Predicting User Behavior over the Web

Amal Adnan Al Safadi

Master of Science in Information Technology
Faculty of Informatics

British University in Dubai

2010

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

(*Amal Adnan Al Safadi*)

To my parents soul, to my husband and my daughters Noor and Nada

Acknowledgements

I would like to express my gratitude to all those who made the completion of this thesis possible.

I am deeply grateful to my supervisor, Dr. Sherief Abdullah for his guideness. His understanding, encouraging and support have provided a real help in passing all the difficulties and obstacals during the thesis research and writing.

I would like to thank Mr. Krzysztof Dembczynski, the winner of ECML/PKDD 2007 Discovery Challenge for his kind response to my questions and providing me with his paper copy.

I want to express my gratitude to Mr. Malik Tahir Hassan, the runner-up of ECML/PKDD 2007 Discovery Challange for providing me with the labelled test set and for his coorporation and answering all my questions.

I would like to express my deep and sincere gratitude to my sisters, brothers and friends for their continueous support and praying.

I owe my loving thanks to my husband, Qusai Al Ja'ouni. Without his encouragement and understanding it would have been impossible for me to finish this work.

Abstract

Web Usage Mining is the application of discovering useful patterns from web data using statistical and data mining techniques. It has recently a wide range of applications in E-commerce web site and E-services such as building interactive marketing strategies, Web recommendation and Web personalization. Due to its importance, the ECML/PKDD conference announced a competition (challenge) where researchers analyze a web-usage data set and attempt to make predictions about user behavior.

The purpose of this thesis is to analyze the first problem of ECML/PKDD 2007 challenge and apply web usage mining techniques in order to predict the user navigation behavior, such as the user visit duration and type of visited pages, based on user real historical data. Toward this goal, I applied web usage mining, data preprocessing, and visualization techniques. I also applied different classification algorithms and studied the effect of attribute selection on each classifier performance. The results I report are comparable to the challenge winner and outperform the runner-up on two out of the three challenge problems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The term Web mining was defined by Etzioni (1996) to denote the use of data mining techniques to automatically discover Web documents and services, extract information from Web resources, and uncover general patterns on the Web. Web mining is an interdisciplinary research area that spans over machine learning, databases, data mining, information retrieval and Web retrieval. Typically, web mining is divided into three types: web content mining, web structure mining and web usage mining.

This dissertation focuses on Web Usage Mining (WUM), which is about using data mining techniques to analyze the traces of web users in order to find interesting patterns. WUM plays an important role in E-commerce web site and E-services. WUM is a very helpful tool to gather the important information from the site visitors. The analysis of this information allows the company to improve its productivity by directing its efforts towards the most effective ways of their promotional strategies and internet traffic.

WUM has a wide range of applications such as building interactive marketing strategies, Web recommendation and Web personalization. In fact, WUM was the topic of *KDDCup 2000* competition and one of the problems proposed in *ECML/PKDD* 2007 Discovery Challenge. This challenge is a good representation of WUM since the dataset is acquired from a real company, not synthesized one.

1.2 Problem Statement

The objective of this study is to solve the first problem of *ECML/PKDD* Discovery Challenge 2007 which was concerned with predicting user behavior by characterizing the nature of a user's visit. The challenge provided a real web traffic data that was collected by *Gemius* (an internet market research company) using cookies technology and special scripts placed on the monitored web-pages (Gemius, 2007). The accumulated information consisted of the user id and the timestamp along with category-view path. User's details such as social and demographic data were collected too. The data were collected after one month of monitoring polish web sites and divided into two data sets, the training data consisted of the first three weeks of the month while the test data consisted of the last week of the month (ECML/PKDD, 2007).

The first discovery challenge has three sub-problems as follows:

- Problem 1: The length of the visit
  A visit is a sequence of page views by one user where each page belongs to a specific category. During one visit, a user may view pages of one or more categories. The goal of problem 1 is to predict whether a user will have a short or a long visit according to the following definition:

  Short visit: is a visit with page views of only one category.

  Long visit: is a visit with views of pages belonging to at least two categories.

  The input for this task is the user id and the timestamp and the output should be the correct predicted class.

- Problem 2: The most probable categories
  Solution of Problem 2 is a list of the most three probable categories in a given visit of a given user.

- Problem 3: The most probable categories and ranges of numbers of page views
  Solution of problem 3 is a list of the most three probable categories in a given visit for a given user with range of number of page views in each category.

1.3 Contributions

This dissertation aims to explore and discover WUM through solving the research problems. This dissertation presents the following contributions:

- Demonstration of how to visualize WUM data and extract useful information from it .
- In-depth analysis of preprocessing techniques on the challenge data and identifying key preprocessing steps (not published by any of the competitors) that significantly improves the performance of all classifiers.
- Comparative study of large number of classifiers, including the ones used by winner and runner up. In addition, trying a new technique (DTNB) that none of the competitors could have used it since it was published in 2008, while the challenge was in 2007.

- Studying the effect of attribute selection on the classifiers performance. In most tasks I outperformed the runner up for the contest and was slightly worse than the winner.

1.4 Organization of the thesis

The rest of this thesis is organized as follows. In Chapter 2, I introduce a short background of Web Mining and the main three phases of WUM process. Then I give a brief description of the ECML/PKDD Discovery Challenge. I outline *ECML/PKDD* discovery challenge 2007 problems, describe the data, and explore the contest results. In addition, the literature review section explores the work of others which is related to web usage mining field, mainly predicting web usage.

Chapter 3 presents the data analysis and visualization and describes the preprocessing strategy I have employed. The implementation part is introduced in chapter 4 where I describe the details of applying the technical part of the selected methodologies for each task. Moreover I present and analyze the results of my approach and compare it with the contest results. Finally, the conclusion and future work is given in Chapter 5.

Chapter 2

Background

This chapter provides a basic knowledge about the Web Usage Mining. It gives a brief background about the main three phases of WUM process. Then It describes the ECML challenge and introduces the research problem in details. Finally It surveys the previous/related work in WUM.

2.1 Web Mining

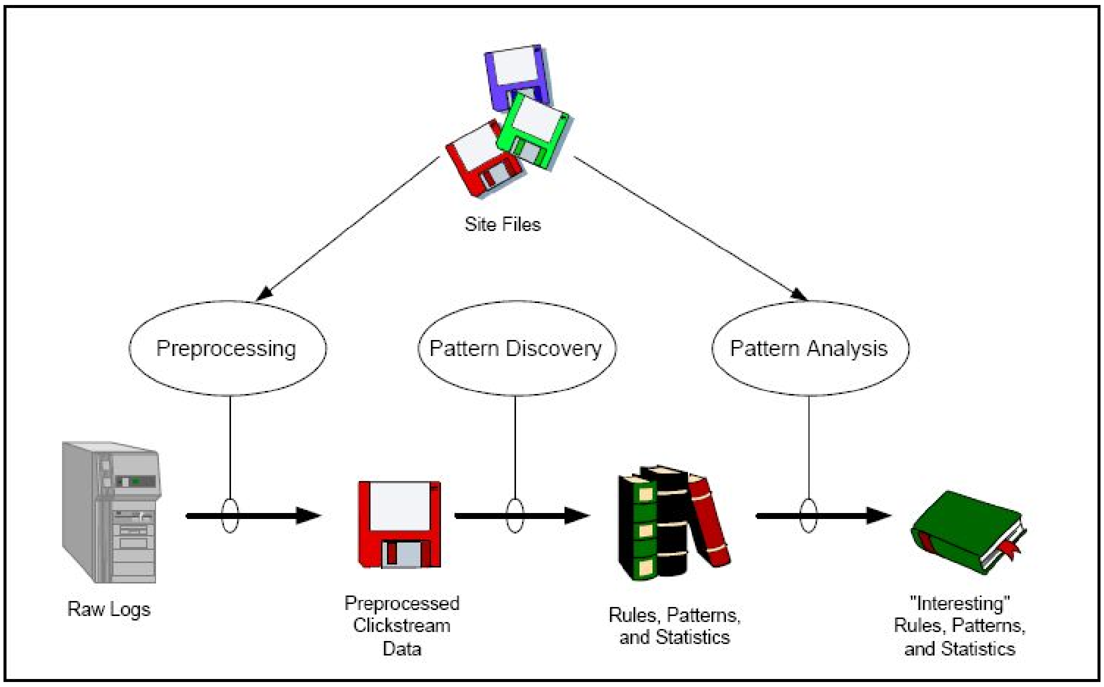
Web Mining is defined as the application of data mining techniques to discover patterns from the Web. Web mining can be divided into three different types, depending on the nature of the data, which are Web content mining, Web structure mining and Web usage mining (Srivastava et al., 2000).

- Web Content Mining: Extracting useful information from page content (text, images, other ... etc). It is used by search engines, agents, recommendation engines to help users find what they are looking for. The technologies that are normally used in web content mining are Natural language processing and Information retrieval.
- Web Structure Mining: using the hyperlink structure of the Web as an additional information source. Graph theory is normally used to analyze the node and connection structure of a web site.
- Web Usage Mining: applying statistical and data mining techniques to the preprocessed Web log data, in order to discover useful patterns.

The task we are addressing in this dissertation falls under the Web Usage Mining type.

2.1.1 Web Usage Mining

Designers and owners of the websites often have a great interest in understanding the patterns of how people navigate their sites. This knowledge could help designers to improve their sites structure, provide better service, increase sales, attract new customers, and retain current customers by personalizing their user profiles. The Web usage mining process can be regarded as a three-phase process, consisting of the data preparation, pattern discovery, and pattern analysis phases (Cooley et al., 2000).



Site Files
Preprocessing
Pattern Discovery
Pattern Analysis
Raw Logs
Preprocessed Clickstream Data
Rules, Patterns, and Statistics
"Interesting" Rules, Patterns, and Statistics

Figure 2.1: Web Usage Mining Process, (Srivastava et al., 2000)

2.1.2 Data Preparation

The data collected to this phase comes from three different locations: the server side, the client side and the proxy side. The first step is to clean the raw web log files from entries involving pages that returned an error or graphics file accesses and insert the processed data into a relational database or a data warehouse (Ferreira, n.d.).

Another important step is the user identification. There are several ways to identify individual visitors. The easiest way is assuming that each IP address identifies a single visitor. Nevertheless, that may be not very accurate because a visitor may access the Web from different computers, or if a proxy is used, many users may use the same IP address. A further assumption can then be made, that consecutive accesses from the same host during a certain time interval come from the same user (Eirinaki and Vazirgiannis, 2003). The best way to identify unique visitors is using cookies or similar mechanisms or the requirement for user registration. However, the reluctance of users to share personal information is a potential problem in using

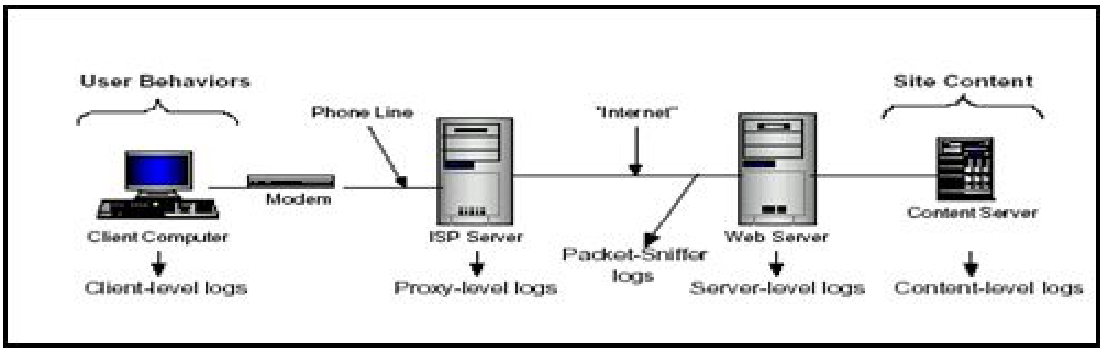


Figure 2.2: Potential Data Sources (Ye, 2003)

such methods.

The next step is performing session identification by dividing the clickstream of each user into sessions. A thirty minute timeout is often used as the default method of breaking a user's clickstream into sessions.

In this dissertation, the log data is already prepared and reprocessed and the users are identified by the cookies technology and the 30 minutes timeout is used to define the sessions. The focus of this dissertation is on analyzing and visualizing the given data to extract useful information to help in the second phase of WUM which is Pattern Discovery.

2.1.3 Pattern Discovery

In the second phase, statistical methods, as well as data mining methods (such as association rules, sequential pattern discovery, clustering, and classification) are applied in order to detect interesting patterns.

There are a wide variety of data mining algorithms, drawn from the fields of statistics, pattern recognition, machine learning, and databases. Here is brief information about some of them:

1. Statistical Analysis: it's the most common method to extract information about web site visitors. By analyzing the session file, many descriptive statistical analyses can be performed such as frequency, medium, mean, etc. this information can be potentially useful in deciding the next step.

2. Link Analysis: is a descriptive approach to explore data that can help identify relationships among values in a database. The two most common approaches to link analysis are Association discovery and Sequence discovery.

   Association rule mining: is a technique for finding frequent patterns, associations, and correlations among sets of items. It is used in order to reveal correlations between pages accessed together during a server session. Such rules indicate the possible relationship between pages that are often viewed together even if they are not directly connected, and can reveal associations between groups of users with specific interests (Eirinaki and Vazirgiannis, 2003).

Sequence discovery mining: is very similar, in that a sequence is an association related over time. Because of ordered nature of click-streams in web log data, this technique can have a very important role in knowledge discovery. We can have an example like this: "If user visits page X, and then page Y, it will visit page Z with c% of chance". The algorithms for sequence mining inherited much from the association mining algorithms (Ferreira, n.d.).

3. Clustering: is used to group together items that have similar characteristics. In Web mining, there are two types of clustering, user clusters and page clusters. Page clustering identifies groups of pages that seem to be conceptually related which is used in Internet Search Engines and Web Assistance Providers (Srivastava et al., 2000). User clustering tends to group users that seem to behave similarly when navigating through a Web site. Such knowledge is used in e-commerce in order to perform market segmentation but is also helpful when the objective is to personalize a web site (Eirinaki and Vazirgiannis, 2003).

4. Classification: is a process that maps a data item into one of several predetermined classes. This pattern can be used both to understand the existing data and to predict how new instances will behave. In the Web domain classes usually represent different user profiles and classification is performed using selected features that describe each user's category. The most common classification algorithms are decision trees, naive bayesian classifier, neural networks (Srivastava et al., 2000).

2.1.4 Pattern Analysis

Pattern Analysis is the final stage of the Web usage mining. The main task of this stage is to evaluate the patterns produced by the data mining algorithms and to eliminate the irrelevant rules or patterns and to extract the interesting rules or patterns from the output of the pattern discovery process.

The most common ways of analyzing such patterns are either by using a query mechanism like SQL on a database where the results are stored, or by loading the results into a data cube and then performing OLAP operations. Additionally, visualization techniques are used for an easier interpretation of the results (Eirinaki and Vazirgiannis, 2003). This stage will not be addressed in this dissertation.

2.2 ECML/PKDD Discovery Challenge

The aim of this dissertation is to solve the first problem of ECML/PKDD discovery challenge of year 2007. The 2007 Discovery Challenge was devoted to three problems: User behavior prediction from web traffic logs, HTTP traffic classification, and Sumerian literature understanding (ECML/PKDD, 2007).

The annual Discovery Challenge constitutes a collection of data and problems as a common ground for better comparisons and discussions of the applicability of KDD methods on a real-world problems with respect to both KDD and application viewpoints.

2.3 Problem Definition

The first problem of 2007 Discovery Challenge is divided into three separate tasks. The first task is predicting the length of the user's visit. The result should be either short for one visited category or long for more than one visited category. Constructing such a model will help web designers to save resources by focusing on users that spend longer time. For example the advertisement or offers can be sent for those specific users instead of sending it randomly for all users. The second task is predicting the first 3 visited categories, which indicates the user's interest. Finally the third task is predicting the number of pages seen in each of the 3 categories in task 2.

2.4 Data

The data (provided by 2007 ECML/PKDD Discovery Challenge) were collected during one month of monitoring Polish web sites. The data had information about 4882 users that generated 545,784 sessions. These sessions had been separated into two datasets; the training data file contains data gathered during the first three weeks of the month (379,485 session paths), while the test data consists of the last week of the month (166,299 session paths). All users are shown in the test data are presented also in the training data.

2.4.1 General Notation

These general notations of the problem have been given by the Discovery Challenge 2007 and have been used in the dissertation (ECML/PKDD, 2007):

- "Page: any web page participating in the research. Pages are distinguishable only by the category they belong to."
- "Page View: the event of displaying the monitored web page."
- "Visit: an uninterrupted series of Page Views on a given web site executed by the same visitor (cookie), counted as a closed whole. This represents an Internet user's total stay on the web site in question for any individual visit. It is assumed that one Page View cannot exceed 30 minutes (a longer Page View duration / gap will result in the series being counted as two separate Visits)."

- "Visit Path: series of web pages visited during one Visit. This represents the clickstream that the user followed in navigating a web site."
- "Category: each page is qualified as a member of a relevant category that is a group of web sites of a similar leading theme., e.g. entertainment, technology, news, communication, education, e-commerce, business, etc. These categories have been assigned certain identifiers."

2.4.2 Data Description

The provided data gives information about users as well as visit paths.

i. Users Table: This table gives demographic information (country, region and city) and system characteristics information (operating system, browsers and versions) where each user has been given a unique id. An example record in Users table is as follows:

UserId	CountryId	RegionId	CityId	SystemId	SysSubId	BrowserId	BrowVerId
10	42	11	44	3	9	1	517

ii. Visit Paths table: This table consists of records of individual visit sessions described by the fields: path-id, user-id, timestamp, path category-id, pageviews-number. An example record in this table is as follows:

pathId	UserId	timestamp	Path(category-id, pageviews-number)
27	1	1169814548	(7,1) (16,2) (17,9) (16,1)

Each session has a unique number, path-id, generated by a single user, user-id, identified based on the user's cookie. The timestamp attribute, given in UNIX timestamp format, is the number of seconds that have elapsed since January 1, 1970 (midnight UTC/GMT), not counting leap seconds. The timestamp records the time of the start of the visit while the path field shows the sequence of visited categories by that user and number of page views during one visit in one category not interrupted by a page view from a different category.

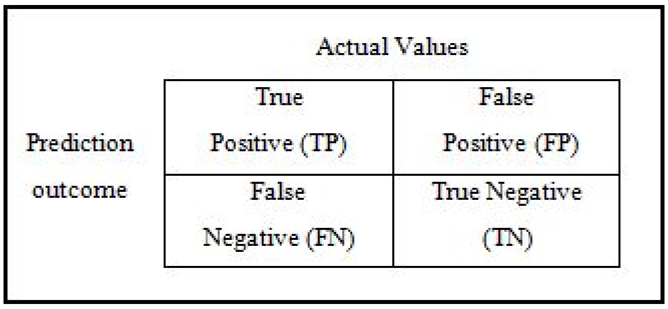
2.5 Evaluation Criteria

To evaluate the three tasks initially I use the accuracy, which defined as:

$$prediction\ accuracy = \frac{the\ number\ of\ correctly\ classified\ instances}{number\ of\ total\ instances} \times 100\%$$

But since the data set is unbalanced (the number of short visits class is greatly more than the long visits class), the accuracy and the error rate of a classifier is not representative of the true performance of the classifier since the classifier can easily be biased towards the major class. That's why we use the Confusion Matrix. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class.

For a binary classification, where the outcomes are labeled either as positive (p) or negative (n) class, there are four possible outcomes from a binary classifier. If the outcome from a prediction is p and the actual value is also p, then it is called a true positive (TP); however if the actual value is n then it is said to be a false positive (FP). Conversely, a true negative has occurred when both the prediction outcome and the actual value are n, and false negative is when the prediction outcome is n while the actual value is p.



	Actual Values	
Prediction outcome	True Positive (TP)	False Positive (FP)
	False Negative (FN)	True Negative (TN)

Figure 2.3: Confusion Matrix

Many statistical measures can be done using the confusion matrix, such as:

ROC Curve: a Receiver Operating Characteristic, which represents the relative trade-offs between true positive (benefits) and false positive (costs). It can be represented by plotting the fraction of true positives (TPR) vs. the fraction of false positives (FPR).

The diagonal line divides the ROC space in areas of good or bad classification. Points above the diagonal line indicate good classification results, while points below the line indicate wrong results.

Precision: for a class is the number of true positives divided by the total number of elements labeled as belonging to the class.

$$Precision = \frac{t_p}{t_p + f_p}$$

Recall: is defined as the number of true positives divided by the total number of elements that actually belong to the class.

$$Recall = \frac{t_p}{t_p + f_n}$$

A Precision score of 1.0 for a class C means that every item labeled as belonging to class C does indeed belong to class C (but says nothing about the number of items from class

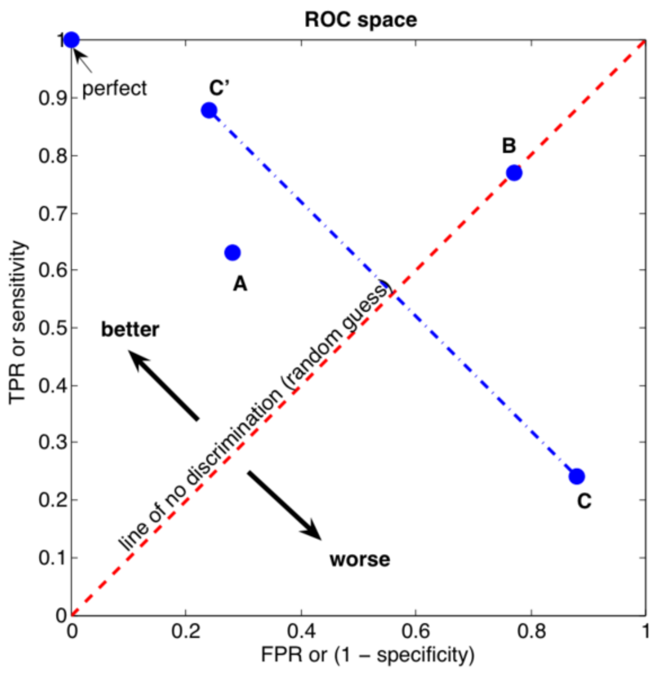


Figure 2.4: ROC Space (Wikipedia, 2007).

C that were not labeled correctly) whereas a Recall of 1.0 means that every item from class C was labeled as belonging to class C (but says nothing about how many other items were incorrectly also labeled as belonging to class C). Precision vs. recall can be plotted in a manner similar to ROC curves.

F-measure: can be used as a single measure of performance of the test. The F-measure is the harmonic mean of precision and recall:

$$F = 2 \times (Precision \times Recall)/(Precision + Recall)$$

F-measure tends to be closer to the smaller one between the precision and recall estimates. F-measure is high if both precision and recall is high.

2.5.1 Score Function

The contest organizers proposed a specific score function to measure the prediction performance of the second and third tasks. For the second task, the estimated vector will be compared with the actual vector of category identifiers according to the formula:

$$CategoryId_{predicted}(i) = CategoryId_{actual}(j)$$

Then two score vectors are created, one for the actual vector and one for the predicted categories vector, accordingly to the following rule: 5 points for the first category, 4 points for second category, 3 for third, etc. (from fifth category on one point is given). Afterwards, minimums of corresponding elements of both vectors are determined. These numbers are summed up and give the final result score.

Since many of the actual vectors and predicted vectors has repeated categories, we assume that when a category appears more than once in the sequence, each time it is regarded as new, different category and replace the repeated category with a new category ID and compute the score accordingly.

For the third task, the values of given categories are checked as described above in the second task and the ranges of numbers of page views are checked according to the condition:

$$number\ \ of\ \ pageviews_{actual}(i) \in range\ \ of\ \ pageviews(i)$$

Similarly as for the second task, the score of actual and predicted vectors are calculated according to the same rule above and at the last step; the award vector is determined by taking minimums of corresponding elements of both vectors and adding 1 point for giving a correct range or 0 points for wrong range of page views. These numbers are summed up and give the final result score.

2.6 Challenge Results

The winner of KDD 2007 first challenge used at the beginning Simple Global Model which related to visit's granule. J48 classifier has been used for the first task by using many selected features such as (user data, week day, hour, etc..). Due to the unsatisfied results of the simple global model, they developed Enhanced Global Model which depends on estimating additional values describing the behavior of the users by using estimation set isolated from the training set. These values are category-based (210 attributes in total) and visit-length based (only two type of visits, short or long). User data such as (country, region, city... etc) and time-based (hour, week-day) were included in this approach which gave better results. Finally, User Models approach was used for the three tasks. This approach used user id number directly without extracting any additional information about the user. the User Models had three versions: the first one was the Simple Classification where the classification process had done by doing a simple call to the look-up table containing probability estimates.

In the second model, Trend Prediction, data for each user was regarded as a short time series with values 0 for short visit or 1 for long visit. Then a polynomial trend was fitted to the time series for each user and was used as a probability estimator. The last and the most sophisticated model was the Auto-regression Model. for each user a separate linear model is fitted to the user's time series, based on the following attributes: normalized timestamp, time from the last visit, average length of the last 2,4 and 8 visits. Almost all the user models have very similar results which was better from the global models results. This motivated the authors to choose the simplest model which was the Simple Classification Model.

Depending on the analysis of the first task, the winner did not perform tests on the global models and choose to use the User Model for the second and third task taking into account only one predictor - user id number.(Dembczynski et al., 2009).

The Runner-Up of KDD 2007 presented a Bayesian Classifier for the three tasks. They model the sequence of page categories visited as a Marcov chain. Naive Bayes classifier was used for the first task by assuming that User Id and the timestamp are conditionally independent. This approach gave the same result which has been got by applying SVM with more efficiency since it took only less than one minute to get the same results of several hours of SVM to learn. For the second task the sequence of page categories visited are modeled as a Marcov chain where the start state is determined by Bayes classifier where the subsequent states are determined by combining the posterior probability estimates given by the Marcov chain with that of the Bayes classifier for that particular position. Using the predicted page categories in Task2, Bayes classifier was used to predict the range of page views made at the first three positions in each visit session (Hassan et al., 2007).

2.7 Related Work

A variety of data mining techniques and algorithms have been used for the purpose of web data analysis.

While this dissertation focus on predicting the expected behaviors of the user, such as his visit period or his preferred subjects, (Ting et al., 2005) presented a new approach called UBB Mining which was based on discovering the unexpected browsing behaviors of the user according to predefined patterns specified by the site designer. The result allowed the designers to review, improve or redesign their sites and could be a good step to model a user browsing behavior. The core of the approach was defining the expected route in the form of continuous common subsequences which was specified by the site owner or designer which is not practical for large sites with different designers.

(Velayathan and Yamada, 2007) presented a method to automatically detect and log user behavior at the client-side by creating a client-side browser. It focused on other factors other than time that evaluate user interest such as scroll action, form action, searching text, copying text etc. In this dissertation, the user identity, timestamp and the categories of visited pages are the main factors that determine user interest.

(Sun et al., 2002) proposed the use of Support Vector Machine (SVM) classifiers to classify web pages using both their text and context feature sets while Dumais and Chen (2000) used SVM classifiers to build a hierarchical structure model for classifying a large, heterogeneous collection of web content to support classification of search results while in this dissertation I use SVM to predict the visit length depending on the user id and timestamp.

(Manavoglu et al., 2003) presented a mixture model based approach for learning individualized behavior models for the web users. To overcome the insufficient data problem they used global model first to capture patterns of general behavior of the users, and once it learned, it was personalized by optimizing the weights of each component for each known user individually. They used Marcov Model and Maximum Entropy Model. In this dissertation, according to the type of data provided, working on the user-level gives better result.

While I use Association Rules as an exploratory tool to discover any significant information helps in choosing the best classifier, (Shyu et al., 2005) applied Association rule mining technique to approximate and construct the predictive model. They used the shortest path algorithm in graph theory to prune the results from the association rule mining to reduce the state-space complexity of the model.

(Khalil et al., 2006) proposed a combination of Markov model and association rules for predicting Web page accesses. The integration is based on a low order Markov model to predict multiple pages to be visited by a user. To resolve ambiguous predictions, sets of subsequence association rules are used to complement the Markov model by using long history data. The integration avoids the complexity of high order Markov model and the limitation of Markov model using short history. This model also reduces the large number of association rules since association rules are only used when ambiguous predictions occur.

The same author presented another integrating of Marcov Model with Clustering to predict web page accesses. The Web pages in the user sessions are first allocated into categories according to Web services that are functionally meaningful. Then, k-means clustering algorithm is implemented using the most appropriate number of clusters and distance measure. Prediction techniques are applied using each cluster as well as using the whole data set (Khalil et al., 2007). In this dissertation I examine a new classifier DTNB, which is introduced by Frank Hall1 (2008) as a hybrid classifier combining Naive Bayes and Decision Tables techniques.

On the other hand, (Gunduz and Ozsu, 2003) proposed a model that considers both the order information of pages in a session and the time spent on them. They clustered user sessions based on their similarity and represented the resulting clusters by a click-stream tree. The new user sessions is then assigned to a cluster based on a similarity measure. The click-stream tree of that cluster is used to generate the recommendation set. In this dissertation I used visit type, long or short, to help in predicting the most desired categories for each user.

Chapter 3

Data Analysis and Reprocessing

In this chapter, the exploratory data analysis and data pre-processing, have been applied on the data.

3.1 Exploratory Data Analysis

The collected data is belonging to 4882 different users and 20 different web page categories. One of the notes about the data is that it has non-uniform data distribution. While some users have only few sessions others have multi sessions. For train data the maximum records for a user is 497 while the minimum is 7 and the average is 77.7 records. While in the test data the maximum records is 215 and the minimum is 1 and the average is 34.06.

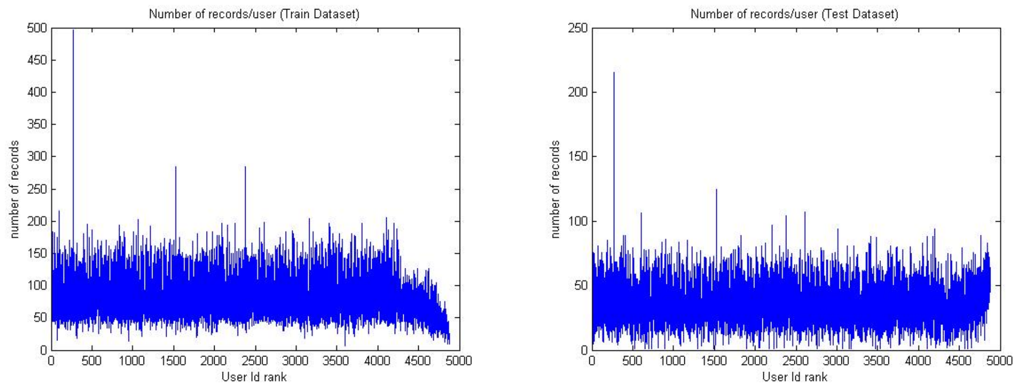


Figure 3.1: Number of records per user

Figure 3.1 shows the number of records per user in both training and testing sets while Figure 3.2 shows the distribution of the two classes in the train data. The blue color represents the short class frequency while the red color represents the long class frequency.

The timestamp attribute is transferred to a readable date/time format. The histograms in Figure 3.3 and Figure 3.4 show the distribution of the sessions of the train and test data on the days of the week and during the hours of the day. The left side of the figures shows the hourly visits

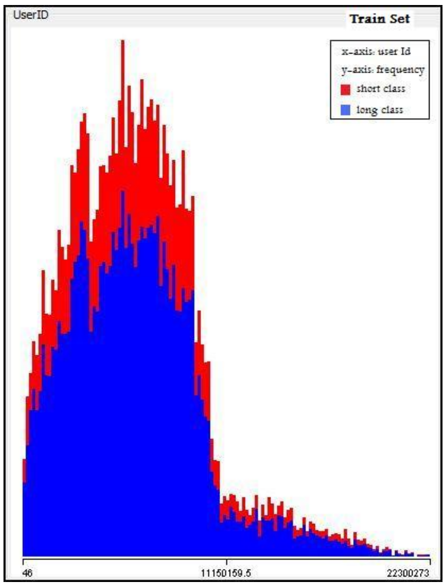


Figure 3.2: Class distribution for train dataset. The x-axis represents user id while the y-axis represents the frequency.

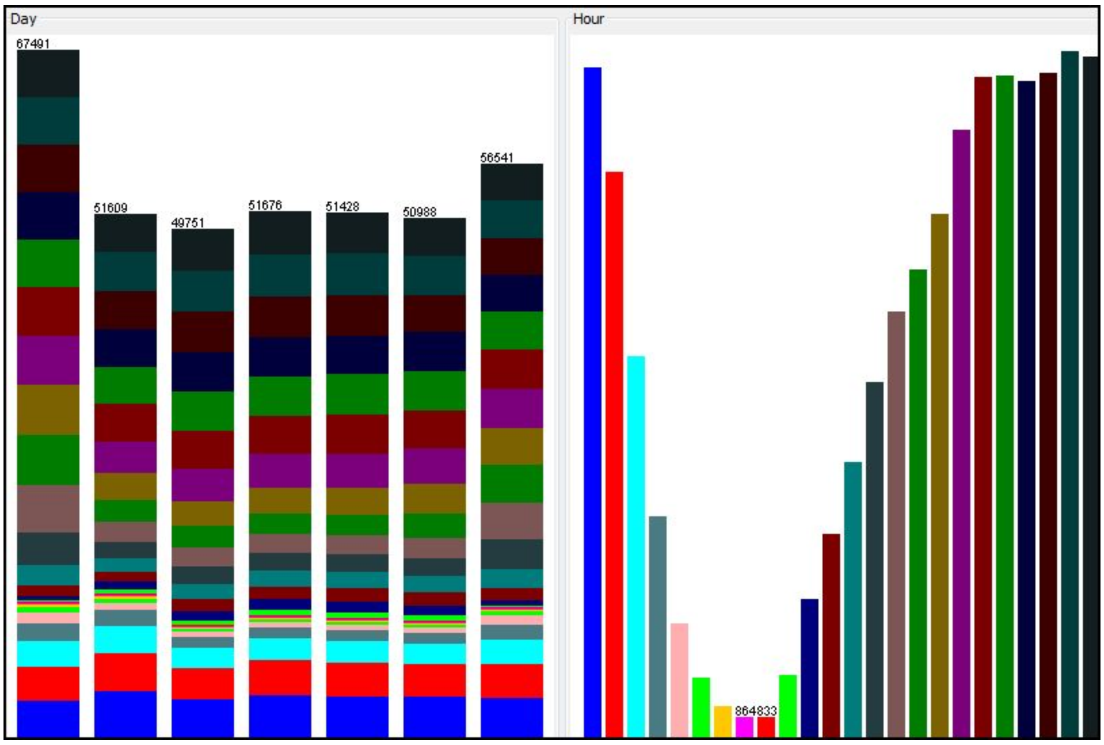


Figure 3.3: Sessions distribution in train data on days of week and during hours of day. The x-axis represents the day and hour attributes while the y-axis represents its frequency. each hour is distinguished by a different color.

frequency where each hour is distinguished by a different color. On the other hand, the right side of the figure shows the days of the week visits frequency where the different colors in each column represent the hourly frequency visits in that day.

For the training data, more than 33% of the visits were on the weekend days (Sun, Sat) while

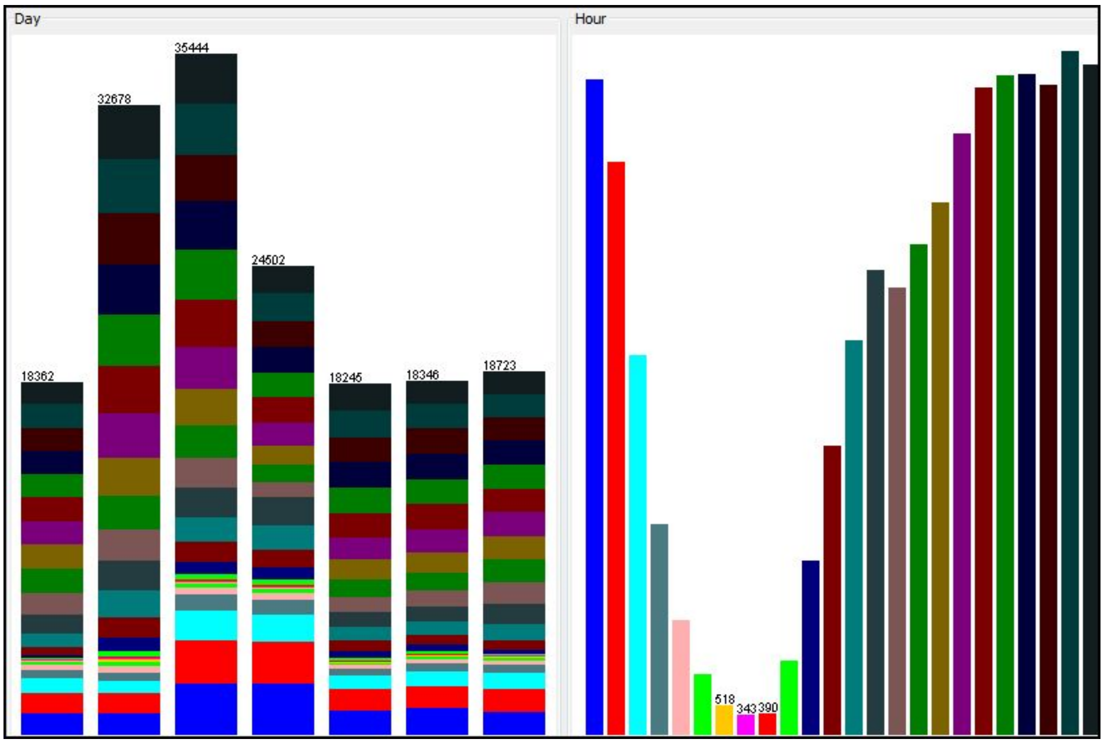
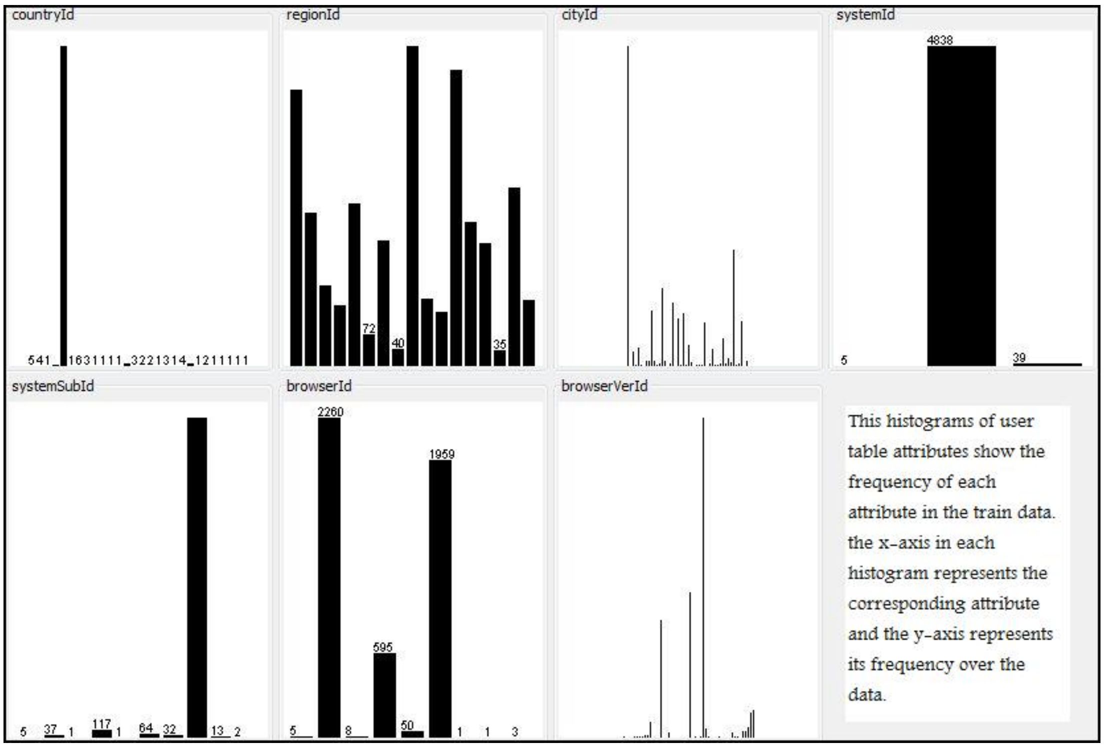


Figure 3.4: Sessions distribution in test data on days of week and during hours of day. The x-axis represents the day and hour attributes while the y-axis represents its frequency. each hour is distinguished by a different color.



This histograms of user table attributes show the frequency of each attribute in the train data. the x-axis in each histogram represents the corresponding attribute and the y-axis represents its frequency over the data.

Figure 3.5: Histogram of user table attributes. The x-axis in each histogram represents the corresponding attribute while the y-axis shows its frequency over the train data

the distribution was almost similar for other week days. Most visits were on the night hours while the minimum visits were between 7:00 to 8:00 morning and it starts rising gradually during the day. This is somehow expectable since it is the time people start their work hours.

However on the test data (Figure 3.4), the maximum activity was on the middle days of the week while the hourly activity is similar to the training set.

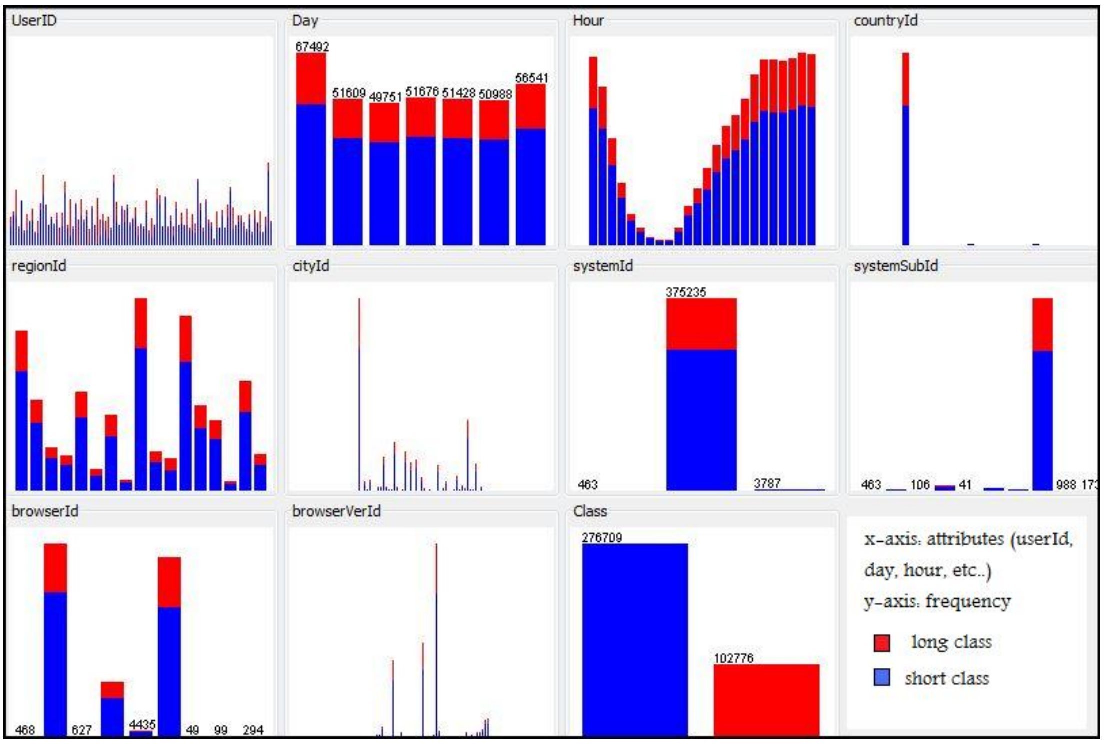


Figure 3.6: The class distribution on user's attributes in train dataset. The x-axis in each histogram represents the corresponding attribute while the y-axis shows its frequency over the train data

Figure 3.5 visualizes the user table attributes. More than 97% of the users are belonging to "Country Id = 42" while the region Id and city Id vary from user to user. Similarly more than 99% of the users are belonging to "system Id = 1". 46% of the users use "Browser Id = 1" while 40% use "Browser Id = 21". These results help out in deciding which attributes to select as input attributes for the selected classifiers since some of it like 'country Id' and 'system Id' do not distinguish the users effectively and doesn't give any extra information about the user characteristics.

The histograms in Figure 3.6 show the distribution of the two classes in the train data. The blue column in the right-bottom histogram represents the short class frequency (276,709 records, 72.917%) while the red column represents the long class frequency (102,776 records, 27.28%). The other histograms show the distribution of the two classes for each attribute (user, day, hour, countryId, regionId, cityId, systemId, systemSubId, browserId, browserVerId).

The x-axis of the histograms represents each of the mentions above attributes respectively and the y-axis is the frequency of each one. Each attribute is represented by two colors (blue, red) which demonstrates the classes' distribution for each. The figure shows evidently that the classes' distribution over each of the attributes is proportional with the classes' distribution on the sessions itself.

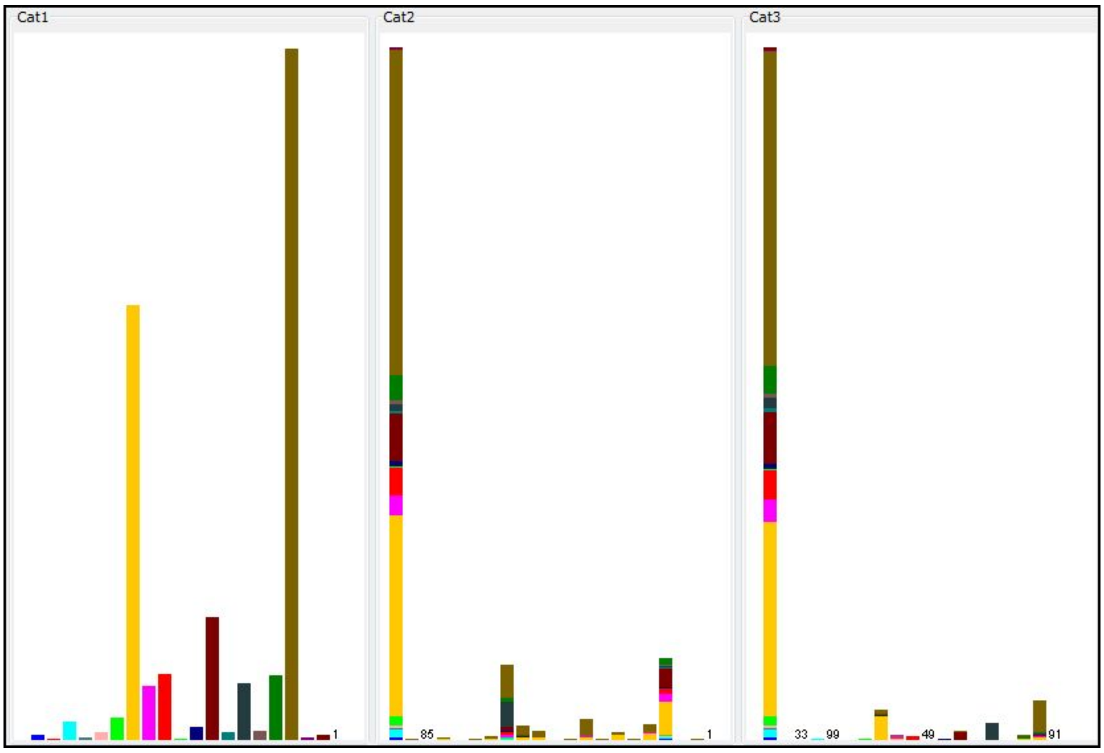


Figure 3.7: Distribution of first three categories visited by the users in the train data. The x-axis represents the attributes (cat1, cat2, cat3) respectively and y-axis represents the frequency of each one.

The histogram in Figure 3.7 shows the distribution of the 20 categories on the first three visited positions (cat1, cat2, cat3) in the train data. The x-axis represents the attributes (cat1, cat2, cat3) respectively and y-axis represents the frequency of each one.

The first visited category is shown in the left histogram where each category Id is represented by a different color. The middle and right histogram illustrate the distribution of the 20 categories on the second and third visited positions and in the same time the colors in each column illustrate the relation between the occurrence of the first visited category (cat1) on the second and third visited categories (cat2 and cat3).

Here is some information that can be extracted from Figure 3.7:

- The first column in the middle and right histogram represent (category Id = 0), which refers to no category visited (i.e. short visit). Most users have a short visit path. The users in 72.92% of the sessions visit only one category while in 27.08% of the sessions have long visits i.e. visit more than one category in a single session.
- During a single visit path, 15.56% of the sessions have two visited categories while only 9.25% of the visits paths have more than 3 visited categories.
- The distribution of the first three categories is homogeneous on the global level. We

note that globally users tend to visit specific categories and usually they re-visit same categories in the next sessions.

- The 1st and 3rd visited categories in the long sessions mostly tend to be the same.
- Category 17 was the most popular category in general. Around 44% of the total sessions started by visiting Category17. It was also the second and third visited category for nearly 32% of the total sessions. Category 7 was the second popular category where around 28% of the total sessions visited this category in the first three categories.
- The first column (category Id = 0) in the 2nd and 3rd position shows that mostly if category 17, category 7 and category 12 are visited in the 1st position, the user will not visit any more categories.

3.2 Data Reprocessing

Depending on the previous data analysis and visualization I have taken some decisions:

1. Since only 9% of the users visit more than 3 categories in each session, I keep only the first three categories in each visit path and remove the remaining of the visit path. One of the reasons to take this decision is the large size of the train data and the limitations WEKA have in memory heap size, beside that the tasks focus on the first 3 categories.
2. Both WEKA and MATLAB read only the data files that have equal columns sizes. So the empty cells of the short visits have to be filled with ”0” to indicate the absence of second and third category for task 2 and task 3.
3. While the timestamp has been transformed to the complete readable time/date; (day, second, minute, hour), only day and hour are selected as input attributes since (minute and second) attributes are implied already in (hour) attribute.

3.2.1 Attributes Selection

According to the data exploratory results, the input data attributes have been chosen to be as the following forms:

1. Using only the {userId} attribute as the input for all tasks.
2. Using the fields {hour and day} from the timestamp attribute with {userId} as data inputs for all tasks.
3. Using the field {browserId} with the previous attributes {userId, hour, day, and browserId} for all tasks.

4. Using the field {browserId} with {userId} as data inputs for all tasks.
5. Using all users' details field beside the userId and timestamp fields.
6. Adding {Cat1} attribute for the different attributes combination mentioned above for the first task.
7. Adding {Class} attribute for the different attributes combination mentioned above for the second and third task.

All attributes have numeric and finite sets of possible values, it has to be discretized or converted to nominal values. Three different filters are used to convert these attributes:

1. **Discretize Filter:** an instance filter that discretizes a range numeric attributes in the dataset into nominal attributes. In this dissertation number of bins=10 for all attributes.
2. **PKIDiscretize Filter:** (Proportional k-interval discretization) tunes discretization bias and variance by adjusting discretized interval size and number proportional to the number of training instances.
3. **NumericToNominal Filter:** a filter for turning numeric attributes into nominal ones. Unlike discretization, it just takes all numeric values and adds them to the list of nominal values of that attribute.

Note: for the three filters (Discretize, PKI Discretize and NumericToNominal) in WEKA, if the training set and test set are filtered separately, a problem of incompatibility between the training set and the supplied test set appears. To overcome this incompatibility problem, both training and testing data are joined in one file and I apply the specific filter on that file, and then split it again for exactly the same original ratio and instances.

3.2.2 Test Options

The trained classifiers are tested using different methods, such as using training data itself, Hold-out (percentage split of the training data), cross validation with 10 folds and finally using the testing data supplied by the organizers.

1. **Training Set:** The classifier is evaluated on how well it predicts the class of the instances it was trained on. This usually gives very optimistic results.
2. **Cross Validation:** In this method, the training set is partitioned into K subsamples where a single subsample of it is retained as the validation data for testing the model, and the remaining $K - 1$ sub-samples are used as training data. The cross-validation process is then repeated K times (the folds), with each of the K subsamples used exactly once as the validation data. The K results from the folds then can be averaged to produce a single

estimation. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation, and each observation is used for validation exactly once. In this dissertation10-fold cross-validation is used.

3. **Hold-out 'Percentage split':** The classifiers are evaluated on how well it predicts a certain percentage of the data which is held out for testing. The ratio of data held out is similar to the ratio between the training data and supplied test data. The total train Data is 379,485 Records, while the supplied test data is 166,299 records which is around one third of the training set. Training data had been divided to 69.53% (263,555 Records) as train data and 30.47% (115,629 Records) for test data.

4. **Supplied Test data:** The classifier is evaluated on how well it predicts the class of a set of instances loaded from a file. The supplied test data is 166,299 records. All the users represented in the training set are as well represented in the testing set.

Chapter 4

Analysis

This chapter describes the analysis for each task. First, the initial analysis for each task and the criteria taken to solve it are presented. Then the result of each task is explored and compared with the contest results.

The three tasks of the contest are classification problems. In each task the unknown value of an attribute y has to be predicted using values of other attributes. These experiments had been done using WEKA3.6 on Intel Centrino T5900 2.2GHz processor and 4GB RAM.

4.1 Task1

4.1.1 Analysis of Task1

The first task is determining length of the visit whether it is long or short visit according to the definition:

$$Length = \begin{pmatrix} short \Leftrightarrow m = 1 \\ long \Leftrightarrow m > 1 \end{pmatrix}$$

where: m stands for number of different Categories in one Visit.

In order to choose the best classifier, Association Rule has been applied on the training data to construct any useful information could help in the decision. Using association rule in WEKA gives these two rules:

$Day = Sun\ \ 67492\ \ \Rightarrow Class = -1\ \ 49361\ \ conf : (0.73)$

$Day = Sat\ \ 56541\ \ \Rightarrow Class = -1\ \ 41019\ \ conf : (0.73)$

This means there is a 73% confidence that the visits on Sunday and Saturday are long visits, which agrees with the results from the previous analysis and visualization. Applying association rule after including the table users' details did not give any significant information.

To solve this task I tried two ways to deal with dataset attributes. The first one is converting the numeric values to nominal values and explore the effect on the classification process using three different filters which are: Discretize PKIDiscretize, and NumericToNominal Filters. The second one is keeping the attributes as numeric values. Different classification methods are used for this task: Naive Bayes, SVM, Decision Tree C4.5 (JR48 in WEKA), Decision Table, and DTNB.

4.1.2 Using Nominal Attributes

1. Using "Discretize" Filter:
The first attempt is discretizing the data using "Discretize" filter. First data is discretized by equal binning where it has been divided to 10 intervals and then by equal frequency. The results are the same for both. Table 4.1 shows the accuracy of each classifier and the time taken of building the model for the four test options using {userId, day and hour} as input attributes for equal binning discretizing. The accuracy shown in this table is exactly the same when we use only {userId} as input attribute.

Test Options	Naive Bayes		J48		SVM	
	accuracy	time	accuracy	time	accuracy	time
Training Data	72.917%	0.37 sec	72.917%	3.48 sec	72.917%	15542.8 sec
Percentage Split	72.914%	0.36 sec	72.914%	3.48 sec	72.914%	17726.9 sec
Cross-Validation	72.917%	0.62 sec	72.917%	2.12 sec	72.917%	19445.9 sec
Supplied Test set	73.313%	0.53 sec	73.313%	3.99 sec	73.313%	13597.13 sec

Table 4.1: Task1: Classification results using {userId, day, hour} after discretizing the data.

The results show that for the discretized data it gives always the majority vote for the three used classifiers. By looking at the confusion matrices we note that all visits is completely classified as short visits which results 100% correctly predicted values for (Class = short) while all classifiers fail to predict any of the long visits class.

a	b	c	d	e	f	g	h	i	j	Classified as
121,919	0	0	0	0	0	0	0	0	0	a = '(-inf–0.8]'
0	0	0	0	0	0	0	0	0	0	b = '(-0.8–0.6]'
0	0	0	0	0	0	0	0	0	0	c = '(-0.6–0.4]'
0	0	0	0	0	0	0	0	0	0	d = '(-0.4–0.2]'
0	0	0	0	0	0	0	0	0	0	e = '(-0.2-0]'
0	0	0	0	0	0	0	0	0	0	f = '(0-0.2]'
0	0	0	0	0	0	0	0	0	0	g = '(0.2-0.4]'
0	0	0	0	0	0	0	0	0	0	h = '(0.4-0.6]'
0	0	0	0	0	0	0	0	0	0	i = '(0.6-0.8]'
44,380	0	0	0	0	0	0	0	0	0	j = '(0.8-inf)'

Table 4.2: Task1: Confussion Matrix for NB, J48, SVM using 'discretize' Filter.

Figure 4.1 is a plot for ROC Curve where (Class=short) is the positive one. The area under the ROC equal 0.5181, which shows the poor of classifiers sensitivity and specificity.

2. Using "PKIDiscretize" Filter:

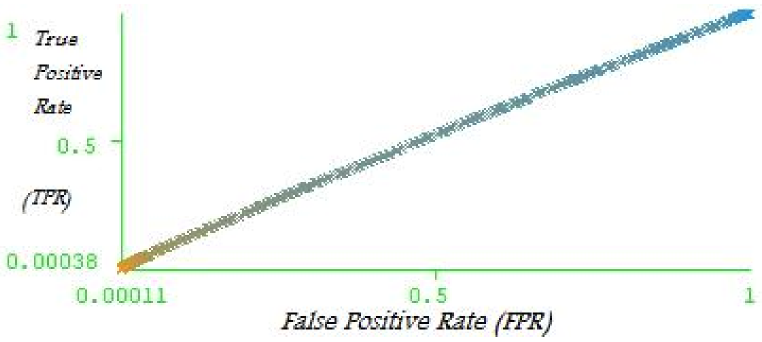


Figure 4.1: Task1: ROC Curve for NB Classifier using 'discretize' filter.

At the beginning the default setting of the filter "Discretize" is used to discretize data to ranges of 10 bins. Later I note that as the bins numbers increase, the accuracy is getting better. But this will result a problem of memory capability in WEKA. This means that discretizing the data makes the input userId lose its individual characteristics. As a second attempt "PKIDiscretize" filter, which discretizes attributes using equal frequency binning, is used.

Test Option	Naive Bayes	
	accuracy	time
Training Data	72.960%	0.51 sec
Percentage Split	72.862%	0.55 sec
Cross-Validation	72.918%	1.37 sec
Supplied Test set	73.342%	0.42 sec

Confusion matrix attribute{userId}		
a	b	Classified as
121,446	474	a = '(-inf-0)' short
43,858	522	b = '(0-inf)' long

Table 4.3: Task1: Results of PKIDiscretize filter/NB Classifier using {userId}.

Table 4.3 and Table 4.4 illustrate the result of using Naive Bayes Classifier using {userId} and {userId, day, hour} attributes respectively, and show the confusion matrix in each case when "PKIDiscretize" filter is used on the data.

Test Option	Naive Bayes	
	accuracy	time
Training Data	72.971 %	0.33 sec
Percentage Split	72.885 %	0.38 sec
Cross-Validation	72.935 %	4.06 sec
Supplied Test set	73.363%	0.61 sec

Confusion matrix {userId, day, hour}		
a	b	Classified as
121,386	534	a = '(-inf-0)' short
43,763	617	b = '(0-inf)' long

Table 4.4: Task1: Results of PKIDiscretize filter/NB Classifier using {userId, day, hour}.

There is a very slight increase in the accuracy and in the elapsed time taken to build the model. However, looking at the confusion matrix shows that the prediction of the minor class (long) has been improved from 0% by using "discretize" filter to 1.4% and the major class (short) accuracy has been decreased to 99.6%. The area under the ROC, where (Class=short) is the positive one, is improved to be 0.609 as shown in Figure 4.2.

However, applying J48 classifier on the data after filtering it by "PKIDiscretize" filter gives exactly the same result we got by using "Discretize" filter without any change or improvement at all.

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.996	0.986	0.735	0.996	0.846	0.609	'(-inf-0)'
	0.014	0.004	0.536	0.014	0.027	0.609	'(0-inf)'
Avg.	0.734	0.724	0.682	0.734	0.627	0.609	

Table 4.5: Task1: Detailed Accuracy by class for PKIDiscretize Filter/NB Classifier

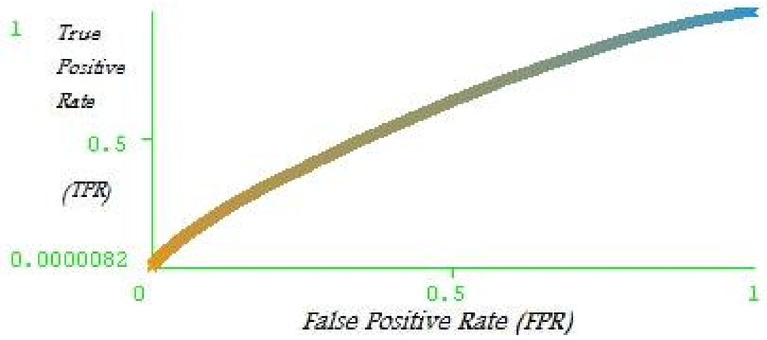


Figure 4.2: Task1: ROC Curve for NB Classifier using "PKIDiscretize" filter.

3. Using "NumericToNominal" Filter:

The slight improve of PKIDiscretize/Naive Bayes Classifier encourages me to try to keep the user's granule by converting the numeric representation of the user identity to nominal values. By using "NumericToNominal" filter there are 4882 distinct users instead of aggregating a range of users into a single value according to number of bins. Table 4.6 shows the results of Naive Bayes classifier using "NumericToNominal" filter.

Test Options	Naive Bayes {userId}		Naive Bayes {userId, day, hour}	
	accuracy	time	accuracy	time
Training Data	77.4489%	0.39 sec	77.4741%	0.64 sec
Percentage Split	76.9219%	0.41 sec	76.9098%	0.7 sec
Cross-validation	77.0304%	0.37 sec	77.0301%	1.39 sec
Supplied Test set	76.6681%	0.42 sec	76.6428%	0.52 sec

Table 4.6: Task1: Naive Bayes Classifier using 'NumericToNominal' filter

There is a good improvement of the classifier and prediction accuracy with a very slight increase of the time taken to build the models. The best result for the test set is 76.668% by considering {userId} attribute only. Including the attributes {day and hour} decrease the accuracy very slightly including a little increase of the time of model building.

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.937	0.701	0.786	0.937	0.855	0.746	'(-inf-0)'
	0.299	0.063	0.633	0.299	0.406	0.746	'(0-inf)'
Avg.	0.767	0.531	0.745	0.767	0.735	0.746	

Table 4.7: Task1: Detailed Accuracy by class for NumericToNominal filter/NB Classifier

Table 4.7 gives more details about the classification performance. These results are the same for {userId} and {userId, day, hour} attributes. The prediction rate of the minor class 'long visits' increases to 29.9%, however the prediction rate of the major class 'short visits' decreases to 93.7%. The area under ROC for the short class increases noticeably to 0.7461 as shown in Figure 4.3.

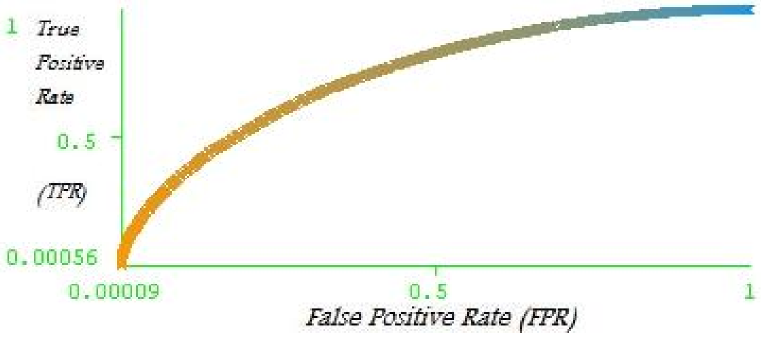


Figure 4.3: Task1: ROC Curve for NB Classifier using NumericToNominal filter.

Confusion Matrix using attribute {userId}			Confusion matrix using attribute {userId, day, hour}		
a	b	Classified as	a	b	Classified as
114,249	7671	a = '(-inf-0)' short	114,204	7716	a = '(-inf-0)' short
31,130	13250	b = '(0-inf)' long	31,127	13253	b = '(0-inf)' long

Table 4.8: Task1: Confusion Matrix for NumericToNominal/NB Classifier

The confusion matrices in Table 4.8 show the very slight difference results between using {userId} and {userId, day, hour} attributes. There is unnoticeable decrease in the short visit prediction and unremarkable increase in the long visit prediction. However, this implies the existence of time effect on user browsing behavior.

Test Options	Naive Bayes {userId, browserId}		Naive Bayes {userId, day, hour, browserId}	
	accuracy	time	accuracy	time
Training Data	77.4207%	0.55 sec	77.427 %	0.64 sec
Percentage Split	76.9037%	0.5 sec	76.929%	0.56 sec
Cross-validation	76.9693%	0.3 sec	76.998%	0.45 sec
Supplied Test set	76.6072%	0.61 sec	76.6282%	0.81 sec

Table 4.9: Task1: Including {browserId} as input attribute for NB Classifier

Including attribute {browserId}, with and without the timestamp, as inputs for Naive Bayes classifier decreases the accuracy a little as shown in Table 4.9, however the confusion matrices in Table 4.10 present improvement in the minor class prediction to 30.7% and the short class prediction decreases to 93.4%.

The final option is using all attributes including timestamp and users' details fields which include 10 attributes: {userId, day, hour, countryId, regionId, cityId, systemId, systemSubId, browserId, browserVerId}. The result follows the previous trend; the overall accuracy decreases to (76.067%) while the minor class prediction accuracy rises to (33.1%) which is the best for that class in all models. The short class prediction accuracy falls to 91.7%.

Confusion Matrix using attribute {userId, browserId}			Confusion matrix using attribute {userId, day, hour, browserId}		
a	b	Classified as	a	b	Classified as
113,999	7,920	a = '(-inf-0)' short	113,826	8,093	a = '(-inf-0)' short
30,982	13,398	b = '(0-inf)' long	30,774	13,606	b = '(0-inf)' long

Table 4.10: Task1: Confusion Matrix for NB Classifier including {browserId}

Classifiers J48 and SVM could not be applied on the data after using NumericToNominal filter because of WEKA memory heap size issues. Instead Decision Table and DTNB classifier are applied on the data. Both classifiers give always the same result regardless of the input attributes combined to {userId}. While Decision Table classifier gives accuracy of 76.6458%, DTNB classifier gives 76.6681% which is exactly the same as Naive Bayes classifier. However DTNB consumes more time than NB classifier.

A very interesting result is shown when I include the first visited category as additional attribute. While it decreases the prediction of Naive Bayes and DTNB classifiers, it highly improves the Decision Table performance. It gives the best result so far which is 77.8904%.

Table 4.11 summarizes the results of Naive Bayes classifier in all input cases by testing it on the supplied test set.

Input Attributes	accuracy	Input Attributes	accuracy
{userId}	76.6681%	{userId,Cat1}	76.4755%
{userId, BrowserId}	76.6072%	{userId, browserId, Cat1}	76.508 %
{userId, day, hour}	76.6428%	{userId, day, hour,Cat1}	76.4953%
{userId, day, hour, browserId}	76.6282%	{userId, day, hour, browserId, Cat1}	76.5314%
{All attributes}	76.067%	{All attributes, Cat1}	76.1953%

Table 4.11: Task1: Summary of NB classifier with different inputs

4.1.3 Using Numeric Attributes

In this experiment, the input attributes remains in its numeric states, however the class attribute has converted to Nominal since most classifiers in WEKA requires nominal class.

Applying Naive Bayes on these attributes gives the majority vote (73.31%) which is the short class prediction. Even using a kernel estimator for numeric attributes rather than a normal distribution did not improve the results at all.

However It becomes possible to apply J48 Classifier on the numeric dataset without memory heap size problem. C4.5 (J48 in Weka) has two parameters to tune: the first is the Confidence value where tree pruning is determined (tree pruning is removing branches that are deemed to provide little or no gain in statistical accuracy of the model). Decreasing the confidence factor will cause more drastic pruning, and a more general model of the data while increasing the confidence factor will decrease the amount of pruning that occurs and provide a more specific modeling based on the training data.

The second parameter determines the minimum number of instances that must be present in the training data for a new leaf to be created in the decision tree to handle those particular instances. This too can create a more generalized or specialized tree; a higher number will create a more generalized tree and a lower number will create a more specialized tree. However changing that parameter has no affect on classifying our data.

Table 4.12 shows different values of the confidence factor and its effects on the classification accuracy.

Attributes/ Confidence Factor	0.15	0.25	0.35	0.45	0.55
userId	76.378%	76.481%	76.556%	76.534%	76.471%
userId, browserId	76.637%	76.653%	76.6685%	76.6884%	76.6288%
userId, day, hour	76.115%	76.119%	76.0299%	76.114%	75.6691%
userId, day, hour, browserId	76.393%	76.233%	76.0997%	75.882%	75.609 %

Table 4.12: Task1: J48 classifier with different pruning

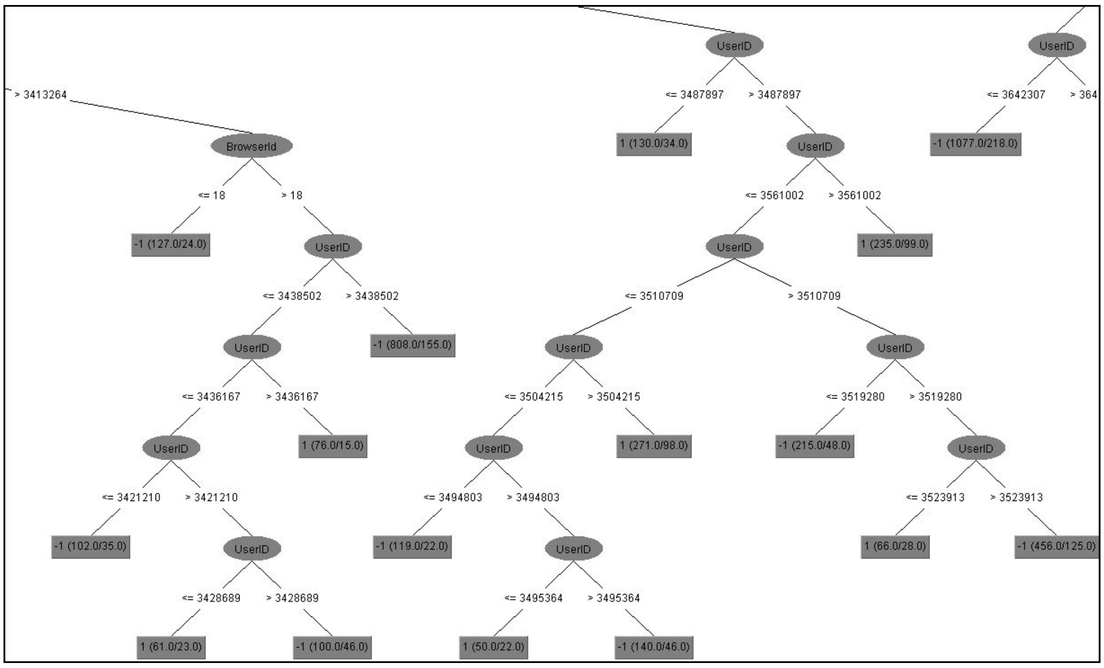
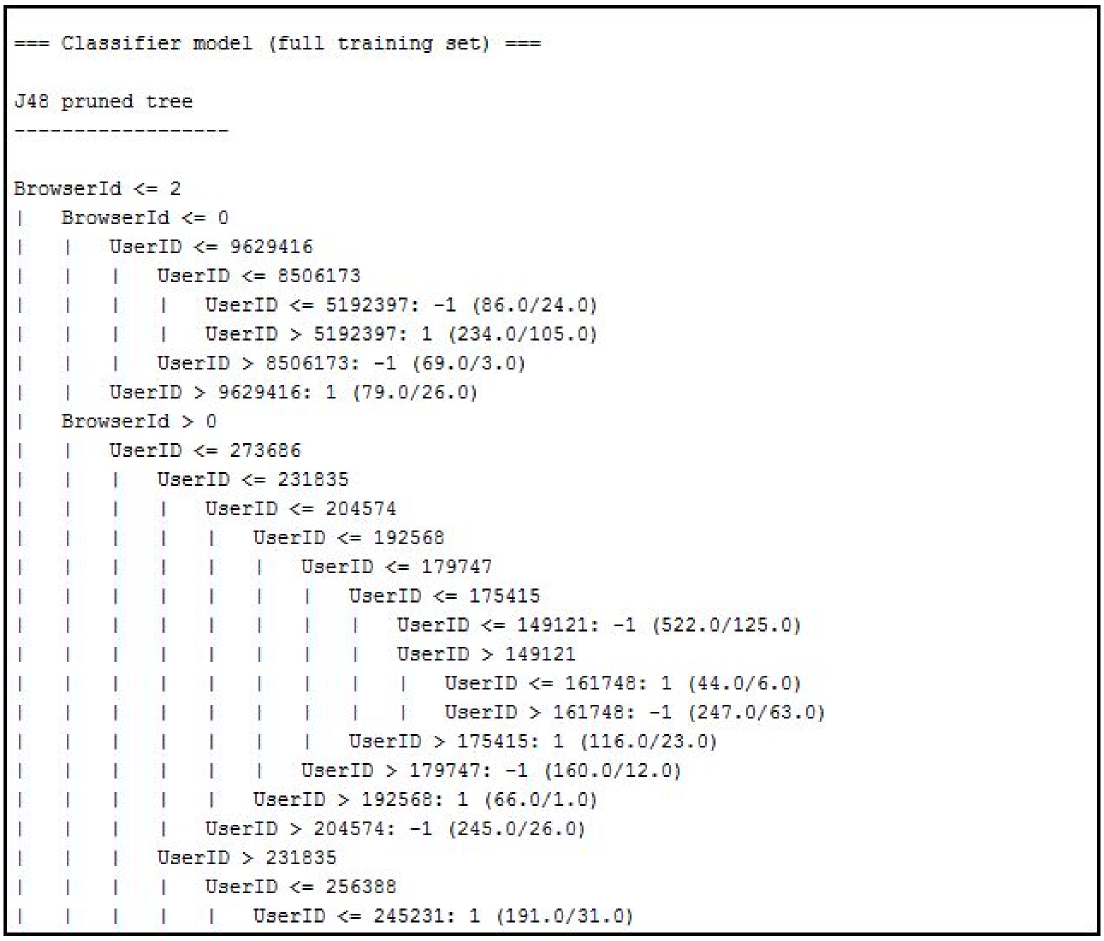


Figure 4.4: Task1: a small section of the induced tree of J48 Classifier, CF 0.43

The best result is 76.7082% using attributes {userId, browserId} only with confidence factor equal to 0.43 (not shown in the table). The empirical experiments show that the classifier accuracy decreases gradually above and under this value. Figure 4.4 views a small section of the induced tree with pruning 0.43 while Figure 4.5 represents a small part of Weka output for this tree. Nodes that generate a classification are followed by a number (sometimes two) in parentheses. The first number tells how many instances are correctly classified by this node. The second number, if it exists (if not, it is taken to be 0.0), represents the number of instances incorrectly classified by the node.

The result in table 4.12 shows the instability of decision tree classifier. Any slight variations in the training data and the attribute selections or the confidence factor produce a significant



```
=== Classifier model (full training set) ===

J48 pruned tree
------------------

BrowserId <= 2
|   BrowserId <= 0
|   |   UserID <= 9629416
|   |   |   UserID <= 8506173
|   |   |   |   UserID <= 5192397: -1 (86.0/24.0)
|   |   |   |   UserID > 5192397: 1 (234.0/105.0)
|   |   |   UserID > 8506173: -1 (69.0/3.0)
|   |   UserID > 9629416: 1 (79.0/26.0)
|   BrowserId > 0
|   |   UserID <= 273686
|   |   |   UserID <= 231835
|   |   |   |   UserID <= 204574
|   |   |   |   |   UserID <= 192568
|   |   |   |   |   |   UserID <= 179747
|   |   |   |   |   |   |   UserID <= 175415
|   |   |   |   |   |   |   |   UserID <= 149121: -1 (522.0/125.0)
|   |   |   |   |   |   |   |   UserID > 149121
|   |   |   |   |   |   |   |   |   UserID <= 161748: 1 (44.0/6.0)
|   |   |   |   |   |   |   |   |   UserID > 161748: -1 (247.0/63.0)
|   |   |   |   |   |   |   UserID > 175415: 1 (116.0/23.0)
|   |   |   |   |   |   UserID > 179747: -1 (160.0/12.0)
|   |   |   |   |   UserID > 192568: 1 (66.0/1.0)
|   |   |   |   UserID > 204574: -1 (245.0/26.0)
|   |   |   UserID > 231835
|   |   |   |   UserID <= 256388
|   |   |   |   |   UserID <= 245231: 1 (191.0/31.0)
```

Figure 4.5: Task1: a small section of Weka output of J48 pruned tree, CF 0.43

difference since attribute choices affect all descendent subtrees. Trees created from numeric data sets can be quite complex since attribute splits for numeric data are binary as shown in Figure 4.4.

Adding {Cat1} attribute improves the performance of J48 classifier for all combinations. Again tuning the confidence factor changes the result in each group. The best result in this case is 77.5471% by using {userId, browserId, Cat1} attributes with confidence factor equal 0.35.

Attributes/ Confidence Factor	0.15	0.25	0.35	0.45
userId,Cat1	77.1478%	77.3005%	77.3138%	77.2795%
userId, browserId, Cat1	77.5332%	77.4833%	77.5471%	77.4978%
userId, day, hour,Cat1	76.8838%	76.8994%	76.8856%	76.8014%
userId, day, hour, browserId, Cat1	77.1159%	77.1039%	77.0059%	76.9367%

Table 4.13: Task1: J48 classifier with different pruning including the first category

4.1.4 Task1 Summary

The results of task 1 can be summarized as follows:

1. Using the ‘Discretize’ filter gives us always the same result which is the majority vote regardless of the used classifier or the test method. It failed to predict any of the minority class (long visits).

2. While using ‘PKIDiscretize’ improves the performance of Naive Bayes classifier very slightly, it did not affect at all the act of J48 classifier. Moreover it increases the time taken to build the models.

3. The time taken to build the models increases as we add more input attributes.

4. Using ‘Discretize’ filter affects the classifiers accuracy since it joints several users in one interval causing the loss of user identity. And since we have only the user id as classifier input, discretizing it leads to very poor classifier.

5. Using the ‘NumericToNominal’ filter helps to keep the individuality of the input userId which improves the accuracy in all classifiers with the different validation methods.

6. Using Naive Bayes with ‘NumericToNominal’ filter and validating it on the supplied test set gives us accuracy of 76.668% which is the same result obtained by DTNB classifier. By looking at the confusion matrix, 93.7% of the short visits predicted correctly while the long visits correct prediction is around 29.9% only.

7. Even though using more input attributes did not give significant difference in the overall accuracy, it improves the prediction accuracy of the minor class ($longclass$). The best prediction for that class is (33.1%) by using all attributes (userId, timestamp and users' details) and (35.3%) by including {Cat1} attribute.

8. Using Naive Bayes classifier with numeric attributes gives only the majority vote while J48 classifier gives better classification (76.7082%) with threshold value equals to 0.43 using {userId, browserId} attributes.

9. Adding the first visited Category for the input attributes helps in predicting user behavior. It gives the best performance 77.8904% using Decision Table Classifier.

10. while changing the specific threshold causes accuracy decreasing, the different iterations shows that increasing the confidence value of J48 tree increase the prediction accuracy of the minor class.

11. Even when the accuracy is similar in all classifiers, the time of building the model varies. Naive Bayes is the fastest classifier while SVM is spending much longer time to achieve the same results.

12. WEKA has a limitation in the used heap size. We increase it to the maximum size allowed for 32-bit operating system which is 1500MB. However we still face the same problem ‘Out of Memory’ when we run some classifiers such as SVM and J48 with Nominal values.

4.1.5 Task1 Evaluation

Table 4.14 shows the results of task1 comparing with the results of the winner and the runner-up of ECML_PKDD 2007 Discovery Challenge.

The best result I obtain is by using J48 classifier on numeric values using {userId, browserId} attributes (76.7082%) with confidence factor equal to (0.43). Naive Bayes classifier on WEKA platform gives accuracy (76.67%) using {userId} attribute only after converting the datasets to nominal attributes.

While the winner results surpass my approach slightly (0.19%) by using attribute input userId my result is outperform slightly the runner-up result (better than it with nearly 0.07%). The winner got his result using (User Model- Trend Prediction) in the first case while they use WEKA-J48 classifier on (Enhanced Global Model) using many attributes in the second case. The runner-up obtain their results using Naive Bayes classifier using their own Java code. However both of the winner and runner-up did not give any information about the distribution of their accuracy, i.e. the prediction performance of the major and minor classes, since the nature of the data is biased toward the short class. On the other hand, all the results agree that including other features such as timestamp or user's details reduce the overall classification accuracy but I also show that using timestamp and user's details, specially browserId, improve the minor class prediction.

I also prove that knowing the first category of the visit affects on predicting the visit length. Except Naive Bayes classifier, adding the first visited category improves the performance of all other classifiers. The best result was 77.8904% using Decision Table classifier. This is important point since it provides website managers more accurate method of predicting user behavior with reasonable assumptions (the category of the first visited page is known prior to predicting whether the user will pursue to other categories or will end his/her session).

Attributes	Contest Winner	Contest Runner-up	My result
userId	76.90%	76.64%	76.67%
userId, timestamp	76.7%	76.6%	76.64%
userId, browserId			76.7082%
userId, Cat1			77.8904%

Table 4.14: Task1 Evaluation

4.2 Task 2

4.2.1 Analysis of Task2

The second task goal is predicting the most three probable categories for a given user. Input data basically will be userId and timestamp of the first Page View. The result should be three identifiers of the most probable categories during a given visit for a given user that appear in the first three places of the visit path according to the following formula:

$$CategoryId_{predicted} = \begin{pmatrix} categoryId\ _{predicted1} \\ categoryId\ _{predicted2} \\ categoryId\ _{predicted3} \end{pmatrix}$$

Association Rule has been applied on the training data using only the first three categories visited by the user. Using only userId as input, we get these three rules:

$Cat2 = 0\ \ 121919 \Rightarrow\ \ Cat3 = 0\ \ 121919\ \ conf : (1)$

$Cat1 = 17\ \ Cat2 = 0\ \ 58130 \Rightarrow\ \ Cat3 = 0\ \ 58130\ \ conf : (1)$

$Cat1 = 7\ \ Cat2 = 0\ \ 34824 \Rightarrow\ \ Cat3 = 0\ \ 34824\ \ conf : (1)$

The extracted rules are straightforward. The first rule assures with 100% confidence that whenever there is no second category, then no third category, which is simply the case in the short visits. The second rule states that in around 22.5% of the cases where $category17$ was the first category to visit, the user will not visit any more pages after it. So it is short visits. As well, the third rule states that in around 13.5% of the cases where $category7$ is visited first by the user, it is a short visit and no more categories will be visited.

Finally, applying association rule using inputs {userId, Day, Hour} gives these rules:

$Day = 1\ \ Cat3 = 0\ \ 57004 \Rightarrow Cat2 = 0\ \ 49361\ \ conf : (0.87)$

$Day = 7\ \ Cat3 = 0\ \ 47437 \Rightarrow Cat2 = 0\ \ 41019\ \ conf : (0.86)$

$Day = 2\ \ 51609 \Rightarrow Cat3 = 0\ \ 43815\ \ conf : (0.85)$

$Day = 4\ \ 51676 \Rightarrow Cat3 = 0\ \ 43812\ \ conf : (0.85)$

$Day = 5\ \ 51428 \Rightarrow Cat3 = 0\ \ 43480\ \ conf : (0.85)$

$Day = 1\ \ 67492 \Rightarrow Cat3 = 0\ \ 57004\ \ conf : (0.84)$

$Day = 3\ \ 49751 \Rightarrow Cat3 = 0\ \ 41983\ \ conf : (0.84)$

$Day = 6\ \ 50988 \Rightarrow Cat3 = 0\ \ 42886\ \ conf : (0.84)$

$Day = 7\ \ 56541 \Rightarrow Cat3 = 0\ \ 47437\ \ conf : (0.84)$

These rules present the same information from our histogram in chapter 3 about the relations between the days of week and categories. It is not satisfactory to extract any information to help in predicting the visited categories; especially that it does not give any information depending on the user itself, which did not help us in choosing the best classifier, however it suggests to try Day and Hour attributes as inputs for the selected classifiers.

4.2.2 Using Nominal Attributes

1. Using "Discretize" Filter:

The data has been discretized using 'Discretize' filter. Both Naive Bayes and J48 classifiers are applied on the data.

a. Naive Bayes Classifier:

The first classifier is Naive Bayes Classifier. It has been applied on data initially using input attribute {userId}. For the third category, 142,931 instances are correctly classified meaning that the accuracy rate is 85.9477%.

Test Options	Category1		Category2		Category3	
Training Data	43.779%	0.64 sec	74.886%	0.3 sec	85.7927%	0.78 sec
Percentage Split	43.755%	0.23 sec	74.812%	0.33 sec	85.7547%	0.73 sec
Cross-Validation	43.779%	0.37 sec	74.886%	0.3 sec	85.7788%	0.84 sec
Supplied Test set	44.729%	0.47 sec	75.082%	0.34 sec	85.9477%	0.6 sec

Table 4.15: Task2: NB Classifier using {userId} with 'Discretize' filter

The reason of this relatively high accuracy maybe because most of visits are short visits hence the 3rd category is mostly zeros. 90.6% of the short visits have been predicted correctly. However if the prediction rate of (category Id=0) is eliminated; only 15,203 correctly predicted instances remain meaning that the actual prediction accuracy for other categories in this position is around 60%. This percentage is distributed almost between all categories as shown in the confusion matrix in Table 4.16.

a	b	c	d	e	f	g	h	i	j	Classified as
127728	27	1	3718	616	0	1728	1061	6046	24	a = '(-inf-1.9]'
187	16	0	57	3	0	6	3	73	0	b = '(1.9-3.8]'
112	1	1	26	6	0	7	1	31	0	c = '(3.8-5.7]'
1720	3	0	4097	48	0	138	39	563	6	d = '(5.7-7.6]'
860	2	0	753	492	0	85	30	299	4	e = '(7.6-9.5]'
144	0	0	34	20	0	101	3	38	0	f = '(9.5-11.4]'
382	0	0	151	22	0	1237	90	165	0	g = '(11.4-3.3]'
134	1	0	64	15	0	32	3309	118	0	h = '(13.3-5.2]'
2757	5	0	474	28	0	41	146	6033	5	i = '(15.2-17.1]'
60	0	0	17	4	0	6	1	27	18	j = '(17.1-inf)'

Table 4.16: Task2: Confusion Matrix of 3rd category prediction using NB Classifier.

As expected the accuracy has been dropped to 75.082% when the 2nd category is predicted depending on {userId and Cat1} as input. In this position, 98.6% of the short visits are predicted correct while only 9.66% of the long visits are correctly classified and they are all belong to $Category$ 7. The classifier fails to predict any of the other categories in that position.

Only accuracy of 44.729% has been got in predicting the 1st category depending only on the userId as input. The $category$ 17 has been predicted correctly with accuracy of 98% and $category$ 7 prediction accuracy is 3.4% while the classifier fails to predict any of the other categories. This somehow supports the extracted rules from applying Association rule on the data. However it is expected results since categories 17 and 7 are the most popular visited categories.

Test Options	Category1		Category2		Category3	
Training Data	43.794%	0.33 sec	74.886%	0.36 sec	85.899%	0.58 sec
Percentage Split	43.767%	0.52 sec	74.812%	0.53 sec	85.7605%	0.69 sec
Cross-Validation	43.781%	0.58 sec	74.886%	0.39 sec	85.8637%	0.46 sec
Supplied Test set	44.708%	0.62 sec	75.082%	0.37 sec	86.0432%	0.44 sec

Table 4.17: Task2: NB Classifier using attributes {userId, day, hour} with 'Discretize' filter

Table 4.17 shows the result of the same classifier using {userId, day, hour}. We note a very slight increase for the three categories using all the test options except for the supplied test set which shows a tiny decrease in the accuracy for the first category.

b. TreeJ48 Classifier:

While using J48 classifier improves slightly the accuracy of Cat3 prediction to 86.745%, the confusion matrix shows that this increase is in predicting (category Id =0) and the prediction of the other categories in the 3rd position falls to only 29.23%. Cat2 accuracy remains the same as Naive Bayes and Cat1 accuracy decreases slightly to 44.511% but here only Category 17 has been predicted correctly instead of categories 7 and 17 in Naive Bayes Classifier.

Test Options	Category1		Category2		Category3	
Training Data	43.7199%	0.42 sec	74.886%	1.17 sec	86.7214%	2.01 sec
Percentage Split	43.7477%	0.44 sec	74.812%	1.12 sec	86.6072%	2.01 sec
Cross-Validation	43.7199%	0.41 sec	74.886%	3.07 sec	86.6835%	1.98 sec
Supplied Test set	44.5105%	0.41 sec	75.082%	1.1 sec	86.745%	2.11 sec

Table 4.18: Task2: J48 Classifier using attributes {userId} with 'Discretize' filter.

Using {userId, day, hour} attributes with J48 classifier mostly make no difference with the results except the increase of time taken to build the models in general as shown in table 4.19.

Test Options	Category1		Category2		Category3	
Training Data	43.7199%	3.88 sec	74.886%	2.4 sec	86.9865%	5.69 sec
Percentage Split	43.7477%	1.65 sec	75.082%	3.63 sec	86.6167%	5.37 sec
Cross-Validation	43.7117%	1.72 sec	74.886%	4.28 sec	86.6698%	6.41 sec
Supplied Test set	44.5108%	1.89 sec	75.082%	2.36 sec	86.7565%	3.85 sec

Table 4.19: Task2: J48 Classifier using attributes {userId, day, hou} with 'Discretize' filter.

I couldn't calculate the score function for these classifiers with this filter because I couldn't get the output prediction exactly since the instants are divided into intervals not individually.

2. Using "NumericToNominal" Filter:

a. Naive Bayes Classifier:

Using filter 'Numeric2Nominal' instead of 'Discretize' filter improves the accuracy noticeably especially for the first category prediction as shown in Table 4.20. For the 3rd category prediction the accuracy is 86.96%. While the prediction of (category Id = 0) stay the same, the prediction accuracy of other categories in that position increase to 64.48%. Similarly the

2nd category prediction accuracy increases to 75.88% where 21.3% only belongs to categories other than ‘0’ category but in this case the percentage is distributed over all the categories not only category 7 as when using ‘Discretize’ filter. The great improve is the remarkable increase of 1st category prediction. The accuracy increases from 44.7% to 74.18% and it is spread over all categories not only category 7 and 17 as in the previous case.

Test Options	Category1		Category2		Category3	
Training Data	76.0622%	0.43 sec	76.2306%	0.45 sec	87.6177%	0.62 sec
Percentage Split	75.3721%	0.25 sec	75.8694%	0.33 sec	86.9583%	0.53 sec
Cross-Validation	75.5086%	0.27 sec	75.9761%	0.66 sec	86.9338%	1.75 sec
Supplied Test set	74.1858%	0.41 sec	75.8876%	0.65 sec	86.9579%	0.66 sec

Table 4.20: Task2: NB Classifier using userId with ‘NumericToNominal’ filter

Some detailed Accuracy by class is shown in Table 4.21.

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.072	0	0.872	0.072	0.134	0.843	1
	0	0	0	0	0	0.822	2
	0.159	0.001	0.589	0.159	0.251	0.883	3
	0.348	0	0.896	0.348	0.501	0.824	4
	0.124	0	0.674	0.124	0.209	0.852	5
	0.558	0.002	0.797	0.558	0.656	0.929	6
	0.775	0.1	0.744	0.775	0.759	0.923	7
	0.411	0.007	0.695	0.411	0.516	0.88	8
	0.74	0.006	0.843	0.74	0.788	0.959	9
	0.145	0	0.771	0.145	0.244	0.794	10
	0.428	0.001	0.774	0.428	0.551	0.905	11
	0.778	0.027	0.699	0.778	0.736	0.972	12
	0.311	0	0.804	0.311	0.449	0.929	13
	0.666	0.013	0.646	0.666	0.656	0.967	14
	0.34	0	0.947	0.34	0.5	0.82	15
	0.302	0.01	0.563	0.302	0.393	0.873	16
	0.845	0.218	0.755	0.845	0.797	0.897	17
	0.383	0	0.921	0.383	0.541	0.938	18
	0.319	0	0.917	0.319	0.473	0.887	19
	0	0	0	0	0	?	20
Avg.	0.742	0.127	0.739	0.742	0.729	0.912	

Table 4.21: Task2: Detailed Accuracy By Class for 1st category prediction

Using other attributes {userId, day, hour} improve the classifier accuracy very slightly as shown in Table 4.22.

Test Options	Category1		Category2		Category3	
Training Data	76.078%	0.65 sec	76.2282%	0.44 sec	87.6519%	0.8 sec
Percentage Split	75.3323%	0.65 sec	75.878%	0.66 sec	86.9799%	0.66 sec
Cross-Validation	75.5004%	1.94 sec	75.9848%	0.7 sec	86.9807%	0.66 sec
Supplied Test set	74.2777%	0.45 sec	75.9217%	0.38 sec	86.9578%	0.42 sec

Table 4.22: Task2: NB Classifier using {userId, day, hour} with ‘NumericToNominal’

The next step is using the users’ details as input attributes. First the most significant factor shown in the visualization and analysis part is {browserId}. The following three forms of input attributes are included:

- {userId, browserId}
- {userId, Day, Hour, browserId}
- {userId, Day, Hour, countryId, regionId, cityId, systemId, systemSubId, browserId, browserVerId}, which means all attributes.

The following table summarizes the results of Naive Bayes classifier in all input cases by testing it on the supplied test set.

Input Attributes	Category1		Category2		Category3	
{userId}	74.19%	0.41 sec	75.89%	0.65 sec	86.96%	0.66 sec
{userId, BrowserId}	74.13%	0.28 sec	75.91%	0.34 sec	86.99%	0.39 sec
{userId, day, hour}	74.28%	0.45 sec	75.92%	0.38 sec	86.96%	0.42 sec
{userId, day, hour, browserId}	74.12%	0.73 sec	75.92%	1.39 sec	87.02%	0.64 sec
{All attributes}	72.09%	1.34 sec	75.04%	1.23 sec	87.03%	3.95 sec

Table 4.23: Task2: Results of NB classifier with different inputs

Even though the results look very near to each other without significant differences, the real change is in predicting the other categories than {categoryId=0} for the second and third categories since this means the improvement of the minor class prediction.

Score Function:

Calculating the specific score function is another way to measure the classifiers and models performance. Table 4.24 shows the score function for Naive Bayes classifier in each case. The maximum possible score is 1,085,524.

Input Attributes	Score	Percentage Score	Average Score
{userId}	715,211	65.89%	4.301
{userId, BrowserId}	750,350	69.12%	4.512
{userId, day, hour}	751,803	69.26%	4.521
{userId, day, hour, browserId}	750,158	69.11%	4.511
{All attributes}	733,814	67.6%	4.413

Table 4.24: Task2: The score function for NB classifier

The score function results prove the analysis I made at the beginning. It assures the relation between {userId, day, hour and browserId} and the users' behavior. The best result was score (4.521) using inputs {userId, day, and hour} followed by including attribute only {browserId} which gives score (4.512). The score drops to (4.413) when all users' details are included but still it is better than using the {userId} attribute alone which gives score (4.3) only.

Adding Class attribute:

Since the actual predicting for the second and third categories was 21.3% and 64.6% respectively for categories other than 'categoryId=0', which is not sufficient, I added *Class* attribute which has two values (long or short) depending on the visit type. Table 4.25 shows the prediction improvement especially for the second category which increased from 21.3% to 64.6% for the actual 20 categories. As well the third category prediction increased from 64.4% to 76.7%.

Input Attributes	Category1	Category2	Category3	Score	Average Score
{userId, Class}	74.661%	90.572%	85.124%	829,429	4.988
{userId, BrowserId, Class}	74.616%	90.575%	85.127%	829,334	4.987
{userId, day, hour, Class}	74.681%	90.565%	85.139%	829,382	4.987
{userId, day, hour, browserId, Class}	74.639%	90.554%	85.135%	829,156	4.986
{All attributes, Class}	72.575%	90.097%	85.168%	808,805	4.864

Table 4.25: Task2: Results of NB classifier with different inputs including Class Attribute

Table 4.25 displays also the score function after adding {Class} attribute to the other different attributes. There is a significant increase comparing to the previous results. However the results of the different attributes are very close. the best score (4.988) is by using userId attribute alone. Using all attributes gives the minimum score function in this group (4.864) but still it is better than any of the results obtained without Class attribute.

Another attempt to improve the prediction for the actual categories is adding new attributes named {ClassAB} which has two values: (2 if there is third category and -2 if no third category in this visit). This improve the classification of the third category, however it decreases the overall effiecency of the classifier. The best score function for this group is 4.837 as shown in 4.26 which is less than the score function in the previous step.

Input Attributes	Category1	Category2	Category3	Score	Average Score
{userId, ClassAB}	74.275%	90.551%	96.276%	804,339	4.836
{userId, BrowserId, ClassAB}	74.247%	90.569%	96.262%	804,237	4.836
{userId, day, hour, ClassAB}	74.257%	90.583%	96.263%	804,418	4.837
{userId, day, hour, browserId, ClassAB}	74.141%	90.568%	96.250%	803,512	4.832
{All attributes, ClassAB}	72.229%	90.149%	96.012%	785,029	4.720

Table 4.26: Task2: Results of NB classifier with different inputs including ClassAB Attribute

I apply J48 on the data after using 'PKIDiscretize' filter and it only works for the prediction of the second and third categories but I face again the memory out problem when I try to predict the first category. Also applying J48 classifier on the data using 'NumericToNominal' filter succeed only to predict the third category with accuracy 88.13% and WEKA fails to continue in predicting the first and second category because of the memory heap size problem. So I decide to try other available classifiers on WEKA such as Decision Table and DNTB.

b. Decision Table Classifier:

Applying Decision table classifier on the nominal data gives approximately the same score function (4.383) for the different input attributes. even though the accuracy of each category prediction is more than Naive Bayes Classifier in each relative input (Cat1=74.46%, Cat2 = 76.99%, Cat3 = 88.39%), this has no affect on the final score function which means that the accuracy increasing is mainly in ‘categoryId = 0’ in the second and third category which is not significant in calculating the score function. Table 4.27 shows that especially the first category prediction didn’t change for all cases. Adding any attributes to userId has no affect at all in calculating the accuracy or the score function.

Input Attributes	Category1	Category2	Category3	Score	Average Score
{userId}	74.459%	76.986%	88.386%	728,391	4.38
{userId, BrowserId}	74.459%	76.95%	88.131%	728,899	4.383
{userId, day, hour}	74.459%	76.931%	88.386%	728,429	4.38
{userId, day, hour, browserId}	74.459%	76.931%	88.387%	728,426	4.38
{All attributes}	74.459%	76.985%	88.386%	728,391	4.38

Table 4.27: Task2: Results of DT classifier with different inputs

As done with Naive Classifier, table 4.28 represents the effect of adding Class and ClassAB attributes on the classifier quality. The table has the result of {userId, Class, ClassAB} attributes only because it is the same for alll other attributes combination. The best score function in this case is (4.841) for ClassAB which still is less than Naive Bayes classifier results.

Input Attributes	Category1	Category2	Category3	Score	Average Score
{userId, Class}	75.203%	89.277%	88.386%	783,030	4.708
{userId, ClassAB}	75.203%	89.277%	96.335%	805067	4.841

Table 4.28: Task2: Results of DT classifier with Class Attributes

c. DTNB Classifier:

Again using DTNB classifier which is a combination of Decision Table and Naive Bayes classifiers didn’t show a significant improvement in the score function. However, DTNB score function is better than what has gotton using Decision Table alone. The accuracy of the second and third category decrease which means that the predicting of the main 20 categories improves. The best score function for this classifier without using Classes attributes is 4.411 by using only the {userId} attribute.

Adding {ClassA} and {ClassAB} attributes improve the score of DTNB classifier for all other input combination. The trend is the same as Naive Bayes and DT classifiers. The best score for this classifier is (4.88) using {UserId, ClassAB} attributes.

4.2.3 Using Numeric Attributes

As done in Task1, the data input remains numeric while the class attribute is converted to nominal. The results obtained using Naive Bayes classifier is almost the same as the result of discretized inputs. While adding Class attributes improve the classification of second and third category, it has no affect on predicting the first category which remains 44.3% only.

Using J48 classifier, for each category, different tuning of confidence factor is experimented. The best result was by using attribute {userId} which gives score function of (4.412) which almost the same of DTNB classifier.

4.2.4 Task2 Summary

- The best score function is (4.988) obtained using Naive Bayes Classifier by Nominal attributes.
- Using Naive Bayes classifier for the numeric attributes is not sufficient especially for the first category prediction.
- Adding Class attributes prevents the false positive (FP) and false negative (FN) in predicting 'categoryId =0' which helps in improving the prediction of other categories.
- J48 classifier and DTNB classifier almost give the same result which is still less than Naive Bayes result.
- In most cases, adding timestamp or user details fields has no remarkable effect on the prediction performance. In contrast, knowing the visit length of the user increase the score function from 4.521 to 4.988.

4.2.5 Task2 Evaluation

The score function of both the winner and runner-up of the contest outperform my result. For the second task the winner score function was 5.56 using the User Model, while the score function of the runner-up was 5.43 using Naive Bayes with Marcov chain rules. The winner used only the user id attribute and did not try other attributes on this task while the runner-up added timestamp attribute.

Contest Winner	Contest Runner-up	My result
5.56	5.43	4.988

Table 4.29: Task2 Evaluation

4.3 Task3

This task is regarding the prediction of the range of the number of page views for the first three page categories visited in a visit session.

$$f_{predicted}\left\{\begin{array}{l} userId \\ timestamp \end{array}\right\} = \left\{\begin{array}{l} categoryId_1, range\ \ of\ \ pageview \\ categoryId_2, range\ \ of\ \ pageview \\ categoryId_3, range\ \ of\ \ pageview \end{array}\right\}$$

Possible ranges and their assigned identifiers:

$$range\ \ Of\ \ Pageview = \left\{ \begin{array}{l} 1 \\ 2-3 \\ 4-\infty \end{array} \right\}$$

By applying the Bayesian rule, the probability of the page views range at position $i (i = 1, 2, 3)$ in a visit session X is:

$$P(R_i|C_i, X) = P(R_i = r_i|C_i = c_i)P(X|R_i = r_i, C_i = c_i)$$

where the page categories ci is what we predicted in Task2.

4.3.1 Task3 Results

For this task I use the predicted categories obtained by Naive Bayes Classifier with {UserID, Class}, which gets the best score for task2. The process is iterative. First the classifier is trained on {userID, Cat1,Pageview1} on training set, and then the model is applied on the test set with same attributes but using the predicted Cat1 and Pageview1 as class label. Then the classifier is trained using {UserID, Cat1, Pageview1, Cat2, Pageview2} from training dataset and the model is applied to test set{UserID, Predicted Cat1, Predicted Pageview1 from previous step, Predicted cat2, and Pageview2 as class label}. Similarly the third step for 3rd pageview. The score function is 6.254 which outperforms the runner-up result and close to the winner result.

Contest Winner	Contest Runner-up	My result
6.3147	5.765	6.254

Table 4.30: Task3 Evaluation

Chapter 5

Conclusion and Further Work

The first challenge of ECML/PKDD Discovery Challenge was a great opportunity to apply WUM techniques and algorithms in a real world problem. In this thesis I introduced a solution to the 2007 ECML/PKDD Discovery Challenge which centered about user behavior prediction over the web. An intensive exploratory data analysis was applied to capture the interesting information from user navigation session. The visualization of the data determined the factors that have influence on predicting user behavior accurately such as the day or hour of navigation process, the browser used to navigate, the type of page visited previously and the time spent on each visit. Based on this, different classification algorithms were applied on different combination of these attributes.

For the first task, I show that while using more input attributes did not increase the overall accuracy, it still improved the prediction of the minor class (long class) considering that the data is biased to the short class. This is important since it allows the website designers to target users who spend more time in their site. I also show that we can predict user visit type (short or long) by knowing the category of first visited page.

For the second task, the results provides an experimental evidence that even though adding timestamp or user details fields has no remarkable effect on categories prediction performance, knowing the visit type (short or long) actually helps in predicting the first three visited categories. In fact it particularly improves the second and third categories prediction. Again this can help in personalizing user profile, redirect online advertising.

In general, the results I report in this thesis are comparable to the challenge winner and surpass the runner-up on two out of the three challenge problems. Given further time, I would attempt to take categories sequence in consideration and study the effect on the classifiers performance.

Appendix A

Methodologies

A.1 Classification Methods

In this section I explore and give a short description of the classifiers I used in the dissertation.

A.1.1 Naive Bayes

A naive Bayes classifier is a simple probabilistic classifier based on applying Bayesian theorem with strong independence assumptions. In spite of its simplicity, Naive Bayes can often outperform more sophisticated classification methods.

Because of the assumption of values independency, only its variances for each class need to be determined and not the entire covariance matrix, which considered as an advantage of Naive Bayes classifier since it requires a small amount of training data to estimate the necessary parameters, i.e. means and variances of the variables, for classification.

Abstractly, the probability model for a classifier is a conditional model $P(C|F_1, ..., F_n)$ over a dependent class variable C with a small number of outcomes or classes, conditional on several feature variables F_1 through F_n. Using Bayes' theorem, we write:

$$P(C|F_1, ..., F_n) = \frac{P(C)P(F_1, ..., F_n|C)}{P(F_1, ..., F_n)}$$

where: $P(C)$ is the prior probability of hypothesis C; $P(F)$ is the prior probability of training data F; $P(C|F)$ is the probability of C given F and; $P(F|C)$ is the probability of F given C.

In practice we are only interested in the numerator of that fraction, since the denominator does not depend on C and the values of the features F_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model $P(C, F_1, ..., F_n)$ which can be rewritten as follows, using repeated applications of the definition of conditional probability:

$$P(C, F_1, ..., F_n) = P(C)P(F_1, ..., F_n|C)$$

$$= P(C)P(F_1|C)P(F_2|C, F_1)P(F_3|C, F_1, F_2)...P(F_n|C, F_1, F_2, F_3, ..., F_{n-1})$$

By assuming that each feature F_i is conditionally independent of every other feature F_j for $j \neq i$. This means that:

$$P(F_i|C, F_j) = P(F_i|C)$$

and so the joint model can be expressed as:

$$P(C, F_1, ..., F_n) = P(C)P(F_1|C)P(F_2|C)P(F_3|C)...$$

$$= P(C)\prod_{i=1}^{n} P(F_i|C)$$

This means that under the above independence assumptions, the conditional distribution over the class variable C can be expressed like this:

$$P(C|F_1, ..., F_n) = \frac{1}{Z}P(C)\prod_{i=1}^{n} P(F_i|C)$$

where Z is a scaling factor dependent only on $F_1, ..., F_n$, i.e., a constant if the values of the feature variables are known (Pop, 2006).

A.1.2 Decision tree

The Decision Tree is one of the most popular classification algorithms in current use in Data Mining and Machine Learning. Decision trees are a way of representing a series of rules that lead to a class or value.

In data mining and machine learning, a decision tree is a predictive model that commonly used to examine the data and induce the tree and its rules that will be used to make predictions (Two Crows Corporation, 2005). Each branch node represents a choice between a number of alternatives, and each leaf node represents a classification or decision. Decision trees which are used to predict categorical variables are called classification trees because they place instances in categories or classes. Decision trees used to predict continuous variables are called regression trees.

A number of different algorithms may be used for building decision trees including CHAID (Chi-squared Automatic Interaction Detection), CART (Classification and Regression Trees), Quest, and C4.5.

C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan. C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier.

Figure A.1 describes the Tree Induction Algorithm:

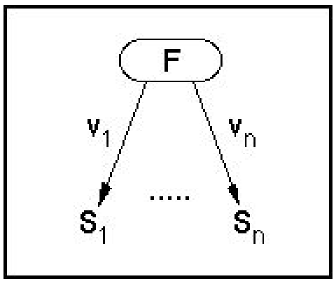


Figure A.1: Tree Induction Algorithm (Ferreira, n.d.).

- The algorithm operates over a set of training instances, C.
- If all instances in C are in class P, create a node P and stop, otherwise select a feature or attribute F and create a decision node.
- Partition the training instances in C into subsets according to the values of V.
- Apply the algorithm recursively to each of the subsets C.

The order in which attributes are chosen determines how complicated the tree is. ID3 uses information theory to determine the most informative attribute.

A.1.3 Decision Table

A decision table has two components: A schema, which is a list of attributes and a body, which is a multi set of labeled instances. Each instance consists of a value for each of the attributes in the schema and a value for the label. The set of instances with the same values for the schema attributes is called a cell (Kohavi and Sommerfield, 1998).

A DT stores the input data in condensed form based on a selected set of attributes and uses it as a lookup table when making predictions. Each entry in the table is associated with class probability estimates based on observed frequencies. The key to learning a DT is to select a subset of highly discriminative attributes (Hall1 and Frank, 2008).

While Decision Trees provide a graphic, tree-like structure with nodes that correspond to conditions, and leaf nodes that correspond to results, Decision Tables present rules in an Excel table like format, with columns that correspond to conditions and actions.

A.1.4 DTNB

Hall and Frank Hall1 and Frank (2008) introduced DTNB Classifier as a hybrid classifier combining Naive Bayes and Decision Tables techniques. At each point in the search, the algorithm evaluates the advantage of dividing the attributes into two disjoint subsets: one for the decision table, the other for Naive Bayes. A forward selection search is used, where at each step, selected attributes are modeled by Naive Bayes and the remainder by the decision table and all attributes are modeled by the decision table initially. At each step, the algorithm also considers dropping an attribute entirely from the model. The class probability estimates of the DT and NB must be combined to generate overall class probability estimates. All probabilities are estimated using Laplace corrected observed counts (Hall1 and Frank, 2008).

A.1.5 SVM

Support Vector Machine (SVM) is a popular machine learning technique proposed by Vapnik and co-workers in 1992 for classification and regression. SVMs allow us to transform the data to a new space where the decision boundary can be linear, mapping back to the original space; we can get a non-linear decision boundary.

Viewing input data as two sets of vectors in an n-dimensional space, an SVM will construct a separating hyperplane in that space, one which maximizes the margin between the two data sets. To calculate the margin, two parallel hyperplanes are constructed, one on each side of the separating hyperplane, which are "pushed up against" the two data sets. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the neighboring data points of both classes, since in general the larger the margin the lower the generalization error of the classifier (Sun et al., 2002) and (Dumais and Chen, 2000). The optimal hyperplane (max margin, lowest capacity) can be uniquely constructed by solving a constrained quadratic optimization problem. The solution $w = \sum v_i x_i$ where only a subset of the training patterns is used. This subset is called support vectors. These vectors carry all the relevant information about the classification problem.

Several existing tools are available for SVM such as: MySVM, SVMlight, Torch, LibSVM. Classifiers built on SVM have shown promising results in text classification

A.1.6 Association Rules

Association rules identify collections of data attributes that are statistically related in the underlying data. An association rule is of the form $X \rightarrow Y$ where X and Y are disjoint conjunctions of attribute-value pairs.

The traditional association rule mining problem can be described as follows: Given a database of transactions, a minimal confidence threshold and a minimal support threshold, find all association rules whose confidence and support are above the corresponding thresholds. The support of the rule is the prior probability of X and Y, $P(XandY)$. Here probability is taken to be the proportion of transactions in the data set which contain the itemset. The confidence of the rule is the conditional probability of Y given X, $P(Y|X)$ and is defined as:

$$conf(X \Rightarrow Y) = supp(X \cup Y)/supp(X)$$

To illustrate the concepts, we use a small example from the supermarket domain. The set of items is I = {milk, bread, butter}. An example rule for the supermarket could be $\{milk, bread \Rightarrow butter\}$ meaning that if milk and bread is bought, customers also buy butter. In practical applications, a rule needs a support of several hundred item sets before it can be considered statistically significant, and datasets often contain thousands or millions of item sets.

While the original goal of association rule mining is to solve market basket problem, it was used in Web data mining (Khalil et al., 2006). Let $P = p_1, p_2, ..., p_m$ be a set of pages in a Web site. Let W be a user session including a sequence of pages visited by the user in a visit, and D includes a collection of user sessions. Let A be a subsequence of W, and p_i be a page. We say that W supports A if A is a subsequence of W, and W supports (A, p_i) if (A, p_i) is a subsequence of W. The support of implication $A \Rightarrow p_i$ is $supp(A, p_i)$ and the confidence of the implication is $supp(A, P)/supp(A)$, denoted by $conf(A \Rightarrow p_i)$. Where $supp(A, p_i) = P(A, p_i)$, and $conf(A, p_i) = P(p_i|A)$.

An implication is called an association rule if its support and confidence are not less than some user specified minimum thresholds. Association rules are like classification rules, except that they can predict any attribute, not just the class.

Many algorithms for generating association rules were presented over time. *Apriori* is the best-known algorithm to mine association rules. It reduces the number of item sets that are considered. The idea is using one-item sets to generate two-item sets, two-item sets to generate three-item sets, ... according to user-specified minimal support and return frequent item set and association rules. Apriori algorithm is fast and efficient when data is sparse (most items are relatively infrequent). When data is dense (a lot of items with the same frequency) Apriori algorithm generates too much frequent item sets.

Finding Association rules can be considered as exploratory data analysis. Association or Sequence rules are not really rules, but rather descriptions of relationships in a particular database. There is no formal testing of models on other data to increase the predictive power of these rules. There is just an implicit assumption that the past behavior will continue in the future. It is often difficult to decide what to do with association rules you've discovered. Analysis and experimentation are usually required to achieve any benefit from association rules (Two Crows Corporation, 2005).

A.2 Discretization

Usually a categorical attribute takes a small number of values, so does the class label. Accordingly $p(C = c)andP(X_i = x_i|C = c)$ can be estimated with reasonable accuracy from corresponding frequencies in the training data. A numeric attribute usually has a large or even an infinite number of values, thus for any particular value $x_i, P(X_i = x_i|C = c)$ might be arbitrarily close to 0. Therefore it is often advisable to aggregate a range of values into a single value for the purpose of estimating probabilities (Yang and Webb, 2009).

The use of different discretization techniques could affect the classification bias and variance. The interval number affects significantly on the classification error. Important distinctions are missed if it is too small and if it is too big, the data are over-discretized and the probability estimation may become unreliable. The best interval number depends upon the size of the training data (Yang and Webb, 2009). Hussain et al. (1999) have proposed that there is a trade-off between the interval number and its effect on the accuracy of classification tasks. A lower number can improve understanding of an attribute but lower learning accuracy. A higher number can degrade understanding but increase learning accuracy.

In this dissertation I use two types of discretization filters presented in WEKA platform for unsupervised learning, which are 'Discretize' filter and 'PKIDiscretize' filter.

Appendix B

Software

I explored several software tools for analyzing and performing data mining techniques. This section provides a brief description of these software tools. I describe the reasons of choosing some of them in my dissertation and neglecting others.

B.1 WEKA

WEKA (Waikato Environment for Knowledge Analysis) is popular free software available under the GNU General Public License for machine learning written in Java, developed at the University of Waikato (Witten and Frank, 2005).

WEKA consists of a large number of learning schemes for classification and regression numeric prediction like decision trees, support vector machines, Bayes classifier, neural networks etc. and clustering beside the Meta classifiers like bagging and boosting, evaluation methods like cross-validation and bootstrapping, numerous attribute selection methods and preprocessing techniques. A graphical user interface provides loading of data, applying machine learning algorithms and visualizing the built models (Dimov, 2007).

WEKA input must be in ARFF (Attribute-Relation File Format). It consists of a header and data section. The first section contains metadata describing the second. It consists of all instances' attributes and their types. The second section consists of attribute values separated by commas (Dimov, 2007). Here is an example of ARFF file:

```
@relation   TrainData
@attribute   userId   46,333,537
@attribute   browserId    1, 25
@attribute   Day   1,2,3,4,5,6,7
@attribute   Class   short,long
@data
46,1,7,long
```

333,21,5,long
537,1,2,short

In addition to the native ARFF data file format, WEKA has the capability to read in '.csv' format files. The first row of the csv file contains the attribute names, separated by commas, followed by each data row with attribute values listed in the same order.

WEKA Problems:

- One of the biggest problems in WEKA is the memory issue. Most classifiers need to see all the data before they can be trained, such as J48 or SMO; hence large datasets can be a problem. WEKA offers several filters for re-sampling a dataset and generating a new dataset reduced in size. However resambling the data didn't help in my dissertation with J48 and SVM classifiers.
- Most Java virtual machines only allocate a certain maximum amount of memory to run Java programs. Usually this is much less than the amount of RAM in the computer. I extend the memory available for the virtual machine (memory heap size) by increasing it to the maximum size allowable for 32bit system, which is 1500MB. This helps with some classifiers with limited input attributes, but when I added more attributes for the inputs I face the same problem again.
- Running a filter twice, once with the train set as input and then the second time with the test set, will create almost certainly two incompatible files. The reason is that whenever a filter is running, it will get initialized based on the input data, and, of course, training and test set will differ, hence creating incompatible output.

B.2 MATLAB

MATLAB is a numerical computing environment and programming language maintained by The MathWorks, MATLAB allows easy matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs in other languages (The MathWorks, 2009).

I used MATLAB initially in building Nave Bayes Classifier for the training data in its original numeric format. This gives me a very poor result since it only gives the majority vote. I tried to apply SVM algorithm on the data but I faced memory problem since in MATLAB 7.4 and earlier, there is a limit on the number of elements in an array of $2^{31} - 2$, or approximately 2×10^9. Adding to this the fact that MATLAB is dealing only with numeric matrices, this push me to use WEKA in my approach instead of MATLAB.

B.3 ARMADA

ARMADA is a data mining tool that extracts Association Rules from numerical data files using a variety of selectable techniques and criteria (Malone, 2003).

The name ARMADA stands for 'Association Rule Miner And Deduction Analysis'. The actual knowledge extracted is presented in the form of easy-to-understand rules, while the details of the process, such as time taken and file size considered, are conveniently summarized in the mining report. The program also can allow the results to be displayed through various graphical representations, such as bar charts and line graphs. It is working under MATLAB version 5.x or greater platform. The selected file to mine must contain numerical data, separated by one of the five delimiting characters (comma, semi-colon, colon, full stop, space).

I used ARMADA under MATLAB platform and WEKA Association Rules to extract any significant information about the data and both of them gives the same results as will be described in the next chapter.

B.4 PERL

Perl is a free general-purpose programming language originally developed for text manipulation and now used for a large variety of tasks, including system administration, web development, network programming, games, and GUI development. ActivePerl is a distribution, ready-to-install package of Perl in windows platform.

I used Active Perl in writing Perl scripts to manipulate some tasks such as:

1. Converting timestamp to a readable format.
2. Identifying the class type (short or long) for the first task and produce new data files with the new format and attributes (userId, Day, Hour, and Class).
3. Filling the empty cells of the short visits with '0' to indicate the absence of second and third category for task 2 and task 3 to get equal-columns files suitable for WEKA and MATLAB.
4. Distinguishing the repeated categories of the visits path with new identifier.
5. Computing the score function for the second and third task.

B.5 Microsoft Excel 2007

Microsoft Excel is a spreadsheet-application written and distributed by Microsoft for Microsoft Windows and Mac OS X. It features calculation, graphing tools, pivot tables and a macro programming language called VBA.

While the oldest versions of Excel had a limitation in the size of their data sets, Version 12.0 (Excel 2007) can handle 1M ($2^{20} = 1048576$) rows, and ($2^{14} = 16384$) columns. This feature helps a lot in dealing with the large train and test dataset and converting the txt files to csv file, the suitable format for WEKA. Moreover it helps in reading the result buffer of WEKA classifiers and extracts the prediction output in order to calculate the score function.

Appendix C

Calculating Score Function

Here are the detailed steps of calculating score function for the second task:

1. The actual visit path vector format is $\{Cat_1, Cat_2, .., Cat_n\}$, where n is the number of visited categories on a single visit for each user. It differs from one to one. The maximum categories number in a single visit for a user in the test data is 200 and the minimum is 1. While the predicted visit path vector format is $\{Cat_1, Cat_2, Cat_3\}$.

2. Some of these categories are repeated, meaning that the user visits a specific category more than once during a single session. In this case the repeated category is considered as a new and different category and has a new category ID.

3. In the case of the predicted vector, each category can appear once or twice or three times for each visit path. So each repeated category is renamed with a new id, which multiply the categories number three times to be 60 categories instead of 20 categories only.

4. In the real vector, the maximum session has 200 visited categories in a single visit path, which means that some categories may be repeated more than three times. In this case renaming each category occurrence will increase the categories numbers a lot. Instead each repeated category is renamed for the first three occurrence and the third new category name is repeated for all the others occurrences.

5. The score function, $Scr(R)$, for the new renamed-categories version of the real vector is calculated according to the score given by the organizers (5 points for first category, 4 for second, etc...).

6. $Scr(P)$ is calculated also for the renamed-categories version of the predicted vector.

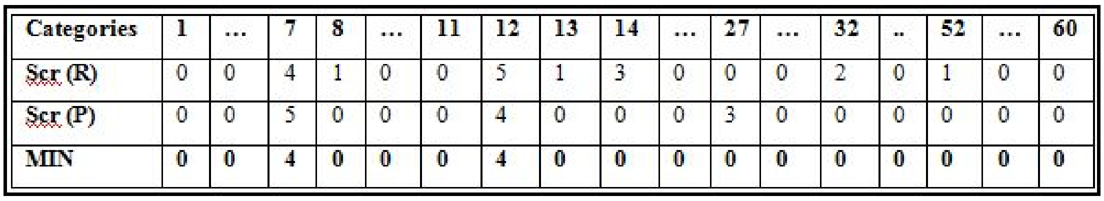
7. The minimums of both score functions $Scr(R)$ and $Scr(P)$ for each category has been chosen to create MIN vector.

8. All the elements of the MIN vector are summed up to get the final score of that visit path vector. The maximum sum of the score the best is the classifier.

Example: Suppose that the actual category vector $R = [12\ 7\ 14\ 12\ 8\ 12\ 13\ 12]$, and the predicted vector $P = [7\ 12\ 7]$. To create the new repeated-categories actual and predicted vectors, we replace the second occurrence of $category$ 12 in actual vector by $(12 + 20 = 32)$, third occurrence of the same category by $(12+40 = 52)$. Any new appearance of this category after three occurrences takes the same last identifier so the fourth occurrence is replaced by (52), so the new repeated-categories actual vector $R = [12\ 7\ 14\ 32\ 8\ 52\ 13\ 52]$.

For the predicted vector, the second occurrence of $category7$ is replaced by $(7 + 20 = 27)$, etc. So the new repeated-categories predicted vector $P = [7\ 12\ 27]$.

The score vector $SCR(R)$ for R and score vector $SCR(P)$ for P are:



Categories	1	...	7	8	...	11	12	13	14	...	27	...	32	..	52	...	60
Scr (R)	0	0	4	1	0	0	5	1	3	0	0	0	2	0	1	0	0
Scr (P)	0	0	5	0	0	0	4	0	0	0	3	0	0	0	0	0	0
MIN	**0**	**0**	**4**	**0**	**0**	**0**	**4**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Figure C.1: Score Function Example

For the actual vector, $category12$ is in the first position so its score is 5, while the repeated $category12$ comes again in the fourth position so the new identifier of it $\{32\}$ gets the score of 2. Same category is visited again in the sixth position, identified as $\{52\}$, gets the score 1. The fourth occurrence of this category in the 8th position takes the score of 1.

As well, for the predicted vector, $category7$ appears in the first place so it gets the score of 5, and when it has been visited again in the third place it is identified as $\{27\}$ and its score is 3. Then the minimums of both vectors MIN is calculated and finally all the elements of MIN vector is summed up and the score $Scr(P) = 8$.

The maximum score of the predicted vector for a single visit path is $(5 + 4 + 3 = 12)$ if all of categories are predicted correctly and the minimum is (0) if all of it predicted incorrect.

The maximum possible overall score for the predicted vector of the test data is calculated as follows:

- If the first category is predicted correctly for all instances:

$$\begin{aligned} Scr_{cat1}(P) &= the\ occurrence\ of\ first\ category \times Score\ of\ first\ position \\ &= 166,299 \times 5 \\ &= 831,495 \end{aligned}$$

- If the second category is predicted correctly for all instances:

$$\begin{aligned} Scr_{cat2}(P) &= the\ occurrence\ of\ second\ category \times Score\ of\ second\ position \\ &= 44,380 \times 4 \\ &= 177,520 \end{aligned}$$

- If the third category is predicted correctly for all instances:

$$\begin{aligned} Scr_{cat2}(P) &= the\ occurrence\ of\ the\ third\ category \times Score\ of\ third\ position \\ &= 25,503 \times 3 \\ &= 76,509 \end{aligned}$$

- $$\begin{aligned} The\ \ maximum\ \ possible\ \ score &= 831,495 + 177,520 + 76509 \\ &= 1,085,524 \end{aligned}$$

Bibliography

Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. *Data Preparation for Mining World Wide Web Browsing Patterns. Knowledge and Information Systems*, 1:5–32, 1999. URL `http://maya.cs.depaul.edu/m̃obasher/papers/webminer-kais.pdf`.

Robert Cooley, Pang ning Tan, and Jaideep Srivastava. *Discovery of Interesting Usage Patterns from Web Data.* In *WEBKDD '99: Revised Papers from the International Workshop on Web Usage Analysis and User Profiling*, pages 163–182, London, UK, 2000. Springer-Verlag. URL `http://www.cs.umn.edu/tech_reports_upload/tr1999/99-022.pdf`.

Krzysztof Dembczynski, Wojciech Kotlowski, and Marcin Sydo. *Effective Prediction of Web User Behaviour with User-Level Models. Fundam. Inf.*, 89(2-3):189–206, 2009.

Rossen Dimov. *Weka: Practical machine learning tools and techniques with Java implementations.* Technical report, University of Saarland, 2007. URL `http://www.dfki.de/k̃ipp/seminar_ws0607/reports/RossenDimov.pdf`.

Susan Dumais and Hao Chen. *Hierarchical classification of Web content.* In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263, New York, NY, USA, 2000. ACM. URL `http://research.microsoft.com/en-us/um/people/sdumais/sigir00.pdf`.

ECML/PKDD. *ECML/PKDD Discovery Challenge 2005, A Collaborative Effort in Knowledge Discovery from Databases*, 2005. URL `http://lisp.vse.cz/challenge/CURRENT/`.

ECML/PKDD. *ECML/PKDD 2007 Discovery Challenge: User's behaviour prediction*, 2007. URL `http://www.ecmlpkdd2007.org/challenge/`.

ECML/PKDD. *ECML/PKDD 2008 Discovery Challenge*, 2008. URL `http://ecmlpkdd2008.org/pastconferences`.

Magdalini Eirinaki and Michalis Vazirgiannis. *Web mining for web personalization. ACM Trans. Internet Technol.*, 3(1):1–27, 2003. URL `http://www.engr.sjsu.edu/meirinaki/papers/EV03_TOIT.pdf`.

Oren Etzioni. The world-wide web: Quagmire or gold mine? *Commun. ACM*, 39(11), 1996.

Pedro G. Ferreira. *Association And Sequence (Mining)In Web Usage Mining*. Technical report, University of Granada, n.d. URL `http://alfa.di.uminho.pt/p̃edrogabriel/papers/webMining.pdf`.

S.A. Company Gemius. *About GemiusTraffic: Web Site Usage Research*, 2007. URL `http://www.gemius.com`.

Sule Gunduz and M. Tamer Ozsu. *A Web page prediction model based on click-stream tree representation of user behavior*. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–540, New York, NY, USA, 2003. ACM. URL `http://www3.itu.edu.tr/ sgunduz/papers/kdd.pdf`.

Steve R. Gunn. *Support vector machines for classification and regression*. Technical report, Faculty of Engineering, Science and Mathematics School of Electronics and Computer Science, 1998. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.9736`.

Mark Hall1 and Eibe Frank. *Combining Naive Bayes and Decision Tables*. Technical report, University of Waikato, 2008. URL `http://www.cs.waikato.ac.nz/ẽibe/pubs/HallAndFrankFLAIRS08.pdf`.

Malik Tahir Hassan, Khurum Nazir Junejo, and Asim Karim. *Bayesian Inference for Web Surfer Behavior Prediction*. Technical report, ECM/PKDD Discovery Challenge Workshop, Warsaw, Poland, 2007. URL `http://www.ecmlpkdd2007.org/CD/DC/articles/c1p2_hassan.pdf`.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. *A practical guide to support vector classification*. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, 2003. URL `http://www.csie.ntu.edu.tw/˜cjlin/papers/guide/guide.pdf`.

Faten Khalil, Jiuyong Li, and Hua Wang. *A framework of combining Markov model with association rules for predicting web page accesses*. In *AusDM '06: Proceedings of the fifth Australasian conference on Data mining and analystics*, pages 177–184, Darlinghurst, Australia, 2006. University of Southern Queensland. URL `http://crpit.com/confpapers/CRPITV61Khalil.pdf`.

Faten Khalil, Jiuyong Li, and Hua Wang. *Integrating Markov Model with Clustering for Predicting Web Page Accesses*. conference paper, AusWeb07 : the Thirteenth Australasian Word Wide Web Conference, Toowoomba, Australia, 2007. URL `http://eprints.usq.edu.au/4026/1/Khalil_Wang_Li.pdfpaper2b.pdf`.

Ron Kohavi. *The Power of Decision Tables*. In *Proceedings of the European Conference on Machine Learning*, pages 174–189. Springer Verlag, 1995.

Ron Kohavi and Daniel Sommerfield. *Targeting Business Users with Decision Table Classifiers*, 1998.

Bing Liu, Yiming Ma, Ching K. Wong, and Philip S. Yu. *Scoring the Data Using Association Rules*. *Applied Intelligence*, 18(2):119–135, 2003. URL `http://www.cs.uic.edu/l̃iub/publications/Scoring.pdf`.

James Malone. *ARMADA: Association Rule Minor And Deduction Analysis*, 2003.

Eren Manavoglu, Dmitry Pavlov, and C. Lee Giles. *Probabilistic user behavior models*. In *In proc. Of International Conference on Data Mining)*. ICDM, 2003. URL `http://clgiles.ist.psu.edu/papers/ICDM-2003-probabilistic-user.pdf`.

Deshpande Mukund and Karypis George. *Selective Markov models for predicting Web page accesses*. *ACM Trans. Internet Technol.*, 4(2):163–184, 2004. URL `http://www.siam.org/meetings/sdm01/pdf/sdm01_04.pdf`.

Ioan Pop. *An approach of the Naive Bayes classifier for the document classification*. *General Mathematics*, 14(4):135–138, 2006. URL `http://ftp.emis.de/journals/GM/vol14nr4/pop/pop.pdf`.

Mei-Ling Shyu, Choochart Haruechaiyasak, Shu-Ching Chen, and Na Zhao. *Collaborative Filtering by Mining Association Rules from User Access Sequences*. In *WIRI '05: Proceedings of the International Workshop on Challenges in Web Information Retrieval and Integration*, pages 128–135, Washington, DC, USA, 2005. IEEE Computer Society.

Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang ning Tan. *Web Usage Mining: Discovery and applications of usage patterns from web data*. *ACM SIGKDD Explorations*, 1(2):12–23, 2000. URL `http://www.sigkdd.org/explorations/issue1-2/srivastava.pdf`.

Aixin Sun, Ee-Peng Lim, and Wee-Keong Ng. *Web classification using support vector machine*. In *WIDM '02: Proceedings of the 4th international workshop on Web information and data management*, pages 96–99, New York, NY, USA, 2002. ACM. URL `http://www.cais.ntu.edu.sg/ãxsun/paper/sun_widm02.pdf`.

Inc. The MathWorks. *MATLAB and Simulink Tutorials*, 2009. URL `http://www.mathworks.com/`.

The University of Waikato. *Weka 3: Data Mining Software in Java*, 2009. URL `http://www.cs.waikato.ac.nz/ml/weka/`.

I-Hsien Ting, Chris Kimble, and Daniel Kudenko. *UBB Mining: Finding Unexpected Browsing Behaviour in Clickstream Data to Improve a Web Site's Design*. In *WI*

'05: Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence, pages 179–185, Washington, DC, USA, 2005. IEEE Computer Society. URL `http://www.chris-kimble.com/Publications/Documents/Ting_2005a.pdf`.

Two Crows Corporation. *Introduction to Data Mining and Knowledge Discovery*. Tutorial booklet, Two Crows Corporation, MD, USA, 2005. URL `http://www.twocrows.com/intro-dm.pdf`.

Ganesan Velayathan and Seiji Yamada. Behavior based web page evaluation. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 1317–1318, New York, NY, USA, 2007. ACM. URL `http://www2007.org/htmlposters/poster1044/`.

Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition edition, 2005.

Ying Yang and Geoffrey I. Webb. *Discretization for naive-Bayes learning: managing discretization bias and variance*. *Mach. Learn.*, 74(1):39–74, 2009. ISSN 0885-6125.

Nong Ye. *The handbook of data mining*. Lawrence Erlbaum Associates, illustrated edition, 2003.