: to minimize error, individualizing the hyper plane which maximizes the margin, considering that, part of the error is tolerated (Parrella).

The basic concepts of SVR which are discussed here can also be found in (Cristianini and Shawe-Taylor Kecman) and (Huang and Tsai). Assume that $x_i \in R^d, i = 1, 2, \cdots, m$ forms a set of input vectors with corresponding response variable $y_i \in R, i = 1, 2, \cdots, m$. SVR builds the linear regression function as shown in Equation 4.1.

$$f(x, w) = w^T x + b \tag{4.1}$$

Equation 4.2 shows Vapnik's linear $\varepsilon-$Insensitivity loss function.

$$|y - f(x, w)|_\varepsilon = \begin{cases} 0, & if \;\; |y - f(x, w)| \leq \varepsilon \\ |y - f(x_i, w)| - \varepsilon, & otherwise \end{cases} \tag{4.2}$$

Based on this, linear regression $f(x, w)$ is estimated by simultaneously minimizing $||w||^2$ and the sum of the linear $\varepsilon-$Insensitivity losses as shown in Equation 4.3. The constant c controls a trade-off between an approximation error and the weight vector norm $||w||$.

$$R = \frac{1}{2}||w||^2 + c(\sum_{i=1}^{m} |y - f(x, w)|_\varepsilon) \tag{4.3}$$

Minimizing the risk R is equivalent to minimizing the risk shown in Equation 4.4 under the constraints illustrated in Equation 4.5, 4.6 and 4.7. Here, ξ_i and ξ_i^* are slack variables, one for exceeding the target value by more than ε and other for being more than ε below the target.

$$R = \frac{1}{2}||w||^2 + c\sum_{i=1}^{m}(\xi + \xi^*) \tag{4.4}$$

$$(w^T x_i + b) - y_i \leq \varepsilon + \xi_i \tag{4.5}$$

$$y_i - (w^T x_i + b) \leq \varepsilon + \xi_i^* \tag{4.6}$$

$$\xi_i, \xi_i^* \geq 0, i = 1, 2, \ldots, m \tag{4.7}$$

Similar to SVM, above constrained optimization problem is solved using Lagrangian theory and the Karush-Kuhn-Tucker conditions to obtain the desired weight vector of the regression function. SVR can map the input vectors $x_i \in R^d$ into a high dimensional feature space $\Phi(x_i) \in H$. A kernel function $K(x_i, x_j)$ performs the mapping $\phi(\cdot)$. The polynomial and radial basis kernel functions are used here and they are shown in Equations 4.8 and 4.9 respectively.

$$Polynomial \quad Function : K(x_i, x_j) = (x_i \cdot x_j + 1)^d \tag{4.8}$$

$$Radial \quad Basis \quad Function : K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2) \tag{4.9}$$

Here, d is the degree of polynomial function and γ is the constant of radial basis function. Choice of kernel function, degree of kernel function (d) in case of polynomial kernel, gamma in kernel function (γ) in case of radial basis kernel and regularization constant (c) are considered as the parameters of SVR. Comprehensive number of experiments are carried out by varying the parameter values as shown in Table 4.2.

Table 4.2: SVR parameters and their values tested

Parameters	Value(s)
Degree of Kernel Function (d)	$1, 2, 3, 4$
Gamma in Kernel Function (γ)	$0, 0.5, 1, 1.5, 2, 2.5, 3, 4, 5, 10, 20, 50, 100$
Regularization Parameter (c)	1

4.2.3 Random Forest

It is already discussed in section 3.4.3. The only difference in the implementation here is that instead of classification tree, regression tree is used as the base learner of the ensemble.

Number of trees (*ntrees*) in the ensemble is considered as the parameter of Random Forest. Experiments are carried out with 50, 100 and 150 number of trees.

4.3 Two Stage Fusion Approach

The basic idea of two stage fusion approach is illustrated in Figure 4.2. The first stage employs SVRs to prepare inputs for the prediction models employed in the second stage.

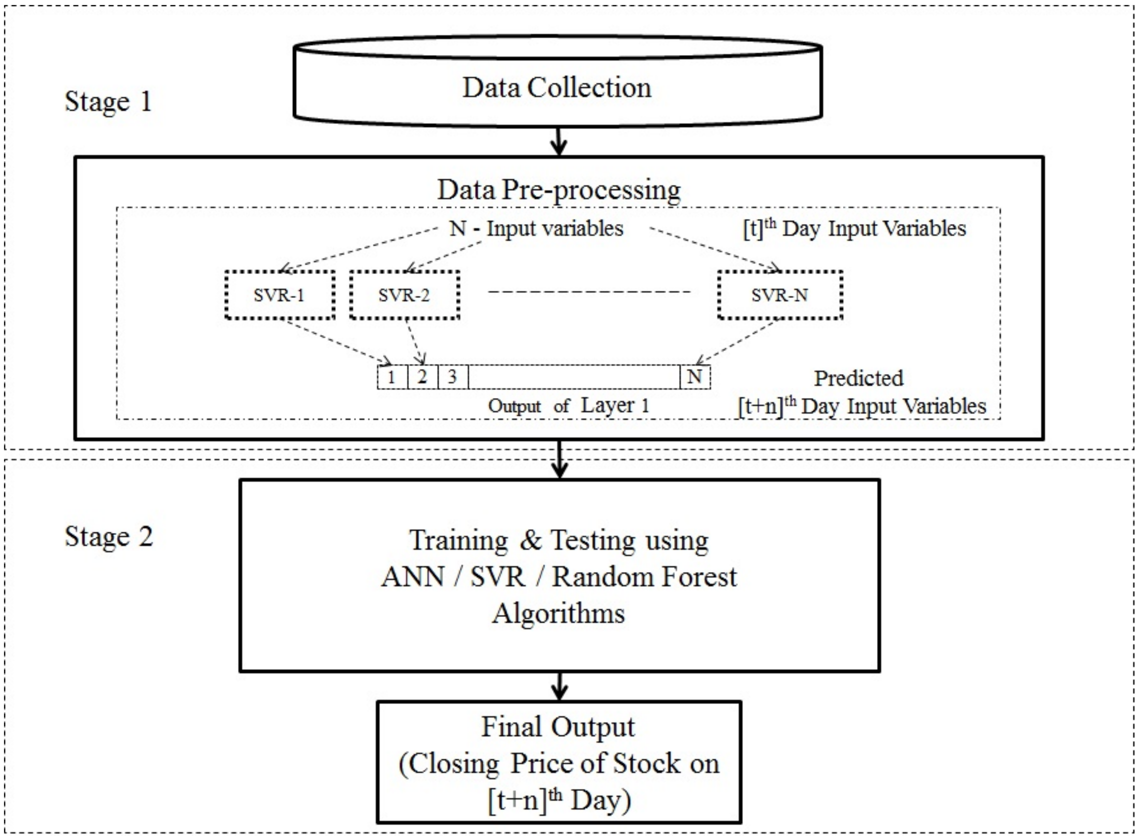

Stage 1
Data Collection
Data Pre-processing
N - Input variables
[t]th Day Input Variables
SVR-1
SVR-2
SVR-N
1 2 3 N
Predicted [t+n]th Day Input Variables
Output of Layer 1
Stage 2
Training & Testing using ANN / SVR / Random Forest Algorithms
Final Output (Closing Price of Stock on [t+n]th Day)

Figure 4.2: General architecture of two stage fusion approach for predicting n day ahead of time

Details about inputs and outputs to these SVRs, for the prediction task of n-day ahead of time, are depicted in Figure 4.3.

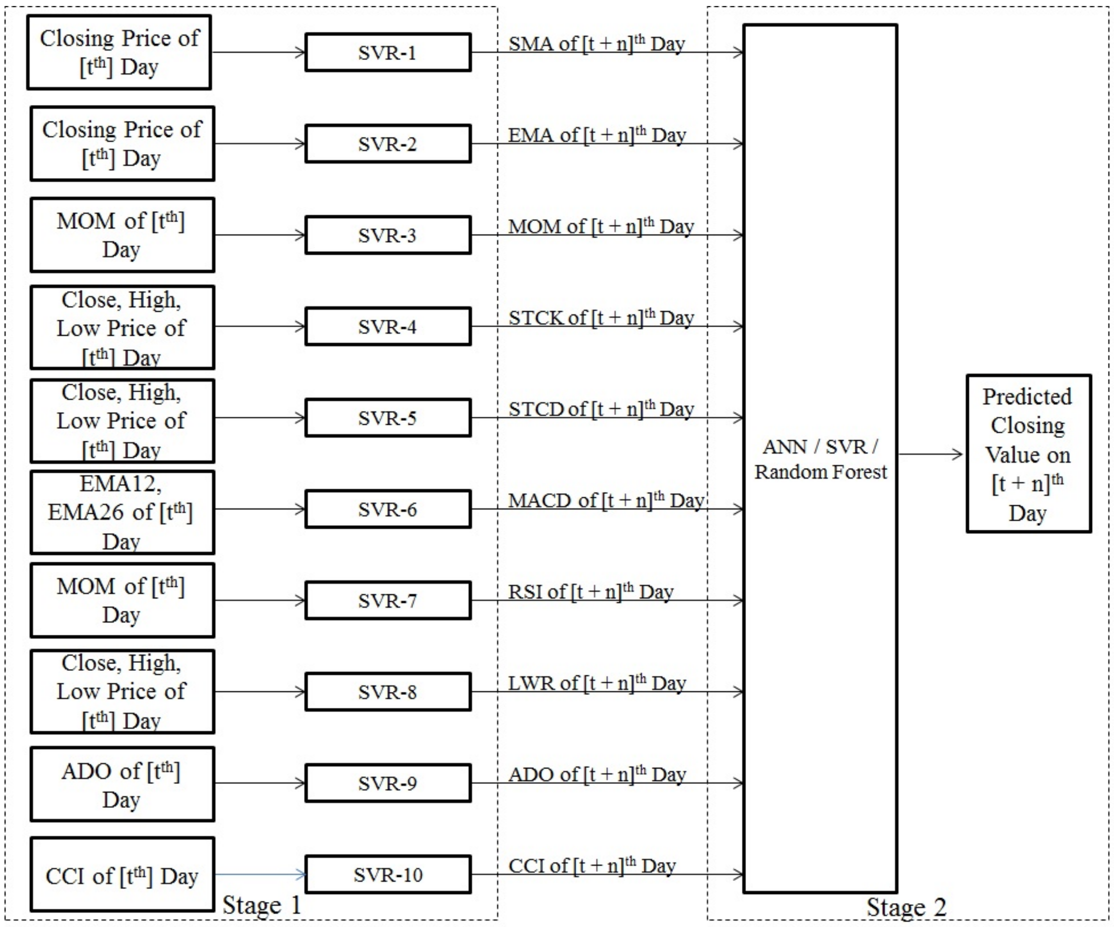


Figure 4.3: Details of two stage fusion approach for predicting n day ahead of time

It is to be noticed that inputs to the SVRs in the first stage describe t^{th} day while outputs of this stage describe $(t+n)^{th}$-day in terms of ten technical indicators. These outputs from the first stage serve as the inputs to the prediction models in the second stage. This leads to the situation where prediction models in the second stage have to identify mapping transformation from technical parameters describing $(t+n)^{th}$ day to $(t+n)^{th}$ day's closing price. This is different from single stage approach, where, prediction models have to identify mapping transformation from technical parameters describing t^{th} day to $(t+n)^{th}$ day's closing price. It can be seen that the final output in both the approaches is the closing value of the $(t+n)^{th}$ day. As shown in Figures 4.2 and 4.3, ANN, SVR and Random Forest are employed as the prediction models in the second stage. Comprehensive number of experiments are carried out for each of the prediction models in second stage by varying the parameter values in the same manner as in the single stage approach.

Table 4.3: Best parameter combination reported by parameter tuning experiments for each of the SVRs in first stage of two stage fusion approach

SVR	**Kernel Function**	**Gamma(γ)**
SVR-1	RBF	2
SVR-2	RBF	2
SVR-3	RBF	2
SVR-4	RBF	100
SVR-5	RBF	100
SVR-6	RBF	10
SVR-7	RBF	4
SVR-8	RBF	100
SVR-9	RBF	2
SVR-10	RBF	1.5

However, parameter tuning experiments are performed for each of the SVRs in the first stage to decide best combination of parameter values. A parameter tuning data set is formed as the 20% data of the entire data set. The parameter tuning data set is further divided in training and testing set. Training data set consists of 80% of the parameter tuning data set while remaining of the parameter tuning data forms the testing data set. By means of experiments on these training and testing set, best combination of parameter values for each of the SVRs in the first stage is identified. In this study, these experiments are called as parameter tuning experiments, for the SVRs in the first stage. The possible values which are considered for each of the parameters of these SVRs are same as shown in Table 4.2.

Results of parameter tuning experiments for each of the SVRs in the first stage show that transformation of input space through RBF kernel performs better than the transformation through polynomial kernel. The best parameter combinations as reported by parameter tuning experiments for each of the SVRs in the first stage are summarized in Table 4.3. It is to be noticed that the aim of the parameter tuning experiments is to identify best parameter combination for each of the SVRs in first stage, so that, error in statistical parameters which are to be used as inputs to the prediction models in second stage is minimized. During the overall experiments of stock market value predictions, SVRs in the first stage are used with the parameters determined during the parameter tuning experiments.

4.4 Experimental Evaluation

4.4.1 Data for Experimentation

This study uses total ten years of historical data from Jan 2003 to Dec 2012 of two stock market indices CNX Nifty and S&P BSE Sensex which are highly voluminous. The ten technical indicators used are calculated from close, high, low and opening prices of these indices. All the data is obtained from http://www.nseindia.com/ and http://www.bseindia.com/ websites.

4.4.2 Evaluation Measures

Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), relative Root Mean Squared Error (rRMSE) and Mean Squared Error (MSE) are used to evaluate the performance of these prediction models and their formulas are shown in Equations 4.10, 4.11, 4.12 and 4.13. It is to be noticed that MAPE is measured in terms of %.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|A_t - F_t|}{|A_t|} \times 100 \tag{4.10}$$

$$MAE = \frac{1}{n}\sum_{t=1}^{n}\frac{|A_t - F_t|}{|A_t|} \tag{4.11}$$

$$rRMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\left(\frac{A_t - F_t}{A_t}\right)^2} \tag{4.12}$$

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(A_t - F_t)^2 \tag{4.13}$$

where A_t is actual value and F_t is forecast value.

4.4.3 Results and Discussions

Three prediction models namely ANN, SVR and RF are used in single stage approach. In two stage fusion approach, the prediction models that are used are SVR-ANN, SVR-SVR and SVR-RF. For both the approaches, prediction experiments for 1 to 10, 15 and 30 days ahead of time are carried out. Results for CNX Nifty are shown in Tables from 4.4 to 4.15. Similar results for S&P BSE Sensex are depicted in Tables 4.16 to 4.27.

It is important to notice that for each of the prediction tasks and prediction models, comprehensive number of experiments are carried out, for different possible combinations of model parameters. The values reported in the tables are the best parameter combinations where minimum prediction error is exhibited.

It is evident from the result that, as predictions are made for more and more number of days in advance, error values increase. This may be obvious for any prediction system. Proposed two stage fusion models SVR-ANN and SVR-RF outperform ANN and RF models for almost all prediction tasks for both the data sets. SVR-SVR outperforms SVR for all the prediction tasks except for the prediction tasks up to 3 to 4 days in advance.

Table 4.28 and 4.29 compares performance of single stage models to two stage fusion models for CNX Nifty. The reported values in these tables are averaged over all 12 prediction tasks (1 to 10, 15 and 30 days in advance). Similar results for S&P BSE Sensex are summarised in Table 4.30 and 4.31. Tables 4.28 & 4.30 compare single stage prediction models to two stage fusion models on the basis of average prediction performance while Tables 4.29 & 4.31 show percentage improvement in performance achieved by two stage fusion prediction models over single stage prediction models.

Table 4.4: Prediction performance of 1-day ahead of time (for CNX Nifty)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	3000	1.91	102.06	2.48	17745.90
SVR-ANN	7000	1.50	79.05	1.93	10006.12
	γ	MAPE	MAE	rRMSE	MSE
SVR	4.00	0.99	52.48	1.26	4427.05
SVR-SVR	5.00	1.47	77.63	1.87	9614.30
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	50	1.36	72.45	1.68	8086.79
SVR-Ranom Forest	150	1.29	69.01	1.64	7710.16

Table 4.5: Prediction performance of 2-days ahead of time (for CNX Nifty)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	3000	1.92	101.79	2.42	16399.21
SVR-ANN	10000	1.66	87.82	2.13	12299.75
	γ	MAPE	MAE	rRMSE	MSE
SVR	2.50	1.40	74.15	1.78	8748.11
SVR-SVR	5.00	1.61	85.14	2.09	12104.73
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	50	1.80	95.69	2.24	14206.36
SVR-Ranom Forest	150	1.55	82.34	1.96	10832.60

Table 4.6: Prediction performance of 3-days ahead of time (for CNX Nifty)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	6000	2.16	113.79	2.75	20668.78
SVR-ANN	4000	1.86	98.25	2.38	15736.44
	γ	MAPE	MAE	rRMSE	MSE
SVR	0.00	1.76	93.13	2.22	13556.21
SVR-SVR	5.00	1.86	98.58	2.39	15833.27
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	150	2.12	112.78	2.67	20043.69
SVR-Ranom Forest	100	1.93	102.07	2.42	16355.13

Table 4.7: Prediction performance of 4-days ahead of time (for CNX Nifty)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	4000	2.49	131.92	3.10	26636.56
SVR-ANN	2000	2.12	112.14	2.72	20455.22
	γ	MAPE	MAE	rRMSE	MSE
SVR	0.00	2.08	109.66	2.59	18445.44
SVR-SVR	5.00	2.06	108.64	2.62	19013.75
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	50	2.40	127.34	2.97	24734.84
SVR-Ranom Forest	100	2.12	112.55	2.69	20067.88

Table 4.8: Prediction performance of 5-days ahead of time (for CNX Nifty)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	2000	2.85	151.81	3.57	36016.34
SVR-ANN	2000	2.32	121.91	2.96	23562.53
	γ	MAPE	MAE	rRMSE	MSE
SVR	4.00	2.34	123.77	2.92	23455.13
SVR-SVR	5.00	2.26	119.59	2.86	22583.42
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	100	2.62	139.01	3.30	30370.98
SVR-Ranom Forest	50	2.39	126.86	3.00	24975.76

Table 4.9: Prediction performance of 6-days ahead of time (for CNX Nifty)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	7000	2.80	149.35	3.47	34308.21
SVR-ANN	5000	2.48	130.52	3.16	26803.00
	γ	MAPE	MAE	rRMSE	MSE
SVR	5.00	2.57	135.44	3.22	28502.25
SVR-SVR	5.00	2.46	130.04	3.08	26297.36
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	50	2.78	147.98	3.55	35171.38
SVR-Ranom Forest	50	2.61	138.29	3.23	28782.80

Table 4.10: Prediction performance of 7-days ahead of time (For CNX Nifty)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	9000	3.02	160.20	3.83	41725.69
SVR-ANN	9000	2.65	139.58	3.36	30552.47
	γ	MAPE	MAE	rRMSE	MSE
SVR	10.00	2.74	144.93	3.48	33231.86
SVR-SVR	5.00	2.61	137.81	3.33	30462.37
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	50	3.08	164.00	3.96	44032.01
SVR-Ranom Forest	100	2.84	150.38	3.55	34948.60

Table 4.11: Prediction performance of 8-days ahead of time (for CNX Nifty)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	3000	3.01	160.14	3.80	40747.70
SVR-ANN	8000	2.82	149.03	3.60	35377.97
	γ	MAPE	MAE	rRMSE	MSE
SVR	0.00	2.90	153.08	3.69	37239.22
SVR-SVR	4.00	2.77	145.99	3.55	34515.93
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	50	3.33	177.80	4.25	51119.66
SVR-Ranom Forest	50	2.90	153.03	3.68	37168.41

Table 4.12: Prediction performance of 9-days ahead of time (for CNX Nifty)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	1000	3.36	178.61	4.22	50724.35
SVR-ANN	4000	3.00	157.63	3.82	39622.51
	γ	MAPE	MAE	rRMSE	MSE
SVR	0.00	3.08	162.50	3.92	41815.60
SVR-SVR	4.00	2.94	154.96	3.77	38621.40
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	150	3.58	190.76	4.56	58407.62
SVR-Ranom Forest	50	3.04	160.58	3.87	41177.90

Table 4.13: Prediction performance of 10-days ahead of time (for CNX Nifty)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	6000	3.54	188.73	4.61	60592.10
SVR-ANN	7000	3.23	170.40	4.15	46788.00
	γ	MAPE	MAE	rRMSE	MSE
SVR	0.00	3.24	170.61	4.15	46764.19
SVR-SVR	4.00	3.11	163.74	3.99	43197.66
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	150	3.73	198.62	4.81	64653.82
SVR-Ranom Forest	150	3.26	172.23	4.15	47132.72

Table 4.14: Prediction performance of 15-days ahead of time (for CNX Nifty)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	8000	4.05	215.26	5.06	71431.74
SVR-ANN	7000	3.75	195.95	4.87	62369.57
	γ	MAPE	MAE	rRMSE	MSE
SVR	0.00	4.04	212.36	5.09	69934.62
SVR-SVR	4.00	3.83	201.51	4.87	63747.90
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	50	4.12	217.61	5.49	82312.31
SVR-Ranom Forest	50	3.82	201.15	4.86	63754.16

Table 4.15: Prediction performance of 30-days ahead of time (for CNX Nifty)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	7000	5.02	267.49	6.21	109479.02
SVR-ANN	7000	4.56	237.59	5.79	86912.03
	γ	MAPE	MAE	rRMSE	MSE
SVR	0.00	5.32	278.37	6.82	124246.62
SVR-SVR	4.00	4.94	258.41	6.26	103710.57
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	150	5.26	276.87	6.96	130770.09
SVR-Ranom Forest	50	4.88	255.23	6.19	101094.69

Table 4.16: Prediction performance of 1-day ahead of time (for S&P BSE Sensex)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	3000	1.78	313.92	2.31	166090.16
SVR-ANN	7000	1.55	272.71	1.96	118395.09
	γ	MAPE	MAE	rRMSE	MSE
SVR	4.00	0.98	172.47	1.25	47558.47
SVR-SVR	0.50	1.48	260.05	1.89	108137.61
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	100	1.25	221.91	1.60	81098.60
SVR-Ranom Forest	50	1.23	216.02	1.55	73483.60

Table 4.17: Prediction performance of 2-days ahead of time (for S&P BSE Sensex)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	7000	1.92	338.77	2.40	179261.33
SVR-ANN	3000	1.69	296.41	2.20	146339.70
	γ	MAPE	MAE	rRMSE	MSE
SVR	10.00	1.38	241.46	1.75	93134.99
SVR-SVR	0.50	1.59	278.92	2.07	131058.24
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	100	1.66	292.50	2.08	134559.91
SVR-Ranom Forest	150	1.62	285.74	2.01	125881.22

Table 4.18: Prediction performance of 3-days ahead of time (for S&P BSE Sensex)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	1000	2.15	378.59	2.75	233075.75
SVR-ANN	9000	2.02	354.01	2.64	212027.62
	γ	MAPE	MAE	rRMSE	MSE
SVR	0.00	1.75	306.11	2.21	148141.01
SVR-SVR	0.50	1.85	324.22	2.37	171209.68
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	100	2.01	353.69	2.54	198459.92
SVR-Ranom Forest	150	1.89	332.20	2.35	169694.03

Table 4.19: Prediction performance of 4-days ahead of time (for S&P BSE Sensex)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	4000	2.28	399.99	2.87	247592.09
SVR-ANN	9000	2.27	392.53	2.99	258336.42
	γ	MAPE	MAE	rRMSE	MSE
SVR	0.00	2.05	358.97	2.56	199730.19
SVR-SVR	0.50	2.04	357.32	2.60	205740.59
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	100	2.42	426.06	3.04	285116.07
SVR-Ranom Forest	100	2.11	370.79	2.64	215054.35

Table 4.20: Prediction performance of 5-days ahead of time (for S&P BSE Sensex)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	2000	2.70	476.36	3.42	363471.55
SVR-ANN	8000	2.30	403.21	2.92	259749.88
	γ	MAPE	MAE	rRMSE	MSE
SVR	2.50	2.31	405.27	2.90	255505.25
SVR-SVR	0.50	2.23	392.30	2.83	244955.34
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	50	2.68	473.25	3.34	344949.70
SVR-Ranom Forest	100	2.32	409.54	2.89	256949.33

Table 4.21: Prediction performance of 6-days ahead of time (for S&P BSE Sensex)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	7000	2.70	478.29	3.37	357530.68
SVR-ANN	6000	2.43	429.90	3.14	312341.99
	γ	MAPE	MAE	rRMSE	MSE
SVR	4.00	2.53	443.13	3.18	308063.70
SVR-SVR	0.50	2.42	425.68	3.04	282869.87
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	50	2.92	516.14	3.68	419168.80
SVR-Ranom Forest	150	2.50	439.29	3.10	293822.81

Table 4.22: Prediction performance of 7-days ahead of time (for S&P BSE Sensex)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	9000	2.82	497.40	3.63	412421.92
SVR-ANN	5000	2.53	445.19	3.23	320178.09
	γ	MAPE	MAE	rRMSE	MSE
SVR	4.00	2.69	472.27	3.42	356444.96
SVR-SVR	1.50	2.55	447.80	3.26	324401.97
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	100	3.20	566.55	4.04	508254.64
SVR-Ranom Forest	150	2.69	473.19	3.33	341307.44

Table 4.23: Prediction performance of 8-days ahead of time (for S&P BSE Sensex)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	6000	3.12	546.67	3.93	465739.08
SVR-ANN	7000	2.64	459.58	3.54	367846.18
	γ	MAPE	MAE	rRMSE	MSE
SVR	0.00	2.84	497.85	3.61	396295.99
SVR-SVR	1.50	2.71	475.37	3.48	368319.67
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	150	3.38	596.67	4.29	569631.81
SVR-Ranom Forest	100	2.88	507.31	3.58	392375.92

Table 4.24: Prediction performance of 9-days ahead of time (for S&P BSE Sensex)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	9000	3.17	554.86	4.05	496313.24
SVR-ANN	6000	2.87	499.37	3.74	411681.57
	γ	MAPE	MAE	rRMSE	MSE
SVR	0.00	3.02	529.53	3.84	444071.59
SVR-SVR	0.50	2.87	502.33	3.68	408835.92
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	50	3.49	616.32	4.46	615743.91
SVR-Ranom Forest	150	3.10	545.80	3.86	454853.43

Table 4.25: Prediction performance of 10-days ahead of time (for S&P BSE Sensex)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	8000	3.45	603.90	4.40	585260.94
SVR-ANN	5000	2.72	474.62	3.60	387086.13
	γ	MAPE	MAE	rRMSE	MSE
SVR	10.00	3.19	557.98	4.10	505260.72
SVR-SVR	0.50	3.00	525.45	3.87	449987.02
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	100	3.62	637.70	4.60	648907.41
SVR-Ranom Forest	150	3.19	561.07	4.06	497755.22

Table 4.26: Prediction performance of 15-days ahead of time (for S&P BSE Sensex)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	5000	3.90	681.64	4.84	700906.09
SVR-ANN	6000	3.58	624.50	4.54	612524.19
	γ	MAPE	MAE	rRMSE	MSE
SVR	10.00	3.94	688.47	4.96	735083.38
SVR-SVR	0.50	3.69	644.66	4.68	651935.80
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	50	4.16	730.56	5.52	916603.04
SVR-Ranom Forest	150	3.80	664.37	4.81	692641.93

Table 4.27: Prediction performance of 30-days ahead of time (for S&P BSE Sensex)

Prediction Models	**Parameters**	**Error Measures**			
	ep	MAPE	MAE	rRMSE	MSE
ANN	8000	4.83	839.92	6.28	1152684.67
SVR-ANN	1000	4.32	745.01	5.58	891384.72
	γ	MAPE	MAE	rRMSE	MSE
SVR	0.00	5.26	913.05	6.75	1341763.55
SVR-SVR	0.50	4.74	822.34	5.99	1047397.92
	ntrees	MAPE	MAE	rRMSE	MSE
Random Forest	150	5.29	926.53	6.73	1357838.40
SVR-Ranom Forest	50	4.67	809.42	5.94	1022031.89

Table 4.28: Average prediction performance for CNX Nifty

Prediction Model	**MAPE**	**MAE**	**rRMSE**	**MSE**
ANN	3.01	160.10	3.79	43872.97
SVR-ANN	2.66	139.99	3.41	34207.13
SVR	2.71	142.54	3.43	37530.53
SVR-SVR	2.66	140.17	3.39	34975.22
Random Forest (RF)	3.02	160.08	3.87	46992.46
SVR-RF	2.72	143.64	3.44	36166.73

Table 4.29: Performance improvement in (%) (Single stage models vs. Two stage fusion models) for CNX Nifty

Models under Comparison	**MAPE**	**MAE**	**rRMSE**	**MSE**
ANN vs. SVR-ANN	11.57	12.56	10.22	22.03
SVR vs. SVR-SVR	1.66	1.66	1.12	6.81
RF vs. SVR-RF	9.81	10.27	11.2	23.04

Table 4.30: Average prediction performance for S&P BSE Sensex

Prediction Model	**MAPE**	**MAE**	**rRMSE**	**MSE**
ANN	2.90	509.19	3.69	446695.60
SVR-ANN	2.58	449.75	3.34	358157.63
SVR	2.66	465.55	3.38	402606.42
SVR-SVR	2.60	454.69	3.31	366287.80
Random Forest (RF)	3.01	529.80	3.83	508104.95
SVR-RF	2.67	467.92	3.35	378614.56

It can be observed that performance of SVR-ANN and SVR-RF models is improved significantly than ANN and RF models. SVR-SVR model exhibits a moderate improvement over SVR model. These results demonstrate the effectiveness of our proposal. Comparison of prediction performance of all the models for both the stock market indices reveals that SVR-ANN model performs the best overall.

Table 4.31: Performance improvement in (%) (Single stage models vs. Two stage fusion models) for S&P BSE Sensex

Models under Comparison	**MAPE**	**MAE**	**rRMSE**	**MSE**
ANN vs. SVR-ANN	11.2	11.67	9.42	19.82
SVR vs. SVR-SVR	2.41	2.33	1.88	9.02
RF vs. SVR-RF	11.31	11.68	12.7	25.48

Figure 4.4 shows the actual value of the CNX Nifty, value predicted by ANN and SVR-ANN models for the task of predicting 5-day ahead of time. The visual representation also validates the effectiveness of the proposed two stage fusion approach. Visual representation for the other prediction tasks (not shown here) also demonstrates the effectiveness of the proposed approach.

The reason behind the improved performance of two stage fusion approach over the single stage approach can be justified as follows. In two stage fusion approach, prediction models in the second stage have to identify transformation from technical parameters describing $(t+n)^{th}$ day to $(t+n)^{th}$ day's closing price, while in single stage approach, prediction models have to identify transformation from technical parameters describing t^{th} day to $(t+n)^{th}$ day's closing price.

The introduction of an additional stage in case of two stage fusion approach takes the responsibility of preparing data for the prediction models in the second stage. Actually it transforms closing and opening price, low and high of t^{th} day to technical parameters representing $(t+n)^{th}$ day. This may reduce the prediction error, as now, the prediction models in second stage have to predict based on predicted technical parameters of $(t+n)^{th}$ day rather than actual technical parameters but of t^{th} day.

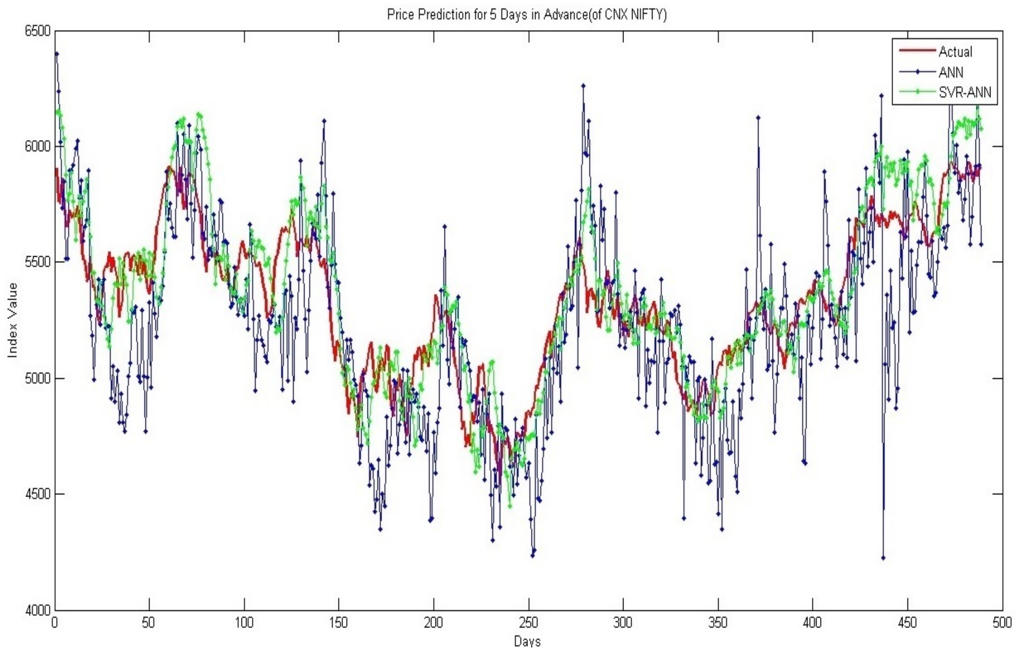


Figure 4.4: Prediction performance comparison of ANN and SVR-ANN for predicting 5 day ahead of time for CNX Nifty

4.5 Conclusions

The task of predicting future values of stock market indices is focused in this study. Experiments are carried out on ten years of historical data of two indices namely CNX Nifty and S&P BSE Sensex from Indian stock markets. The predictions are made for 1 to 10, 15 and 30 days in advance.

Review of the existing literature on the topic revealed that existing methods for predicting stock market index's value\price have used a single stage prediction approach. In these existing methods, the technical\statistical parameters' value of $(t)^{th}$ day is used as inputs to predict the $(t+n)^{th}$ day's closing price\value (t is a current day). In such scenarios, as the value of n increases, predictions are based on increasingly older values of statistical parameters and thereby not accurate enough. It is clear from this discussion that there is a need to address this problem and two stage prediction scheme which can bridge this gap and minimize the error stage wise may be helpful.

Some of the literatures on the focused topic have tried to hybridize various machine learning techniques but none has tried to bridge the identified gap, rather, in these literatures, generally it is found that one machine learning technique is used to tune the design parameters of the other technique.

A two stage fusion approach involving support vector regression (SVR) in the first stage and ANN, random forest and SVR in the second stage is proposed in this chapter to address the problem identified. Experiments are carried out with single stage and two stage fusion prediction models. The results show that two stage hybrid models perform better than that of the single stage prediction models. The performance improvement is significant in case when ANN and RF are hybridized with SVR. A moderate improvement in the performance is observed when SVR is hybridized with itself. The best overall prediction performance is achieved by SVR-ANN model.

The proposal of two stage prediction scheme is a significant research contribution of this chapter as this scheme provides a kind of new way of feeding adequate information to prediction models. To accomplish this, machine learning methods are used in cascade in two stages. First stage uses SVR to predict future values of statistical parameters which are fed as the inputs to the prediction models in the second stage. Experimental results are promising and demonstrates the usefulness of the proposed approach. The proposed approach is not only successful but also useful and adaptable for other prediction tasks such as weather forecasting, energy consumption forecasting and GDP forecasting. This generalizability of the proposed approach definitely makes the proposal a significant contribution to the research.

Chapter 5

Social Resource Recommendation using Learning from Positive and Unlabeled Examples

Bookmarks submitted on social bookmarking system delicious (http://www.delicious.com/) and artists on online music system last.fm (http://www.last.fm/) are some of the examples of social resources. The memory based collaborative filtering has served as the most widely used algorithm for recommending social resources. However its predictions are based on some ad hoc heuristic rules and its success depends on the availability of a critical mass of users. This encourages to investigate some alternative approach for recommending social resources. It is to be noticed that user bookmarking a resource (e.g. URL) or submitting a resource (e.g. research article) on this system, implicitly indicates his likings to this resource. These resources are considered as positive examples of the user preference. Other resources (e.g. URLs/research articles), however, do not imply negative preference of the user about them. This leads to the situation where we have positive examples but no negative examples for user preference. If a learning based approach is to be devised as the alternative to memory based collaborative filtering for the task of social resource recommendation, it requires to learn the recommender from positive and unlabeled examples. Therefore, the focus of this work is to build the social resource recommender using learning from positive and unlabeled examples.

Model based two-step techniques to learn a classifier using positive and unlabeled examples is proposed in this study to address personalized resource recommendations. In the first step of these techniques, Naive Bayes (NB) classifier is employed to identify reliable negative resources. In the second step, to generate effective resource recommender, Classification and Regression Tree (CART) and Least Squares-Support Vector Machine (LS-SVM) are exercised (Breiman et al. Suykens and Vandewalle). A direct method based on LS-SVM is also put forward to realize the recommendation task. In direct method, LS-SVM is customized for learning from positive and unlabelled data. Furthermore, the impact of feature selection on the proposed techniques is also studied. Memory based collaborative filtering as well as proposed techniques exploit usage data to generate personalized recommendations.

The motivation behind the work is (Liu et al., "Partially supervised classification of text documents") and (Liu). It was shown theoretically by them that by using positive and unlabeled resource sets, accurate classifiers could be built with high probability provided that sufficient positive and unlabeled resources were available.

5.1 Introduction

Social bookmarking system is a web based resource sharing system that allows users to upload, share and organize their resources i.e. bookmarks and publications. The system has changed the organization of bookmarks from an individual activity limited to a desktop to collective attempt over the web. Users can submit their resources that lead to large communities of users to collaboratively create accessible repositories of web resources. User bookmarking a resource (URL) on this system implicitly indicates his likings to the resource (URL). These resources are considered as positive examples of the user preference. Other resources (URLs) however do not imply negative preference of the user about them. This leads to the situation where we have positive examples but no negative examples for user preference.

Online music system such as last.fm constructs detailed profile of its user by analysing details of the track the user listens to, either from internet radio stations, or the user's computer or many portable music devices. User listening to a particular artist many times is an implicit indicator about his positive preference about the artist. The artists who have not been listened by user, however, are not the indicators of user's negative preference. This again leads to the situation where we have positive examples but no negative examples for user preference.

Conventional classification techniques require both labeled positive and labeled negative examples to build a recommender, they are, thus not suitable to the problem.

The memory based collaborative filtering has served as the most widely used technique for resource recommendations, but it has its own limitations of reliance on ad hoc heuristic rules and dependence of success on availability of a critical mass of users.

This study proposes novel techniques to solve the problem. The first set of techniques employ naive Bayes method in the first step while Classification and Regression Tree (CART) and LS-SVM are utilized in the second step. The proposal is to first use naive Bayes classifier to extract some reliable negative resources from the unlabeled set and then apply CART or LS-SVM along with feature selection to build a recommender. The study also put forwards a direct method based on LS-SVM to build a resource recommender. LS-SVM has been tailored to learn from positive and unlabeled data. Experimental results show that the proposed techniques outperform existing method significantly.

5.2 Related Work

The task of social resource recommendation entails retrieving and recommending interesting social resource to the user. Recommendations can be based on a range of information sources about the user and the resource, contributing information at different representation levels. In many systems, tags present an additional level of representation, linking users to resource through an alternative route.

The past few years have seen an escalating number of approaches for resource recommendation that exploits these two types of data representations (Hotho et al. Clements, de Vries, and Reinders Tso-Sutter, Marinho, and Schmidt-Thieme Wetzker, Umbrath, and Said). Usage, tag and metadata of resources were used in (Bogers and Van Den Bosch) for recommendation generation. They also investigated about how to fuse different recommendation approaches together to further improve recommendation accuracy. These approaches typically used a memory based Collaborative Filtering (CF) algorithm to make their recommendations.

It is recognised in this study that the data which is part of social resource sharing systems such as delicious (http://www.delicious.com/) and last.fm (http://www.last.fm/) are actually examples of positive and unlabeled data. Hence, it is felt that social resource recommendation can be modelled as learning from positive and unlabeled examples. Learning from positive and unlabeled examples (PU learning) was also used in (Davis et al.) and (Cerulo, Elkan, and Ceccarelli) for named entity disambiguation in streaming data and learning gene regularity networks respectively. Some new methods for identifying set of reliable negative documents and classifying text documents were proposed in (Liu et al., "Building text classifiers using positive and unlabelled examples").

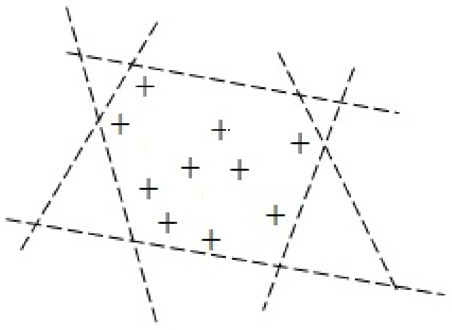
5.3 Collaborative Filtering for Resource Recommendation

Collaborative filtering algorithms use usage data for generating recommendations. (Bogers and Van Den Bosch) extended one specific CF algorithm: the k-Nearest Neighbor (k-NN) algorithm. In their proposal, first step involved locating users that were most similar to the active user, i.e., the user to whom the new items are to be recommended. Similarity between the active user and all other users in the system was calculated by considering the overlap in resources they had preferred.

Each user u_k was represented as a Boolean user profile vector u_k. All the resources, preferred by him/her, were represented with a value 1 in this vector. Cosine similarity metric was used to determine the similarity between two users u_k and the active user u_a as $sim_{cosine}(u_a, u_k) = \frac{u_a \cdot u_k}{||u_a|| ||u_k||}$ (Bogers and Van Den Bosch). Cosine similarity had been used effectively with data sets with implicit ratings (Breese, Heckerman, and Kadie).

In the second step, the resources of the most like-minded users were gathered to determine the most suitable recommendations for the active user. The supposition here was, more similar two users are in the resources they share, the more of similar mind they are. The top k most similar users for the active user u_a formed the Set of Similar Users $SSU(u_a)$. Taking into consideration this set of nearest neighbors, the final prediction scores $S_{a,l}$ for each of the SSU's resources i_l was computed as $S_{a,l} = \sum_{u_k \in SSU(u_a)} sim_{cosine}(u_a, u_k)$.

Thus, the predicted score of a resource i_l was the sum of similarity values (be-


(a) With only positive examples

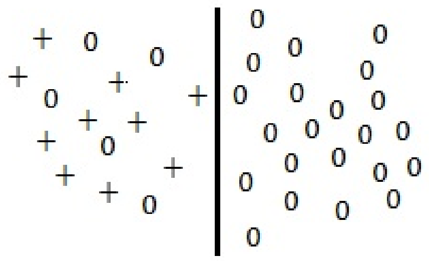

(b) with both positive and unlabeled examples

Figure 5.1: Learning from positive and unlabeled examples (Liu)

tween 0 and 1) of all N nearest neighbors that actually posted resource i_l.

Lastly, all resources were ranked by their predicted score $S_{a,l}$. Resources already posted by the active user were filtered out to generate the final list of recommendations for the active user.

5.4 Learning from Positive and Unlabeled Examples

In this section, the usefulness of unlabeled data and theoretical foundations of learning from positive and unlabeled examples is discussed as convoluted in (Liu et al., "Partially supervised classification of text documents" Liu).

5.4.1 Usefulness of Unlabeled Data

In this segment, some perception on why learning from positive and unlabeled examples is feasible and why unlabeled examples are useful is built up (Liu). Figure 5.1 depicts the idea.

In Figure 5.1a, only positive resources/examples are represented. Symbol '+' is used to indicate the positive resource. It is assumed that a linear classifier is adequate for the classification. In this scenario, it is difficult to identify where to draw the line that separates positive and negative examples because it is not known where the negative examples might be. There are infinite possibilities. Conversely, if the unlabeled data are added to the space as shown in Figure 5.1b, it becomes very apparent where the separation line should be. Symbol '0' is used to represent unlabeled data.

5.4.2 Theoretical Foundations of PU Learning

Let (x_i, y_i) be random variables drawn independently from probability distribution $I_{(x,y)}$ where $y \in \{-1, 1\}$ is the conditional random variable that is to be approximated given x. x_i is used to represent a resource while y_i is used to represent its class. Positive resources can be symbolized by class $+1$ while -1 can be used to characterize negative resources. Assume that the positive and unlabeled resources are independently drawn from the conditional distribution $I_{x|y}$ and the marginal distribution I_x. Aim is to learn a classification function f, that can separate positive and negative resources with minimum probability of error, $Pr(f(x) \neq y)$. Rewriting it into more useful form,

$$Pr(f(x) \neq y) = Pr(f(x) = 1 \quad and \quad y = -1) + Pr(f(x) = -1 \quad and \quad y = 1) \tag{5.1}$$

The first expression in Equation 5.1 can be further expressed as

$$\begin{aligned} Pr(f(x) = 1 \quad and \quad y = -1) &= Pr(f(x) = 1) - Pr(f(x) = 1 \quad and \quad y = 1) & (5.2) \\ &= Pr(f(x) = 1) - (Pr(y = 1) - Pr(f(x) = -1 \quad and \quad y = 1)) & (5.3) \end{aligned}$$

Substituting results from Equations 5.2 and 5.3 into Equation 5.1,

$$Pr(f(x) \neq y) = Pr(f(x) = 1) - Pr(y = 1) + 2Pr(f(x) = -1|y = 1)Pr(y = 1) \tag{5.4}$$

Since $Pr(y = 1)$ is constant, the probability of error can be minimized by minimizing

$$Pr(f(x) = 1) + 2Pr(f(x) = -1|y = 1)Pr(y = 1) \tag{5.5}$$

If $Pr(f(x) = -1|y = 1)$ can be held small, minimizing the probability of error is same as minimizing $Pr(f(x) = 1)$. This is just about same as minimizing $Pr_U(f(x) = 1)$ (because number of positive examples are very small compared to number of unlabeled examples) while holding $Pr_P(f(x) = 1) \geq r$ where r is the recall, i.e. $Pr(f(x) = 1|y = 1)$. Note that $(Pr_P(f(x) = 1) \geq r)$ is same as $(Pr_P(f(x) = -1) \leq (1 - r))$.

Above formulations allow to model the problem as constrained optimization problem where, the objective is to minimize the number of unlabeled examples labeled as positive, subject to the constraint that the fraction of errors on the positive examples is no more than $1 - r$ (Liu).

5.5 The Proposed Two Step Techniques

In the previous section, it has been shown theoretically that by using positive and unlabeled resource sets, accurate classifiers can be built with high probability provided that sufficient positive and unlabeled resources are available. Nevertheless, the discussed theoretical method has two drawbacks: (i) The constrained optimization problem may not be easy to solve (ii) Given a practical problem, it seems to be difficult to select a preferred recall level that will give a good classifier. This section proposes practical two step heuristic techniques inspired from the work in (Liu et al., "Building text classifiers using positive and unlabelled examples") and (Liu). In the first step of the techniques, reliable negative resources are extracted from the unlabeled set using naive Bayesian method. In the second step, CART and LS-SVM are experimented in conjunction with feature selection (by means of Information Gain method (Han, Kamber, and Pei)) to build the recommender. Lastly, the built recommender is used to generate personalized recommendations.

5.5.1 Finding Reliable Negative Resources

Naive Bayesian method is one of the popular techniques for classification. Even though the postulation that features are independent given class label of a resource is not realistic in this domain, it has been shown to perform very well in practice by many researchers (Domingos and Pazzani McCallum, Nigam, et al.).

Given a set of training resources I, each resource i is considered as a vector of n attribute values $[i.e., i = (i_1, i_2, i_3, \cdots, i_n)]$. In this work, each attribute represents a distinct user and the value of the attribute indicates whether the corresponding user has liked the corresponding resource or disliked it. This means, all the attributes are modelled as Boolean attributes. This allows us to fit the multivariate Bernoulli distribution to the data. The naive Bayesian method decides the class C of resource i as the one that maximizes the conditional probability $P(C|i)$. According to Bayes' rule,

$$P(C|i) = \frac{P(i|C)P(C)}{P(i)} \tag{5.6}$$

To determine $P(i|C)$, naive Bayesian assumption, that attributes are statistically independent is made. As i is a vector of n attribute values, this supposition leads to,

$$P(i|C) = P(i_1, i_2, i_3, \cdots, i_n|C) = \prod_{k=1}^{n} P(i_k|C) \tag{5.7}$$

The proportion of resources from class C that includes attribute value i_k is used to calculate each $P(i_k|C)$. $P(C)$ is the probability of resources for class C, which is calculated as the fraction of the training resources that fall in class C. $P(i)$ is common denominator, which is not required in the calculation as only the class label is to be decided. Laplacian prior is also used in the actual calculation of conditional probability to avoid probability estimation to 0.

To identify set of reliable negative (RN) resources from the unlabeled set U, steps shown in Algorithm 2 are followed.

Algorithm 2 Algorithm to identify reliable negative (RN) resources

```
Input: positive set $P$, unlabeled set $U$
Output: reliable negative set $RN$
 1: $RN \leftarrow \emptyset$
 2: Assign the class label 1 to each resource in positive set $P$
 3: Assign the class label $-1$ to each resource in unlabeled set $U$
 4: Build a naive Bayes classifier using $P$ and $U$. Fit multivariate Bernoulli distribu-
    tion to the data
 5: for all resource $r \in U$ do
 6:     if its probability $P(1|r) < 0.5$ then
 7:         $RN = RN \cup \{r\}$
 8:     end if
 9: end for
10: return $RN$
```

5.5.2 Building and using Recommender to make Predictions

Resources which are part of P and RN form the data for building the recommender. Before this data is used to build the recommender, it is partitioned into training and test set. Information Gain (Han, Kamber, and Pei) is also used to identify those features which are important. Data with only important features are used to learn the recommender. The number of important features is varied in the experiment to study its impact on the final outcome. CART (Han, Kamber, and Pei) and LS-SVM are used to build recommender in second step. Performance of these methods is also compared.

Decision tree learners build a decision tree by recursively partitioning examples into subgroups until those subgroups include examples of a single class. A partition is formed by a test on the selected attribute. In this study, CART is used to construct a regression tree. CART uses Gini index which selects the attribute that has the minimum Gini index or maximizes the reduction in impurity (Han, Kamber, and Pei).

In SVM, the idea is to identify maximum marginal hyper plane $< w \cdot x > +b = 0$ to separate the data belonging to different class y_i where $y_i \in \{1, -1\}$. Here, a direction perpendicular to the hyper plane is defined by w, $b \in R$ is a bias and x is a set of input vectors. $w \cdot x$ is a dot product of w and x. In classical SVM, one needs to solve convex quadratic programming problem. LS-SVM solves set of linear Equations instead of a convex quadratic programming problem for classical SVMs (Suykens and Vandewalle). Linear LS-SVM under separable case is the best condition. In practice, though, the training data is more or less always noisy. It has been shown by means of extensive empirical studies that LS-SVM is comparable to SVM in terms of generalization performance (Van Gestel et al.), (Zhang, Pena, and Robles). However, the underlying reason for their similarity is not well understood yet (Ye and Xiong). In this study, LS-SVM is considered under non-separable case.

Definition (Linear LS-SVM: Non-Separable Case): Given a set of training examples which are linearly separable, $T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, learning is to solve the following constrained minimization problem,

$$\begin{aligned} Minimize : \frac{< w \cdot w >}{2} + \frac{C}{2}\sum_{i=1}^{n} \xi_i^2 \\ Subject \quad to : y_i(< w \cdot x_i > +b) = 1 - \xi_i, \quad i = 1, 2, \ldots, n \end{aligned} \tag{5.8}$$

where $C \geq 0$ are user defined parameters and ξ_i is a slack variable (Liu).

Solving the constrained minimization problem in Equation 5.8 produces the solutions for w and b, which in turn give us the maximum margin hyper plane $< w \cdot x_i > +b = 0$ with the margin $\frac{2}{||w||}$.

In a nutshell, the procedure to be followed, for the second step of two step techniques, is depicted in Algorithm 3.

Algorithm 3 Algorithm for building the recommender

```
Input: positive set $P$, reliable negative set $RN$, number of features $N$ to be considered to build the recommender, choice $CH$ to specify the method to be used in second step to build the recommender
Output: Built Recommender $R$
1: Assign the class label 1 to each resource in positive set $P$
2: Assign the class label $-1$ to each resource in reliable negative set $RN$
3: Construct training and testing set.
4: Identify $N$ most important features using Information Gain on the training data
5: Build recommender $R$ (using the training set and selected $N$ features) through the method specified by means of the value of $CH$
6: Return Recommender $R$
```

Once the personalized recommender is built for the user, it can be used for that user to predict the confidence by which each of the existing resource belongs to positive set. For each of the resource classified as positive, confidence of classification is calculated based on the distance from the decision boundary.

5.5.3 Feature Selection

Information Gain is used to select the best features. Number of selected features are varied to see the impact on the performance of the recommender. Furhter, in this section, how information gain can be used for feature selection is discussed.

Assume that D denotes the set of examples which are part of training set. The expected information required to classify an example in D is given by

$$Info(D) = \sum_{i=1}^{m} p_i log_2 p_i \tag{5.9}$$

where p_i is the probability that an arbitrary example in D belongs to class CL_i and is estimated by $|CL_{(i,D)}|/|D|$ and m denotes the number of classes. In this chapter, for the given dataset and task on the hand, $m = 2$. $Info(D)$ is also known as the entropy of D.

Assume that A is one of the attribute of examples in D and it has v distinct values, $\{a_1, a_2, a_3, , a_v\}$. In this chapter, for the given dataset and task on the hand,

$v = 2$. Let $Info_A(D)$ denote the expected information required to classify an example from D based on the partitioning by A. It is defined as

$$Info_A(D) = \sum_{j=1}^{v} (|D_j|)/(|D|) \times Info(D_j) \tag{5.10}$$

The term $(|D_j|)/(|D|)$ serves as the weight of the j^{th} partition. Information gain is defined as the difference between the original information requirements and the new requirements and it is defined as

$$Gain(A) = Info(D) - Info_A(D) \tag{5.11}$$

In other words, $Gain(A)$ quantifies the importance of feature A. More the gain better the feature. Using the above formulations, information gain for each of the features is found and importance of each of the features is identified. The idea is to use best n features, if the recommender is to be learnt from n features.

5.6 The Proposed Direct Method

In this section, a direct method is proposed, inspired from the work in (Liu et al., "Building text classifiers using positive and unlabelled examples") and (Liu). Direct method is based on LS-SVM and eliminates the need of identifying reliable negative resources. Let the set of training examples be $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, where x_i is the input vector and y_i is its corresponding class label and $y_i \in \{1, -1\}$. Presume that first $k - 1$ examples are positive examples $(+1)$, while the rest are unlabeled examples, which are labelled as negative (-1). Thus, the negative set has noise, i.e., contains positive examples. In practice, the positive set may also contain some noise. If noise is allowed in positive examples, then learning is to solve the following constrained optimization problem.

$$\begin{gathered} Minimize : \frac{< w \cdot w >}{2} + \frac{C_+}{2} \sum_{i=1}^{k-1} \xi_i^2 + \frac{C_-}{2} \sum_{i=k}^{n} \xi_i^2 \\ Subject \quad to : y_i(< w \cdot x_i > + b) = 1 - \xi_i, \quad i = 1, 2, \dots, n \end{gathered} \tag{5.12}$$

where $C_+, C_- \geq 0$ are user defined parameters and ξ_i is a slack variable. C_+, C_- can be varied to achieve the objective. Intuitively, a bigger value is assigned to C_+ while a smaller value to C_- because the unlabeled set, which is assumed to be negative, may contain positive data. In all the experiments performed in this study, C_+ and C_- are set to 0.9 and 0.1 respectively on the basis of empirical evidence. Lagrangian Primal corresponding to the optimization problem with equality constraints in Equation 5.13 is

$$\begin{aligned} L_p = \frac{1}{2} < w \cdot w > + \frac{C_+}{2} \sum_{i=1}^{k-1} \xi_i^2 + \frac{C_-}{2} \sum_{i=k}^{n} \xi_i^2 + \\ \sum_{i=1}^{n} \alpha_i [1 - \xi_i - y_i (< w \cdot x_i > + b)], \quad \alpha_i \in R \end{aligned} \tag{5.13}$$

As discussed in (Rao and Rao), conditions for the optimal solution give us following expressions:

$$\frac{\partial L_p}{\partial w} = 0 \Rightarrow w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{5.14}$$

$$\frac{\partial L_p}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{5.15}$$

$$\frac{\partial L_p}{\partial \xi_i} = 0 \Rightarrow C_+ \xi_i - \alpha_i = 0 \Rightarrow \xi_i = \frac{\alpha_i}{C_+}, \quad i = 1, 2, \ldots, k-1 \tag{5.16}$$

$$\frac{\partial L_p}{\partial \xi_i} = 0 \Rightarrow C_- \xi_i - \alpha_i = 0 \Rightarrow \xi_i = \frac{\alpha_i}{C_-}, \quad i = k, k+1, \ldots, n \tag{5.17}$$

$$\frac{\partial L_p}{\partial \alpha_i} = 0 \Rightarrow y_i (< w \cdot x_i > + b) - 1 + \xi_i = 0, \quad i = 1, 2, \ldots, n \tag{5.18}$$

Eliminating w and ξ_i from Equation 5.18 using Equations 5.14 and 5.16 and then using this result along with Equation 5.15 gives linear system as shown in Equation 5.19 ,

$$\begin{bmatrix} 0 & Y_{k-1}^T \\ Y_{k-1} & \Omega_{k-1} + C_+^{-1} I_{k-1} \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 1_{k-1} \end{bmatrix} \tag{5.19}$$

where, $Y_{k-1} = [y_1, y_2, \ldots, y_{k-1}], 1_{k-1} = [1, 1, \ldots, 1], \alpha = [\alpha_1, \alpha_2, \ldots, \alpha_{k-1}]$. I_{k-1} is $(k-1) \times (k-1)$ identity matrix and $\Omega \in R^{(k-1)\times(k-1)}$ is the kernel matrix defined by $\Omega_{ij} = y_i y_j < x_i \cdot x_j >$. $i, j = 1, 2, \ldots, k-1$. Eliminating w and ξ_i from Equation 5.18

using Equations 5.14 and 5.17 and then using this result along with Equation 5.15 gives the linear system as shown in Equation 5.20,

$$\begin{bmatrix} 0 & Y_{n-(k-1)}^T \\ Y_{n-(k-1)} & \Omega_{n-(k-1)} + C_-^{-1} I_{n-(k-1)} \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 1_{n-(k-1)} \end{bmatrix} \tag{5.20}$$

where, $Y_{n-(k-1)} = [y_k, y_{k+1}, \ldots, y_n], 1_{n-(k-1)} = [1, 1, \ldots, 1], \alpha = [\alpha_k, \alpha_{k+1}, \ldots, \alpha_n]$. $I_{n-(k-1)}$ is $(n-(k-1)) \times (n-(k-1))$ identity matrix and $\Omega \in R^{(n-(k-1)) \times (n-(k-1))}$ is the kernel matrix defined by $\Omega_{ij} = y_i y_j < x_i \cdot x_j > . \quad i, j = k, k+1, \ldots, n$.

Solving the linear systems in 5.19 and 5.20 give values of α_i for $i = 1, 2, \ldots, k-1$ and $i = k, k+1, \ldots, n$ respectively in addition to the value of b. This in turn, gives the maximum margin hyper plane $< w \cdot x_i > +b = 0$ with the margin $\frac{2}{||w||}$.

5.7 Experimental Evaluation

In this section, proposed techniques are evaluated and compared with memory based collaborative filtering which is the most widely used technique for social resource recommendations.

5.7.1 Data set

Two data sets are used in all the experiments. The first one is the Hetrec2011-delicious-2k (http://www.delicious.com/) and the second one is Hetrec2011-lastfm-2k (http://www.last.fm/). Both the data sets were released in the framework of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011) (http://ir.ii.uam.es/hetrec2011/) at the 5th ACM Conference on Recommender Systems (RecSys 2011) (http://recsys.acm.org/2011/).

In *delicious* data set, 7 files are given in addition to the readme file. Usage data is extracted from user_taggedbookmarks.dat file. This extraction resulted in to 1867 unique users and 69226 unique URLs. Each user is modelled as Boolean feature vector where the number of features is 69226. A value 1 of a specific feature in feature vector of the user indicates that the user has bookmarked the corresponding URL. In $lastfm$ data set, 6 files are given in addition to the readme file. Listen count of users is extracted from user_artists.dat. There are 1892 unique users and 17632 unique artists. If the user had listened to a specific artist more than some listen count threshold (one, in this study) times, it is considered that the user has positive preference for that artist, else he has negative preference. This consideration resulted in Boolean user feature vectors.

5.7.2 Experimental Methodology

In the experiments, techniques proposed in this study and a typical memory based collaborative filtering as described in (Bogers and Van Den Bosch) are evaluated. Twenty test users who have bookmarked at least 45 URLs from the *delicious* data set are selected for experiments on *delicious* data set. Similarly twenty test users who have shown positive preference (based on listen count threshold) for at least 45 artists from the $lastfm$ data set are selected for the experiments on $lastfm$ data set. For each data set, stratified 3-fold cross validation is carried out. In both the approaches after calculating predicted score (using the built recommender) for each of the resources in the test set, they are arranged in descending order and recommended from the top of the list based on Top 5, Top 10, or Top 15 recommendations. Experiments are also carried out by changing neighborhood size in memory based collaborative filtering and selecting different number of features in the proposed techniques to study the impact of these parameters on the quality of recommendations.

5.7.3 Evaluation Measures

Let I_T be an evaluation data set consisting of $|I_T|$ examples (x_i, Y_i), $i = 1, 2, \cdots, |I_T|$, $Y_i \in \{-1, 1\}$. Let f be the recommender and $Z_i = f(x_i)$ be the prediction by f for example x_i. In these experiments, Precision, Recall and F-measure for the recommender f on the test data set I_T are calculated using the below mentioned formulas.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{5.21}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegative} \tag{5.22}$$

$$F - measure(f, I_T) = \frac{2 \times Precison \times Recall}{Precision + Recall} \tag{5.23}$$

Precision(P) @ TopN, Recall(R) @ TopN and F-measure(F) @ TopN are used as the performance measures. They are calculated by considering only the topmost results returned by the classifier in the Equations 5.21, 5.22 and 5.23.

5.7.4 Results and Discussions

This section discusses about the experimental results. Tables 5.1 and 5.2 show the results on *delicious* and *lastfm* data sets respectively. Results for memory based collaborative filtering, proposed two step techniques and proposed direct technique are shown.

Experiments are carried out with varying number of nearest users in case of memory based collaborative filtering and varying number of features in case of the proposed techniques. However, tables show results for only that value of number of Nearest Neighbors (NN) / features (NOF), where individual technique has performed the best overall.

Table 5.1: Results on *delicious*

P/R/F @TopN	**Collaborative Filtering (NN=30)**	**Proposed Two Step Technique (CART in Second Step) (NOF=1000)**	**Proposed Two Step Technique (LS-SVM in Second Step) (NOF=1000)**	**Proposed Direct Method Based on Modified LS-SVM (NOF=1000)**
P/R/F @Top5	0.5267/0.2980/ 0.3806	0.7400/0.4116/ 0.5290	0.7933/0.4522/ 0.5760	0.8200/0.4697/ 0.5973
P/R/F @Top10	0.3900/0.4202/ 0.4045	0.5867/0.6263/ 0.6059	0.6000/0.6495/ 0.6238	0.6933/0.7594/ 0.7249
P/R/F @Top15	0.3489/0.5596/ 0.4298	0.4578/0.7018/ 0.5541	0.4511/0.7090/ 0.5514	0.5311/0.8377/ 0.6501

It is apparent from the experimental results that the proposed techniques produce significantly better results than memory based collaborative filtering which is the most widely used technique for resource recommendations. It is also evident that the best results are achieved when number of Nearest Neighbor (NN) is set to 30 and 10 for *delicious* and *lastfm* data sets respectively, in case of collaborative filtering. The proposed techniques perform best at 1000 features for *delicious* data set while, they perform best at 10 (CART) and 15 (LS-SVM and direct approach) features in case of *lastfm* data set.

Table 5.2: Results on *lastfm*

P/R/F @TopN	**Collaborative Filtering (NN=10)**	**Proposed Two Step Technique (CART in Second Step) (NOF=10)**	**Proposed Two Step Technique (LS-SVM in Second Step) (NOF=15)**	**Proposed Direct Method Based on Modified LS-SVM (NOF=15)**
P/R/F @Top5	0.2778/0.0906/ 0.1366	0.3067/0.1010/ 0.1520	0.3800/0.1231/ 0.1860	0.4067/0.1320/ 0.1993
P/R/F @Top10	0.1978/0.1289/ 0.1561	0.2400/0.1575/ 0.1902	0.2600/0.1680/ 0.2041	0.3100/0.2022/ 0.2448
P/R/F @Top15	0.1578/0.1547/ 0.1562	0.1867/0.1832/ 0.1849	0.2022/0.1961/ 0.1991	0.2467/0.2426/ 0.2446

Figures 5.2 and 5.3 illustrate the impact of number of nearest neighbors used to make predictions in memory based collaborative filtering approach.

Figures 5.4, 5.5 and 5.6 depict significance of number of features (NOF) used to generate TOP 5, Top 10 and TOP 15 recommendations respectively through these proposed techniques. These figures show results for *delicious* data set.

Similarly, results for *lastfm* data set are shown in Figures 5.7, 5.8 and 5.9. It is apparent from the figures that memory based techniques as well as the proposed techniques are sensitive to the number of nearest neighbors / features.

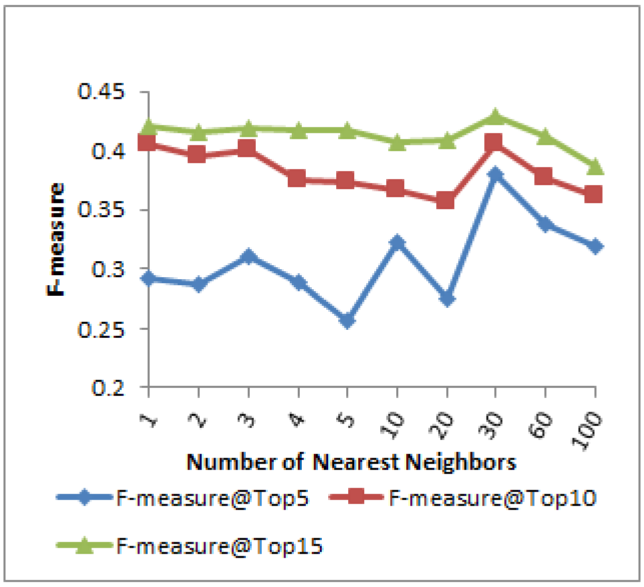


Figure 5.2: Collaborative filtering - *delicious* data set

It is palpable that selecting the right value for this parameter definitely affects the performance of the recommender.

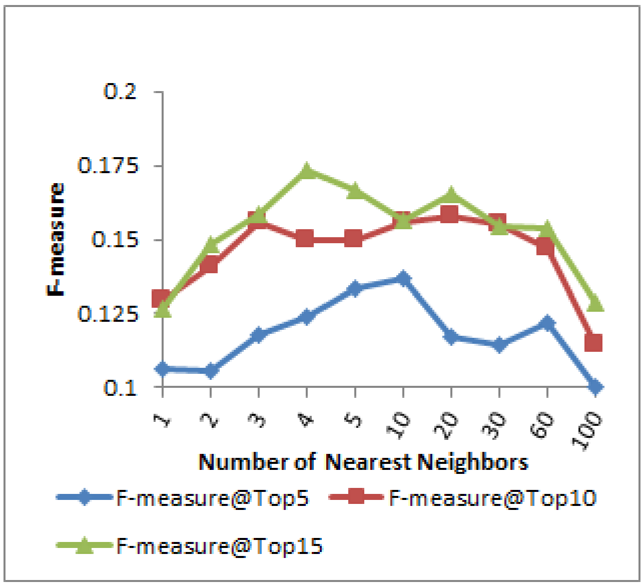


Figure 5.3: Collaborative filtering - *lastfm* data set

While learning from positive and unlabeled examples, it is assumed that unlabeled examples contain errors. In practice, the positive set (set of positive examples) may also contain some errors. This should be considered while learning a classifier from positive and unlabeled examples. In two step techniques, there is no way to incorporate this consideration and all positive examples are considered noise free.

However, as stated earlier, positive examples may also have errors and these erroneous examples may degrade the quality of learnt model through two step techniques which in turn can degrade the performance of two step techniques.

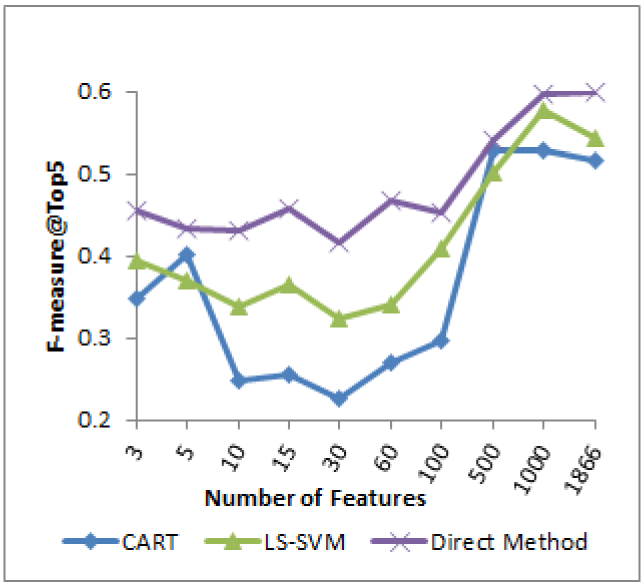


Figure 5.4: F-measure @ Top5 - *delicious* data set

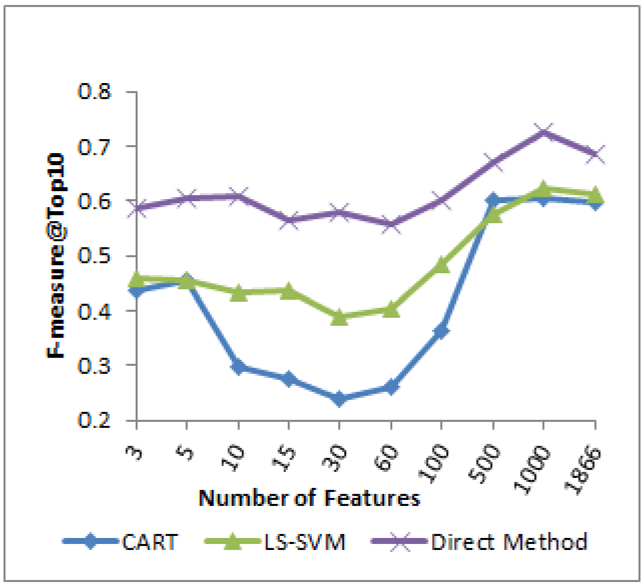


Figure 5.5: F-measure @ Top10 - *delicious* data set

In regular LS-SVM, regularization parameter C is used to weigh errors. There is no way to weigh positive and negative errors differently. In modified LS-SVM, proposed in this thesis, two regularization parameters C_+ and C_- are introduced to weigh positive and negative errors differently. These parameters can be controlled individually and thereby positive and negative errors can be controlled individually while learning the LS-SVM model. This is the reason behind the improved results of modified LS-SVM.

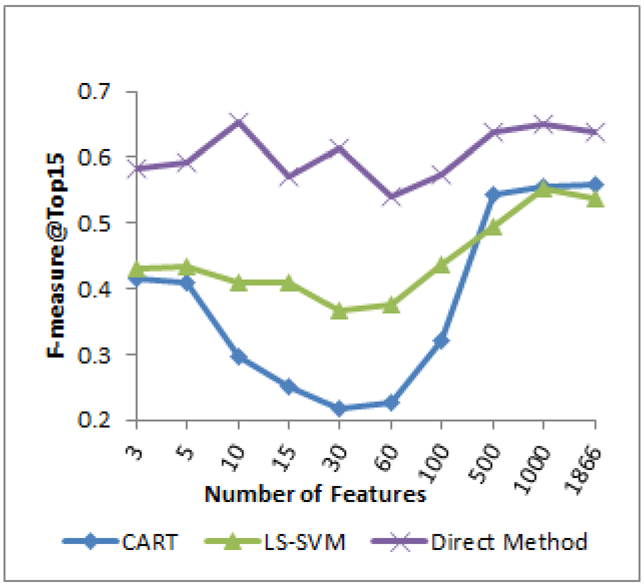


Figure 5.6: F-measure @ Top15 - *delicious* data set

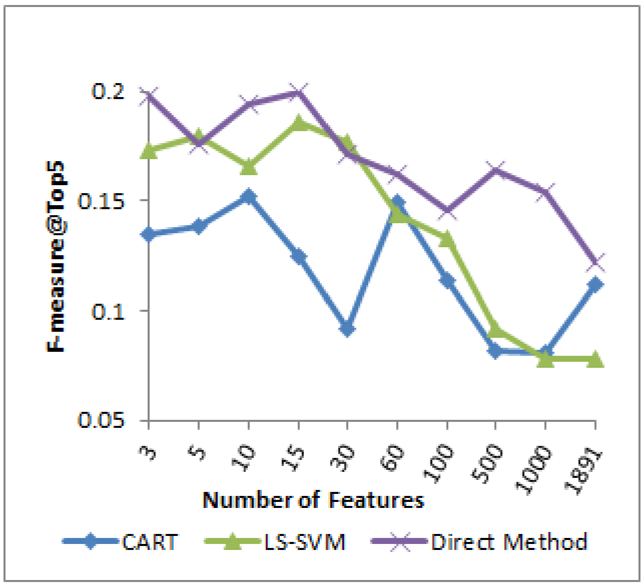


Figure 5.7: Fmeasure @ Top5 - *lastfm* data set

5.8 Conclusions

In this chapter, the problem of personalized resource recommendations under the situation where only positive and unlabeled examples are available is discussed. Methods based on naive Bayes classifier and CART/LS-SVM to learn a recommender using positive and unlabeled (PU) examples are proposed.

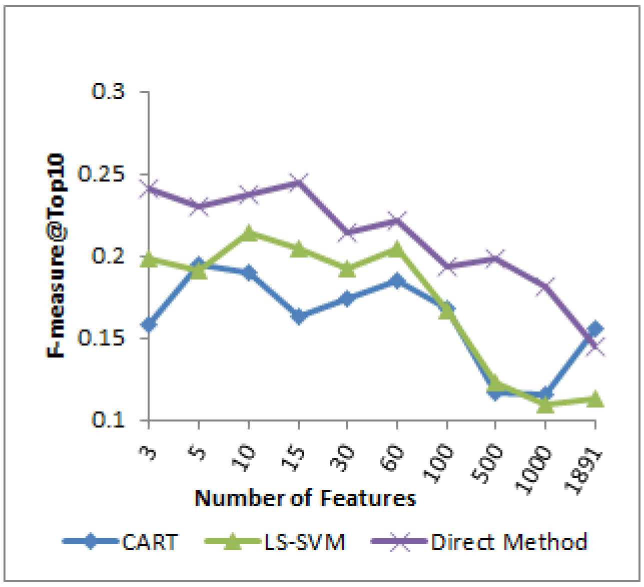


Figure 5.8: Fmeasure @ Top10 - *lastfm* data set

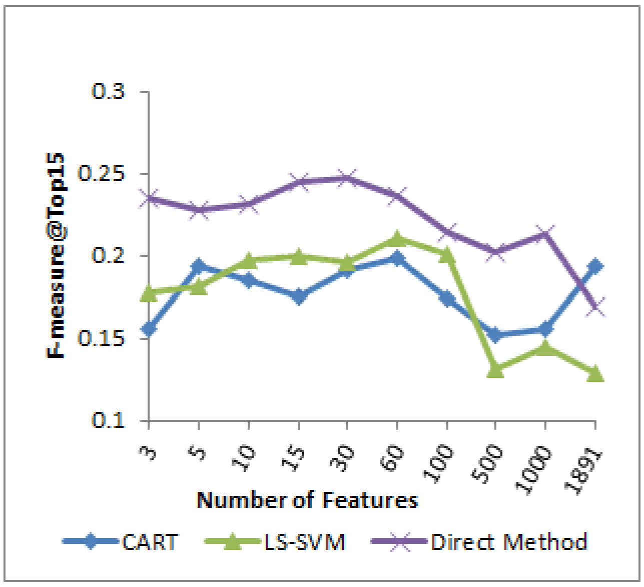


Figure 5.9: Fmeasure @ Top15 - *lastfm* data set

Moreover, a direct method in which LS-SVM is adapted to learn from PU examples is also proposed. The proposed methods also use feature selection to its advantage. Experimental results show that all the proposed techniques perform considerably better than memory based collaborative filtering. Direct method performs the best among all the techniques discussed.

It is furthermore inferred that selecting right number of features definitely affects the accuracy of the recommender.

The tailoring of LS-SVM to enable it learn from positive and unlabeled examples is a unique contribution of this work. To the best of our knowledge, this is the first attempt which models the social resource recommendation as learning from the positive and unlabeled examples.

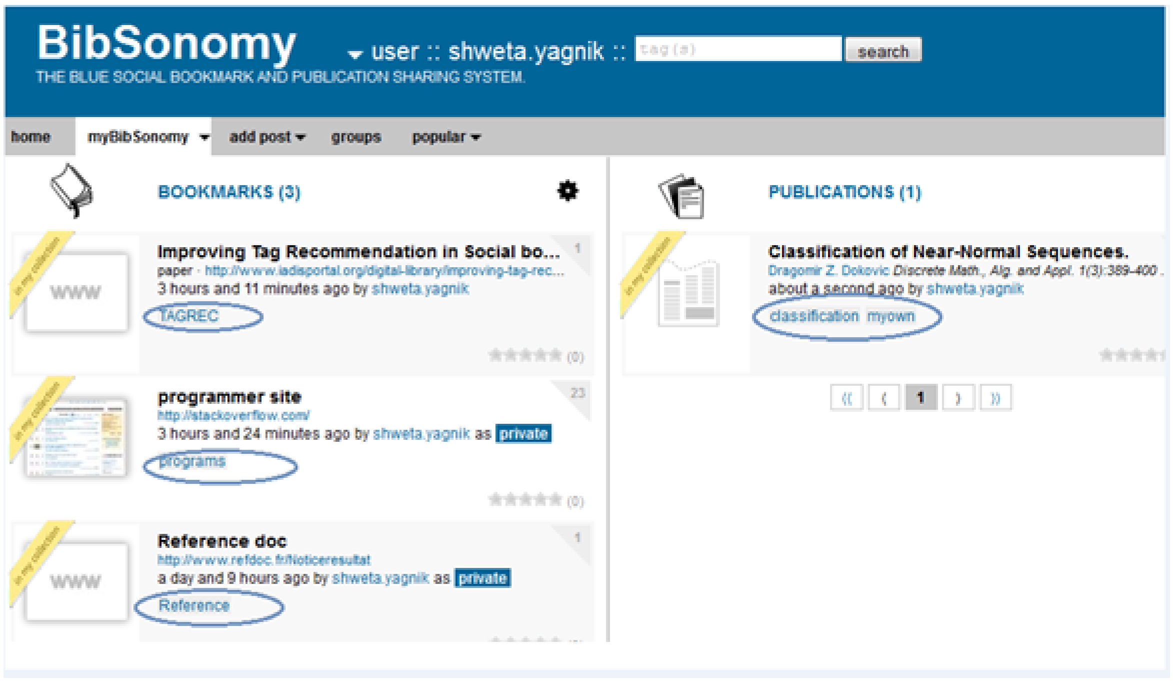
Chapter 6

Recommending Tags for New Resources in Social Bookmarking System

Social bookmarking system is a web-based resource sharing system that allows users to upload, share and organize their resources i.e. bookmarks and publications. The system has shifted the paradigm of bookmarking from an individual activity limited to desktop to a collective activity on the web. It also facilitates user to annotate his/her resource with free form tags that leads to large communities of users to collaboratively create accessible repositories of web resources. Tagging process has its own challenges like ambiguity, redundancy or misspelled tags and sometimes user tends to avoid it as he/she has to describe tag at his/her own. The resultant tag space is noisy or very sparse and dilutes the purpose of tagging. The effective solution is Tag Recommendation System that automatically suggests appropriate set of tags to user while annotating resource. This study proposes a framework that does not depend on tagging history of the resource or user, and hence, capable of suggesting tags to the resources which are being submitted to the system for the first time. Tag recommendation task is modelled as multi-label text classification problem and naive Bayes classifier is used as the base learner of the multi-label classifier. Experiments with Boolean, bag-of-words and Term Frequency-Inverse Document Frequency (TFIDF) representation of the resources are carried out by fitting appropriate distribution to the data based on the representation used. Impact of feature selection on the effectiveness of the tag recommendation is also studied. Effectiveness of the proposed framework is evaluated through Precision (P), Recall (R) and F-measure (F) metrics.

6.1 Introduction

Social bookmarking system allows user to collect, organize, share and label the resources, here, bookmarks or publications, with arbitrary words i.e. Tags. Figure 6.1 shows a snapshot of BibSonomy http://www.bibsonomy.org/ (Benz et al.), a social bookmark and publication sharing system that supports collaborative tagging where user can post his resources and categorize them from his personal point of view by providing tags.



Improving Tag Recommendation in Social bo...
paper · http://www.iadisportal.org/digital-library/improving-tag-rec...
3 hours and 11 minutes ago by shweta.yagnik
TAGREC

programmer site
http://stackoverflow.com/
3 hours and 24 minutes ago by shweta.yagnik as private
programs

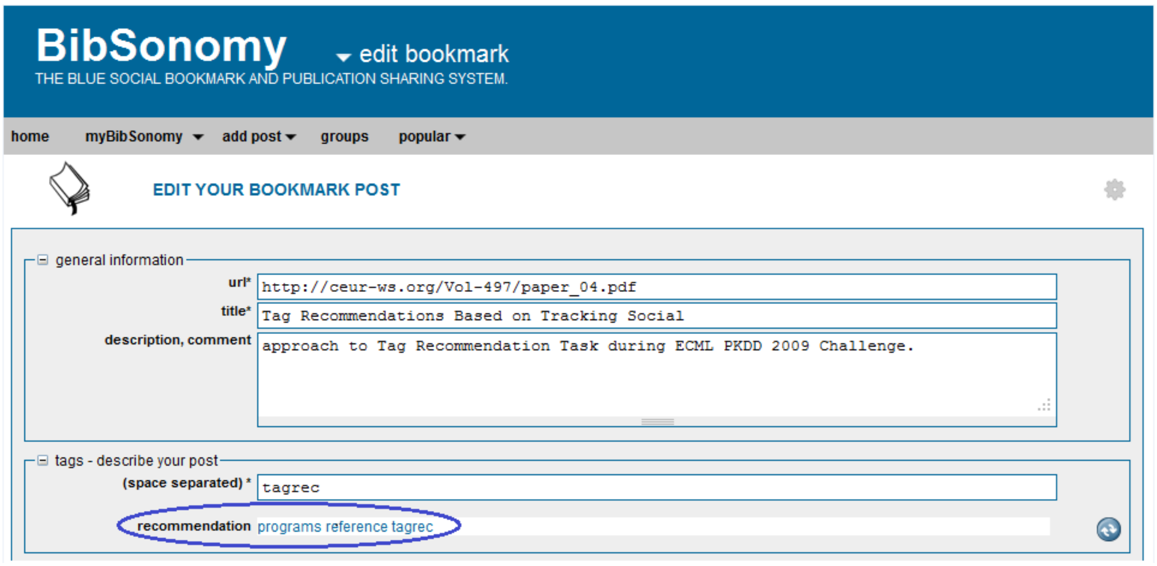
Reference doc
http://www.refdoc.fr/Noticeresultat
a day and 9 hours ago by shweta.yagnik as private
Reference

Classification of Near-Normal Sequences.
Dragomir Z. Dokovic Discrete Math., Alg. and Appl. 1(3):389-400 .
about a second ago by shweta.yagnik
classification myown

Figure 6.1: BibSonomy: social bookmark and publication sharing system

The simplicity of collaborative tagging for user-centric content publishing and management comes at the cost of some challenges. The freedom of selecting tags compels user to write descriptive tags on his own to define his viewpoint which is burdensome and time consuming task (Marchetti et al.). Hence, user may avoid or assign very small number of tags to resource, resulting in very sparse tag space. Further, different users may choose tags based on their knowledge background and preferences i.e. they may describe the same resource based on different granularity level resulting into noisy tag space and create difficulty to find relevant material based on such tags. It is also important to notice that synonymous tags increase data redundancy and polysemous tags i.e. a tag that has many contextual meanings, lead to inappropriate connections between resources.

These hurdles in tagging process create very sparse or noisy tag-space that ultimately dilutes the purpose of tagging for information organization. However, it inspires to develop methods that help users while tagging by automatically recommending an appropriate set of tags. The objective of tag recommendation mechanisms is to ease the process of finding useful tags for a resource by reducing efforts from a manual entry to a mouse click and hence, increasing the chances of getting a resource annotated. The system also helps in consolidating the vocabulary across users which exposes different aspects of a resource. Enriched set of tags help user in reminding what a resource is about. Figure 6.2 shows tag recommendation in BibSonomy. It can be seen that, when user posts a bookmark or publication, the system gives suggestion for tags which are appropriate to the resource being submitted.



url* http://ceur-ws.org/Vol-497/paper_04.pdf
title* Tag Recommendations Based on Tracking Social
description, comment approach to Tag Recommendation Task during ECML PKDD 2009 Challenge.
(space separated) * tagrec
recommendation programs reference tagrec

Figure 6.2: Tag recommendation in BibSonomy

6.2 Related Work

In (Katakis, Tsoumakas, and Vlahavas), tag recommendation problem was modelled as multi-label text classification task. They used tagging history and represented resources as Boolean feature vectors. It is felt that alternative representations of these resources in conjunction with feature selection are worth exploring options. Two tag recommendation approaches were compared in (Jäschke et al.). First was a classic Collaborative Filtering (CF) and other was a graph-based tag recommendation system based on FolkRank algorithm. To reduce sparsity of folksonomy graph, which is main limitation of graph-based methods, p-core processing i.e. graph pruning was used. The evaluation tests were performed on resultant dense part of data set, which may not be representative of real life data.

K-Nearest Neighbour algorithm was adapted for tag recommendation in (Gemmell et al.). They also used p-core processing to deal with noisy tag space and effectively worked on dense part of the data set. (Tatu, Srikanth, and DSilva) derived document and user models from the textual content of the post. In this model, tag suggestions were not only from the existing tag space but also from the metadata provided by user to the resource, like title, description and the content of the document. User model was derived from user's tagging behaviour. (Lipczak) showed in his studies that CF based on cosine similarity between users, calculated from resource content was not a good idea for recommending tags. They proved that there was no correlation in cosine similarity between two users calculated based on tags and content of item. They also suggested potential sources of tags for recommendation focusing user's personomy.

(Gendarmi, Lanubile, and White) designed "Prompter"-A recommender system which suggested tags according to three facets of a social bookmarking system: the personal tagging history, the social tagging behaviour and the textual content of the resource. Evaluation against a snapshot of the BibSonomy data set revealed that, the combination of these three different tag sources improved the precision of generated tag suggestions in the case where users already had a plentiful tagging history and bookmarks that point to popular resources within the community (http://www.kde.cs.unikassel.de/bibsonomy/dumps/). (Hamouda and Wanas) suggested personalized tag recommendation for social bookmarking system based on finding similar users and similar bookmarks. A generalized tag recommendation framework was proposed in (Alepidou, Vavliakis, and Mitkas). It conveyed the semantics of resources according to different user profiles. System was built upon resource's title, user's tagging history and other tags. (Ju and Hwang) exploited previously annotated tags on the same resource, resource descriptions and previously annotated tags by the same person. They devised and deployed a filtering scheme for removing inappropriate candidates and a weighting scheme for combining information from multiple sources.

In this study, tag recommendation is modelled as multi-label text classification problem. Impact of different possible representation of the resource and feature selection is experimentally evaluated.

6.3 Proposed Approach

In the proposed approach, tag recommendation task is modelled as multi-label text classification problem. Multi-label classification is a supervised learning problem where an instance may be associated with multiple labels. Tag recommendation task can be modelled as multi-label classification problem, as, one resource may be annotated with multiple tags based on the relevance with the resource. These different relevant tags also facilitate in exposing multiple aspects of a resource. To handle multi-label classification problem, there are mainly two approaches. First is problem transformation methods that convert the multi-label classification problem into a set of binary classification problems (Read et al.). Binary Relevance (BR), label combination or label power-set method and classifier chains are examples of problem transformation methods. Other approach is algorithm adaptation methods that modify learning ;algorithm to directly perform multi-label classification.

BR problem transformation method is used in the proposed tag recommendation framework. It is a simple classifier that scales linearly with the number of classes in a multi-label classification data set (Read et al.). It considers the prediction of each label as an independent binary classification task, thus each binary model is trained to predict the relevance of one of the labels. To accomplish this, the original data set is transformed into total $|L|$ sets, where L is the set of labels i.e. set of unique tags in this task. Each data set D_λ contains all the examples that are labelled as λ in the original data set. It learns a binary classifier $C_\lambda : X \rightarrow \{\lambda, -\lambda\}$ for each of the labels.

Naive Bayes is used as a base learner because it is computationally efficient as well as optimal for classification tasks even when the conditional independence between attributes assumption is invalid (Zhang, Pena, and Robles). Experiments are carried out with Boolean, bag-of-words and TFIDF representations of resources and accordingly Multivariate Bernoulli distribution (MVBD), Multinomial Distribution (MND) or Normal Distribution (ND) is fitted to the data (Markov and Larose). Mulan package is used for all the experiments reported here (Tsoumakas et al.).

Performance of multi-label classification is measured based on standard information retrieval metrics called precision, recall and f-measure as mentioned in Equations 6.1, 6.2 and 6.3, respectively, where m denotes total number of test instances, P_i is a set of predicted labels and Y_i is set of actual labels for instance x_i (Tsoumakas et al.).

Precision is the number of correct tags retrieved divided by the total number of retrieved tags. Thus, it gives the percentage of correctly recommended tags among all tags recommended by the tag recommendation algorithm (Katakis, Tsoumakas, and Vlahavas).

$$Precision = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap P_i|}{|P_i|} \tag{6.1}$$

Recall is the number of correct tags retrieved divided by the total number of correct tags. Thus, it is the percentage of correctly recommended tags among all tags annotated by the users i.e. actual tags (Katakis, Tsoumakas, and Vlahavas).

$$Recall = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap P_i|}{|Y_i|} \tag{6.2}$$

It is hard to compare two classifiers using two different evaluation metrics. F-measure is harmonic mean of precision and recall which gives a single metric for comparison. F-measure tends to be closer to smaller of the two (Katakis, Tsoumakas, and Vlahavas).

$$F - measure = \frac{1}{m}\sum_{i=1}^{m}\frac{2|Y_i \cap P_i|}{|P_i| + |Y_i|} \tag{6.3}$$

6.4 Data Preprocessing

During the ECML/PKDD Discovery Challenge, Belgium 2008, organizers provided data set of BibSonomy system (http://www.kde.cs.unikassel.de/bibsonomy/dumps/

Benz et al. http://www.bibsonomy.org/). It contains three training files named tas, bookmark and BibTex. Table 6.1 reflects the attributes of all three training files.

Table 6.1: Attributes of three files

File	**Attributes**
tas	user, tag, content_id, content_type, date
bookmark	content_id, url_hash, url, description, extended description, date
BibTex	content_id, journal volume, chapter, edition, month, day, booktitle, howPublished, institution, organization, publisher, address, school, series, bibtexKey, url, type, description, annote, note, pages, bKey, number, crossref, misc, bibtexAbstract, simhash0, simhash1, simhash2, entrytype, title, author, editor, year

The original training tas file contains 816,197 records, bookmark file contains 176,147 and BibTex file contains 92,545 instances. The tas file describes tag assignments made by a user to resource and contains other details like user_id, tag, conten_id (bookmark.content_id or BibTex.content_id), content_type (1 = Bookmark Resource, 2 = BibTeX Resource) and date. For instance, user's tag assignment record is shown in Table 6.2.

In the snapshot of BibSonomy data set used in this study, tas file contains total 304,118 records, where (user_id, content_id) pair appears multiple times based on number of tag assignments by the user to resource. It reveals total amount of tags assigned by users, as each record represents a single tag assigned to the resource. As a part of preprocessing, all tags are converted to lower case and punctuation marks and non-English characters are removed from tag string. After this step, tas file is left with 303,670 records with plain text tag assignments. Thus, each resource is associated with plain text tags, that can be accurately processed to generate recommendation. There are total 173,568 posts of Bookmark resource and 130,102 posts of BibTeX resource. These posts contain 50,000 unique items from each type of resource. For Bookmark data set total 11,067 unique tags and for BibTeX data set 10,878 unique tags are found in the preprocessed data set. Average tag assignment to Bookmark is 3.4 tags and to BibTeX is 2.6 tags.

Table 6.2: Tag assignment to resource

Example	**Attribute**	**Value**
Example 1	user_id	27
	tag	computer
	content_id	938977
	content_type	1
	date	10/10/2005 10:40
Example 2	user_id	27
	tag	quiet
	content_id	938977
	content_type	1
	date	10/10/2005 10:40

Table 6.3: Bookmarked web page in Bibsonomy

Attribute	**Value**
content_id	4145011
url_hash	1a4e59c781ba7f9b9dfb63d493738a1a
url	http://www.epyxmobile.com/
description	Mobile Internet Telephony :: Skype for the road!
extended description	Take Skype with your for the road! Use your mobile phone to call Skype users or receive calls from them, for free! Make phone calls between mobile phones for free, even across country borders!
date	1/5/1989 10:40

The bookmark file contains bookmarked post related information in fields like content_id, url_hash, url, description, extended description and date. Url_hash field uniquely identifies bookmark resource. For instance, one of the web pages bookmarked by user is described by the information shown in Table 6.3.

Table 6.4: Bookmarked publication in Bibsonomy

Attribute	**Value**
content_id	688717
journal volume	Computer Networks and ISDN Systems
chapter	30
bibtexKey	brin1998web
url	http://citeseer.ist.psu.edu/brin98anatomy.html
pages	107-117
number	17
misc	keywords = google pagerank searchengine, priority = 3,citeulike-article-id = 922
bibtexAbstract	In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems
simhash0	7a736d3fbe3935f4a95181ca5fa0368f
simhash1	1234ad3633d435ef79d8a7f36dafa0a9
simhash2	1779c82bd34bbf1ca62956d136a22adf
entrtype	Article
title	The Anatomy of a Large-Scale Hypertextual Web Search Engine
author	Sergey Brin and Lawrence Page
year	1998

The BibTeX file contains bookmarked publication related information in fields like content_id, journal volume, chapter, edition, month, day, booktitle, howPublished, institution, organization, publisher, address, school, series, bibtexKey, url, type, description, annote, note, pages, bKey, number, crossref, misc, bibtexAbstract, simhash0-2, entrytype, title, author, editor and year. Simhash1 uniquely identifies BibTeX entry. Miscellaneous information is collected in the misc field, which may include user comments, non-standard BibTeX fields like isbn, bibdate etc. For instance, one of the publications bookmarked by user is described by the information shown in Table 6.4.

As there exist a huge amount of posted tags, a right number of tags should be selected for reducing the computational cost and avoiding the over fitting problem. Histogram for Bookmark and BibTeX data set is plotted to analyse the frequency of occurrence of tags.

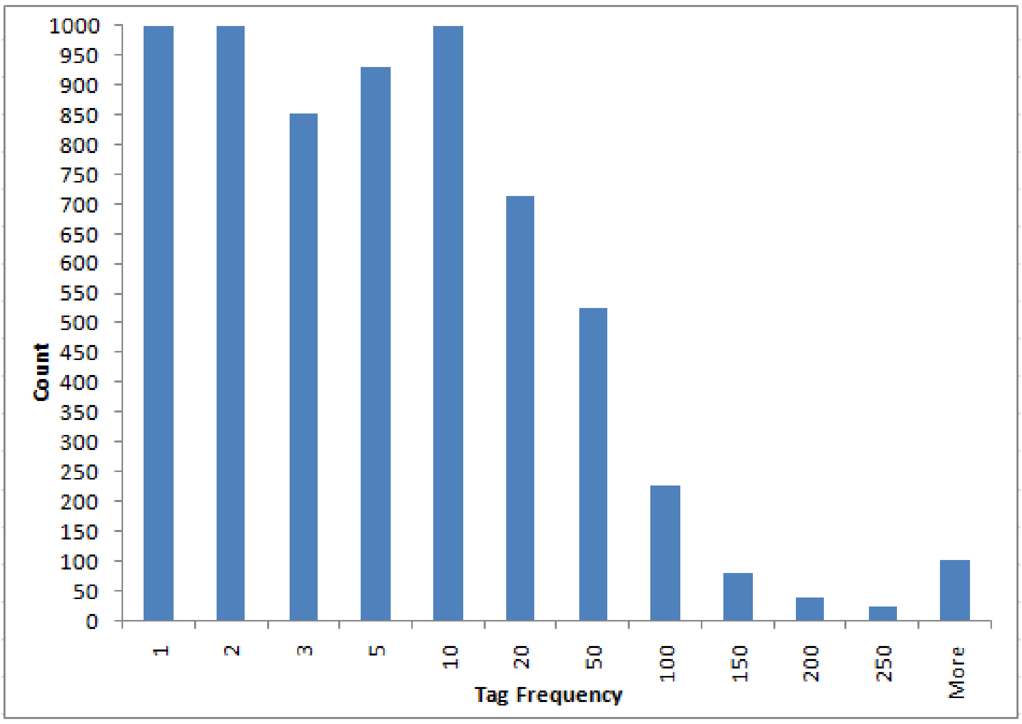


Figure 6.3: Histogram for tag frequency distribution in Bookmark data set

Histograms shown in Figures 6.3 and 6.4 reflect that low frequency tags i.e. tags which are used single, twice, 5-10 times etc. have dominating count. It reveals the fact that, repository has a big number of low-frequency tags, which increases sparsity and complicates the process of retrieving good recommendations. High frequency tags should be considered when designing an effective tag recommender. In order to decrease the dimensionality of the problem, high frequency tags are considered resulting in moderate unique tag count. For bookmark data set, keeping the tag frequency $\geq$ 100 results into 245 unique tags and for BibTeX data set keeping tag frequency $\geq$ 100 results into 111 unique tags. 80% of the resultant data set is kept as training and remaining 20% as testing.

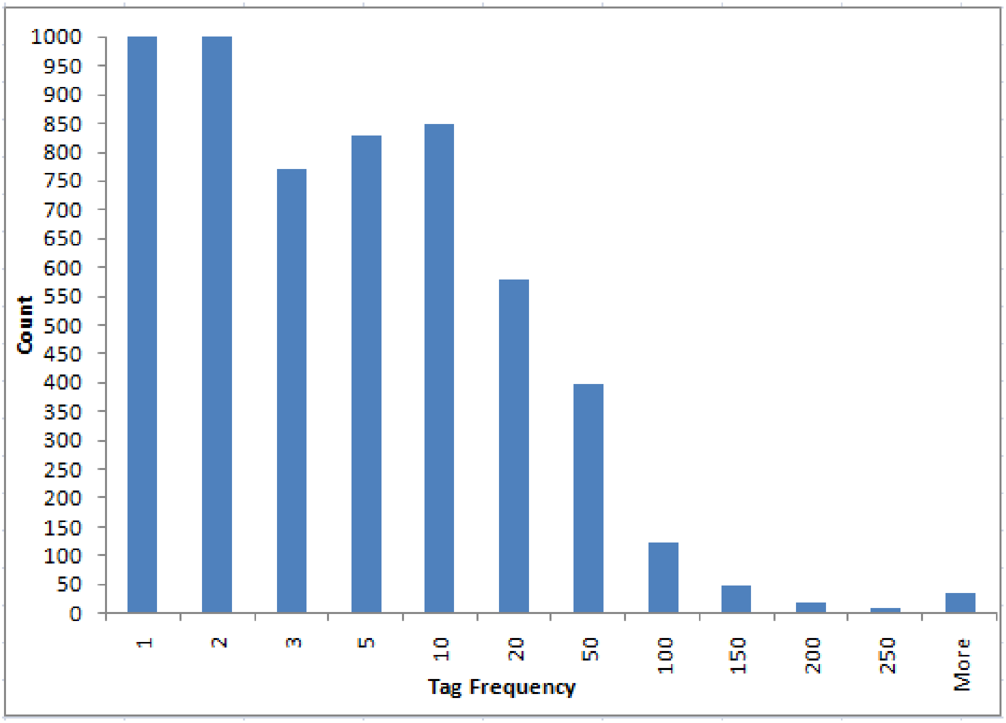


Figure 6.4: Histogram for tag frequency distribution in BibTeX data set

The classifier considers the text representation of the resource for which tags are to be recommended. Experiments are carried out with Boolean, bag-of-words and TFIDF representations of the resources (Markov and Larose). In order to create textual representation for the Bookmark resources, description and extended description fields are used, while for BibTeX resources, journal, booktitle, bitexAbstract and title fields are used from the data set. Weka (Hall et al.) is used to convert string attributes into a set of attributes representing presence/absence, count or TFIDF (Markov and Larose) of words. Bookmark and BibTeX resources are represented with 1,439 and 1,173 attributes, respectively. Figure 6.5 shows the conceptual flow of preprocessing steps followed for the proposed system.

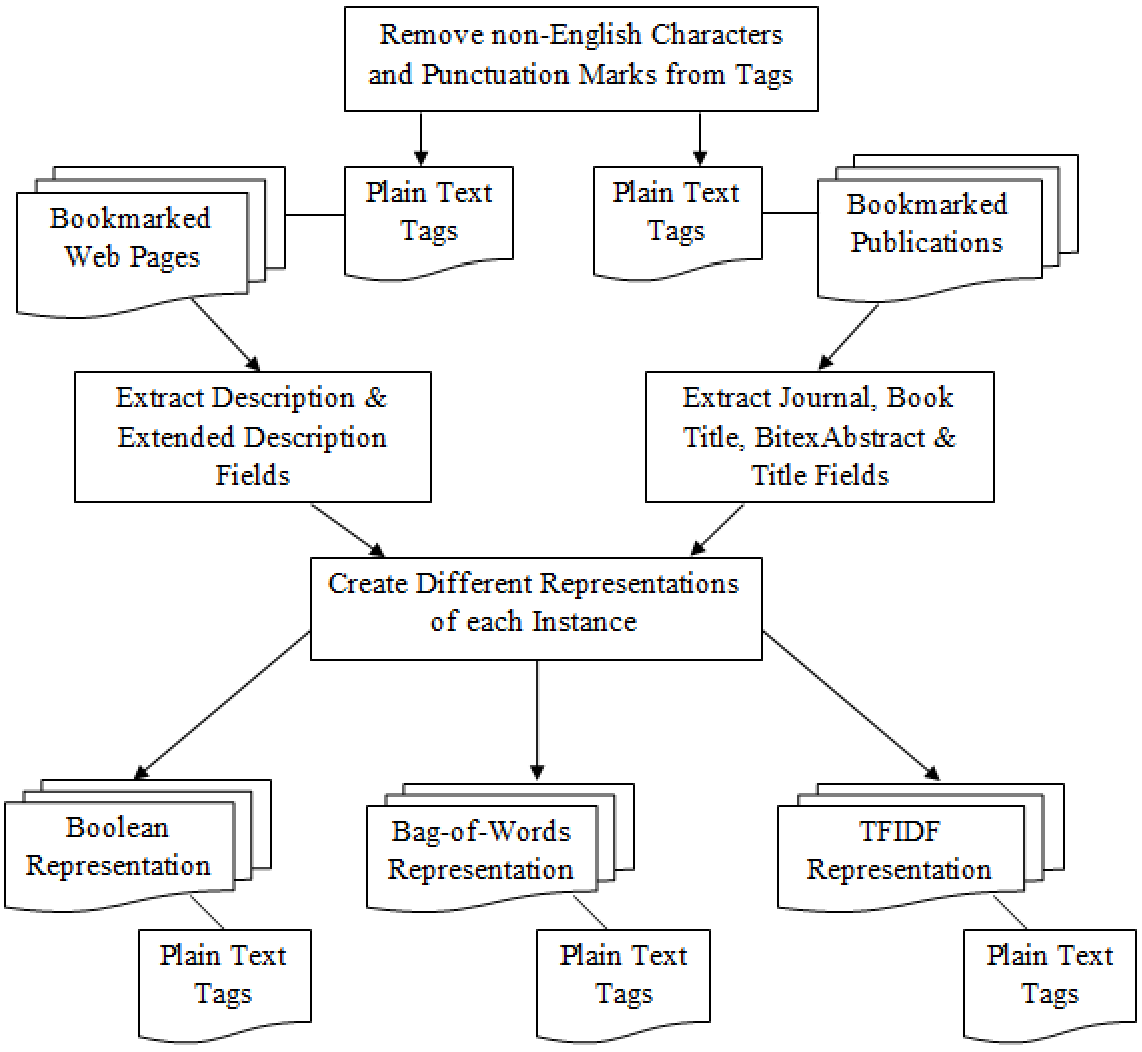

Remove non-English Characters and Punctuation Marks from Tags
Plain Text Tags
Plain Text Tags
Bookmarked Web Pages
Bookmarked Publications
Extract Description & Extended Description Fields
Extract Journal, Book Title, BitexAbstract & Title Fields
Create Different Representations of each Instance
Boolean Representation
Bag-of-Words Representation
TFIDF Representation
Plain Text Tags
Plain Text Tags
Plain Text Tags

Figure 6.5: Conceptual flow of preprocessing steps

6.5 Proposed Tag Recommendation Algorithm

The objective is to predict tags that a user would assign to a particular resource. The important observation from the given data set is that particular resource is submitted only once by the user i.e. only once, any Bookmark or BibTeX resource is submitted by any user. Hence, every test is a new unseen resource, for which, set of tags is to be recommended. In this work, BR classifier from the Mulan (Tsoumakas et al.) package is used. Naive Bayes classifier is used as the base learner of the Binary Relevance (BR) classifier. The proposal in this study is to use bag-of-words representation rather than Boolean or TFIDF representation of resources which allows to fit Multinomial Distribution (MND) rather than Multivariate Bernoulli Distribution (MVBD) or Normal Distribution (ND) to the data. In bag-of-words representation, each attribute is represented as a natural number, indicating the number of occurrences of term in the document. The multinomial distribution defines the posterior probability $P(C|d_j)$ of document d_j belonging to class C as depicted in Equation 6.4 (Markov and Larose).

$$P(C|d_j) = \frac{P(d_j|C)P(C)}{P(d_j)} \tag{6.4}$$

Assume that there are m attributes $a_1, a_2, \ldots, a_m$ and n documents $d_1, d_2, \ldots, d_n$ from class C. Number of times attribute a_i occurs in document d_j is denoted as n_{ij}, and the probability with which attribute a_i occurs in all documents from class C is denoted as $P(a_i|C)$. It is defined by Equation 6.5.

$$P(a_i|C) = \frac{\sum_{j=1}^{n} n_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} n_{ij}} \tag{6.5}$$

The multinomial distribution defines the probability of document d_j given class C as $P(d_j|C)$ and it is as in Equation 6.6.

$$P(d_j|C) = \left(\sum_{i=1}^{m} n_{ij}\right)! \prod_{i=1}^{m} \frac{P(t_i|C)^{n_{ij}}}{n_{ij}!} \tag{6.6}$$

The ordering of words is ignored in the bag-of-words model. To consider all possible orderings of each word ($n_{ij}!$) and all words in the document, $(\sum_{j=1}^{m} n_{ij})!$ is added (Markov and Larose).

As stated earlier, experiments are also carried out with Boolean and TFIDF representation of the resources. In Boolean representation, each attribute is represented by a value 0 or 1 depending on whether or not the corresponding term occurs in the textual representation of the resource. The resources in this representation are binary vectors following the Multivariate Bernoulli Distribution (MVBD). In TFIDF representation, each attribute is represented by a value indicating its term frequency-inverse document frequency. This leads to the continuous-valued vectors following the Normal Distribution (ND). In case of both the data sets, resources are represented with large number of attributes. It is possible that some of the attributes may not be relevant to the classification task and others may be redundant. Feature selection can help to incorporate only those features, which are important for classification task (Zhang, Pena, and Robles). This may improve performance of classification. To incorporate feature selection, BinaryRelevanceAttributeEvaluator is used from Mulan package (Tsoumakas et al.). It evaluates individual attribute based on Weka's

GainRatioAttributeEval evaluation metrics (Hall et al.). Ranker class is used to give ranking to each attribute. Parameter M decides the number of features to be selected. The proposed tag recommender is trained and tested with varying number of features.

A set of K binary classifiers are trained where each classifier c_k corresponds to tag $t_k \in T$, where T is the set of all available tags. T is decided in the preprocessing step for both the data sets. For any new resource d, each classifier c_k predicts the probability/confidence with which it should be annotated with t_k. The final outcome of the process is the set of $TopN$ tags recommended by the classifiers from the available tags.

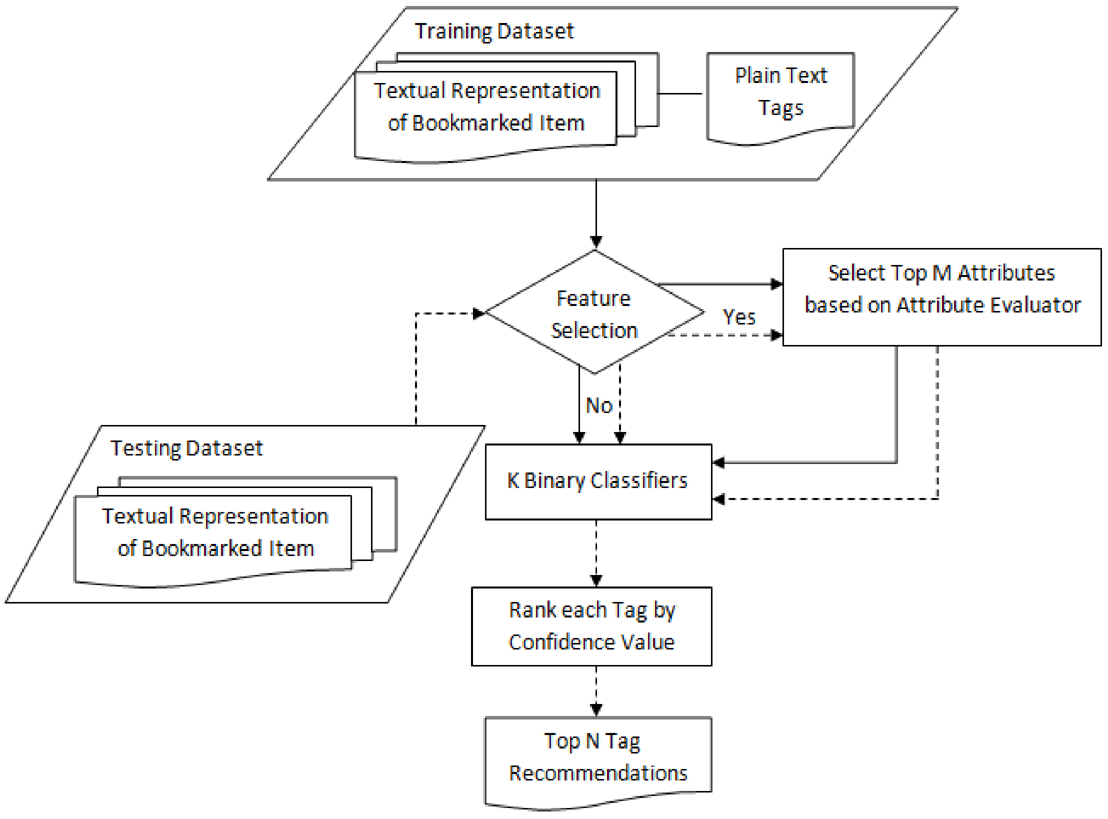

Training Dataset
Textual Representation of Bookmarked Item
Plain Text Tags
Feature Selection
Yes
No
Select Top M Attributes based on Attribute Evaluator
Testing Dataset
Textual Representation of Bookmarked Item
K Binary Classifiers
Rank each Tag by Confidence Value
Top N Tag Recommendations

Figure 6.6: Flowchart of proposed tag recommendation system

The set of resources used to train the classifier is 80% of the set of all the resources previously annotated by the users. During training, each resource that is tagged with t_k is considered as a positive example for c_k, while all other resources which are not tagged with t_k are considered as negative examples for c_k. For each test instance, classifier predicts set of tags and gives ranking to each tag based on probability/confidence value. Figure 6.6 shows the flowchart of the proposed tag recommendation system, where solid lines indicate the learning step, while dotted lines indicate the classification step. To calculate Precision (P), Recall (R) and F-measure (F), tags predicted by the system for each test instance are compared with their true tag assignments. To calculate value of these measures at Top N, only top N resources are retrieved.

6.6 Experimental Evaluation

Using Precision (P), Recall (R) and F-measure (F) as the evaluation measures, performance of Tag Recommender is compared for Boolean, bag-of-words and TFIDF representation of resources.

Table 6.5: Tag recommendation in social bookmarking system - results on Bookmark data set

P/R/F @TopN	**MVBD**	**MND**	**ND**
P/R/F @Top1	0.218/0.078/0.107	0.234/0.080/0.112	0.078/0.030/0.040
P/R/F @Top3	0.136/0.145/0.127	0.148/0.149/0.135	0.057/0.064/0.054
P/R/F @Top5	0.103/0.173/0.117	0.117/0.188/0.131	0.046/0.082/0.054
P/R/F @Top10	0.071/0.224/0.100	0.083/0.249/0.115	0.035/0.114/0.049

Experiments are carried out in two stages. In the first stage, proposed tag recommender is evaluated without using feature selection. Results for bookmark and BibTeX data set are shown in Tables 6.5 and 6.6 respectively. It is clear from the results, that tag recommender performs best when multinomial distribution is fitted to the data.

In the second stage, proposed tag recommender is evaluated with feature selection. Results for different Number of Features (NOF) are summarized in Figures 6.7 and 6.8.

Table 6.6: Tag recommendation in social bookmarking system - results on BibTeX data set

P/R/F @TopN	MVBD	MND	ND
P/R/F @Top1	0.501/0.454/0.465	0.619/0.569/0.581	0.180/0.170/0.172
P/R/F @Top3	0.246/0.629/0.339	0.278/0.712/0.383	0.090/0.235/0.125
P/R/F @Top5	0.165/0.691/0.256	0.187/0.784/0.289	0.069/0.288/0.107
P/R/F @Top10	0.094/0.780/0.163	0.102/0.835/0.175	0.052/0.413/0.088

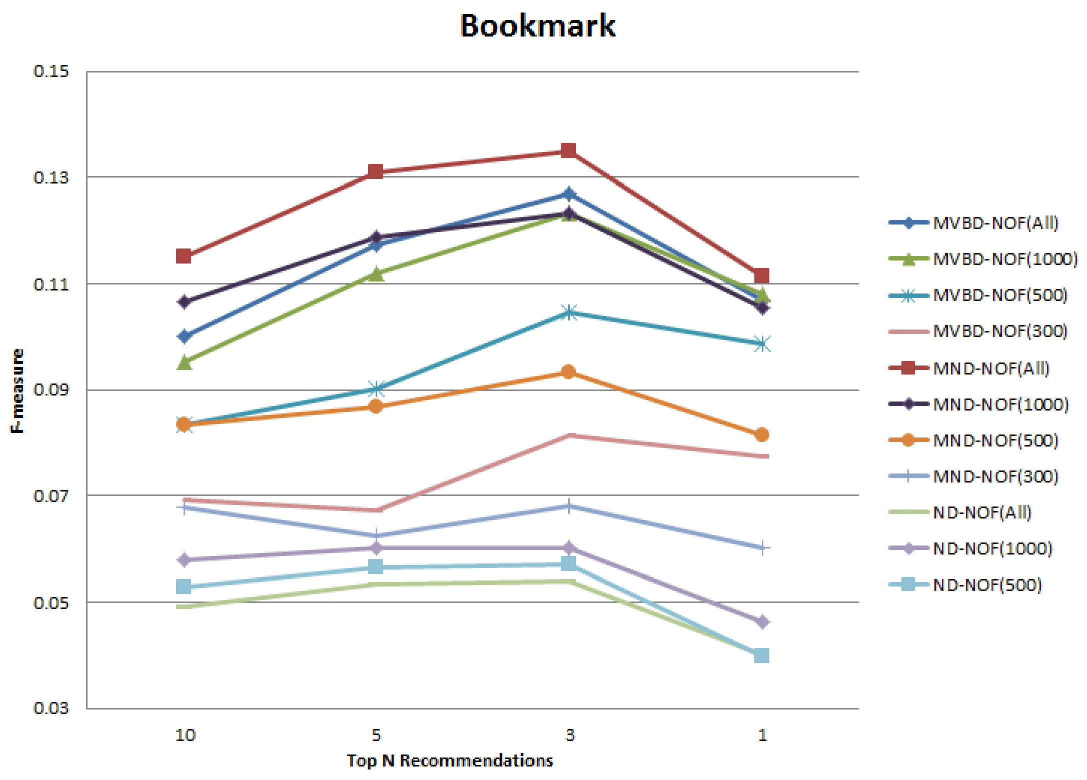


Figure 6.7: Impact of feature selection on Bookmark data set

It can be seen that while using feature selection, results are favourable in case of BibTeX data set, but performance of the tag recommender is degraded in case of Bookmark data set.

6.7 Conclusions

This study models tag recommendation task as multi-label text classification problem. It is evident from the experimental results that, when feature selection is not used, tag recommender performs the best, when multinomial distribution is fitted to the data rather than the scenarios when multivariate Bernoulli distribution or normal distribution is fitted.

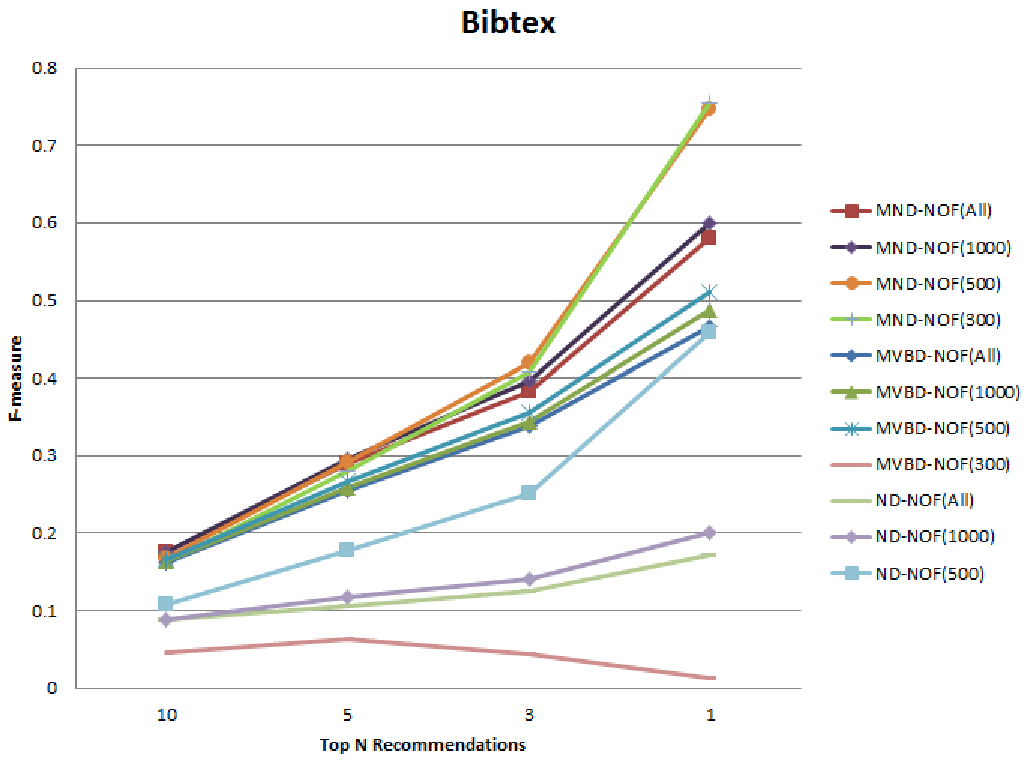


Figure 6.8: Impact of feature selection on BibTex data set

Incorporation of feature selection further improves the performance of tag recommender in case of BibTeX data set but it affects adversely in case of Bookmark data set.

The important fact which should be mentioned here is the generalizability of the proposed approach. In the proposed approach, bookmark and BibTex are modeled using general features which are maintained by all bookmarking system. This enables the proposed approach to be used by any such similar system.

The proposal of modeling the resource using the occurance frequency of words from its textual description and then fitting the multinomial distribution while modeling the problem as multi-label classification problem is a distict contribution to the research owing to the improved performance of the tag recommender.

Chapter 7

Movie Recommender System - Hybrid Filtering Approach

Recommender System can be built using approaches like: (i) Collaborative Filtering (ii) Content Based Filtering and (iii) Hybrid Filtering. In Collaborative Recommender System, ratings of the most similar users (in case of user based collaborative filtering) or items (in case of item based collaborative filtering) are used to predict the rating of the new item (movie, in this study). In Content Based Filtering, user profile is constructed based on the content of the items liked by the user in the past, and then, based on similarity between user and item profile, recommendations are made. Hybrid Filtering combines collaborative and content based approaches. The focus of this study is movie recommendation task. Prediction task is modelled as classification task where the aim is to predict whether the movie will be liked or disliked by the user. An item based recommender which combines usage, tag and movie specific data such as genres, star cast and directors is proposed in this study. The proposed recommender is tested using Hetrec2011-movielens-2k data set. Accuracy and F-measure is used to evaluate the performance of the proposed recommender.

7.1 Introduction

Various movie businesses like Netflix http://www.netflix.com/, IMDB http://www.imdb.com/, and Hulu http://www.hulu.com/ etc. recommend movies to their customers. Although there are several factors which affect the quality of recommender system, recommendations based on common viewpoints of user have become more and more trustworthy and widely used. The recommendation task is often, reduced to the problem of estimating what rating a user would give for an unseen item, or to find a list of items that the user is most likely to enjoy. Movie recommendation is an open research area with unanswered problems and with growing social networking data. There is a need of systematically fusing different types of data about movies and users from various sources to improve the quality of recommendations. As stated earlier, recommendation systems are categorized as content-based, collaborative or hybrid recommender system (Adomavicius and Tuzhilin).

Content-based recommendation system recommends user, items similar to the ones, the user favoured in the past. However, it suffers from the problem such as limited content analysis, over-specialization and new user problem (Adomavicius and Tuzhilin).

User-based Collaborative Filtering (CF) is a technique for producing personalized recommendations by computing the similarity between the current user and other users with similar choices. Thus, the current user choice is predicted by gathering choice information from other users with similar preferences. If choices matched in the past, it is assumed that they will match in future as well. However, it suffers from the problem such as sparsity, new user problem and new item problem (Adomavicius and Tuzhilin). In item-based collaborative filtering, first, similarity between items is found and then to predict the rating of item i by user u, ratings of u, for most similar items of i are used.

Hybrid approaches combine collaborative and content-based methods to overcome certain limitations of these individual techniques. Hybrid Recommender can be built by different ways such as: combining separate recommenders, adding content-based characteristics to collaborative models, adding collaborative characteristics to content-based models and developing a single unifying recommendation model (Adomavicius and Tuzhilin).

In this study, an item based hybrid recommender is proposed that combines usage, tag and movie specific data such as genres, star cast and directors to improve the accuracy of the recommender system.

7.2 Related Work

A separate collaborative and content-based system can be implemented and then can be used to build the hybrid recommender system. Outputs obtained from individual recommendation systems were combined linearly in (Claypool et al.) while (Jäschke et al.) used the voting scheme for the same. In (Melville, Mooney, and Nagarajan), additional ratings were calculated using a pure content-based predictor. These ratings were then used to augment the user's rating vector in collaborative filtering. Latent Semantic Indexing was used in (Soboroff and Nicholas) to generate a collaborative view of a collection of user profiles. A rule-based classifier using content-based and collaborative characteristics was proposed in (Basu, Hirsh, Cohen, et al.).

The book recommender system proposed by (Liang et al.) was built from tag information only. The authors stated that tags could capture the content information of items. However, tags are sometimes meaningful only to the users that assigned them. They can be ambiguous and can also have a lot of synonyms. Authors proposed a way to address this problem by expanding the tag set.

Weighted Tag Rating Recommender (WTRR) proposed in (Nagar) was an extension to the work carried out in Weighted Tag Recommender (WTR) (Liang et al.). WTR exploited tag data but did not use ratings' data and other information available about the items. However, it is to be noticed that tags may not always capture the true preferences of users. This was addressed in WTRR by using actual ratings with tags. One main difference between WTR and WTRR was that, instead of simply counting the number of times a user u_i has tagged an item with the tag t_x, ratings were also considered of the movies which were tagged with t_x by user u_i. Two key observations about WTRR are: (i) it is a user-based recommender system and it does not use all the information available about items apart from tags and ratings and (ii) during prediction, it only uses ratings of only those movies which have also been tagged. These key observations are good candidates for further explorations.

In the approach proposed in this study, movie specific information like genre, star cast and director of the movie is used. This information is used along with ratings and tags to find similarity between items. During rating prediction, all the available ratings are used rather than considering ratings of only those movies, which are tagged also.

7.3 Item-Based Collaborative Filtering

The first objective in item-based collaborative filtering is to find similarity between items (Sarwar et al.). In the implementation of basic item-based collaborative filtering, Pearson Correlation is used to find similarity between items (movies), where, items' profile is in terms of ratings given to them by different users. Similarity between items i and j is calculated using Equation 7.1.

$$sim(i,j) = \frac{\sum_{u \in U} (r_{u,i} - \overline{r_i})(r_{u,j} - \overline{r_j})}{\sqrt{\sum_{u \in U} (r_{u,i} - \overline{r_i})^2} \sqrt{\sum_{u \in U} (r_{u,j} - \overline{r_j})^2}} \tag{7.1}$$

Here, U is the set of users who have rated both i and j, $\overline{r_i}$ is the average rating of item i and $r_{u,i}$ is the rating of item i by user u. Rating of user u for an unseen movie m is predicted using Equation 7.2.

$$r_{u,m} = \frac{\sum_{v \in N(m)} sim(m,v) r_{u,v}}{\sum_{v \in N(m)} |sim(m,v)|} \tag{7.2}$$

Here, $N(m)$ is the ordered set of movies which are most similar to m and rated by user u. In this study, it is considered that if the predicted rating is more than 3, the user likes the movie, otherwise it is considered that user dislikes the movie.

7.4 Proposed Approach

As stated earlier, Hetrec2011-movielens-2k data set is used in this work. From this data set, first of all, user-movie rating matrix and user-movie-tag matrix are constructed. User-movie rating matrix stores ratings of users to movies, while user-movie-tag matrix stores number of times a tag is assigned to the movie by the user. A third matrix, user-movie sub-rating matrix is then constructed from the two matrices. This matrix stores only those movies for which every user has provided a tag as well as a rating. After preparing matrices as above, under mentioned steps are followed. The approach proposed in this study is inspired from the work done in (Nagar) and (Liang et al.).

7.4.1 Movie's Tag Profile Generation

All the users that tagged movies in hetrec2011-movielens-2k data set (Cantador, Brusilovsky, and Kuflik), (http://www.grouplens.org/), (http:

//www.imdb.com/), (www.rottentomatoes.com/) are confined in the user set $U = \{u_1, u_2, \ldots, u_{|U|}\}$. All the movies from the corpus are contained in the movie set $M = \{m_1, m_2, \ldots, m_{|M|}\}$, while all the tags used by the users in U to label movies in M are enclosed in the tag set $T = \{t_1, t_2, \ldots, t_{|T|}\}$.

$R = \{0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$ is used to denote the set of all possible ratings that users can give. Following steps are performed to construct Movie's Tag Profile.

- Calculate the relevance of a tag to a movie as a weight.
- Calculate relevance of a tag to a user as a weight.
- Estimate relatedness between two tags using these weights.
- Construct the tag profile of the movie using relatedness.

Relevance of a Tag to a Movie

To find relevance of a tag to a movie which captures ratings in addition to the tag, (Nagar) proposed the formulation as shown in Equation 7.3.

$$w_{m_i}(t_x) = \frac{\sum_{u_j \in U_{m_i,t_x}} r_{u_j,t_x}(m_i)}{\sum_{u_j \in U_{m_i},t_y \in T_{m_i}} r_{u_j,t_y}(m_i)} \tag{7.3}$$

Here, the numerator represents a summation of the ratings $r_{u_j,t_x}(m_i)$ assigned to the movie m_i by all the users u_j, who used t_x to annotate it. U_{m_i,t_x} denotes the set of users who used t_x to tag m_i. A summation of all the ratings from the users who tagged m_i is represented by the denominator. The true popularity of the tag t_x with respect to a movie m_i is now captured by the value of $w_{m_i}(t_x)$.

Relevance of a Tag to a User

(Nagar) defined relevance of a tag to a user which signified how strongly the user felt about the tag as shown in Equation 7.4.

$$w_{u_i}(t_x) = \frac{\sum_{m_j \in M_{u_i,t_x}} r_{u_j,t_x}(m_j)}{\sum_{m_j \in M_{u_i},t_y \in T_{u_i}} r_{u_i,t_y}(m_j)} \tag{7.4}$$

where, a summation of the ratings assigned to the movie m_j by all the users who used t_x to annotate it, is represented by the numerator. Summation over all ratings assigned to the movie m_j by all the users who tagged it is signified by the denominator.

Tag Relatedness Metric for the Movie

The relatedness of two tags with respect to a movie can be calculated given the relevance of a tag with respect to the user. The relatedness metric is used to avoid semantic ambiguity while constructing the movie profiles. The relatedness metric between two tags t_x and t_y is denoted by $c_{m_i}(t_x, t_y)$ and it represents the degree of correspondence (or connection) between tags with respect to movie m_i. It measures similarity between tags t_x and t_y in the context of the movie m_i. It is computed using Equation 7.5

$$c_{m_i}(t_x, t_y) = \frac{1}{|U_{m_i,t_x}|} \sum_{u_j \in U_{m_i,t_x}} w_{u_j}(t_y) \tag{7.5}$$

Movie's Tag Profile

Assuming tag t_y as the representative for the movie m_i, the weight or relevance of tag t_y to the movie m_i is calculated as a summation of relatedness between the tags used by movie m_i (i.e., $t_x \in T_{m_i}$) and target tag t_y. The total relevance weight of t_y for the movie m_i is denoted as $W_{m_i}(t_y)$. It is defined in Equation 7.6.

$$W_{m_i}(t_y) = \sum_{t_x \in T_{m_i}} w_{m_i}(t_x) c_{m_i}(t_x, t_y) \tag{7.6}$$

Similar to the concept of the Inverse Document Frequency (IDF) (Markov and Larose) in information retrieval, to measure the general importance of the tag in the topic preference identification of the movie, a tag's occurrence for all movies must be taken into consideration. $imf(t_y)$ is used to denote the inverse movie frequency of tag t_y and it is defined in Equation 7.7.

$$imf(t_y) = \frac{1}{log(e + |M_{t_y}|)} \tag{7.7}$$

Here, $|M_{t_y}|$ is the number of movies that is tagged with t_y and e is the Euler's number. It is easy to note that $0 \leq imf(t_y) \leq 1$. Tag profile for each movie is then defined as in Equation 7.8.

$$M_i^T = \{W_{m_i}(t_y) \cdot imf(t_y) | t_y \in T\} \tag{7.8}$$

7.4.2 Preference Profile Generation

Movie profile is two-faceted, comprising of the weighted tag profile (M^T) and of the user preference profile (M^U). Using tags, content-based quality of the approach is captured. Collaborative filtering idea is acquired into the proposed system, by using the rating again, but in a straightforward manner: M^U is a vector with U elements, each corresponding to a user in the corpus. The value of the elements are either 0 or 1, depending on whether or not the movie has been rated by the user.

7.4.3 Neighborhood Formation

In order to predict user's rating for an unseen movie m, first, list of movies similar to m is found. The fundamental idea is to recognize for each movie m, an ordered list of N most similar movies, $M = \{m_1, m_2, \ldots, m_N\}$ such that $m \in M$ and $sim(m, m_1)$ is maximum, $sim(m, m_2)$ is the second highest and so on. The $N-$nearest movies are selected based on the similarity value.

Each movie is encoded with its own topic preferences and user preferences, where topic preferences are captured by tags while user preferences are captured by simplified ratings. The similarity between two movies m_i and m_j based on tags is denoted as $sim^T_m(m_i, m_j)$, where T is the set of tags used to tag movies m_i and m_j. Pearson correlation coefficient is used to measure this similarity between two movies which are represented by the set of all weighted tags.

The similarity between two movies m_i and m_j based on user preference is denoted as $sim^U_m(m_i, m_j)$ and is defined in Equation 7.9.

$$sim^U_m(m_i, m_j) = \frac{\sum_{u_k \in U_{m_i} \cap U_{m_j}} imf(u_k)}{\sqrt{|U_{m_i}||U_{m_j}|}} \tag{7.9}$$

where, U is the set of all users, $|U_{m_i}|$ specifies the number of users who have rated movie m_i, $imf(u_k)$ designates the inverse movie frequency of user u_k and it is defined in Equation 7.10.

$$imf(u_k) = \frac{1}{log(e + |M_{u_k}|)} \tag{7.10}$$

where, $|M_{(u_k)}|$ indicates the number of movies which have been rated by user u_k.

Given the tag and rating profiles of movies m_i and m_j, the similarity between these two, based on the tag and rating profile is given by Equation 7.11.

$$sim(m_i, m_j) = \omega \cdot sim^T_m(m_i, m_j) + (1 - \omega) \cdot sim^U_m(m_i, m_j) \tag{7.11}$$

ω is a weighting parameter such that $0 \leq \omega \leq 1$. It controls the extent of the collaborative dimension of the algorithm. As the value of ω is decreased, the algorithm becomes predominantly collaborative, and the contribution from the movies user preferences dominates. During the experimental phase, ω is varied to see the impact on the quality of recommendations. Similarity between movies is also found from their genre, star cast and director profile. It is to be noticed that genre, star cast and director profiles of a movie are actually Boolean vectors indicating association between them and a movie. Any combination of these profiles of a movie is also represented by a Boolean vector which is obtained by concatenating Boolean vectors of the profiles which are being combined. Experiments are also carried out where weighted combinations of these profiles is used in calculation of similarity between movies. In star-cast profile of an item, only first five actors of each movie (according to the order in which they appear on the IMDB's cast page of the movie) are considered. Pearson correlation is used throughout to calculate similarity between movies.

7.4.4 Rating Prediction Formula

Equation 7.2 is used to predict the rating of user u for an unseen movie m. If the predicted rating is greater than 3, it is considered that the user will like the movie otherwise it is considered that user will dislike the movie. These steps are summarized in the Figure 7.1.

7.5 Experimental Evaluation

7.5.1 Data Set

In all experiments presented here, data set hetrec2011-movielens-2k dated May 2011 is used (Cantador, Brusilovsky, and Kuflik), http://www.rottentomatoes.com/, http://www.imdb.com/, http://www.grouplens.com/. (Cantador, Brusilovsky, and Kuflik) have made it available to the public. It is based on the original MovieLens10M data set, published by the GroupLens research group. In this data set, movies also refer to their corresponding web pages at the IMDB website. The data set contains 2,113 users, 10,197 movies and a total of 13,222 unique tags. These tags fall into 47,957 tag assignment tuples of the form [user, tag, movie]. It also contains 855,598 user ratings ranging from 0.5 to 5.0, in increments of 0.5, leading to a total of 10 distinct rating

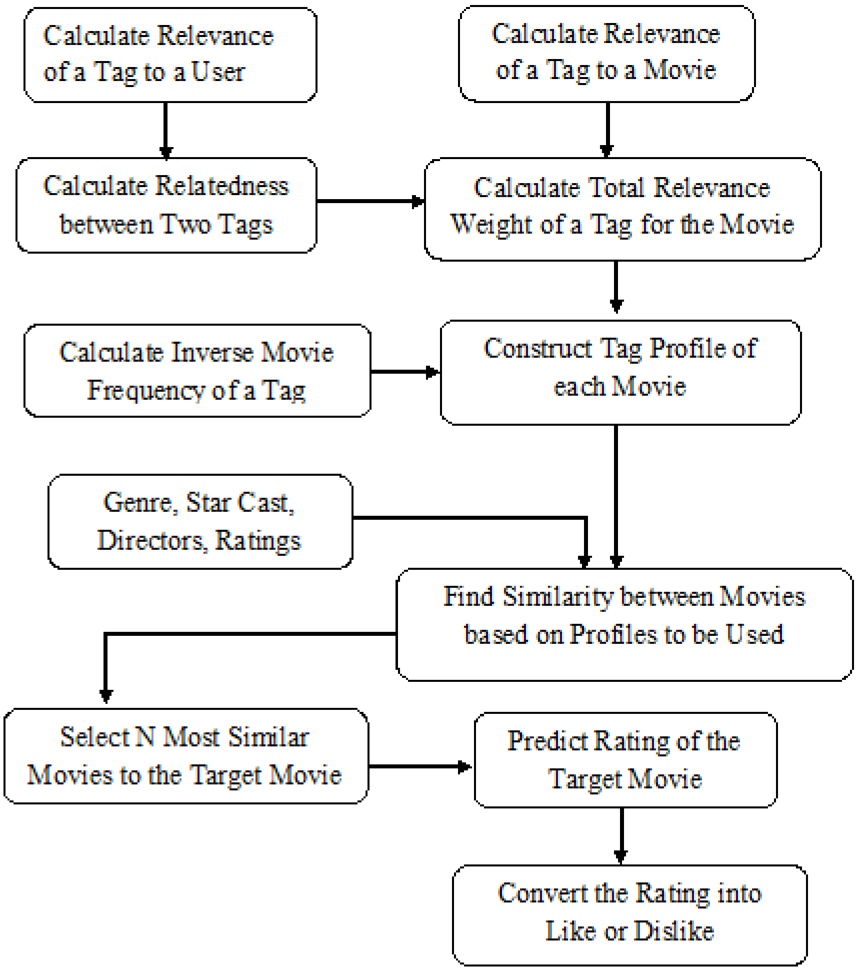

Calculate Relevance of a Tag to a User
Calculate Relevance of a Tag to a Movie
Calculate Relatedness between Two Tags
Calculate Total Relevance Weight of a Tag for the Movie
Calculate Inverse Movie Frequency of a Tag
Construct Tag Profile of each Movie
Genre, Star Cast, Directors, Ratings
Find Similarity between Movies based on Profiles to be Used
Select N Most Similar Movies to the Target Movie
Predict Rating of the Target Movie
Convert the Rating into Like or Dislike

Figure 7.1: Conceptual flow of the proposed recommender system

values. There is an average of 405 ratings per user, and 85 per movie. There are 20 genre types, 20,809 movie-genre assignments, 4060 directors and 95,321 actors. There are average 22 actors per movie. The data is pre-processed to construct user-movie rating matrix and user-movie-tag matrix. A third matrix, user-movie sub-rating matrix is then constructed from the two matrices. This matrix stores rating of only those movies which have been tagged. In construction of star-cast profile of movie, only those actors who have worked in more than 2 movies are considered. This data set has been previously used in (Bothos et al.), (Said et al.), (Jones, Ghosh, and Sharma).

7.5.2 Experimental Methodology and Results

To evaluate and compare outcome of experiments, 5-fold cross validation is carried out for all the experiments performed in this study. For each of the experiments, 20 items are selected as the target items. These selected items are rated and tagged by minimum of 20 and maximum of 50 users. Accuracy and f-measure are used as the performance evaluators. Following experiments are carried out in this study.

- Experiment 1: Basic item-based collaborative filtering, where, similarity between movies is found using user-movie rating matrix, and predictions are also made using user-movie rating matrix.

- Experiment 2: Hybrid filtering, where, similarity between movies is found using genre profile of the movies, and predictions are made using user-movie rating matrix.

- Experiment 3: Hybrid filtering, where, similarity between movies is found using genre and star cast profile of the movies, and predictions are made using user-movie rating matrix.

- Experiment 4: Hybrid filtering, where, similarity between movies is found using genre, star cast and director profile of the movies, and predictions are made using user-movie rating matrix.

- Experiment 5: Hybrid filtering, where, similarity between movies is found using Boolean tag profile of the movies, and predictions are made using user-movie rating matrix.

- Experiment 6: Hybrid filtering, where, similarity between movies is found using bag-of-words tag profile of the movies and predictions are made using user-movie rating matrix.

- Experiment 7: Hybrid filtering, where, similarity between movies is found using term-frequency (TF) (Markov and Larose) tag profile of the movies, and predictions are made using user-movie rating matrix.

- Experiment 8: Hybrid filtering, where, similarity between movies is found using term-frequency-inverse document frequency (TFIDF) (Markov and Larose) tag profile of the movies, and predictions are made using user-movie rating matrix.

- Experiment 9: Hybrid filtering, where, similarity between movies is found by setting $\omega = 0.9$ in Equation 7.11, and predictions are made using user-movie sub-rating matrix.
- Experiment 10: Hybrid filtering, where, similarity between movies is found by setting $\omega = 0.9$ in Equation 7.11, and predictions are made using user-movie rating matrix.
- Experiment 11: Hybrid filtering, where, similarity between movies is found by setting $\omega = 1.0$ in Equation 7.11, and predictions are made using user-movie rating matrix.
- Experiment 12: Hybrid filtering, where, similarity between movies is found by modelling movie profiles as combination of tag profiles ($\omega = 1.0$ in Equation (7.11) and ratings, and predictions are made using user-movie rating matrix.
- Experiment 13: Hybrid filtering, where, similarity between movies is found by modelling movie profiles as combination of tag profiles ($\omega = 1.0$ in Equation 7.11) and genre profile, and predictions are made using user-movie rating matrix.
- Experiment 14: Hybrid filtering, where, similarity between movies is found by modelling movie profiles as combination of ratings and genre profile, and predictions are made using user-movie rating matrix.
- Experiment 15: Hybrid filtering, where, similarity between movies is found first using movie's tag profile and then using their genre profile. To compute the final similarity between movies these two similarities are combined with weight 0.8 and 0.2 respectively. Predictions are made using user-movie rating matrix.

Experiments are performed with varying size of the neighborhood. However, for each of the techniques, results for that size of neighborhood where the technique has performed the best is reported. It is evident from the result that hybrid recommender system outperforms the basic item-based collaborative filtering in all settings apart from that in experiment 5, 6, 7 and 9. The approach proposed in (Nagar) used the user-movie sub-rating matrix for the calculation of rating to be predicted. Approach proposed in this study uses user-movie rating matrix to calculate ratings to be predicted. Use of user-movie sub-rating matrix is obvious for the construction of movie profile and finding similarity between movies, but using user-movie rating matrix rather than user-movie sub-rating matrix during the phase of rating prediction is advocated in this work. This allows to predict based on more number of ratings which leads to the improvement in the performance.

Performance of the recommender under various settings as discussed above is shown in Table 7.1.

Table 7.1: Experimental results

Experiment	**No. of Nearest Neighbors**	**Accuracy**	**F-measure**
Experiment 1	100	0.7024	0.6880
Experiment 2	100	0.7198	0.7220
Experiment 3	500	0.7454	0.7510
Experiment 4	500	0.7454	0.7510
Experiment 5	40	0.7101	0.6775
Experiment 6	40	0.7090	0.6567
Experiment 7	40	0.7090	0.6567
Experiment 8	40	0.7465	0.6956
Experiment 9	5	0.6698	0.5957
Experiment 10	20	0.7570	0.7384
Experiment 11	20	0.7570	0.7384
Experiment 12	100	0.7117	0.6921
Experiment 13	100	0.7430	0.7258
Experiment 14	100	0.7198	0.7220
Experiment 15	10	0.7726	0.7511

7.6 Conclusions

For the task of movie recommendation, an item-based hybrid filtering approach which combines usage, tag and content data of movies is proposed. Movie recommendation task is modelled as classification problem where it is predicted that the user will like or dislike the movie. Movie Recommender system proposed in this study exploits movie specific data such as movie genres, star cast and directors in addition to the ratings and weighted tags. Item profiles are constructed from the careful combinations of different types of data. Similarity between items is then calculated based on these profiles. The improvement in the quality of the recommendations is evident from the experimental results.

An attempt to combine weighted tags, ratings and movie specific data to build an item-based recommender for the task of movie recommendation can definitely be considered as useful conntribution to the research.

Chapter 8

Evaluating a Recommender learnt from Labeled and Unlabeled Data

Supervised learning algorithms require labeled training examples from every class to engender a classification function. One of the shortcomings of this classical paradigm is that in order to learn the function accurately, a large number of labeled examples are needed. There are many situations (e.g. a new user in an online recommender system) where for every class, only a small set of labeled examples is available. Situations such as these encourage to investigate about the usefulness of unlabeled examples in learning a recommender. The main objective of this study is to examine the influence on the accuracy of the recommender when it is built using unlabeled examples in addition to the labeled examples. Co-Training algorithm allows to incorporate unlabeled examples while learning a classifier/recommender (Liu Blum and Mitchell). Usefulness of this algorithm is investigated by means of experimental study using hetrec2011-movielens-2k data set. Accuracy and f-measure are used as the evaluation measures.

8.1 Introduction

Recommender system is one of the applications to predict rating or preference for the items that have not been seen by a user. This system typically produces a list of recommendations. As stated earlier, recommending books, CDs, and other products at amazon.com, movies by MovieLens, and news at VERSIFI Technologies (formerly adaptiveInfo.com) are examples of such applications to name a few (Adomavicius and

Tuzhilin).

In a recommender system, the items which are liked or disliked by the user are the labeled examples about the users' preference. However, there are many other items which are not rated by the user. These items form the set of unlabeled examples. One of the major problem with recommender system is that it does not have adequate number of labeled examples for the new user.

In supervised learning, labeled training examples from every class are used by the learning algorithms to generate a classification function. The biggest problem of this common prototype is that, in order to learn accurately, large number of labeled examples are needed. Labelling can be very labor intensive and time consuming, since it is often done manually. In many applications, such as movie recommender system, for many user only small number of ratings (i.e. labeled examples) are available. A recommender learnt through these small number of ratings may not be adequately accurate and useful. The number of ratings of a user increases only gradually. A partially supervised learning algorithm as its name suggests, does not need full supervision, and thus is able to lessen the labelling effort. This type of learning exhibits a small set of labeled examples of every class, and a large set of unlabeled examples. The idea is to make use of the unlabeled examples to increase the prediction accuracy of a learner. Co-training algorithm belongs to the family of partially supervised learning algorithms and is able to incorporate unlabeled data in addition to the labeled data in learning a classifier/recommender. Efficacy of co-training algorithm in learning a recommender by exploiting unlabeled examples in the presence of small number of labeled examples is critical.

8.2 Related Work

Co-Training algorithm is special in a way that it allows learning from labeled and unlabeled examples. Co-training was formalized in (Blum and Mitchell). They provided a theoretical guarantee for accurate learning, subject to certain assumptions. In (Nigam and Ghani), it was shown that co-training produced more accurate classifiers than the Expectation-Maximization (EM) algorithm, even for data sets which did not satisfy the strict underlying assumptions. Co-training algorithm basically allows a classifier/recommender to exploit unlabeled data in addition to the labeled data during the learning process. A new user of the collaborative recommender system exhibits small number of labeled examples about his/her interest. It is difficult to learn an accurate recommender based on this few labeled examples. Therefore, the use of the unlabeled examples to improve the performance of a recommender through co-training algorithm is worth investigating.

Application of co-training was investigated for word sense disambiguation in (Mihalcea). Author investigated optimal and empirical parameter setting methods. Author proposed a new method that combined co-training with majority voting. A PAC analysis on co-training style algorithm was presented in (Wang and Zhou). They showed that co-training process could succeed even without two views, given that two learners had large differences. They also proved theoretically that co-training process could not improve the performance further, after a number of rounds. A spectral clustering algorithm having a flavor of co-training was proposed in (Kumar and Daumé). The major advantage of their algorithm was that there were no hyperparameters to set. They empirically compared their proposal with number of baseline methods on synthetic and real-world data set. Cross-lingual sentiment classification was addressed with the help of co-training in (Wan). Experimental results showed that their approach performed better than that of standard inductive and transductive classifiers.

8.3 Co-Training Algorithm

Co-training assumes that the set of features can be partitioned into two subsets (x_1 and x_2). For learning the target classification function, each of them is sufficient. This division of features is not considered by the traditional learning algorithms. This feature division is exploited by co-training to learn separate classifiers over each of the feature sets.

The co-training algorithm employed in this study is shown in Algorithm 4 (Liu Blum and Mitchell). In the implementation reported here, x_1 portion refers to rating profile, while x_2 portion refers to genre profile of a movie.

Algorithm 4 Co-training algorithm

```
Input: labeled set $L$, unlabelled set $U$
Output: two classifiers
1: Create a pool $U'$ of examples by selecting $u$ examples at random from $U$
2: Learn a classifier $f$ using $L$ based on $x_1 + x_2$ (i.e. all features) of the examples $x$
3: repeat
4:     Learn a classifier $f_1$ using $L$ based on only $x_1$ portion of the examples $x$
5:     Learn a classifier $f_2$ using $L$ based on only $x_2$ portion of the examples $x$
6:     Apply $f_1$ to classify the examples in $U'$, for each class $c_i$, pick $n_i$ examples that
       $f_1$ has most confidently classified as class $c_i$, and add them to $L$
7:     Apply $f_2$ to classify the examples in $U'$, for each class $c_i$, pick $n_i$ examples that
       $f_2$ has most confidently classified as class $c_i$, and add them to $L$
8:     Randomly select $\sum_{i=1}^{|C|} 2n_i$ examples from $U$ to replenish $U'$ ($|C|$ represents
       number of classes)
9: until $k$ iterations
10: return $f_1$ and $f_2$.
```

Co-training returns two classifiers. At classification time, the two classifiers are applied separately for each test example and their scores are combined to decide the final class. For naive Bayesian classifiers, two probability scores are multiplied, i.e.,

$$Pr(c_j|x) = Pr(c_j|x_1)Pr(c_j|x_2) \tag{8.1}$$

The probability calculated in Equation 8.1 represents results of the recommender based on labelled and unlabeled data. Let this recommender be denoted as f^*.

8.3.1 Experimental Evaluation

8.3.2 Data set

Data set hetrec2011-movielens-2k is used for experimental evaluation. It is already discussed in section 7.5.1.

8.3.3 Evaluation Measures

Precision, Recall, F-measure and Accuracy are used as the evaluation measures. They are already discussed in section 3.5.1.

8.3.4 Experimental Methodology, Results and Discussions

For experimentation purpose, 24 users, who have rated at least 100 items are selected as the test users. For each of the test users, 80% of the movies rated by him/her are considered as a part of training set. Remaining movies form the testing set. It is to be noted that when x_1 portion of the features is considered, each of the observation in training set and testing set is a real-valued vector describing a movie in terms of ratings of users other than the test user under focus. When x_2 portion of the features is considered, each of the observation in training set and testing set is a Boolean vector describing a movie in terms of its associated genres. This implies that each vector based on x_1 portion of the features is a real-valued vector but with large number of missing values. A missing value at a particular position in this vector indicates that corresponding user has not rated the movie represented by the vector. Experiments are carried out for all the test users and reported results are aggregated over the results of each of the test users. Co-training algorithm is executed with parameter values of $u = 200$, $n_i = 5$ and $k = 50$. It is to be noted that $|C|$ denotes the number of classes and $|C| = 2$ in the experiments performed in this study. All three classifiers f, f_1, and f_2 are learnt through naive Bayes algorithm.

For each of the test users, two kinds of experiments are carried out. In the first experiment, recommender (f in the algorithm) is learnt only using the labeled data while in the second experiment, recommenders (f_1 and f_2) are learnt through labeled and some number of unlabeled data. Experiments are carried out by changing the number of labeled data, to see the impact of unlabeled data in the presence of different number of labeled data. These results are summarized in Table 8.1. It can be seen from the results that using unlabeled data in addition to the labeled data while learning the recommender improves the performance of the recommender. It is also evident that the performance of the recommenders improve when number of labeled data increases.

Impact of feature selection on recommender while using only labeled data and labeled plus unlabeled data is also studied. In this experiment, for each of the test user, all the ratings which are available are used as the labeled data. Feature selection is performed only on the rating profile (e.g. x_1 portion of the features). Results

Table 8.1: Aggregated results

Number of Labeled Examples	**Aggregated Results of a Recommender (f) (Only Labeled Data)**		**Aggregated Results of a Recommender (f*) (Labeled + Unlabeled Data) using Co-training**	
	Accuracy	F-measure	Accuracy	F-measure
10	0.5644	0.5905	0.6884	0.6957
20	0.5718	0.5986	0.6917	0.6986
30	0.6032	0.6236	0.6773	0.6849
50	0.6213	0.6428	0.6981	0.7056
100	0.6245	0.6418	0.7056	0.7120
All	0.6539	0.6601	0.7136	0.7131

are depicted in Figures 8.1 and 8.2. Figures clearly suggest that learning a recommender using appropriate number of features definitely affects the performance of the recommender. This is not only true for the conventional recommender that uses only labeled data, but also true for the recommender based on co-training algorithm, which incorporates unlabeled data in learning process.

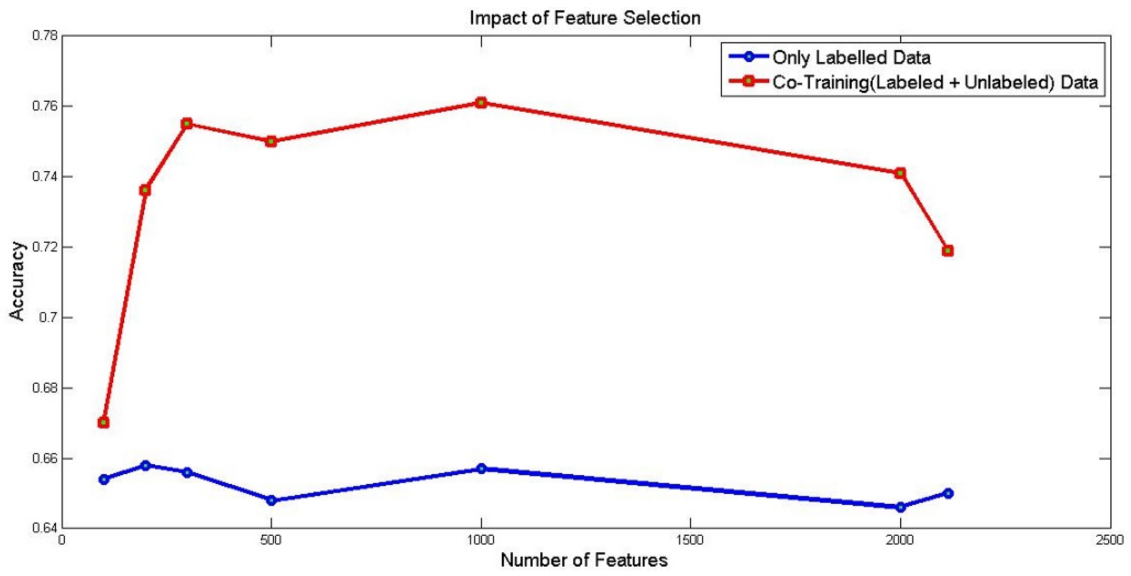


Figure 8.1: Impact of feature selection on accuracy

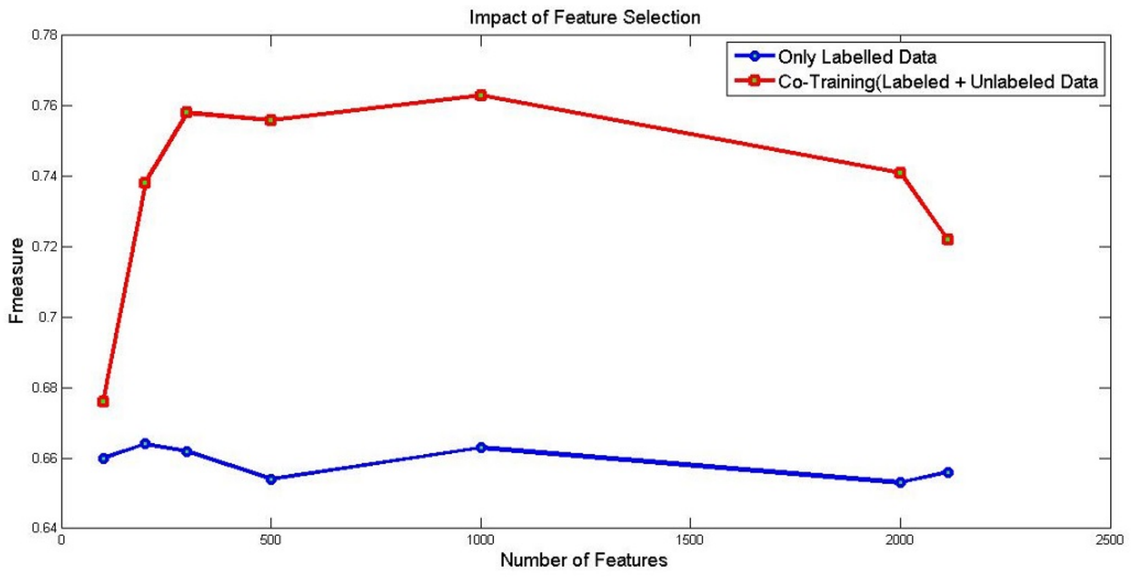


Figure 8.2: Impact of feature selection on f-measure

8.4 Conclusions

The usefulness of unlabeled examples in learning the recommender in the presence of varying number of labeled examples is evaluated in this study. From the experimental results, it is evident that accuracy of the recommender is improved, when it is learnt using unlabeled data in addition to the labeled data. The improvement in the performance is almost 12% when the labeled data is as few as 10 to 20. Results also demonstrate that selecting right number of features while learning a recommender further improves the performance. One potential direction for the future work can be the experimentation on other data sets.

The performance of a movie recommender system is improved using unlabeled examples and to the best of our knowledge this is the first attmept of its kind in this domain. Owing to the success of the proposed approach, it can be considered as the unique contribution to the research.

Chapter 9

Conclusions and Future Work

Having done the exhaustive study of recommender systems belonging to various domains, stock market prediction systems, social resource recommender, tag recommender for social bookmarking system and hybrid movie recommender have been developed. These developments ensure the application of recommender system in various domains. Summary of work done, conclusions derived therein and possible future work are mentioned herewith in detail.

- Forecasting in stock markets is one of the domains focused in the thesis. Two forecasting systems are developed for prediction in stock markets. First system addresses the problem of predicting direction of movement of stock market index and stock price. The focus of the second forecasting system is on predicting future values of stock market index. Predictions from these forecasting systems can be used to provide a kind of financial service to the users. This may enable the users to make correct decisions while investing in stock markets.

  It was identified from the literature, that, machine learning techniques can be used to develop prediction models for stock markets. Artificial Neural Network (ANN), Support Vector Machine, Random Forest and Naive Bayes classifier are used in this thesis to develop the prediction models. It is also felt that improvement in representing and preparing the data which is to be used as the input to the prediction models, can improve the prediction performance. This is achieved by means of Trend Deterministic Data Preparation Layer and Two Stage Fusion Models for the task of predicting direction of movement and value of stock market index respectively. Experimental results are in alignment with the intuition behind which these Trend Deterministic Data Preparation Layer and two stage fusion models are proposed.

Improvement of accuracy with the help of Trend Deterministic Data Preparation Layer which is based on common investor's methods for stock investing, promotes the idea of pre-processing the data based on the domain in which machine learning algorithms are used. This idea can be further extended not only in stock domain by incorporating other human approaches of investing but also in various other domains where recommender systems and machine learning techniques are used. Ten technical indicators are used to construct the knowledge base, however, other macro-economic variables like currency exchange rates, inflation, government policies, interest rates etc. that affect stock market can also be used as the inputs to the models or in construction of the knowledge base of a recommender system. Average volume of a stock is also a potential candidate that may be useful in deciding the trend. It is worth noticing that at Trend Deterministic Data Preparation Layer, technical indicators' opinion about stock price movement is categorized as either 'up' or 'down'. Multiple categories like 'highly possible to go up', 'highly possible to go down', 'less possible to go up', 'less possible to go down' and 'neutral signal' are worth exploring. This may give more accurate input to prediction algorithms. The other careful observation reveals that the focus while predicting movement is short term prediction. Long term prediction can also be thought as one of the future directions which may involve analysis of stock's quarterly performance, revenue, profit returns, companies organizational stability etc. In this thesis, for predicting in stock market, technical indicators are derived based on the period of last 10 days (e.g. SMA, WMA, etc.). It is worth exploring the significance of the length of this period, particularly, when the objective is long term prediction.

In the work related to predicting future values of stock market indices, design parameters of SVRs in the first stage are determined experimentally, however it may be worth exploring algorithms such as genetic algorithm to tune the design parameters of these SVRs. This may lead to more accurate prediction of statistical parameters by these SVRs. Another direction for future work can be to use more statistical parameters as inputs to find much better correlation.

- The problem of social resource recommendation under the situation, where only positive and unlabelled examples are available, is also addressed in this thesis. Social bookmarking sites such as Bibsonomy or Delicious allow users to bookmark URLs and submit the research articles. User bookmarking a resource (URL) or submitting a resource (research article) on this system, implicitly indicates his likings to the resource. Other resources (URLs/research articles), however, do not imply negative preference of the user about them. This leads to the situation where we have positive examples, but no negative examples for user preference.

  The memory based collaborative filtering has served as the most widely used technique for resource recommendations, but it has its own limitations of reliance on ad hoc heuristic rules and dependence of success on availability of a critical mass of users. If a learning based approach is to be devised, as the alternative to memory based collaborative filtering for the task of social resource recommendation, it requires to learn the recommender from positive and unlabeled examples.

  Hence, two step methods based on naive Bayes classifier and CART/LS-SVM to learn a recommender using positive and unlabeled examples are proposed. Moreover, a direct method in which LS-SVM is adapted to learn from positive and unlabeled examples is also proposed. Experimental results validate the theoretic assumption behind the proposal of this scheme and optimal results are achieved.

  A recommender that can exploit content data of the resources to generate recommendation may be an interesting direction for the future work. Another direction for the future work is to fuse the recommendations from the usage based and content based recommender to improve the performance.

- It is known that, social bookmarking system allows users to upload, share and organize their resources. It also facilitates user to annotate his resource with free form tags. The freedom of selecting tags compels user to write descriptive tags on his own to define his viewpoint, which is burdensome and time consuming task. Hence, user may avoid or assign very small number of tags to resource, resulting in very sparse tag space. Further, different users may choose tags based on their knowledge background and preferences i.e. they may describe the same resource based on different granularity level resulting into noisy tag space and create difficulty to find relevant material based on such tags. It is important to notice that synonymous tags increase data redundancy and polysemous tags i.e. a tag that has many contextual meanings, lead to inappropriate connections between resources.

The effective solution is Tag Recommendation System that automatically suggests appropriate set of tags to user while annotating resource. Hence, a tag recommender is implemented to assist the user in tagging process. The recommendation task is modelled as multi-label text classification problem. Textual content of the resources is used to learn the tag recommender. Using the textual content, resources are represented with various Information Retrieval models such as Boolean, bag-of-words and TFIDF. It is evident from the result that, the tag recommender performs the best when it is learnt through the bag-of-words representation of the resources.

A personalized tag recommender is definitely an interesting direction for future work.

- Movie recommendation is an open research area with unanswered problems and with growing social networking data. A movie recommender system, that is solely based on ratings or content of the movies may not be accurate enough. There is a need of systematically fusing different types of data about movies and users from various sources to improve the quality of recommendations.

  Hence, an item-based hybrid filtering approach which combines usage, tag and content data of movies is proposed. Movie recommendation task is modelled as classification problem where it is predicted that the user will like or dislike the movie. Movie Recommender system proposed in the thesis exploits movie specific data such as movie genres, star cast and directors in addition to the ratings and weighted tags. Item profiles are constructed from the careful combinations of different types of data. Similarity between items is then calculated based on these profiles. It is evident from the results that fusing the different kind of data related to movie, appropriately, improves the accuracy of the recommender.

A possible direction for the future work may be the use of machine learning techniques for exploiting these profiles and generating recommendations.

- In a recommender system, the items which are liked or disliked by the user are the labeled examples about the users' preference. However, there are many other items which are not rated by the user. These items form the set of unlabeled examples. One of the major problems with recommender system is that, if it does not have adequate number of labeled examples for the user for whom recommendations are to be made, it may not be adequately accurate and useful.

  The usefulness of unlabeled examples in learning the recommender in the presence of small number of labeled examples is crucial and hence, studied in this thesis. From the experimental results, it is evident that, accuracy of the recommender is improved when it is learnt using unlabeled data in addition to the labeled data.

  A interesting issue is that researchers have yet not shown that when the labeled data set is sufficiently large, the unlabeled data still help. This can serve as the useful future direction. There is also a need to evaluate the usefulness of unlabeled examples in different domains and on different datasets in order to reach to some concrete conclusions.

Works Cited

Abraham, Ajith, Baikunth Nath, and Prabhat Kumar Mahanti. "Hybrid intelligent systems for stock market analysis." *Computational Science-ICCS 2001*. Springer, 2001. 337–345.

Adomavicius, Gediminas and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions." *Knowledge and Data Engineering, IEEE Transactions on* 17.6 (2005): 734–749.

Ahmed, Shahid. "Aggregate economic variables and stock markets in India." *International Research Journal of Finance and Economics* 14 (2008): 141–164.

Aldin, Mahniood Moem, Hasan Dehghan Dehnavr, and Somayye Entezari. "Evaluating the Employment of Technical Indicators in Predicting Stock Price Index Variations Using Artificial Neural Networks (Case Study: Tehran Stock Exchange)." *International Journal of Business & Management* 7.15 (2012).

Alepidou, Zinovia I, Konstantinos N Vavliakis, and Pericles A Mitkas. "A semantic tag recommendation framework for collaborative tagging systems." *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)*. IEEE, 2011. 633–636.

Araújo, Ricardo de A and Tiago AE Ferreira. "A morphological-rank-linear evolutionary method for stock market prediction." *Information Sciences* 237 (2013): 3–17.

Asadi, Shahrokh, et al. "Hybridization of evolutionary Levenberg–Marquardt neural networks and data pre-processing for stock market prediction." *Knowledge-Based Systems* 35 (2012): 245–258.

Baeza-Yates, Ricardo, Berthier Ribeiro-Neto, et al. *Modern information retrieval.* Vol. 463. ACM press New York, 1999.

Balabanović, Marko and Yoav Shoham. “Fab: content-based, collaborative recommendation.” *Communications of the ACM* 40.3 (1997): 66–72.

Basu, Chumki, Haym Hirsh, William Cohen, et al. “Recommendation as classification: Using social and content-based information in recommendation.” *AAAI/IAAI*. 1998. 714–720.

Belkin, Nicholas J and W Bruce Croft. “Information filtering and information retrieval: two sides of the same coin?” *Communications of the ACM* 35.12 (1992): 29–38.

Benz, Dominik, et al. “The social bookmark and publication management system bibsonomy.” *The VLDB JournalThe International Journal on Very Large Data Bases* 19.6 (2010): 849–875.

Billsus, Daniel and Michael J Pazzani. “Learning Collaborative Information Filters.” *ICML*. 1998. 46–54.

—. “User modeling for adaptive news access.” *User modeling and user-adapted interaction* 10.2-3 (2000): 147–180.

Blum, Avrim and Tom Mitchell. “Combining labeled and unlabeled data with co-training.” *Proceedings of the eleventh annual conference on Computational learning theory*. ACM, 1998. 92–100.

Bogers, Toine and Antal Van den Bosch. “Comparing and evaluating information retrieval algorithms for news recommendation.” *Proceedings of the 2007 ACM conference on Recommender systems*. ACM, 2007. 141–144.

Bogers, Toine and Antal Van Den Bosch. “Fusing recommendations for social bookmarking web sites.” *International Journal of Electronic Commerce* 15.3 (2011): 31–72.

Bollerslev, Tim. “Generalized autoregressive conditional heteroskedasticity.” *Journal of econometrics* 31.3 (1986): 307–327.

Bothos, Efthimios, et al. “Information market based recommender systems fusion.” *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*. ACM, 2011. 1–8.

Breese, John S, David Heckerman, and Carl Kadie. “Empirical analysis of predictive algorithms for collaborative filtering.” *Proceedings of the Fourteenth conference* *on Uncertainty in artificial intelligence.* Morgan Kaufmann Publishers Inc., 1998. 43–52.

Breiman, Leo. "Random forests." *Machine learning* 45.1 (2001): 5–32.

Breiman, Leo, et al. *Classification and regression trees.* CRC press, 1984.

Burke, Robin. "Hybrid recommender systems: Survey and experiments." *User modeling and user-adapted interaction* 12.4 (2002): 331–370.

Cantador, Ivan, Peter Brusilovsky, and Tsvi Kuflik. "Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011)." *RecSys.* 2011. 387–388.

Cerulo, Luigi, Charles Elkan, and Michele Ceccarelli. "Learning gene regulatory networks from only positive and unlabeled data." *Bmc Bioinformatics* 11.1 (2010): 228.

Chen, An-Sing, Mark T Leung, and Hazem Daouk. "Application of neural networks to an emerging financial market: forecasting and trading the Taiwan Stock Index." *Computers & Operations Research* 30.6 (2003): 901–923.

Cheng, Cheng, Wei Xu, and Jiajia Wang. "A Comparison of Ensemble Methods in Financial Market Prediction." *Computational Sciences and Optimization (CSO), 2012 Fifth International Joint Conference on.* IEEE, 2012. 755–759.

Claypool, Mark, et al. "Combining content-based and collaborative filters in an online newspaper." *Proceedings of ACM SIGIR workshop on recommender systems.* Citeseer, 1999.

Clements, Maarten, Arjen P de Vries, and Marcel JT Reinders. "Optimizing single term queries using a personalized Markov random walk over the social graph." *Workshop on Exploiting Semantic Annotations in Information Retrieval (ESAIR).* 2008.

Davis, Alexandre, et al. "Named entity disambiguation in streaming data." *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1.* Association for Computational Linguistics, 2012. 815–824.

Delgado, Joaquin and Naohiro Ishii. "Memory-based weighted majority prediction." *SIGIR Workshop Recomm. Syst. Citeseer.* Citeseer, 1999.

Domingos, Pedro and Michael Pazzani. "On the optimality of the simple Bayesian classifier under zero-one loss." *Machine learning* 29.2-3 (1997): 103–130.

Eijlander, Ph and Antonius Marinus Bogers. "Recommender systems for social bookmarking." (2009).

Garg, A, S Sriram, and K Tai. "Empirical analysis of model selection criteria for genetic programming in modeling of time series system." *Computational Intelligence for Financial Engineering & Economics (CIFEr), 2013 IEEE Conference on.* IEEE, 2013. 90–94.

Gemmell, Jonathan, et al. "Adapting K-Nearest Neighbor for Tag Recommendation in Folksonomies." *ITWP* 528 (2009).

Gendarmi, Domenico, Filippo Lanubile, and Bebo White. "Improving tag recommendations in social bookmarking systems: a preliminary study." *Proc. of the IADIS International Conference WWW/Internet.* 2009. 133–140.

Getoor, Lise and Mehran Sahami. "Using probabilistic relational models for collaborative filtering." *Workshop on Web Usage Analysis and User Profiling (WEBKDD'99).* Citeseer, 1999.

Goldberg, Ken, et al. "Eigentaste: A constant time collaborative filtering algorithm." *Information Retrieval* 4.2 (2001): 133–151.

Good, Nathaniel, et al. "Combining collaborative filtering with personal agents for better recommendations." *AAAI/IAAI.* 1999. 439–446.

Hadavandi, Esmaeil, Arash Ghanbari, and Salman Abbasian-Naghneh. "Developing an evolutionary neural network model for stock index forecasting." *Advanced Intelligent Computing Theories and Applications.* Springer, 2010. 407–415.

Hadavandi, Esmaeil, Hassan Shavandi, and Arash Ghanbari. "Integration of genetic fuzzy systems and artificial neural networks for stock price forecasting." *Knowledge-Based Systems* 23.8 (2010): 800–808.

Hall, Mark, et al. "The WEKA data mining software: an update." *ACM SIGKDD explorations newsletter* 11.1 (2009): 10–18.

Hamouda, Sally and Nayer Wanas. "PUT-Tag: personalized user-centric tag recommendation for social bookmarking systems." *Social network analysis and mining* 1.4 (2011): 377–385.

Han, Jiawei, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques.* Morgan kaufmann, 2006.

Hanani, Uri, Bracha Shapira, and Peretz Shoval. “Information filtering: Overview of issues, research and systems.” *User Modeling and User-Adapted Interaction* 11.3 (2001): 203–259.

Hassan, Md Rafiul, Baikunth Nath, and Michael Kirley. “A fusion model of HMM, ANN and GA for stock market forecasting.” *Expert Systems with Applications* 33.1 (2007): 171–180.

Hofmann, Thomas. “Collaborative filtering via gaussian probabilistic latent semantic analysis.” *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval.* ACM, 2003. 259–266.

—.“Probabilistic latent semantic indexing.” *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 1999. 50–57.

Hotho, Andreas, et al. “Information retrieval in folksonomies: Search and ranking.” *The semantic web: research and applications.* Springer, 2006. 411–426.

Hsieh, David A. “Chaos and nonlinear dynamics: application to financial markets.” *The journal of finance* 46.5 (1991): 1839–1877.

Hsu, Sheng-Hsun, et al. “A two-stage architecture for stock price forecasting by integrating self-organizing map and support vector regression.” *Expert Systems with Applications* 36.4 (2009): 7947–7951.

http://www.bibsonomy.org/.

http://www.kde.cs.unikassel.de/bibsonomy/dumps/. “Knowledge and data engineering group, University of Kassel: Benchmark folksonomy data from BibSonomy.”

Huang, Cheng-Lung and Cheng-Yi Tsai. “A hybrid SOFM-SVR with a filter-based feature selection for stock market forecasting.” *Expert Systems with Applications* 36.2 (2009): 1529–1539.

Huang, Shian-Chang and Tung-Kuang Wu. “Integrating GA-based time-scale feature extractions with SVMs for stock index forecasting.” *Expert Systems with Applications* 35.4 (2008): 2080–2088.

Huang, Wei, Yoshiteru Nakamori, and Shou-Yang Wang. "Forecasting stock market movement direction with support vector machine." *Computers & Operations Research* 32.10 (2005): 2513–2522.

Jäschke, Robert, et al. "Tag recommendations in social bookmarking systems." *Ai Communications* 21.4 (2008): 231–247.

Jones, Clinton, Joydeep Ghosh, and Aayush Sharma. "Learning multiple models for exploiting predictive heterogeneity in recommender systems." *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems.* ACM, 2011. 17–24.

Ju, Sanghun and Kyu-Baek Hwang. "A weighting scheme for tag recommendation in social bookmarking systems." *Proc. the ECML/PKDD 2009 Discovery Challenge Workshop.* 2009. 109–118.

Kara, Yakup, Melek Acar Boyacioglu, and Ömer Kaan Baykan. "Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange." *Expert systems with Applications* 38.5 (2011): 5311–5319.

Katakis, Ioannis, Grigorios Tsoumakas, and Ioannis Vlahavas. "Multilabel text classification for automated tag suggestion." *Proceedings of the ECML/PKDD.* 2008.

Kazem, Ahmad, et al. "Support vector regression with chaos-based firefly algorithm for stock market price forecasting." *Applied Soft Computing* 13.2 (2013): 947–958.

Khemchandani, Reshma, Suresh Chandra, et al. "Knowledge based proximal support vector machines." *European Journal of Operational Research* 195.3 (2009): 914–923.

Kim, Kyoung-jae. "Financial time series forecasting using support vector machines." *Neurocomputing* 55.1 (2003): 307–319.

Kumar, Abhishek and Hal Daumé. "A co-training approach for multi-view spectral clustering." *Proceedings of the 28th International Conference on Machine Learning (ICML-11).* 2011. 393–400.

Lang, Ken. "Newsweeder: Learning to filter netnews." *In Proceedings of the Twelfth International Conference on Machine Learning.* Citeseer, 1995.

Lee, Yi-Shian and Lee-Ing Tong. “Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming.” *Knowledge-Based Systems* 24.1 (2011): 66–72.

Liang, Huizhi, et al. “A hybrid recommender systems based on weighted tags.” (2010).

Lipczak, Marek. “Tag recommendation for folksonomies oriented towards individual users.” *ECML PKDD discovery challenge* 84 (2008).

Liu, Bing. *Web data mining.* Springer, 2007.

Liu, Bing, et al. “Building text classifiers using positive and unlabelled examples.” *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on.* IEEE, 2003. 179–186.

Liu, Bing, et al. “Partially supervised classification of text documents.” *ICML.* Citeseer, 2002. 387–394.

Liu, Fajiang and Jun Wang. “Fluctuation prediction of stock market index by Legendre neural network with random time strength function.” *Neurocomputing* 83 (2012): 12–21.

Malkiel, Burton G and Eugene F Fama. “Efficient capital markets: A review of theory and empirical work*.” *The journal of Finance* 25.2 (1970): 383–417.

Mantri, Jibendu Kumar, P Gahan, and BB Nayak. “Artificial neural networks-an application to stock market volatility.” *International Journal of Engineering Science and Technology* 2.5 (2010): 1451–1460.

Marchetti, Andrea, et al. “Semkey: A semantic collaborative tagging system.” *Workshop on Tagging and Metadata for Social Information Organization at WWW.* 2007. 8–12.

Markov, Zdravko and Daniel T Larose. *Data mining the Web: uncovering patterns in Web content, structure, and usage.* John Wiley & Sons, 2007.

Marlin, Benjamin M. “Modeling User Rating Profiles For Collaborative Filtering.” *NIPS.* 2003.

McCallum, Andrew, Kamal Nigam, et al. “A comparison of event models for naive bayes text classification.” *AAAI-98 workshop on learning for text categorization.* Citeseer, 1998. 41–48.

Mehrotra, Kishan, Chilukuri K Mohan, and Sanjay Ranka. *Elements of artificial neural networks.* MIT press, 1997.

Melville, Prem, Raymond J Mooney, and Ramadass Nagarajan. "Content-boosted collaborative filtering for improved recommendations." *AAAI/IAAI*. 2002. 187–192.

MIAO, Kai, Fang CHEN, and Zhi-gang ZHAO. "Stock Price Forecast Based on Bacterial Colony RBF Neural Network [J]." *Journal of Qingdao University (Natural Science Edition)* 2 (2007): 011.

Mihalcea, Rada. "Co-training and self-training for word sense disambiguation." *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2004)*. 2004.

Mishra, Ritesh Kumar, Sanjay Sehgal, and NR Bhanumurthy. "A search for long-range dependence and chaotic structure in Indian stock market." *Review of Financial Economics* 20.2 (2011): 96–104.

Montaner Rigall, Miquel, et al. *Collaborative recommender agents based on case-based reasoning and trust*. Universitat de Girona, 2003.

Mooney, Raymond J and Loriene Roy. "Content-based book recommending using learning for text categorization." *Proceedings of the fifth ACM conference on Digital libraries*. ACM, 2000. 195–204.

Nagar, Swapnil. "A hybrid recommender: user profiling from tags/keywords and ratings." Diss. Kansas State University, 2012.

Nair, Binoy B, et al. "A GA-artificial neural network hybrid system for financial time series forecasting." *Information Technology and Mobile Communication*. Springer, 2011. 499–506.

Nakamura, Atsuyoshi and Naoki Abe. "Collaborative Filtering Using Weighted Majority Prediction Algorithms." *ICML*. 1998. 395–403.

Nigam, Kamal and Rayid Ghani. "Analyzing the effectiveness and applicability of co-training." *Proceedings of the ninth international conference on Information and knowledge management*. ACM, 2000. 86–93.

Ou, Phichhang and Hengshan Wang. "Prediction of stock market index movement by ten data mining techniques." *Modern Applied Science* 3.12 (2009): P28.

Pai, Ping-Feng, et al. "Time series forecasting by a seasonal support vector regression model." *Expert Systems with Applications* 37.6 (2010): 4261–4265.

Parrella, Francesco. "Online support vector regression." *Master's Thesis, Department of Information Science, University of Genoa, Italy* (2007).

Pavlov, Dmitry and David M Pennock. "A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains." *NIPS*. 2002. 1441–1448.

Pazzani, Michael and Daniel Billsus. "Learning and revising user profiles: The identification of interesting web sites." *Machine learning* 27.3 (1997): 313–331.

Pazzani, Michael J. "A framework for collaborative, content-based and demographic filtering." *Artificial Intelligence Review* 13.5-6 (1999): 393–408.

Popescul, Alexandrin, David M Pennock, and Steve Lawrence. "Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments." *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2001. 437–444.

Quinlan, J. Ross. "Induction of decision trees." *Machine learning* 1.1 (1986): 81–106.

Quinlan, John Ross. *C4. 5: programs for machine learning*. Vol. 1. Morgan kaufmann, 1993.

Rao, Singiresu S and SS Rao. *Engineering optimization: theory and practice*. John Wiley & Sons, 2009.

Read, Jesse, et al. "Classifier chains for multi-label classification." *Machine learning* 85.3 (2011): 333–359.

Resnick, Paul, et al. "GroupLens: an open architecture for collaborative filtering of netnews." *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. ACM, 1994. 175–186.

Said, Alan, et al. "KMulE: a framework for user-based comparison of recommender algorithms." *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*. ACM, 2012. 323–324.

Salton, Gerard. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of*. Addison-Wesley, 1989.

Sarwar, Badrul, et al. "Item-based collaborative filtering recommendation algorithms." *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001. 285–295.

Schein, Andrew I, et al. “Methods and metrics for cold-start recommendations.” *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 2002. 253–260.

Shardanand, Upendra and Pattie Maes. “Social information filtering: algorithms for automating word of mouth.” *Proceedings of the SIGCHI conference on Human factors in computing systems.* ACM Press/Addison-Wesley Publishing Co., 1995. 210–217.

Sharma, Prashant. “Core characteristics of web 2.0 services.” *Tech Pluto* (2008).

Shen, Wei, et al. “Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm.” *Knowledge-Based Systems* 24.3 (2011): 378–385.

Soboroff, Ian and Charles Nicholas. “Combining content and collaboration in text filtering.” *Proceedings of the IJCAI.* 1999. 86–91.

Sun, Jie and Hui Li. “Financial distress prediction using support vector machines: Ensemble vs. individual.” *Applied Soft Computing* 12.8 (2012): 2254–2265.

Suykens, Johan AK and Joos Vandewalle. “Least squares support vector machine classifiers.” *Neural processing letters* 9.3 (1999): 293–300.

Tatu, Marta, Munirathnam Srikanth, and Thomas DSilva. “Rsdc08: Tag recommendations using bookmark content.” *ECML PKDD discovery challenge* 2008 (2008): 96–107.

Tran, Thomas and Robin Cohen. “Hybrid recommender systems for electronic commerce.” *Proc. Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04, AAAI Press.* 2000.

Tsai, Chih-Fong, et al. “Predicting stock returns by classifier ensembles.” *Applied Soft Computing* 11.2 (2011): 2452–2459.

Tso-Sutter, Karen HL, Leandro Balby Marinho, and Lars Schmidt-Thieme. “Tag-aware recommender systems by fusion of collaborative filtering algorithms.” *Proceedings of the 2008 ACM symposium on Applied computing.* ACM, 2008. 1995–1999.

Tsoumakas, Grigorios, et al. “Mulan: A java library for multi-label learning.” *The Journal of Machine Learning Research* 12 (2011): 2411–2414.

Ungar, Lyle H and Dean P Foster. "Clustering methods for collaborative filtering." *AAAI Workshop on Recommendation Systems.* 1998.

Van Gestel, Tony, et al. "Benchmarking least squares support vector machine classifiers." *Machine Learning* 54.1 (2004): 5–32.

Vapnik, Vladimir N. "An overview of statistical learning theory." *Neural Networks, IEEE Transactions on* 10.5 (1999): 988–999.

Wan, Xiaojun. "Co-training for cross-lingual sentiment classification." *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1.* Association for Computational Linguistics, 2009. 235–243.

Wang, Jung-Hua and Jia-Yann Leu. "Stock market trend prediction using ARIMA-based neural networks." *Neural Networks, 1996., IEEE International Conference on.* IEEE, 1996. 2160–2165.

Wang, Wei and Zhi-Hua Zhou. "Analyzing co-training style algorithms." *Machine Learning: ECML 2007.* Springer, 2007. 454–465.

Wetzker, Robert, Winfried Umbrath, and Alan Said. "A hybrid approach to item recommendation in folksonomies." *Proceedings of the WSDM'09 Workshop on Exploiting Semantic Annotations in Information Retrieval.* ACM, 2009. 25–29.

Whitman, Brian and Steve Lawrence. "Inferring descriptions and similarity for music from community metadata." *Proceedings of the 2002 International Computer Music Conference.* Citeseer, 2002. 591–598.

Xu, Xiujuan, Chunguang Zhou, and Zhe Wang. "Credit scoring algorithm based on link analysis ranking with support vector machine." *Expert Systems with Applications* 36.2 (2009): 2625–2632.

Ye, Jieping and Tao Xiong. "SVM versus least squares SVM." *International Conference on Artificial Intelligence and Statistics.* 2007. 644–651.

Zarandi, MH, Esmaeil Hadavandi, and IB Turksen. "A hybrid fuzzy intelligent agent-based system for stock price prediction." *International Journal of Intelligent Systems* 27.11 (2012): 947–969.

Zhang, Min-Ling, Jose M Pena, and Victor Robles. "Feature selection for multi-label naive Bayes classification." *Information Sciences* 179.19 (2009): 3218–3229.

Zhang, Yudong and Lenan Wu. "Stock market prediction of S&P 500 via combination of improved BCO approach and BP neural network." *Expert systems with applications* 36.5 (2009): 8849–8854.

Index

Accuracy, 28

Adaptive Gradient Descent, 43

ANN Model, 21

ANN Parameters, 23

Co-Training Algorithm, 118

Collaborative Recommenders, 8

Content-based Recommenders, 6

Continuous Representation, 19

Direct Method, 75

F-measure, 28, 79

Gradient Descent with Momentum, 22

Hybrid Recommenders, 9

Information Filtering, 5

Information Retrieval, 5

Item-Based Collaborative Filtering, 106

Learning from Positive and Unlabeled Examples, 70

Linear LS-SVM, 74

Movie's Rating Profile Generation, 109

Movie's Tag Profile, 108

Multi-label Classification, 90

Multinomial Distribution, 97

Multivariate Bernoulli Distribution, 98

Naive Bayes Classifier, 26

Neighborhood Formation, 109

Normal Distribution, 98

Precision, 79

Prediction Models, 21

Publications related to Thesis, iii

Random Forest, 25

Rating Prediction Formula, 110

Recall, 79

Recommender Systems, 5

Relevance of a Tag to a Movie, 107

Relevance of a Tag to a User, 107

Reliable Negative Resources, 72

Single Stage Approach, 42

Social Bookmarking System, 67

Support Vector Regression, 44

SVM Model, 23

Tag Recommendation Algorithm, 97

Tag Relatedness Metric for the Movie, 108

Technical Indicators, 20

Theoretical Foundations of PU Learning, 71

Trend Deterministic Data, 19

Two Stage Fusion Approach, 46

Usefulness of Unlabeled Data, 70