**M S Ramaiah Institute of Technology**

(An Autonomous Institute, Affiliated to VTU)

MSR nagar, MSRIT post, Bangalore-54

A Dissertation Report on

**The Repeat Shopper Prediction Analysis**

Submitted by

Manasa M 1MS12CS052

Mounica Naidu M 1MS12CS059

Nikhita Jayakumar 1MS12CS067

Rajath Jain 1MS12CS086

*in partial fulfillment for the award of the degree of*

# *Bachelor of Engineering in Computer Science & Engineering*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**M.S.RAMAIAH INSTITUTE OF TECHNOLOGY**

**(Autonomous Institute, Affiliated to VTU)**

**BANGALORE-560054**

[www.msrit.edu](http://www.msrit.edu), **May 2015**

# Abstract

Our main aim is to predict which shoppers will become repeat buyers.

Consumer brands often offer discounts to attract new shoppers to buy their products. The most valuable customers are those who return after this initial incented purchase.  With enough purchase history, it is possible to predict which shoppers, when presented an offer, will buy a new item. However, identifying the shopper who will become a loyal buyer prior to the initial purchase is the challenging task.

The Repeat Shopper Prediction Analysis asks participants involves in predicting which shoppers are most likely to repeat the purchase. To aid with algorithmic development, a complete, basket-level, pre-offer shopping history for a large set of shoppers is extracted. The incentive offered to that shopper and their post-incentive behavior is also observed.

#### Contents

***Declaration i***

***Acknowledgements ii***

***Abstract iii***

***List of Figures***

***List of Tables***

1 **INTRODUCTION**

* 1. General Introduction……………….
  2. Statement of the Problem…………..
  3. Objectives of the project……………
  4. Project deliverables……………
  5. Current Scope………………………
  6. Future Scope……………………….

1. **PROJECT ORGANIZATION**
   1. Software Process Models
   2. Roles and Responsibilities
2. **LITERATURE SURVEY**

3.1….Introduction

3.2…Main Body

3.3 Conclusion of Survey

1. **SOFTWARE REQUIREMENT SPECIFICATIONS**

4.1 External Interface Requirements

4.1.1 User Interfaces

4.1.2 Hardware Interfaces

4.1.3 Software Interfaces

4.2 Functional Requirements

4.3 Software System Attributes

4.3.1 Reliability

4.3.2 Availability

4.3.3 Security

4.3.4 Portability

4.3.5 Maintainability

4.3.6 Performance

4.4 Performance Requirements

4.5 Design Constraints

4.6 Other Requirements

1. **DESIGN**
   1. Introduction
   2. Data flow diagram

5.3 Sequence Diagram

1. **IMPLEMENTATION**
   1. Tools Introduction
   2. Technology Introduction
   3. Overall view of the project in terms of implementation
   4. Code
2. **TESTING**

7.1 Results and Snapshots

1. **CONCLUSION & SCOPE FOR FUTURE WORK**
2. **REFERENCES**

**1. INRODUCTION**

* 1. **General Introduction:**

**Data analytics** (DA) is the science of examining raw data with the purpose of drawing conclusions about that information. It is used in many industries to allow companies and organization to make better business decisions and in the sciences to verify or disprove existing models or theories.

Reference to qualitative and quantitative techniques and processes used to enhance productivity and business gain. Data is extracted and categorized to identify and analyze behavioral data and patterns, and techniques vary according to organizational requirements.

**1.2 Statement of the Problem:**

The Repeat Shopper Prediction Analysis involves the prediction on which shoppers are most likely to repeat their purchase. The most important customers are those who return after this initial incented purchase. With algorithmic development, the complete, basket-level, pre-offer shopping history for a large set of shoppers is given to us. The incentive offered to that shopper and their post-incentive behavior is also observed.

**1.3 Objectives of the project:**

Our project objective is to understand which shoppers are most likely to repeat a purchase based on our prediction analysis technique used. The more times a customer purchases items, the stronger the relationship becomes. It’s these strong producer-consumer relationships that cause customers to champion your products and services, effectively creating powerful referrals and word-of-mouth marketing that will help drive new customers.

* 1. **Project deliverables:**

The Revisiting Shopper Prediction Analysis predicts which shoppers are most likely to repeat purchase. To create this prediction, you are given a minimum of a year of shopping history prior to each customer's incentive, as well as the purchase histories of many other shoppers.

* Customers who are most likely to purchase again
* The brand which a revisiting customer may choose

**1.5 Current Scope**

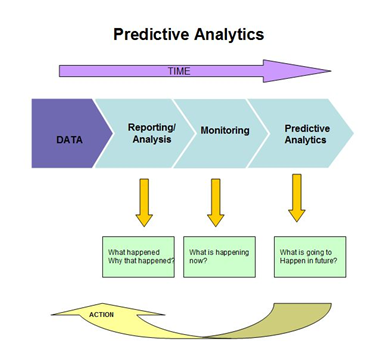
In the present study, some of the data mining techniques have been used effectively to classify the customers of a retail shopping store and then to describe the characteristics of the derived customer segments. Further analysis has been done in order to reduce the store attributes to a manageable level and to understand the customer perception of the stores image. Not only does it help to reduce the number of attributes but also gives the values which can finally be used to develop the predictive model for store loyalty through multiple regression.

**1.6 Future Scope**

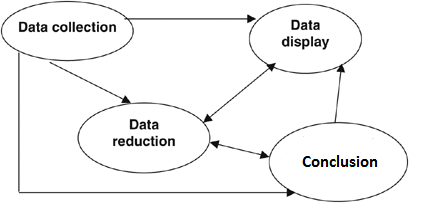
In terms of future scope, a variety of data mining techniques can be used by researchers to simplify customer perceptions and attitudes. Every day, every hour and every minute data gets generated from millions of shoppers, yet, retail managers/ business executives always grapple with relevant information that can help retailers/ researchers design strategies to generate customer loyalty. Utilization of this data is done to generate certain knowledge that can help them in modeling and predicting customer behavior and further in order to know their customers better. Data mining can be applied through a variety of other techniques such as concept description, cluster analysis, classification-prediction, association analysis and evolution analysis.

1. **PROJECT ORGANIZATION**
   1. **Software Process Models:**

**Predictive analysis model:**

****

**Iterative data analytics model:**

****

**2.2 Roles and Responsibilities:**

Our group members analyzed and worked on the provided dataset.

The Repeat Shopper Prediction Analysis was then followed by the data reduction and normalization technique, where we successfully reduced our dataset and normalized them. i.e., the unwanted rows were discarded from the transactions data and for the minimization of data redundancy.

The team worked with VMWare Workstation using Ubuntu and also worked on Mac terminal and managed to analyze the frequency of the repeat buyers/customers for a certain given category of items.

The key purpose of this project was to understand the shoppers loyalty to the brand/company of the products made available.

We were able to execute the code via python programming using R tool as well Vowpal Wabbit.

The results were tabulated and understood by our team and the required analysis was made.

1. **LITERATURE SURVEY**

As per The revisiting shopper prediction analysis the shoppers frequency for purchase of the required items. The main aim of this prediction analysis is to understand the customer’s loyalty and to predict their next purchase.

With the increasing internet literacy, the prospect of online marketing is increasing on a large scale. The consumers indulging in online shopping consider many factors If companies analyses the factors affecting consumer behavior towards online shopping and the relationships between these factors and the type of online buyers, then they can devise effective marketing strategies to convert potential customers into active ones, while retaining existing online customers.

This project is a part of study, and focuses on factors which buyers keep in mind while shopping online. This research found that information, perceived usefulness, ease of use; perceived enjoyment and security/privacy are the five dominant factors which influence consumer perceptions of online purchasing.

Testing of the hypotheses consists of two steps. We first estimate a demand model of purchase incidence and brand choice to get coefficients on brand loyalty, size loyalty and price for the online and offline channels. Importantly, we use store visits and purchases in both channels and estimate the online and offline coefficients jointly in the same demand model. We then compute elasticity’s for brand loyalty, size loyalty and price sensitivity. Next, we regress the online and offline elasticity’s on household and product characteristics to see whether there exist systematic differences across households and products

The finding that product characteristics influence consumer behavior in the online store differently from the offline stores should help manufacturers and retailers tailor their marketing strategies to different channels.

Vladimir Nikulin from Department of Mathematical Methods in Economy, Vyatka State University, Kirov, Russia in his paper has discussed On the Method for Data Streams Aggregation to Predict Shoppers Loyalty.

The consumer is usually not sure how much he will use a service in the future (e.g. mobile phones or credit cards), but nevertheless, he will frequently renew subscriptions to these services. The factors that affect consumer/supplier relationships and customer retention have been studied (White & Yanamandram, 2007). Understanding how consumers respond to a supplier’s offering of reinforcements and utilities in the contractual behaviour context will define the main factors that maintain retention behaviour (Beckett et al., 2000; Choi et al., 2006; Pearlman, 2007)

**4. SOFTWARE REQUIREMENT SPECIFICATIONS**

**4.1 External Interface Requirements:**

**4.1.1 User Interfaces**

The user interface we are using for our application involves the project’s output being displayed on the console. The output containing the repeat shopper analysis will be displayed on the screen based on the data that has been provided.

**4.1.2 Hardware Interfaces**

Intel processor and minimum 1GB RAM and a minimum of 22 GB memory to store the data files is being used to perform the data analysis.

**4.1.3 Software Interfaces**

OS X El Capitan

The Ubuntu 14.04 LTS is used for integrating data.

VMWare Workstation

Vowpal-Wabbit

R tool

Python

**4.2 Functional Requirements**

**a. Retrieving Input:**

We have provided complete, basket-level; pre-offer shopping history for a large set of shoppers who were targeted for an offer campaign. A year of shopping history prior to each customer's incentive, as well as the purchase histories of many other shoppers. The transaction history contains all items purchased, not just items related to the offer. Only one offer per customer is included in the data. The training set is comprised of offers issued before 2013-05-01. The test set is offers issued on or after 2013-05-01.

**b. Processing of data:**

Inputting datasets, this will be processed and reduced using data reduction algorithms and techniques.

**c. Shopper Analysis:**

From the data that has been extracted the analysis is done and with the use of R programming various statistics of the data can be extracted.

**d. Output:**

The output shows for each customer (id) in testHistory.csv, predicting a probability that the customer repeat-purchased the product from the promotion they received.

**4.3 Software System Attributes**

4.3.1 **Reliability**:

The software will meet all of the functional requirements without any unexpected behavior. At no time should the output display incorrect or outdated information without alerting the user of potential errors.

4.3.2 **Availability:**

The software reviewing the data will be available at all times, as long as the device we are using is in proper working condition.

4.4.3 Security: The software will be secure without any chances of data getting corrupted or data being lost.

4.3.4 **Maintainability**:

The software should be written clearly. The code should be well documented. Care should be taken to design the software modularly to ensure that easy maintenance.

4.3.5 **Portability**:

This software will run on any Linux based system which supports the required software’s and tools.

4.3.6 **Performance:**

The system must be interactive and the delays involved must be less. The delay in the system increases with the amount of data sets which we are to be read and processed.

**4.4 Performance Requirements**

**Real-Time**: The software will provide up-to-date information, limited only by the rate of Twitter input. The gauge output should display the latest results at all times, and if it lags behind, the user should be notified.

**System Resource Consumption**: Resource consumption of this application should not reach an amount that renders the mobile device unusable. The application should be capable of operating in the background should the user wish to utilize other applications.

**4.5 Design Constraints**

The API has some limitations such as it does not accept large data and we need to modify our code to accommodate the large data.

**4.6 Other Requirements**

All the requirements are mentioned for the working of the application, and no more requirements are needed.

**5. DESIGN**

**5.1 Introduction:**

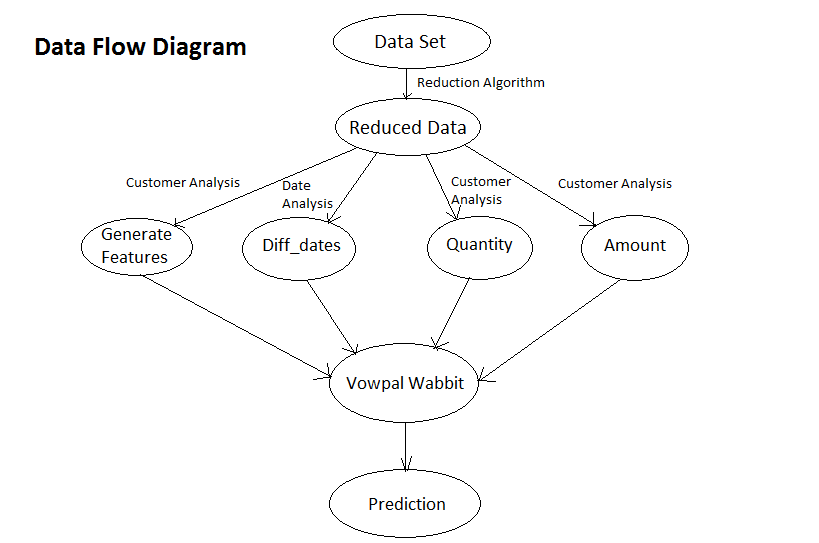
Design in Data Analytics is required to understand the process of working on the desired data sets. The key features focused on are as follows:

Data Exploration-Getting the feel of the data and understanding what outcome is require from it.

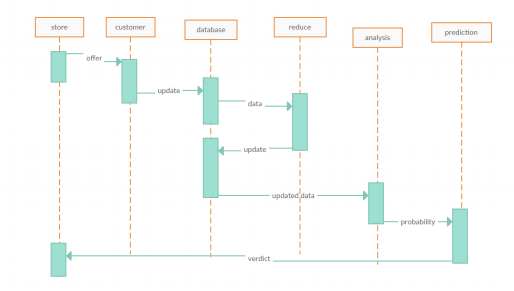
Data Validation- Validating the chosen data to be precise and consistent.

Data Sanitization- If inaccuracies arise, how do we tackle those issues**?**

**5.2 Data flow diagram:**



**5.3 Sequence Diagram**

****

**6. IMPLEMENTATION**

* 1. **Tools Introduction**

**R Tool:**

R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language provides a wide variety of statistical (linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering) and graphical techniques, and is highly extensible. R provides an Open Source route to participation in that activity. One of R’s strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. It is available as Free Software in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and Mac OS.

**Python:**

Python supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming) and [functional programming](https://en.wikipedia.org/wiki/Functional_programming) or [procedural](https://en.wikipedia.org/wiki/Procedural_programming) styles. It features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management) and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library)

This package allows the user to call [Python](http://www.python.org) from R. rPython is intended for running Python code from R. R programs and packages can:

* Pass data to Python: vectors of various types (logical, character, numeric), lists, etc.
* Get data from Python.
* Call Python code, call Python functions and methods.

**Vowpal Wabbit:**

It is an open source fast out-of-core learning system library and program. Vowpal Wabbit is notable as an efficient scalable implementation of [online machine learning](https://en.wikipedia.org/wiki/Online_machine_learning) and support for a number of [machine learning reductions](https://en.wikipedia.org/w/index.php?title=Machine_learning_reductions&action=edit&redlink=1), importance weighting, and a selection of different [loss functions](https://en.wikipedia.org/wiki/Loss_function) and optimization algorithms.

 VW supports online learning and optimization by default. Online learning is an old approach which is becoming much more common. Various alterations to standard [stochastic gradient descent](http://en.wikipedia.org/wiki/Stochastic_gradient_descent?WT.mc_id=Blog_MachLearn_General_DI) make the default rule more robust across many datasets, and [progressive validation](http://hunch.net/%7Ejl/projects/prediction_bounds/progressive_validation/coltfinal.pdf?WT.mc_id=Blog_MachLearn_General_DI) allows debugging learning applications in sub-linear time.

 VW does [Feature Hashing](http://en.wikipedia.org/wiki/Feature_hashing?WT.mc_id=Blog_MachLearn_General_DI) which allows learning from dramatically rawer representations, reducing the need to preprocess data, speeding execution, and sometimes even improving accuracy.

The conjunction of online learning and feature hashing imply the ability to learn from any amount of information via network streaming.  This makes the system a reliable baseline tool.

 VW has also been parallelized to be the most scalable public ML algorithm.

 VW has a reduction stack which allows the basic core learning algorithm to address many advanced problem types such as cost-sensitive multiclass classification. Some of these advanced problem types, such as for interactive learning exist only in VW

* 1. **Technology Introduction:**

**Ubuntu Operating System:** Ubuntu Linux operating system and distribution, with Unity as its default desktop environment for personal computers including smart phones in later versions.

**VMware Workstation with Ubuntu:** is a hypervisor that runs on x64 computers;it enables users to set up one or more virtual machines (VMs) on a single physical machine, and use them simultaneously along with the actual machine.

* 1. **Overall view of the project in terms of implementation:**

**Data sets (test & train data sets**): Test data is data which has been specifically identified for use in tests, typically of a computer program. Some data may be used in a confirmatory way, typically to verify that a given set of input to a given function produces some expected result. Training data is the data on which the machine learning programs learn to perform co relational tasks (classify, cluster, learn the attributes).

## Data reduction

The offers.csv file has all the categories and companies a coupon offer can have. The rows can be discarded from the transactions data which don’t have a category id or a company id which is on offer.

The function reduce\_data () in the messy code can be used. It runs in about 5-10 minutes and will reduce it further. This makes our future model code more manageable (around 1.6GB).

**Feature Engineering**

A large part of our project is feature engineering: Creating good indicative features from the purchase history. From the benchmark we already have 4 features: Has bought in the coupon offer category before, has bought the brand before, has bought from the company before and has bought company + category + brand on offer.

Putting these into a binary feature (1 or 0), but we have all the transaction data; we can count how many times someone has bought inside a category.

### Has bought from company on offer

Feature: has\_bought\_company where we count how many times the shopper has bought a product from the company on offer.

Related feature: has\_bought\_company\_q which holds the quantity bought (sometimes shoppers buy multiple items at once).

Another feature that counts the total amount spent on a company on offer: has\_bought\_company\_a.

We also generate features that count the days between the previous purchases and the date of the coupon offer. So if for instance the shopper spend 50$ on a company in the last 3 months we would set has\_bought\_company\_a\_90 to 50. We generate these features for last 30 days, last 60, last 90 and last 180 days.

If the shopper has never bought a product from a company on the coupon offer then we generate a negative feature: has\_not\_bought\_company.

### Has bought from category on the coupon offer

This is basically the same as above, only for the category. We also generate features for date ranges and generate negative features if the shopper has never bought from the category on offer.

### Has bought brand on the coupon offer

We check if the user has bought the brand before that is on the coupon offer. We then generate the same features as above.

### Combinations of brand, category and company on offer

If the shopper has bought from the brand, category and company before we generate a specific feature for that. Also for individual combinations like brand + company. And we again generate negative features, like: has\_not\_bought\_brand\_company.

### The offer value and offer quantity

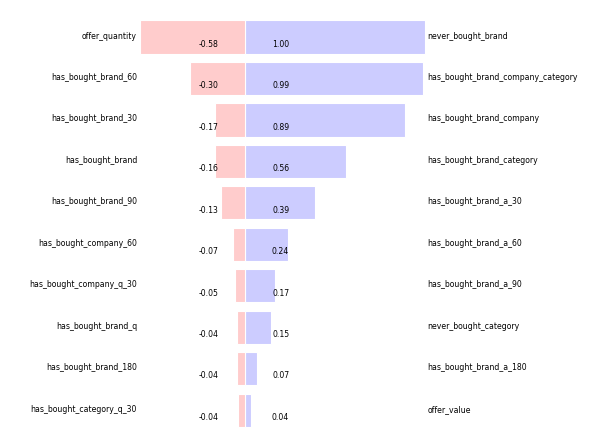
This is a constant for every offer. The offer value might influence the number of repeat buyers. The offer value fluctuates between about 5 and 0.75.

We also get the offer quantity (how many items can be redeemed with the coupon). Offer\_quantity is always 1 in the train data.

### Total shopper spend

We are interested to see how much the shopper spend. We hope this won’t change too much from the original data set, if we only count the amounts from the reduced data set. For every transaction still in the reduced data set we take the amount and add it all up. We think that total shopper spend will influence future chance of repeat buys.

**GRAPHICAL REPRESENTATION:**

****

* 1. **Code:**

**1)**

|  |
| --- |
| """ |
| from datetime import datetime, date | |
|  | |
|  | |
| from collections import defaultdict | |
|  | |
|  | |
|  | |
|  | |
|  | |
| loc\_offers = "data/offers.csv" | |
|  | |
|  | |
| loc\_transactions = "data/transactions.csv" | |
|  | |
|  | |
| loc\_train = "data/trainHistory.csv" | |
|  | |
|  | |
| loc\_test = "data/testHistory.csv" | |
|  | |
|  | |
|  | |
|  | |
|  | |
| # will be created | |
|  | |
|  | |
| loc\_reduced = "data/reduced.csv" | |
|  | |
|  | |
| loc\_out\_train = "data/train.vw" | |
|  | |
|  | |
| loc\_out\_test = "data/test.vw" | |
|  | |
|  | |
|  | |
|  | |
|  | |
| ### | |
|  | |
|  | |
|  | |
|  | |
|  | |
| def reduce\_data(loc\_offers, loc\_transactions, loc\_reduced): | |
|  | |
|  | |
| start = datetime.now() | |
|  | |
|  | |
| #get all categories and comps on offer in a dict | |
|  | |
|  | |
| offers\_cat = {} | |
|  | |
|  | |
| offers\_co = {} | |
|  | |
|  | |
| for e, line in enumerate( open(loc\_offers) ): | |
|  | |
|  | |
| offers\_cat[ line.split(",")[1] ] = 1 | |
|  | |
|  | |
| offers\_co[ line.split(",")[3] ] = 1 | |
|  | |
|  | |
| #open output file | |
|  | |
|  | |
| with open(loc\_reduced, "wb") as outfile: | |
|  | |
|  | |
| #go through transactions file and reduce | |
|  | |
|  | |
| reduced = 0 | |
|  | |
|  | |
| for e, line in enumerate( open(loc\_transactions) ): | |
|  | |
|  | |
| if e == 0: | |
|  | |
|  | |
| outfile.write( line ) #print header | |
|  | |
|  | |
| else: | |
|  | |
|  | |
| #only write when if category in offers dict | |
|  | |
|  | |
| if line.split(",")[3] in offers\_cat or line.split(",")[4] in offers\_co: | |
|  | |
|  | |
| outfile.write( line ) | |
|  | |
|  | |
| reduced += 1 | |
|  | |
|  | |
| #progress | |
|  | |
|  | |
| if e % 5000000 == 0: | |
|  | |
|  | |
| print e, reduced, datetime.now() - start | |
|  | |
|  | |
| print e, reduced, datetime.now() - start | |
|  | |
|  | |
|  | |
|  | |
|  | |
| #reduce\_data(loc\_offers, loc\_transactions, loc\_reduced) | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
| def diff\_days(s1,s2): | |
|  | |
|  | |
| date\_format = "%Y-%m-%d" | |
|  | |
|  | |
| a = datetime.strptime(s1, date\_format) | |
|  | |
|  | |
| b = datetime.strptime(s2, date\_format) | |
|  | |
|  | |
| delta = b - a | |
|  | |
|  | |
| return delta.days | |
|  | |
|  | |
|  | |
|  | |
|  | |
| def generate\_features(loc\_train, loc\_test, loc\_transactions, loc\_out\_train, loc\_out\_test): | |
|  | |
|  | |
| #keep a dictionary with the offerdata | |
|  | |
|  | |
| offers = {} | |
|  | |
|  | |
| for e, line in enumerate( open(loc\_offers) ): | |
|  | |
|  | |
| row = line.strip().split(",") | |
|  | |
|  | |
| offers[ row[0] ] = row | |
|  | |
|  | |
|  | |
|  | |
|  | |
| #keep two dictionaries with the shopper id's from test and train | |
|  | |
|  | |
| train\_ids = {} | |
|  | |
|  | |
| test\_ids = {} | |
|  | |
|  | |
| for e, line in enumerate( open(loc\_train) ): | |
|  | |
|  | |
| if e > 0: | |
|  | |
|  | |
| row = line.strip().split(",") | |
|  | |
|  | |
| train\_ids[row[0]] = row | |
|  | |
|  | |
| for e, line in enumerate( open(loc\_test) ): | |
|  | |
|  | |
| if e > 0: | |
|  | |
|  | |
| row = line.strip().split(",") | |
|  | |
|  | |
| test\_ids[row[0]] = row | |
|  | |
|  | |
| #open two output files | |
|  | |
|  | |
| with open(loc\_out\_train, "wb") as out\_train, open(loc\_out\_test, "wb") as out\_test: | |
|  | |
|  | |
| #iterate through reduced dataset | |
|  | |
|  | |
| last\_id = 0 | |
|  | |
|  | |
| features = defaultdict(float) | |
|  | |
|  | |
| for e, line in enumerate( open(loc\_transactions) ): | |
|  | |
|  | |
| if e > 0: #skip header | |
|  | |
|  | |
| #poor man's csv reader | |
|  | |
|  | |
| row = line.strip().split(",") | |
|  | |
|  | |
| #write away the features when we get to a new shopper id | |
|  | |
|  | |
| if last\_id != row[0] and e != 1: | |
|  | |
|  | |
|  | |
|  | |
|  | |
| #generate negative features | |
|  | |
|  | |
| if "has\_bought\_company" not in features: | |
|  | |
|  | |
| features['never\_bought\_company'] = 1 | |
|  | |
|  | |
|  | |
|  | |
|  | |
| if "has\_bought\_category" not in features: | |
|  | |
|  | |
| features['never\_bought\_category'] = 1 | |
|  | |
|  | |
|  | |
|  | |
|  | |
| if "has\_bought\_brand" not in features: | |
|  | |
|  | |
| features['never\_bought\_brand'] = 1 | |
|  | |
|  | |
|  | |
|  | |
|  | |
| if "has\_bought\_brand" in features and "has\_bought\_category" in features and "has\_bought\_company" in features: | |
|  | |
|  | |
| features['has\_bought\_brand\_company\_category'] = 1 | |
|  | |
|  | |
|  | |
|  | |
|  | |
| if "has\_bought\_brand" in features and "has\_bought\_category" in features: | |
|  | |
|  | |
| features['has\_bought\_brand\_category'] = 1 | |
|  | |
|  | |
|  | |
|  | |
|  | |
| if "has\_bought\_brand" in features and "has\_bought\_company" in features: | |
|  | |
|  | |
| features['has\_bought\_brand\_company'] = 1 | |
|  | |
|  | |
|  | |
|  | |
|  | |
| outline = "" | |
|  | |
|  | |
| test = False | |
|  | |
|  | |
| for k, v in features.items(): | |
|  | |
|  | |
|  | |
|  | |
|  | |
| if k == "label" and v == 0.5: | |
|  | |
|  | |
| #test | |
|  | |
|  | |
| outline = "1 '" + last\_id + " |f" + outline | |
|  | |
|  | |
| test = True | |
|  | |
|  | |
| elif k == "label": | |
|  | |
|  | |
| outline = str(v) + " '" + last\_id + " |f" + outline | |
|  | |
|  | |
| else: | |
|  | |
|  | |
| outline += " " + k+":"+str(v) | |
|  | |
|  | |
| outline += "\n" | |
|  | |
|  | |
| if test: | |
|  | |
|  | |
| out\_test.write( outline ) | |
|  | |
|  | |
| else: | |
|  | |
|  | |
| out\_train.write( outline ) | |
|  | |
|  | |
| #print "Writing features or storing them in an array" | |
|  | |
|  | |
| #reset features | |
|  | |
|  | |
| features = defaultdict(float) | |
|  | |
|  | |
| #generate features from transaction record | |
|  | |
|  | |
| #check if we have a test sample or train sample | |
|  | |
|  | |
| if row[0] in train\_ids or row[0] in test\_ids: | |
|  | |
|  | |
| #generate label and history | |
|  | |
|  | |
| if row[0] in train\_ids: | |
|  | |
|  | |
| history = train\_ids[row[0]] | |
|  | |
|  | |
| if train\_ids[row[0]][5] == "t": | |
|  | |
|  | |
| features['label'] = 1 | |
|  | |
|  | |
| else: | |
|  | |
|  | |
| features['label'] = 0 | |
|  | |
|  | |
| else: | |
|  | |
|  | |
| history = test\_ids[row[0]] | |
|  | |
|  | |
| features['label'] = 0.5 | |
|  | |
|  | |
|  | |
|  | |
|  | |
| #print "label", label | |
|  | |
|  | |
| #print "trainhistory", train\_ids[row[0]] | |
|  | |
|  | |
| #print "transaction", row | |
|  | |
|  | |
| #print "offers", offers[ train\_ids[row[0]][2] ] | |
|  | |
|  | |
| #print | |
|  | |
|  | |
|  | |
|  | |
|  | |
| features['offer\_value'] = offers[ history[2] ][4] | |
|  | |
|  | |
| features['offer\_quantity'] = offers[ history[2] ][2] | |
|  | |
|  | |
| offervalue = offers[ history[2] ][4] | |
|  | |
|  | |
|  | |
|  | |
|  | |
| features['total\_spend'] += float( row[10] ) | |
|  | |
|  | |
|  | |
|  | |
|  | |
| if offers[ history[2] ][3] == row[4]: | |
|  | |
|  | |
| features['has\_bought\_company'] += 1.0 | |
|  | |
|  | |
| features['has\_bought\_company\_q'] += float( row[9] ) | |
|  | |
|  | |
| features['has\_bought\_company\_a'] += float( row[10] ) | |
|  | |
|  | |
|  | |
|  | |
|  | |
| date\_diff\_days = diff\_days(row[6],history[-1]) | |
|  | |
|  | |
| if date\_diff\_days < 30: | |
|  | |
|  | |
| features['has\_bought\_company\_30'] += 1.0 | |
|  | |
|  | |
| features['has\_bought\_company\_q\_30'] += float( row[9] ) | |
|  | |
|  | |
| features['has\_bought\_company\_a\_30'] += float( row[10] ) | |
|  | |
|  | |
| if date\_diff\_days < 60: | |
|  | |
|  | |
| features['has\_bought\_company\_60'] += 1.0 | |
|  | |
|  | |
| features['has\_bought\_company\_q\_60'] += float( row[9] ) | |
|  | |
|  | |
| features['has\_bought\_company\_a\_60'] += float( row[10] ) | |
|  | |
|  | |
| if date\_diff\_days < 90: | |
|  | |
|  | |
| features['has\_bought\_company\_90'] += 1.0 | |
|  | |
|  | |
| features['has\_bought\_company\_q\_90'] += float( row[9] ) | |
|  | |
|  | |
| features['has\_bought\_company\_a\_90'] += float( row[10] ) | |
|  | |
|  | |
| if date\_diff\_days < 180: | |
|  | |
|  | |
| features['has\_bought\_company\_180'] += 1.0 | |
|  | |
|  | |
| features['has\_bought\_company\_q\_180'] += float( row[9] ) | |
|  | |
|  | |
| features['has\_bought\_company\_a\_180'] += float( row[10] ) | |
|  | |
|  | |
|  | |
|  | |
|  | |
| if offers[ history[2] ][1] == row[3]: | |
|  | |
|  | |
|  | |
|  | |
|  | |
| features['has\_bought\_category'] += 1.0 | |
|  | |
|  | |
| features['has\_bought\_category\_q'] += float( row[9] ) | |
|  | |
|  | |
| features['has\_bought\_category\_a'] += float( row[10] ) | |
|  | |
|  | |
| date\_diff\_days = diff\_days(row[6],history[-1]) | |
|  | |
|  | |
| if date\_diff\_days < 30: | |
|  | |
|  | |
| features['has\_bought\_category\_30'] += 1.0 | |
|  | |
|  | |
| features['has\_bought\_category\_q\_30'] += float( row[9] ) | |
|  | |
|  | |
| features['has\_bought\_category\_a\_30'] += float( row[10] ) | |
|  | |
|  | |
| if date\_diff\_days < 60: | |
|  | |
|  | |
| features['has\_bought\_category\_60'] += 1.0 | |
|  | |
|  | |
| features['has\_bought\_category\_q\_60'] += float( row[9] ) | |
|  | |
|  | |
| features['has\_bought\_category\_a\_60'] += float( row[10] ) | |
|  | |
|  | |
| if date\_diff\_days < 90: | |
|  | |
|  | |
| features['has\_bought\_category\_90'] += 1.0 | |
|  | |
|  | |
| features['has\_bought\_category\_q\_90'] += float( row[9] ) | |
|  | |
|  | |
| features['has\_bought\_category\_a\_90'] += float( row[10] ) | |
|  | |
|  | |
| if date\_diff\_days < 180: | |
|  | |
|  | |
| features['has\_bought\_category\_180'] += 1.0 | |
|  | |
|  | |
| features['has\_bought\_category\_q\_180'] += float( row[9] ) | |
|  | |
|  | |
| features['has\_bought\_category\_a\_180'] += float( row[10] ) | |
|  | |
|  | |
| if offers[ history[2] ][5] == row[5]: | |
|  | |
|  | |
| features['has\_bought\_brand'] += 1.0 | |
|  | |
|  | |
| features['has\_bought\_brand\_q'] += float( row[9] ) | |
|  | |
|  | |
| features['has\_bought\_brand\_a'] += float( row[10] ) | |
|  | |
|  | |
| date\_diff\_days = diff\_days(row[6],history[-1]) | |
|  | |
|  | |
| if date\_diff\_days < 30: | |
|  | |
|  | |
| features['has\_bought\_brand\_30'] += 1.0 | |
|  | |
|  | |
| features['has\_bought\_brand\_q\_30'] += float( row[9] ) | |
|  | |
|  | |
| features['has\_bought\_brand\_a\_30'] += float( row[10] ) | |
|  | |
|  | |
| if date\_diff\_days < 60: | |
|  | |
|  | |
| features['has\_bought\_brand\_60'] += 1.0 | |
|  | |
|  | |
| features['has\_bought\_brand\_q\_60'] += float( row[9] ) | |
|  | |
|  | |
| features['has\_bought\_brand\_a\_60'] += float( row[10] ) | |
|  | |
|  | |
| if date\_diff\_days < 90: | |
|  | |
|  | |
| features['has\_bought\_brand\_90'] += 1.0 | |
|  | |
|  | |
| features['has\_bought\_brand\_q\_90'] += float( row[9] ) | |
|  | |
|  | |
| features['has\_bought\_brand\_a\_90'] += float( row[10] ) | |
|  | |
|  | |
| if date\_diff\_days < 180: | |
|  | |
|  | |
| features['has\_bought\_brand\_180'] += 1.0 | |
|  | |
|  | |
| features['has\_bought\_brand\_q\_180'] += float( row[9] ) | |
|  | |
|  | |
| features['has\_bought\_brand\_a\_180'] += float( row[10] ) | |
|  | |
|  | |
| last\_id = row[0] | |
|  | |
|  | |
| if e % 100000 == 0: | |
|  | |
|  | |
| print e | |
|  | |
|  | |
|  | |
|  | |
|  | |
| #generate\_features(loc\_train, loc\_test, loc\_transactions, loc\_out\_train, loc\_out\_test) | |
|  | |
| if \_\_name\_\_ == '\_\_main\_\_': | |
|  | |
|  | |
| reduce\_data(loc\_offers, loc\_transactions, loc\_reduced) | |
|  | |

|  |
| --- |
| generate\_features(loc\_train, loc\_test, loc\_reduced, loc\_out\_train, loc\_out\_test) |

1. **TESTING**
   1. **Results and Snapshots:**

### Training

Now we run Vowpal Wabbit (version 7.1) and train a model with our train set.

vw train.vw -c -k --passes 40 -l 0.85 -f shop.model.vw --loss\_function quantile --quantile\_tau 0.6

Where:

* -c -k --passes 40 says to use a cache, kill any previous cache and run 40 passes
* -l 0.85 sets the learning rate to 0.85
* -f shop.model.vw saves the model
* --loss\_function quantile says to use quantile regression
* --quantile\_tau 0.6 is a parameter to tweak when using the quantile loss function.

We get an average loss of 0.149688.

### Testing

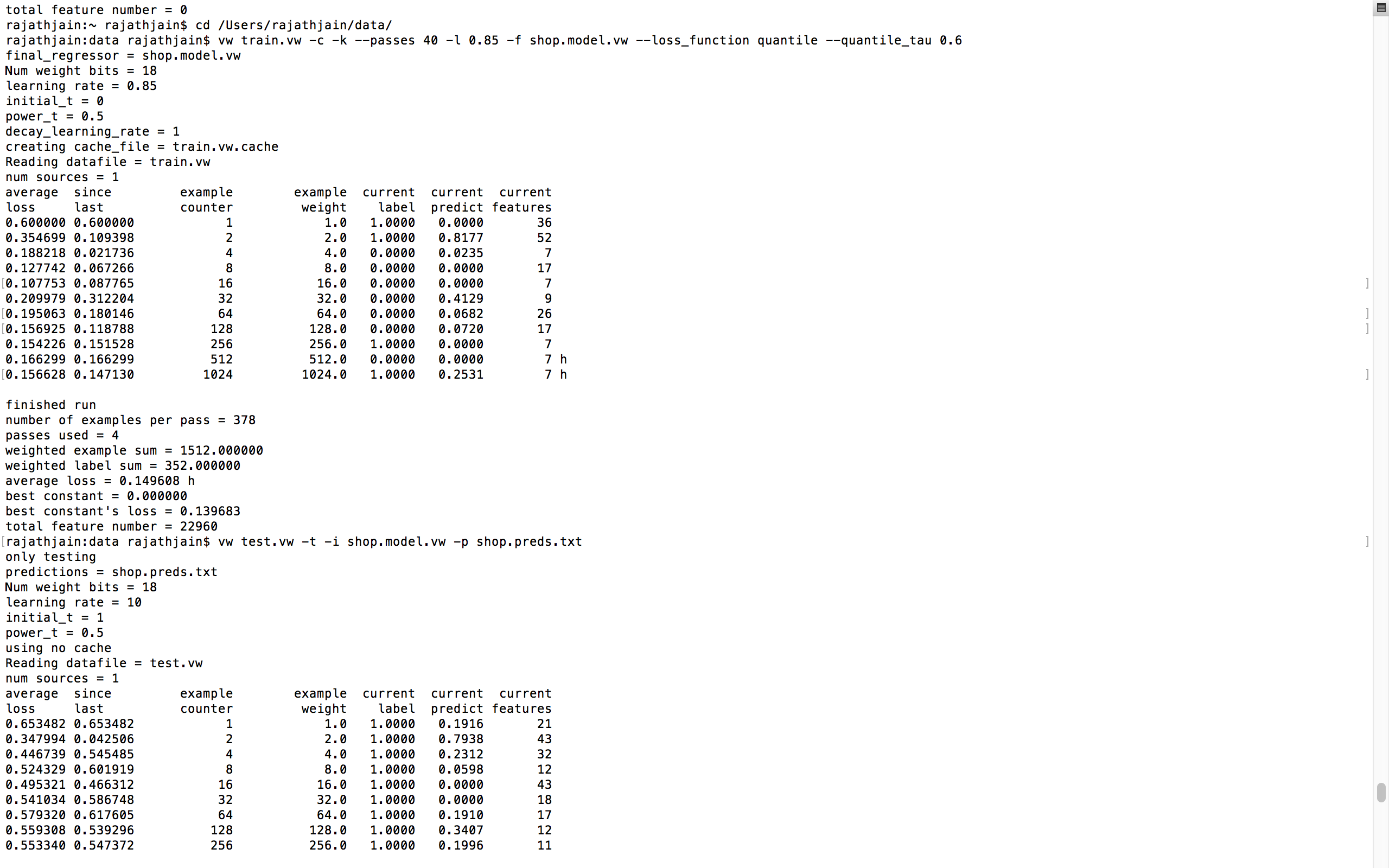
Now we use the model and the train set to get us predictions:

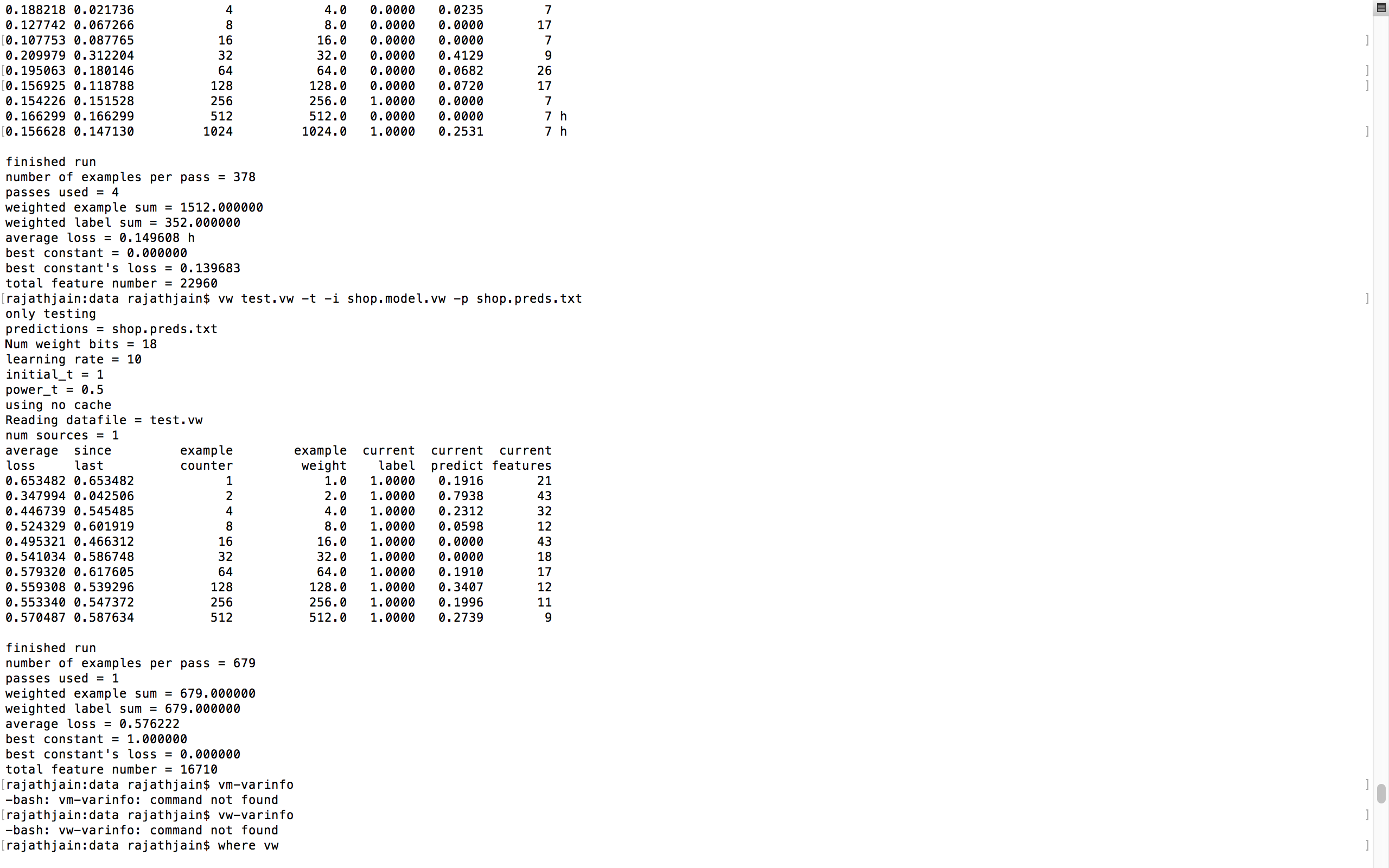
vw shop.test.vw -t -i shop.model.vw -p shop.preds.txt

Where:

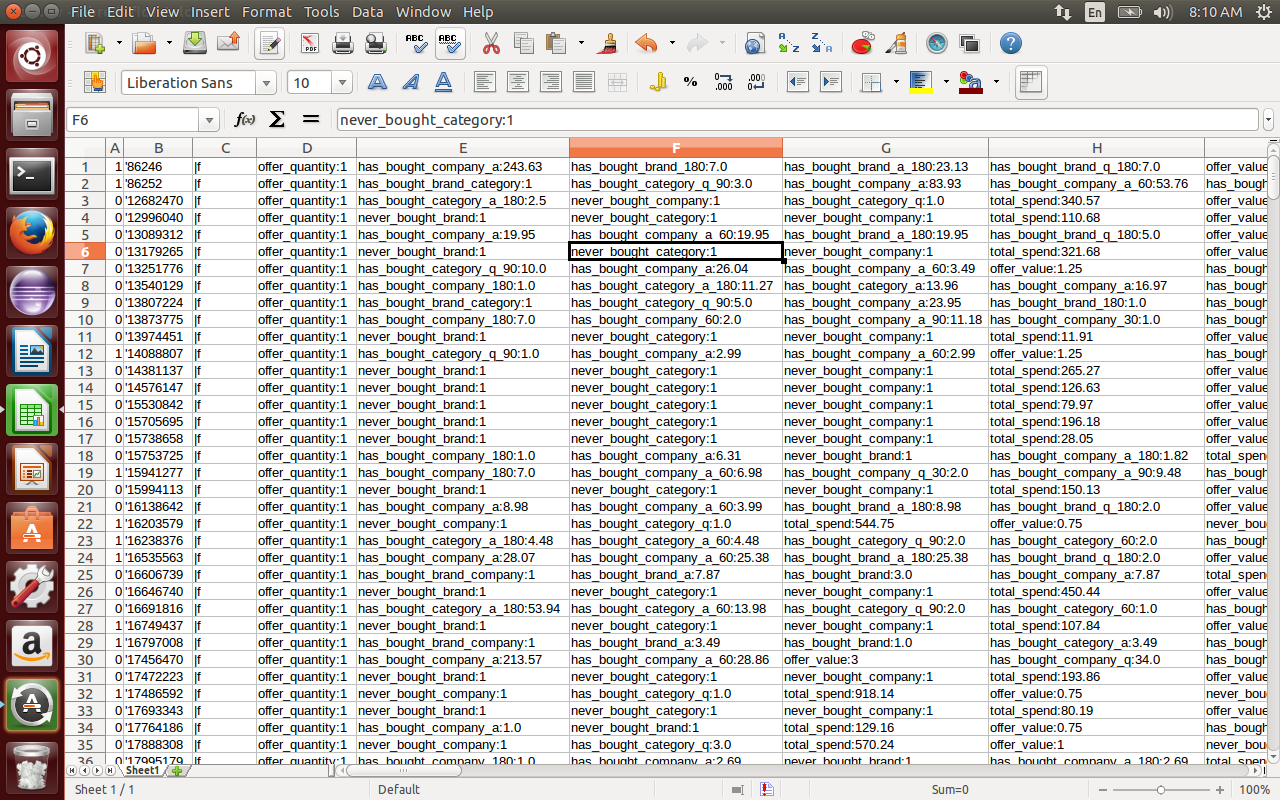
* -t says to test only
* -i says to load a certain model
* -p says to store predictions

**SNAPSHOTS**

Terminal:



Feature set obtained:



1. **CONCLUSION & SCOPE FOR FUTURE WORK**

The Revisiting shopper prediction analysis is a very useful system for understanding the dealer’s perspective and more importantly the customer’s loyalty. Be it the selling of a single brand or a multi brand, it helps analyze which is the bestselling brand in the store or website. Prediction plus providing frequent customers with offers on items, enables us to understand if that particular shopper will revisit the store only on an offer day or also on a normal day. This enables the various brands as well as sellers to categorize the items and prices of these items more accurately.

The future scope of this analysis would be to identify the customer’s loyalty and behavior towards shopping in general. We would further want to make the prediction more precise so as to allow online/offline shopping sectors to benefit from our ideology.

1. **REFERENCES**

<http://stackoverflow.com/questions/32156970/how-to-install-vowpal-wabbit-python-interface>

<http://praveenkumarsrinivasan.com/tech_n_gizmos/tech/data_analytics/2014/06/20/vowpal-wabbit/>

<http://nbviewer.ipython.org/github/hal3/vowpal_wabbit/blob/master/python/Learning_to_Search.ipynb>

<https://analyticsacademy.withgoogle.com/course03/unit?unit=3&lesson=3>

<http://www.vmware.com/in/products/vrealize-log-insight>

<https://www.r-project.org/about.html>

<http://blog.datacamp.com/r-or-python-for-data-analysis/>