Topics In Computer Science – Machine Learning
CSCI 6905 Spring 2018, Group 1

Final Report

William Clark
Sumati Kulkarni
Babak Maleki Shoja
Venkatesh Reddy Pala
Vishwa Patel

Table of Contents

Project Proposal Customer Churn Prediction

1.1 Learning Goals

In this project we as a team discussed potential semester long project to help facilitate the learning and practicing of Machine Learning. Our team goal is to come up with a proposed team project and to create an Analytics Base Table (ABT) for our proposed project. The team discussed and planned out the project workload for the next couple of weeks. This allows us to learn to work as a group and depend on one another to complete tasks assigned to us in the future. Since this is our first team project we also had goals to achieving good communication and collaboration.

1.1.1 Abstract

In this document, the project proposal for group project of Machine Learning CSCI 6905 is presented. First, the business problem that requires predictive data analysis and machine learning methods to be solved is described. Afterward, we developed Analytics Base Table (ABT) for the problem. We hereby acknowledge that the data is from a Kaggle challenge which is available on www.kaggle.com. Data is provided by KKBOX which is introduced in the document.

1.2 Business problem – project proposal

KKBOX is an Asian music streaming service. They hold the most comprehensive Asia-Pop music library in the world with more than 30 million music tracks. They are supported by advertising and paid subscriptions. For a subscription business, it is vital for long term business success to predict churn. It is stated that even slight variations in churn can significantly and drastically affect profits. KKBOX wants to predict whether a subscribed user will churn based on existing dataset regarding their subscribers.

Here, we start to convert the business problem to analytic solutions. The problem is to develop an algorithm that helps predicting if a paid user will churn after subscription expires. In addition to predicting user churn, solving this problem will provide KKBOX insights about the reasons that users leave. This will help KKBOX to be proactive in keeping its users dancing as long as possible.

1.3. Analytics Base Table (ABT)

1.3.1 Prediction subject

In order to develop ABT, we first need to identify the prediction subject. In our problem, prediction subject is a paid user (or subscriber). Afterwards, we need to determine descriptive features and a target feature.

1.3.2 Target feature

As explained in Section 1, the target feature is clearly introduced. In this project, we need to predict whether a paid user churn when the subscription expires. We call this feature “Churn” which is going to be a binary feature because churn will happen (Churn value equal to 1)” or the user renew his or her subscription (Churn value equal to 0).

1.3.3 Descriptive features

In order to determine descriptive features, domain concepts is introduced and features regarding each domain is identified. One domain is related to target feature explained in the previous subsection which is whether a user churned or not.

Second domain includes information regarding user basic details. This contains user IDs, age, city, gender, registration method (webpage, mobile, etc.), and registration date.

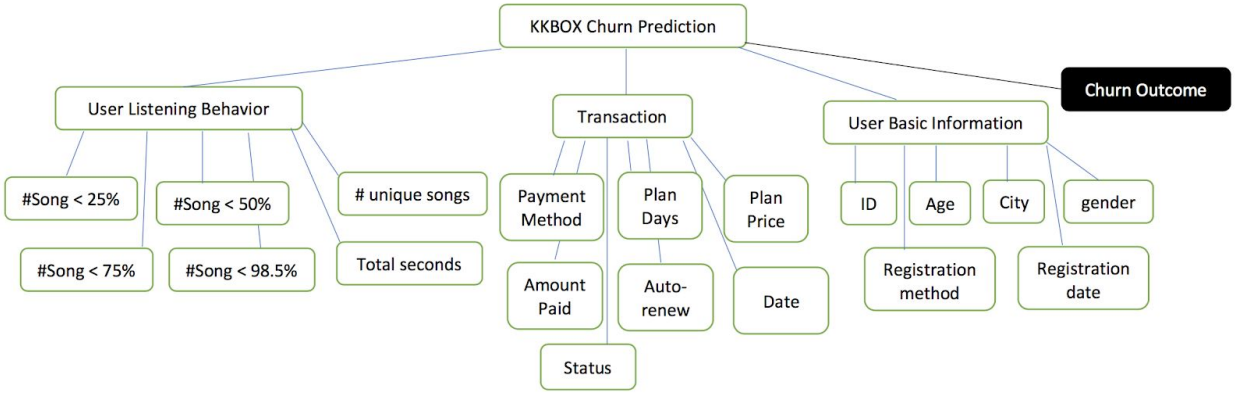
Third domain is related to payment method the user selected and whether it is now canceled, or it is still active. We call this domain “transaction” domain. It includes payment method ID, payment plan days, plan list price, actual amount paid, auto-renew (0 or 1), transaction date, and status (canceled or active).

Last domain is called “user listening behavior”. This is daily user logs describing listening behaviors of a user. This domain includes following features.

- Date
- Number of songs played by the user less than 25% of the song length
- Number of songs played by the user less than 50% of the song length
- Number of songs played by the user less than 75% of the song length

- Number of songs played by the user less than 98.5% of the song length
- Number of unique songs played by the user
- Total seconds of music played by the user

1.3.4 Domain concepts and related features hierarchy for KKBOX Churn prediction



1.3.5 Analytics Base Table for KKBOX Churn prediction

ABT for KKBOX Churn Prediction																			
User Basic Information						Transaction							User Listening Behavior						Target
ID	age	city	gender	Reg. Method	Reg. Date	Pay. Method	Plan days	Plan price	Amount paid	Auto-renew	date	status	# unq	# 25%	# 50%	# 75%	# 98.5%	Tot. Sec.	Churn

1.4 Individual Contributions

Arjun Aneja	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed and Compiled Report
William Clark	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
Sumati Kulkarni	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
Babak Maleki Shoja	Provided invaluable contributions to the completion of the tasks assigned to the group Researched, defined & drafted proposal Reviewed Report

Venkatesh Reedy Pala	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
Vishwa Patel	Provided invaluable contributions to the completion of the tasks assigned to the group for this project Reviewed Report

1.5 Team Summary

This assignment gave us all a chance to find a problem that requires predictive data analysis and required us to define Analytics Base Table (ABT) for the problem. This assignment was more difficult than it first appeared, because it was challenging to find data analytics machine learning applied to predictive data analytics. The proposal that is ultimately chose will serve as the foundation for our work for the rest of the course.

1.6 References

- Kelleher, John, Namee, Brian Mac, Arcy, Aoife D'. (2015). Fundamentals of Machine Learning for Predictive Data Analytics. The MIT Press Cambridge, Massachusetts London, England
- Kaggle, WSGM- KKBox's Churn Prediction Challenge https://www.kaggle.com/c/kkbox-churn-prediction-challenge#description 2018 Kaggle Inc

Decision Trees for Classification: A Machine Learning Algorithm

2.1 Learning Goals

In this assignment, the team had to read and discuss “Decision Trees for Classification: A Machine Learning Algorithm” by Mayur Kulkarni, a blog post on Xoriant.com. The purpose of this assignment was to get a better understanding of Decision Tree by going through an example of how to create a decision tree using various equations and calculations. The blog post had sample python code that showed how to create a decision tree. The blog touched on various concepts related to decision tree like various types of trees, Entropy, Information Gain (ID3 algorithm) and how to perform calculation for these concepts.

2.2 Decision Trees for Classification

The blog first begins by defining a decision tree which is a “type of Supervised Machine learning where data is split according to certain parameters”. These trees are typically defined by decision nodes and leaves. You can see an example of a decision tree in Figure 2. There are two types of decision tree, Classification tree, where the decision variable is categorical, and Regression tree which is defined as having a decision or the outcome variable to be a Continuous. Typically, a classification tree is defined by Boolean operators to determine if the outcome of the decision is “fit or unfit”. An example of a classification tree is like the game guess who. Where the output is the person you are getting, and the decisions are best off of yes and no answers. While regression trees use continuous variable to do a comparison between two scalar numerical values. An example of this might be house prices and square feet. Regression trees are also useful for analyzing data that has a timestamp. Decision tree can be constructed with various algorithms, but the blog focuses on Iterative before discussing ID3 algorithm which it states is one of the best one.

To best understand the ID3 algorithm we must break down the mathematical formula that are associated with it. The first step in this mathematical process it to start with an original set ‘S’ the root node and on each iteration of the algorithm, it iterates through every unused attribute of the set and calculates the entropy or information gain. Entropy is shown as H(S) and it is defined as a measure of the amount of uncertainty or randomness in the data and predictability of specific events. The equation is defined as:

$$H(S) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

Formula 1: Entropy

Information gain also called Kullback-Leibler divergence is the change in entropy after deciding on a particular attribute based on a independent variable. The equation is represented as:

$$IG(S,A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

Figure 2: Information Gain

The pseudo code for the code that was ran is the following. 1) Create root node for the tree, 2) If all examples are positive, return leaf node ‘positive’, 3) Else if all examples are negative, return leaf node ‘negative’, 4) Calculate the entropy of current state H(S), 5) For each attribute, calculate the entropy with respect to the attribute ‘x’ denoted by H(S, x), 6) Select the attribute which has maximum value of IG(S, x), 7) Remove the attribute that offers highest IG from the set of attributes, 8) Repeat until we run out of all attributes, or the decision tree has all leaf nodes.

Using this pseudo code the author of the blog then used sample data based on weather condition over 14 days and defined the outcome variable as weather Golf was played on those days. To get the rest of the tree, recursion is applied, and the equations and calculations are done to complete the tree. The next Information Gain that was done for Wind. Using formula one and formula two we calculate each of the weak and strong levels of the wind. These formulas being used can be shown in Figure 1. steps can be shown be shown in Figure 1-5.

1. $H(S_{weak})$
2. $H(S_{strong})$
3. $P(S_{weak})$
4. $P(S_{strong})$
5. $H(S) = 0.94$ which we had already calculated in the previous example

Figure 1: The levels and calculation of wind in the example

$$Entropy(S_{weak}) = -\left(\frac{6}{8}\right)\log_2\left(\frac{6}{8}\right) - \left(\frac{2}{8}\right)\log_2\left(\frac{2}{8}\right)$$

$$= 0.811$$

Figure 2: Entropy calculation of week level wind

$$Entropy(S_{strong}) = -\left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right) - \left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right)$$

$$= 1.000$$

Figure 3: Entropy Calculation of the strong level wind

$$IG(S, Wind) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

$$IG(S, Wind) = H(S) - P(S_{weak}) * H(S_{weak}) - P(S_{strong}) * H(S_{strong})$$

$$= 0.940 - \left(\frac{8}{14}\right)(0.811) - \left(\frac{6}{14}\right)(1.00)$$

$$= 0.048$$

Figure 4: Information gain calculation of wind

The ID3 algorithm showed how it performed that tasks recursively by defining a plan. Based on the plan it first showed the entropy of the current state and since the result was 0.94 it means the distribution was random. The next step calculated the attribute that gave the highest possible Information Gain which was Wind. Based on that calculation performed, the results were out of 8 weeks that had windy condition and 6 of them were Yes to play golf and 2 were No. Finally, all pieces were gathered, and Information Gain calculation was performed on all the feature. The result of the calculation looked like:

IG(S, Outlook) = 0.246
IG(S, Temperature) = 0.029
IG(S, Humidity) = 0.151
IG(S, Wind) = 0.048 (Previous Example)

Since outlook had the highest Information Gain, it would be the root node, sunny, overcast and rain would be the branches. The blog concluded by stating whenever outlook is overcast, play golf is always Yes.

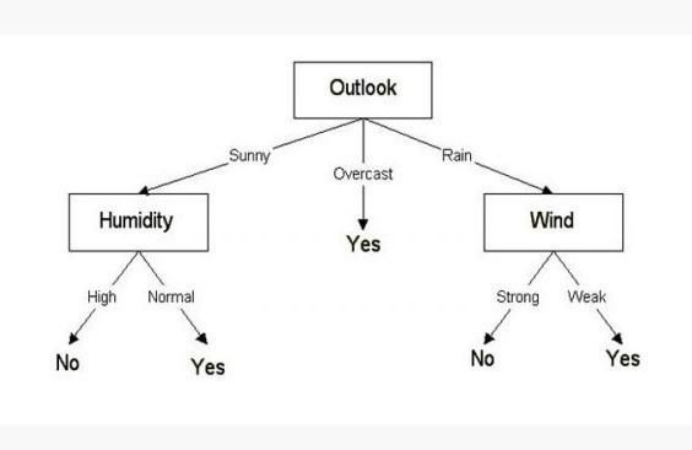


Figure 5: The complete decision tree based off of information gain

The blog then shows the python code that was used to generate this analysis. As a team we ran this code with the associated data. We concluded that he code did work and we were able to reproduce the outputs that are within this blog.

2.3 Team Summary

The Xoriant Blog about “Decision Trees for Classification: A Machine Learning Algorithm” gave the team a chance to read and discuss about Decision Trees and how to go through the process of performing Entropy and Information Gain calculation for a given set of data. The blog used a real-world example that showed the team how to create a binary tree on actual set of data. It was very detailed on how to perform the calculation by breaking it down into several steps. The blog also demoed sample python code on how to create a decision tree which was very helpful in understanding how the algorithm was created. Several team members ran the python code and we discussed and compared our thoughts on the output of the decision tree. In conclusion, the team felt this was a great assignment and it showed us in detail about how to use entropy “to measure discriminatory power of an attribute for classification task” and how to perform Information Gain to “rank attribute for filtering at a given node in a tree”.

2.4 Individual Contributions

Arjun Aneja	Provided invaluable contributions to the completion of the tasks assigned to the group Created and Compiled Report Review Report
William Clark	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
Sumati Kulkarni	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
Babak Maleki Shoja	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
Venkatesh Reedy Pala	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report

Vishwa Patel

Provided invaluable contributions to the completion of the tasks assigned to the group for this project Reviewed Report

2.5 References

- Kulkarni, Mayur (2017, September 7) Decision Trees for Classification: A Machine Learning Algorithm. Retrieved from https://www.xoriant.com/blog/product-engineering/decision-treesmachine-learning-algorithm.html
- Kelleher, John, Namee, Brian Mac, Arcy, Aoife D'. (2015). Fundamentals of Machine Learning for Predictive Data Analytics. The MIT Press Cambridge, Massachusetts London, England

Classification And Regression Trees: A Practical
Guide for Describing a Dataset

3.1 Learning Goals

In this assignment, the team had to read and discuss “Classification and Regression Trees: A Practical Guide for Describing a Dataset” by Leo Pekelis, a paper from Bicostal Datafest from Stanford University. The purpose of this assignment was to get a better understanding of decision trees. The understanding was gained in two ways. First an understanding of the benefits and uses of a regression tree is done. Second a breakdown of the foundational mathematical principles of a regression tree. This is done by defining the structural model. The structural model is broken down by the nodes and leaves of the decision tree. From there it is able to access the continuous variables in vector space. This is the basic process of the decision tree and allows for one to actually apply this method in R.

3.1.1 **Abstract**

The paper starts with the discussion of Decision Trees, Decision trees are further categorized for deciding the sets of possible outcomes. Decision trees break down the data by making the decision based on series of descriptive features. In the proposed model, it is done through a series of questions.

Based on the features in the set, the decision tree infers from the samples and labels them. Using the algorithm in this paper, they split the data on the feature which results in largest information gain. There is a repetitive process that repeats itself in a loop where child nodes are extracted for the pure data. In this example the data leafs are deep and since it may result in overfitting, a limit has set to make the data prune the tree. Data pruning is applied manually in the paper.

3.2 Introduction

Before breaking down the mathematical principles of a regression tree. It is important to understand what a regression tree is from a high-level. The definition of what a regression is to describe the approximate similarities between two variables. These similarities are used to determine the correlation and strength of relativeness between two variables. Regression trees take this step one step further by creating a structural model of if then results. This can be used to predict the outcome features of a dataset. This is of course the feed of continuous data has the same features as the tree.

Another benefit of using regression trees is that they can “work” and produce accurate predictions or predicted classifications based on few logical if then conditions. This is not the same as other classification methods that need a lot of logical operators to have the same accurate results. This makes the results very clear and interpretable of where they can come from. This interpretability typically makes a prediction model easy to manage. This kind of management allows for a quick analysis of the accuracy of your model. If the model is inaccurate given the interpretability it would not be hard to go back into the model and change the appropriate logical operator.

3.3 Methods

Breaking down what needs to be done to create a regression tree. The first thing that must be done is to understand the structural model that is a decision tree. Decision trees are a function of the summation of the features and all real numbers associated with the vector space and vector input variable. This can be shown in Formula One.

$$F(x) = \sum_{i=1}^{M} c_m I(x \in R_m)$$

Formula One: The structural model of a regression tree.

The vector space has a specific size. So the next part of defining a regression tree is to define the size of the tree. This is broken down in a three step derivation The automatic way to select the size of the tree is denoted by Formula Two.

$$e_m = \frac{1}{N} \sum_{x_i \in R_m} (y_i - \bar{y}_m)^2$$

Formula Two: Feature specific Tree Sizing

Then the following formulation is applied.

$$Imp_m = e_m - e_{ml} - e_{mr}$$

Formula Three: Cumulative Features Tree Sizing

And finally, it is solved by the following equation.

$$\min_{T} \left[e(T) + cp|T| \right]$$

Formula Four: Minimum tree size

Where |T| is the number of terminal nodes of the tree. Tree size is important because it can severely impact the amount of pruning that needs to be done to the tree. If the size of the tree is to big it may be hard to prune the appropriate factors. This would in turn decrease the amount of potentially accuracy that the tree has. Thus it is important to compute the minimum tree size.

Once the structure model and the size of the tree are defined the statistical programming language R is used to make the decision tree. R has specific libraries to make these kind of decisions. They have imported the libraries and the data is provided as well. As the first step, they have developed an algorithm and a form of outcomes. The data outcome is in the form of matrix where the columns represent the variables and rows demonstrate observations.

To export the results and display it with visualization, other frameworks are being used in the R. The node indexes of the child node are represented by the rows of matrix, these nodes were unique in nature, several abbreviations are used for several objects in the matrix, frame is denoted with 1 row per node of the tree, ‘var’ describes the name of the variable used in the leaf of the node, n represents the number of observations reaching the node, and yval shows the fitted outcome value at the node. This formula explains that the entry “cptable” gives tree statistics for each child node, “rel error” is the ratio of the objective, e(T), to that of single root tree. Last but not least, they described some libraries outside of R to do the same. The output of the R algorithms that were provided in this assessment would produce a tree as shown in Figure One.

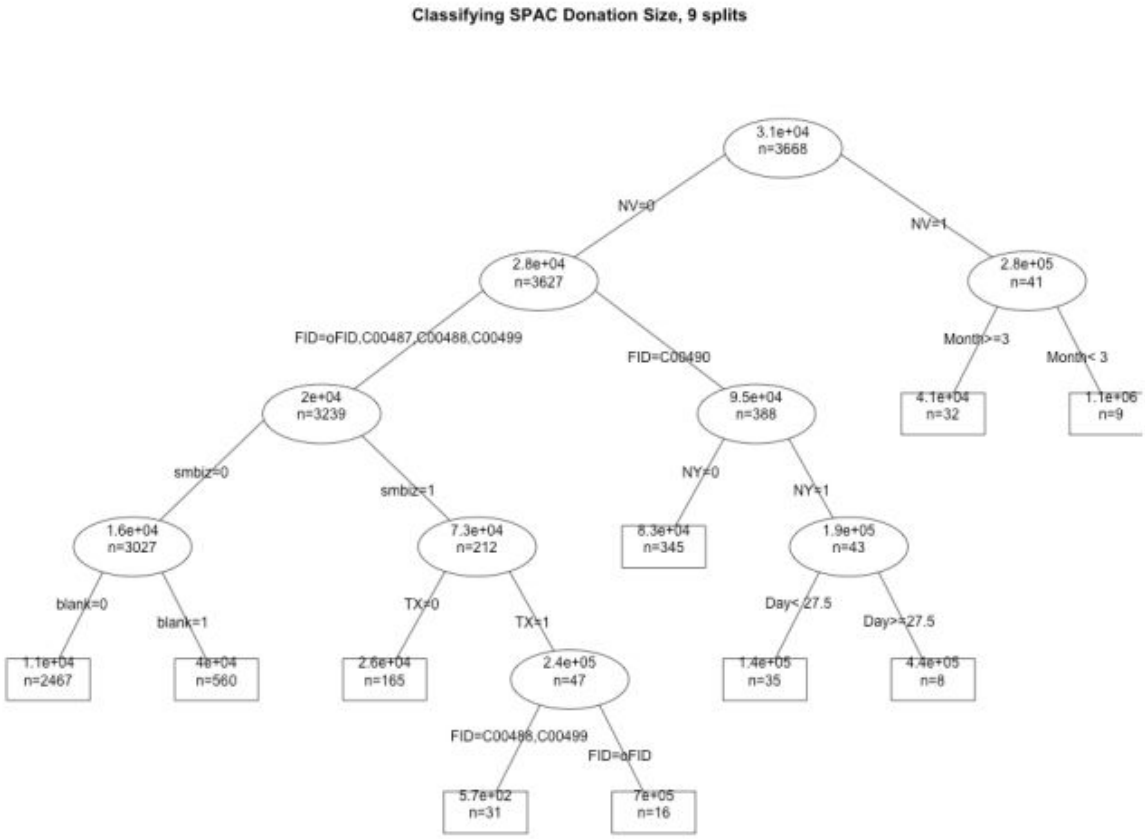


Figure 1: The final decision tree made in R

3.4 Conclusion

Overall trees have advantages and disadvantages. Trees are fast, invariant, resistance to irrelevant variables, handling the missing data and outcomes are categorized. On the contrary, there are also some disadvantages such as low accuracy and also sensitive variance i.e. a small change in dataset would largely affect the tree. In this study we were able to create a regression tree using R and multiple libraries that are associated with R. As well as break down and understand the mathematical principles that are associated with building a regression tree.

3.5 References

- Pekelis, Leo (2013, February 2) Classification And Regression Trees : A Practical Guide for Describing a Dataset. Retrieved from https://statweb.stanford.edu/~lpekelis/talks/13_datafest_cart_talk.pdf

- Kelleher, John, Namee, Brian Mac, Arcy, Aoife D'. (2015). Fundamentals of Machine Learning for Predictive Data Analytics. The MIT Press Cambridge, Massachusetts London, England

Cars Data Report

4.1 Learning Goals

In this assignment, the team had to take an excel sheet called “cars.xls” which contains automobile data that had to be evaluated by the team. This report is grounded on the first two processes in the CRISP-DM phase. Thus, a business problem is created that could use these data in a predictive analytic scenario and thus to solve this problem, with these data, a data understanding is created. This understanding is defined by a data quality report.

4.2 Business Understanding

Currently, a car dealership's sales are low. They figure that it is due to the types of cars that they are selling. To potentially increase sales, they would like to figure out which cars are most recommended. If the car dealership has more of a chance to put cars on the lot that are highly recommended, they believe that the car dealerships sales will increase. The car dealership company have a dataset that has a variety of features of cars. Since these data consist of is so sparse, a data understanding must be gained.

4.3 Data Understanding

To best solve this business problem using predictive analytics an assessment of the data must be done. This assessment is also referred to a data understanding. The data provided has 18 features within it. These features are made, model, recommend, where, body, price, manufacturer, city miles per gallon, highway miles per gallon, range, reliability, satisfied, weight, horse, rpm, gas tank volume, engine type, and length. Since the target feature is recommended we have 17 descriptive features to work with. With the target feature being of a categorical data type. The continuous features that are within this data set are 10 total. These are city mpg, hwy mpg, weight, horse, rpm, gas tank, engine, price, and length. While the categorical descriptive features with a total of 7 are satisfied, reliable, model, where, body, manuf, and where.

4.3.1 Data Quality Assessment - Statistical Exploration and Visualization

A data quality report was created based off a central tendency and variation exploration of these data. This report is used to show the best descriptive features to use for our prediction of the target feature recommended. To make sure we are using the best descriptive features for the prediction of the target feature we first do an assessment of count, missing, card, mode, mode frequency, mode %, 2nd nd Mode, 2nd Mode Frequency, and 2 mode percentage. The results of this assessment are within Table 1. Next, a data exploration of the continuous features is done by an assessment of count, missing values, cardinality, minimum, maximum, 1st, and 3rd quartiles, max, and standard deviation for continuous variables. The results of this

assessment are within Table 2. These explorations and visualization were done within a Python environment and Microsoft Excel.

4.3.2 Data Quality Report

Using the resulting quality report tables and figure report key issues about these data arise. To begin an analysis of each of the key data issue variables is done. This information can be gathered from the tables. Assessing the cardinality of the descriptive features city miles per gallon and highway miles per gallons. It can be shown that these features are not as unique as they should be. On the other side of the spectrum, it seems that the descriptive feature model has too many unique values. Next, an assessment of percent missing shows that satisfy, reliable, city miles per gallon, and highway miles per gallons are missing a high percentage of values. Just based on these two quality assessments of missing values and cardinality. The descriptive features satisfy, reliable, city miles per gallon, highway miles per gallon, and model should be removed or manipulated to correct these issues. Some outliers within these data are the price, weight and engine, descriptive features. This can be solved by clamping the data.

4.4 Conclusion

A report of the car data was done by the team. This report created a business problem and reported the quality issues of these data. The business problem was a lack of sales at a car dealership. The solution to this problem was to use predictive data analytics to increase sales at a car dealership. To create a predictive model though, is imperative to gain a full understanding of the data. An understanding of the data is defined by understanding key quality issues that are typically within datasets. These issues usually are assessed by looking to see missing data and cardinality. As well as creating and assessing a statistical overview of the data, and statistical visualizations of the data. Our understanding showed that some descriptors such as car model, city mpg, highway mpg, satisfied, and reliable were weak descriptors to use in a predictive model. Some Next steps for these data is to create a predictive model in hopes to increase sales at the car dealership.

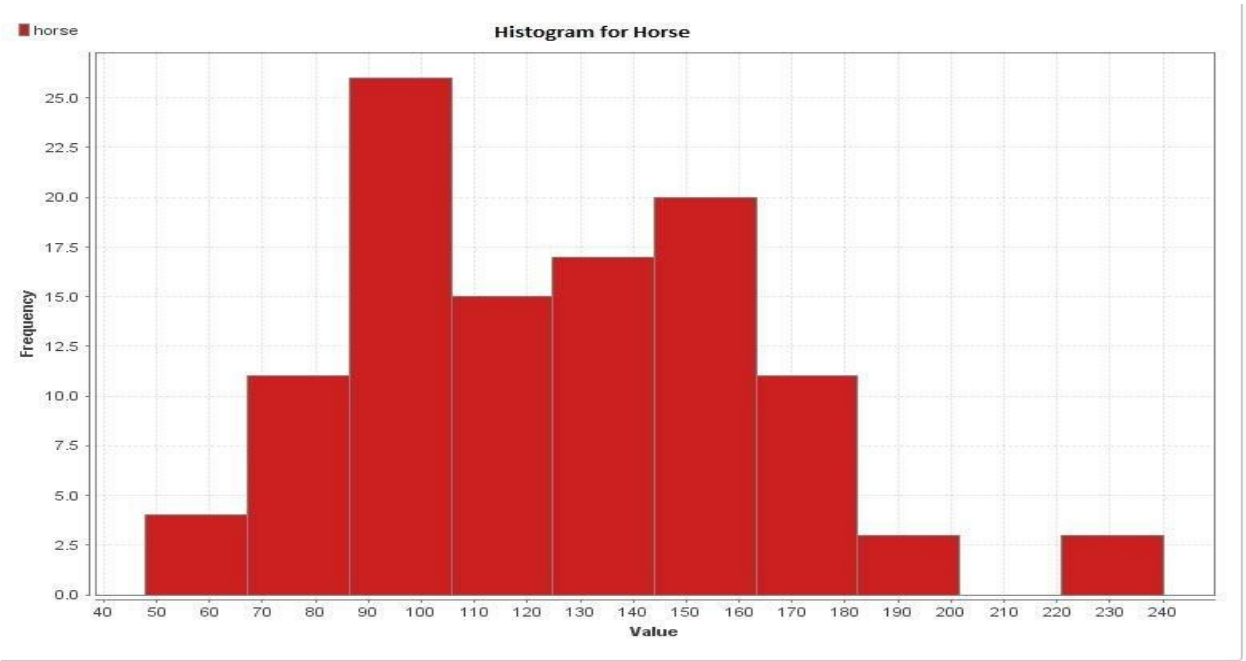
4.5 Tables and Figures

4.5.1 Table 1: Categorical Descriptive Features Data Quality Report

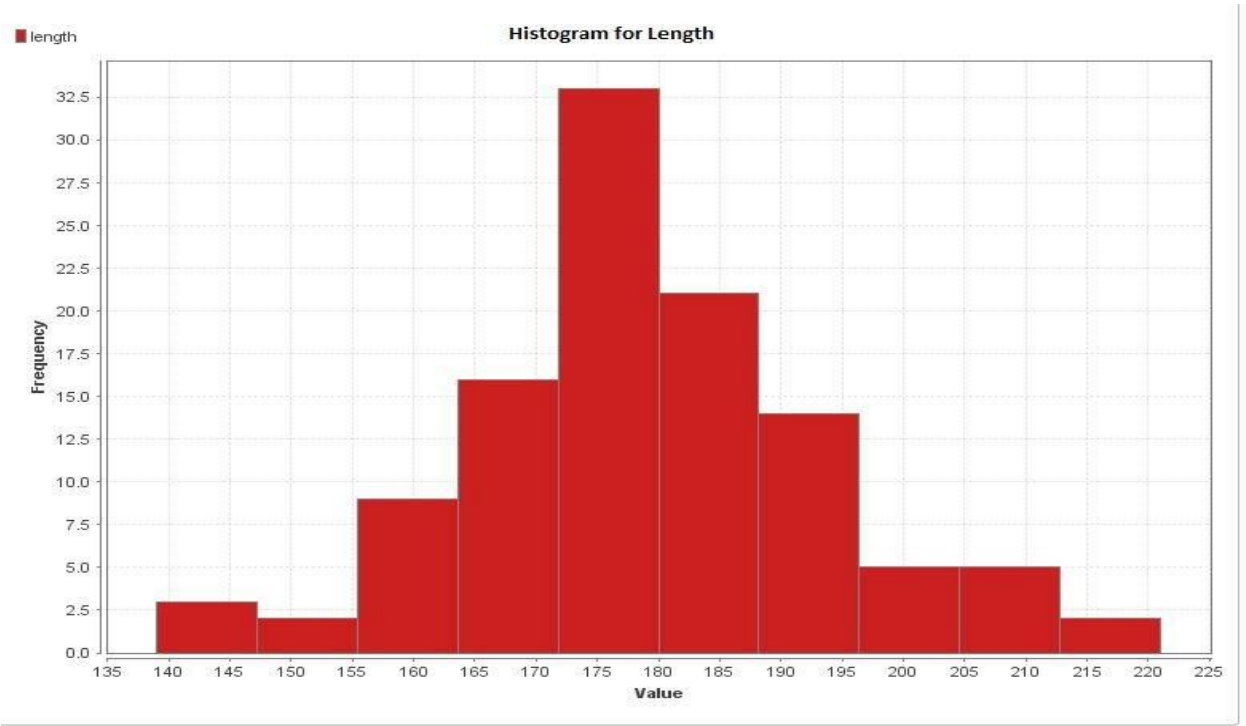
Feature	Count	Missing %	Card.	Mode	Mode Freq	Mode %	2nd Mode	2nd Mode Freq	Mode %
make	111	0	32	For	9	8.108108	Toy	9	8.108108
model	111	0	110	N/A	N/A	N/A	N/A	N/A	N/A
where	111	0	3	Asian	47	42.34234	American	40	36.03604
body	111	0	6	Subcompact	26	23.42342	Midsize	25	22.52252
manuf	111	0	25	GM	20	18.01802	Chr/For	14	12.61261
satisfiy	111	30.63063	5	3 Average	35	31.53153	4 Above Average	20	18.01802
reliable	111	17.11712	5	3 Average	30	27.02703	1 Poor/5 Excellent	20	18.01802

Feature	Missing	Card.	Min	Max	1st	3rd	Mean	Median	Std
price	0	107	4349	44850	9667.5	18543	14727.05	12247	7455.165
citympg	48.64865	16	12	31	15	20	17.8214 3	18	4.169618
hwympg	48.64865	21	27	58	32	39	36	35	6.128918
range	48.64865	31	270	580	373.75	435	406.696 4	405	56.62698
weight	0	98	1640	4190	2412.5	3325	2892.955 5	2907.5	556.3632
horse	0	54	48	240	93.25	153	124.981 8	122	38.22315
rpm	0	81	1410	3650	2180	2768.75	2405.409	2467.5	484.1208
gastank	0	44	8.4	27	13.6	18	15.8490 9	15.9	3.178879
engine	0	48	61	350	109	181	155.954 5	140.5	65.09603
length	0	48	139	221	171	188	179.190 9	178	14.83951

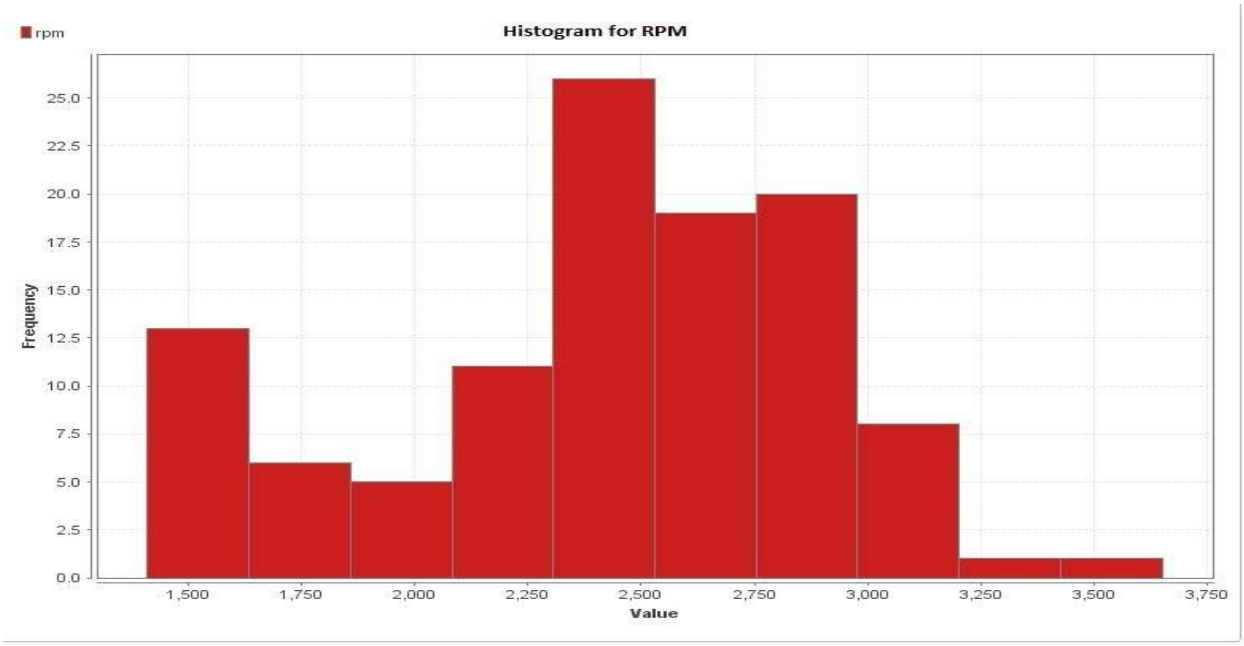
4.5.2 Histogram 1: Horsepower



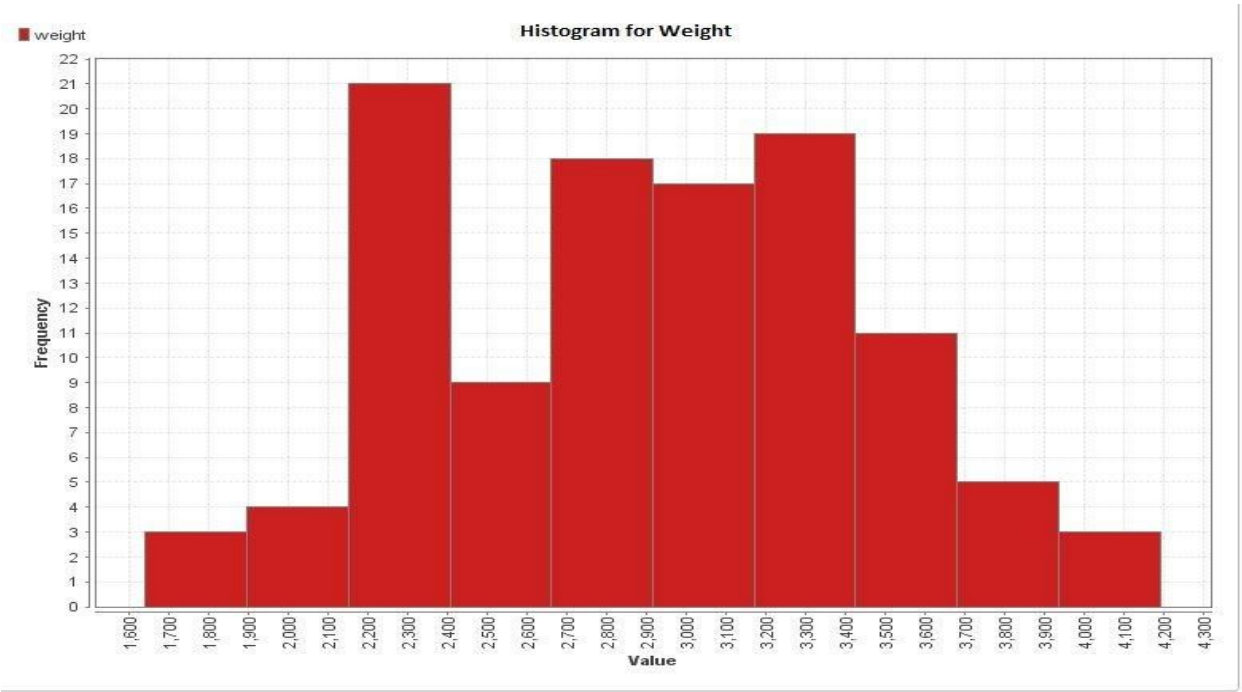
4.5.3 Histogram 2: Length



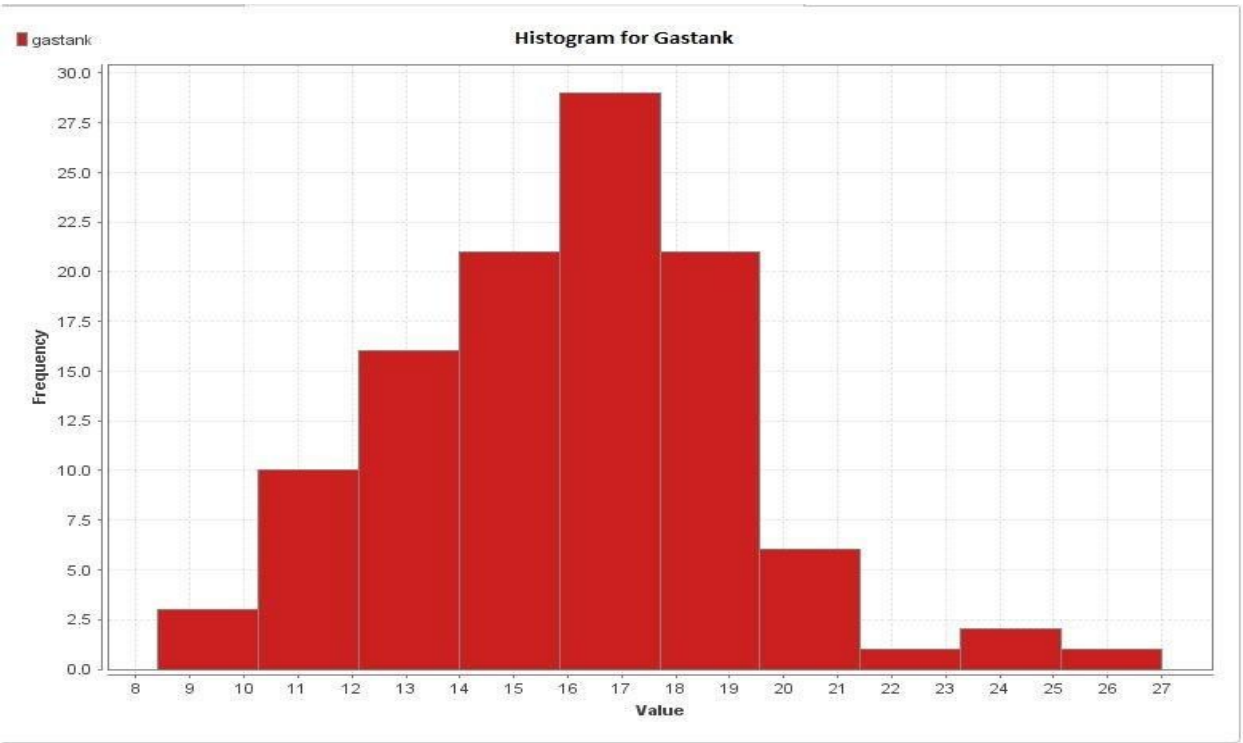
4.5.4 Histogram 3: RPM



4.5.5 Histogram 4: Weight

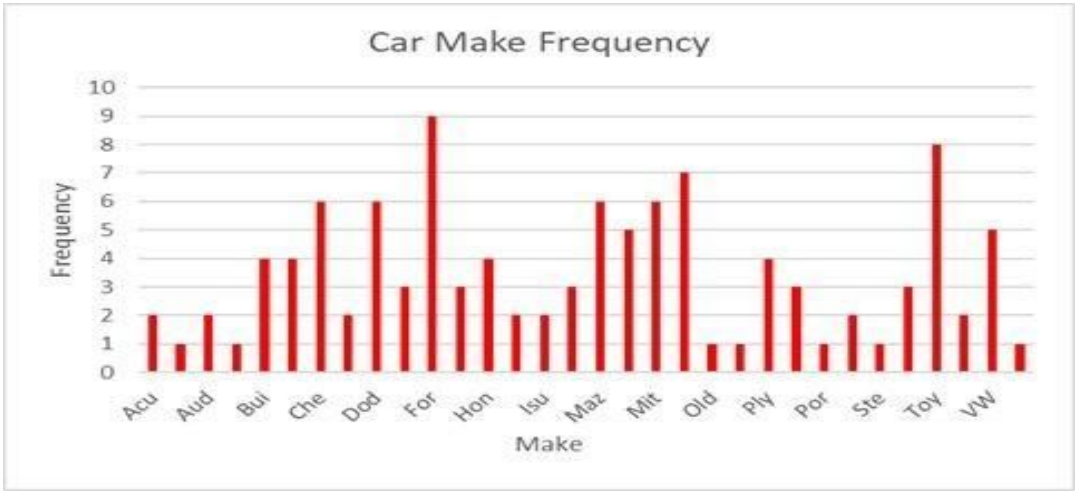


4.5.6 Histogram 5 : Length

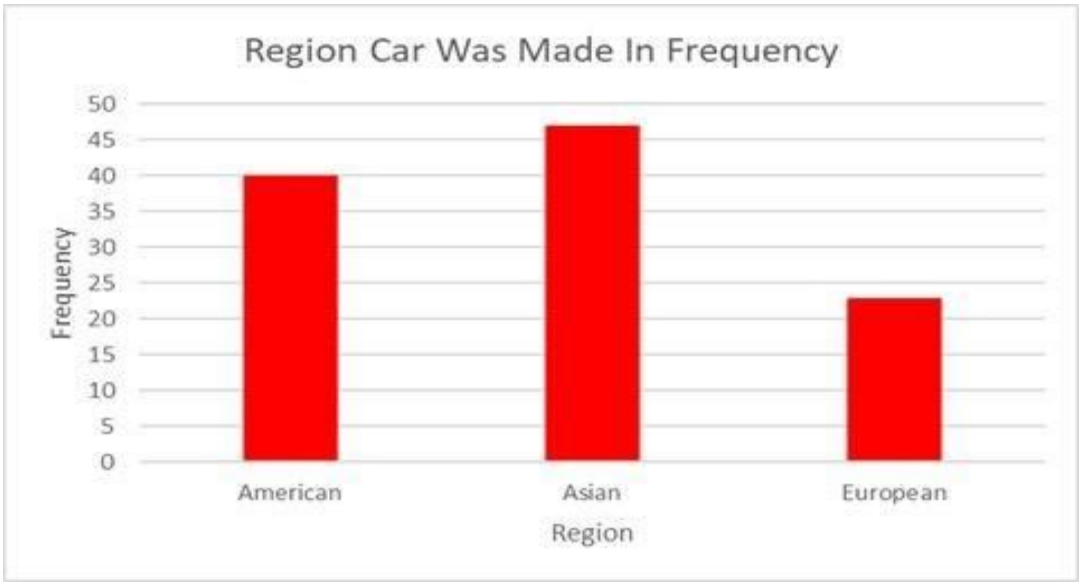


4.5.7 Histogram 6 : Price

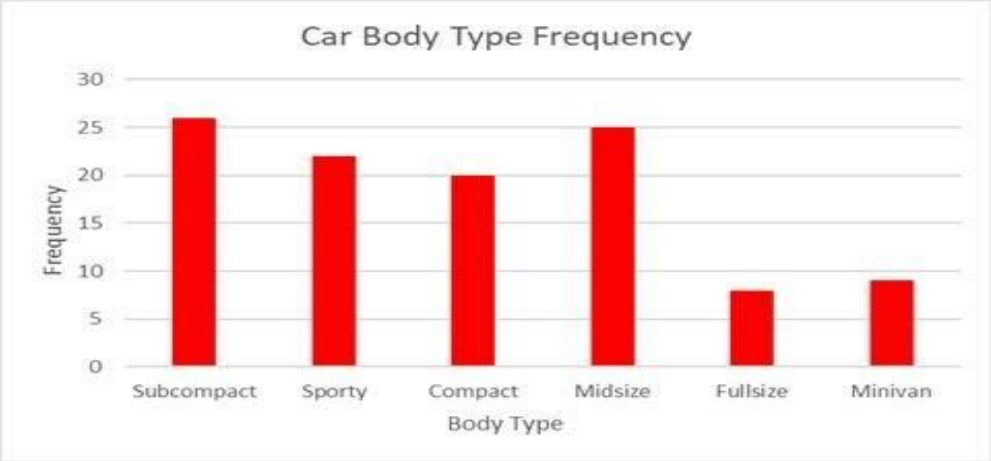
4.5.8 Frequency Bar Graph 1 : Make



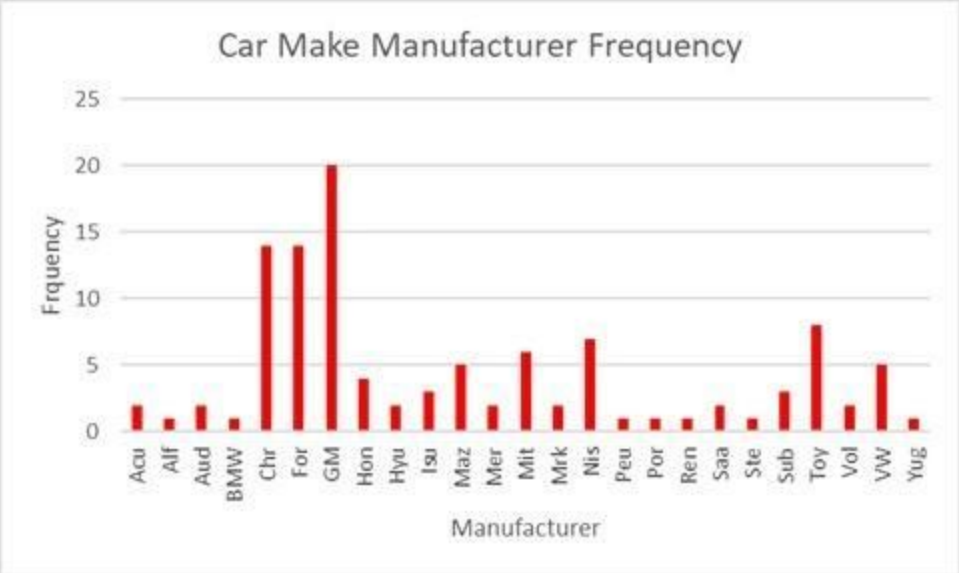
4.5.9 Frequency Bar Graph 2: Where



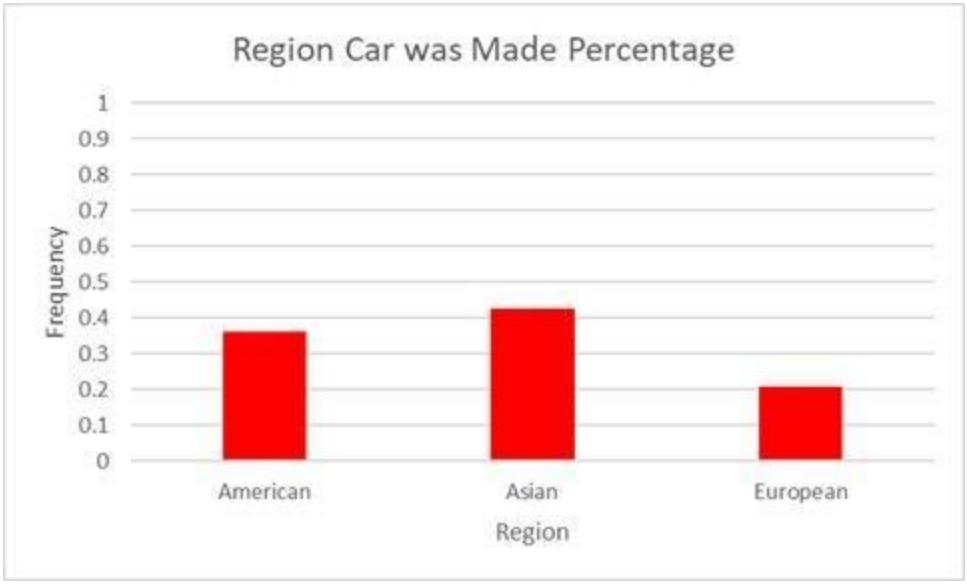
4.5.10 Frequency Bar Graph 3: Body



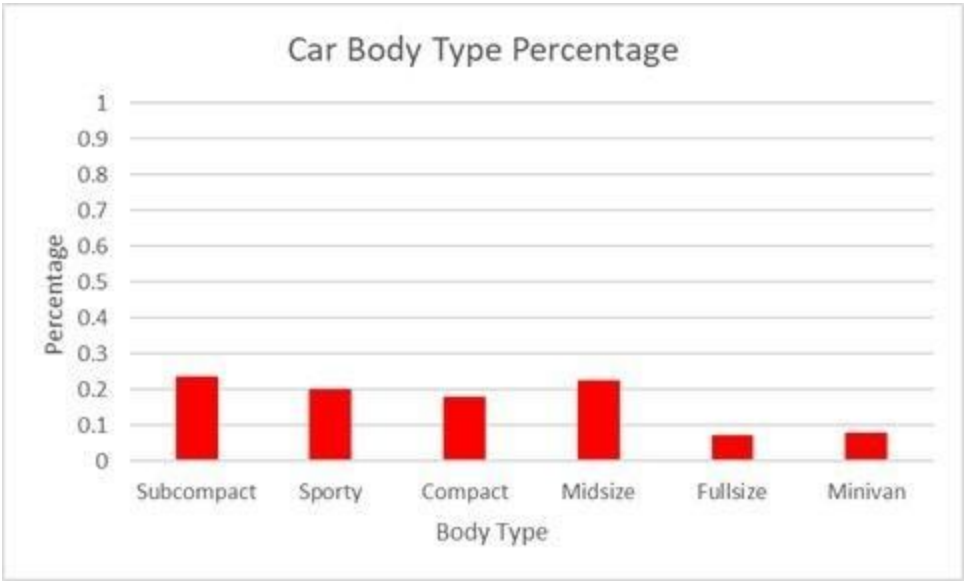
4.5.11 Frequency Bar Graph 4: Manuf



4.5.12 Percentage Bar Graph 1: Where



4.5.13 Percentage Bar Graph 2: Body



4.6 References

- Kelleher, John, Namee, Brian Mac, Arcy, Aoife D'. (2015). Fundamentals of Machine Learning for Predictive Data Analytics. The MIT Press Cambridge, Massachusetts London, England

Kn-Neighbour Model Implementation

5.1 Learning Goals

In this assignment, the team developed a model for KKBOX Music Streaming Service Provider as our group project problem in order to predict churn of a subscribed user using descriptive features provided by KKBOX. The purpose of this assignment was to get familiar with applying a prediction model to an existing problem, how we can evaluate the goodness of fit and to determine features that can be excluded from the data.

5.1.1 **Abstract**

In this document, the model developed and used to predict churn of a subscriber based on descriptive features for KKBOX music streaming service provider is defined. First, we briefly review the business problem and descriptive and target features provided by the KKBOX for this problem. Then, we describe the model we used for predicting churn based on descriptive features. Next, the results of applying the model is provided. Finally, we draw conclusion regarding churn prediction problem.

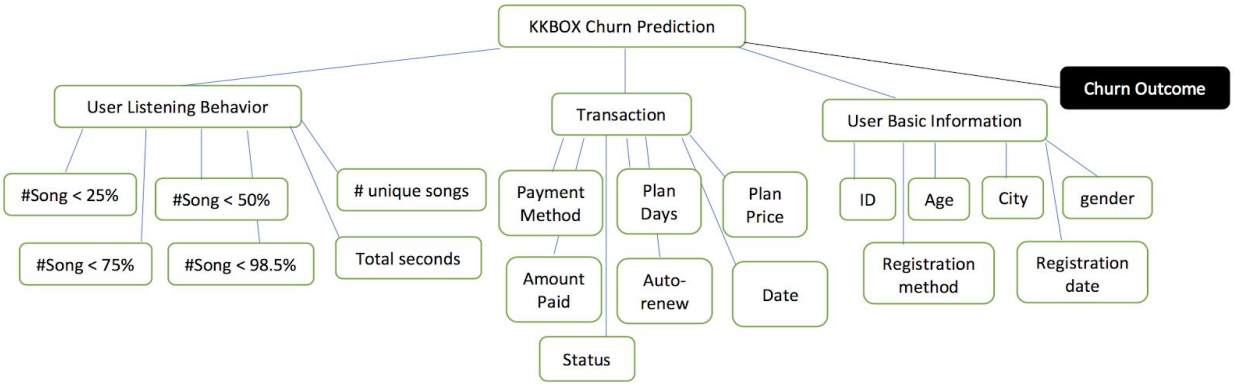
5.2 Business problem

As provided in the project proposal, KKBOX, an Asian music streaming service provider, is facing the challenge of predicting whether a subscriber churn after his or her subscription expires or their decision to extend their subscription. This is a critical problem for such businesses and even a slight deviation from the predictions KKBOX has gathered a large amount of data from its users to resolve this problem.

The analytic solution to this problem defined as follows: We proposed to develop a model to predict the churn of a paid user after subscription expires. We take this into consideration to apply the prediction model and evaluate the results.

5.3 Descriptive and Target Features

Before describing the features, we should note that the prediction subject is defined as a paid subscriber. Here, we briefly review the features for the problem. The features of this problem and corresponding ABT developed as follows.



ABT for KKBOX Churn Prediction																			
User Basic Information						Transaction							User Listening Behavior						Target
ID	age	city	gender	Reg. Method	Reg. Date	Pay. Method	Plan days	Plan price	Amount paid	Auto-renew	date	status	# unq	# 25%	# 50%	# 75%	# 98.5%	Tot. Sec.	Churn

As explained in previous report, we need to predict whether a paid user churn when the subscription expires. We call this feature “Churn” which is going to be a binary feature because churn will happen (Churn value equal to 1)” or the user renew his or her subscription (Churn value equal to 0).

Descriptive Feature	Description	Type
ID	The ID of a user which is unique for each user	Int, continuous
AGE	Age of user	Int, continuous
CITY	Location of User	Categorical
Gender	Gender of User	Categorical with Cardinality of 2
Registration method	Method user used to register for this service	categorical
Registration Date	Date on which User registered	Continuous
Payment Method ID	Method used by user to pay	Categorical
Payment plan days	Plan chosen by user	Categorical with relation to available plans
Plan List Price	Price of plan	Categorical with relation to available plans
Actual amount paid	Amount user actually paid	Continuous
Auto-renew	The feature denotes that if a user has activated auto renew	Categorical feature with cardinality of 2
Transaction date	Similar to registration date	Continuous

Status	Denotes if a user is still active or canceled subscription	Categorical with the cardinality of 2
Number of songs	Songs played by user less than 25% or 50% or 75% or 98.5%	Continuous
Number of unique songs	Unique songs played by the user	Continuous
Total seconds of music	Total seconds of music played by the user	Continuous

5.4 Model Implementation Steps

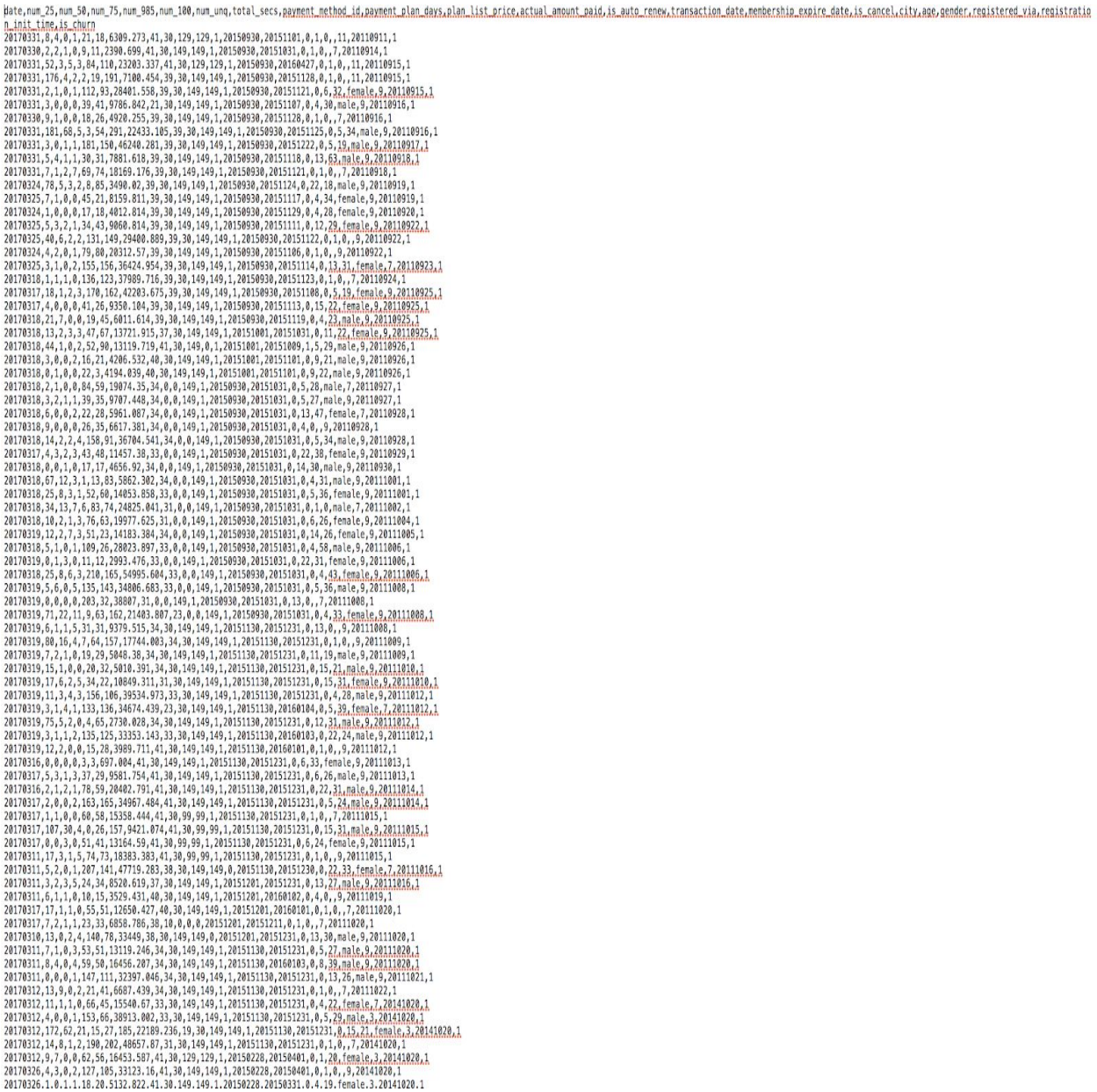
For our dataset (KKBOX customer churn prediction), we have used python language and Spyder IDE for obtaining results. Prediction model steps that we have done to achieve required results is described below:

1) Firstly, we copy (read) the data from .csv file format and store it in a variable.
2) We create two different variables, X and y in which descriptive features and target feature is stored respectively
3) We used “sklearn” library, preprocessing package and LabelEncoder class which provides functionality to convert Categorical data into Numerical data (from string to int). There were some missing values in the column “source_screen_name”, those values are calculated to numerical value, as in our case ‘nan’ is converted to numerical value “17”.
4) Now the derived data is then converted into normalized data and then stored into same variable.
5) The normalized data is stored in an external file “.csv”.
6) Using train_test_split class in sklearn.cross_validation package, we have split dataset into training sets and testing sets. In which we have given 20% of the total dataset is given into the testing part while the rest of the data is given into the training part.
7) We have used K_neighboursclassifiers algorithm to train the machine, which includes both descriptive features and target features.
8) Once the training is done, then we have given the x_test (descriptive features) to predict the outcome based on what model (machine) has learned in training.
9) We have then computed the accuracy of the data by comparing the actual target feature and what we received by our machine learning model.
10) To see how many features are predicted right and how many predicted wrong, we have used confusion matrix to achieve that.

5.5 Results

Dataset we have used in our model is 10,000 data. In which 8000 is given in training part and 2000 is given in testing phase. Following screenshots are our original dataset and the obtained result:

1) Original Dataset of our project:



```
date,num_25,num_50,num_75,num_985,num_100,num_unq,total_secs,payment_method_id,payment_plan_days,plan_list_price,actual_amount_paid,is_auto_renew,transaction_date,membership_expire_date,is_cancel,city,age,gender,registered_via,registration_init_time,is_churn
20170331,8,4,0,1,21,18,6309.273,41,30,129,129,1,20150930,20151101,0,1,0,,11,20110911,1
20170330,2,2,1,0,9,11,2390.699,41,30,149,149,1,20150930,20151031,0,1,0,,7,20110914,1
20170331,52,3,5,3,84,110,23203.337,41,30,129,129,1,20150930,20160427,0,1,0,,11,20110915,1
20170331,176,4,2,2,19,191,7100.454,39,30,149,149,1,20150930,20151128,0,1,0,,11,20110915,1
20170331,2,1,0,1,112,93,28401.558,39,30,149,149,1,20150930,20151121,0,6,32,female,9,20110915,1
20170331,3,0,0,0,39,41,9786.842,21,30,149,149,1,20150930,20151107,0,4,30,male,9,20110916,1
20170330,9,1,0,0,18,26,4920.255,39,30,149,149,1,20150930,20151128,0,1,0,,7,20110916,1
20170331,181,68,5,3,54,291,22433.105,39,30,149,149,1,20150930,20151125,0,5,34,male,9,20110916,1
20170331,3,0,1,1,181,150,46240.281,39,30,149,149,1,20150930,20151222,0,5,19,male,9,20110917,1
20170331,5,4,1,1,30,31,7881.618,39,30,149,149,1,20150930,20151118,0,13,63,male,9,20110918,1
20170331,7,1,2,7,69,74,18169.176,39,30,149,149,1,20150930,20151121,0,1,0,,7,20110918,1
20170324,78,5,3,2,8,85,3490.02,39,30,149,149,1,20150930,20151124,0,22,18,male,9,20110919,1
20170325,7,1,0,0,45,21,8159.811,39,30,149,149,1,20150930,20151117,0,4,34,female,9,20110919,1
20170324,1,0,0,0,17,18,4012.814,39,30,149,149,1,20150930,20151129,0,4,28,female,9,20110920,1
20170325,5,3,2,1,34,43,9860.814,39,30,149,149,1,20150930,20151111,0,12,29,female,9,20110922,1
20170325,40,6,2,2,131,149,29400.889,39,30,149,149,1,20150930,20151122,0,1,0,,9,20110922,1
20170324,4,2,0,1,79,80,20312.57,39,30,149,149,1,20150930,20151106,0,1,0,,9,20110922,1
20170325,3,1,0,2,155,156,36424.954,39,30,149,149,1,20150930,20151114,0,13,31,female,7,20110923,1
20170318,1,1,1,0,136,123,37989.716,39,30,149,149,1,20150930,20151123,0,1,0,,7,20110924,1
20170317,18,1,2,3,170,162,42203.675,39,30,149,149,1,20150930,20151108,0,5,19,female,9,20110925,1
20170317,4,0,0,0,41,26,9350.104,39,30,149,149,1,20150930,20151113,0,15,22,female,9,20110925,1
20170318,21,7,0,0,19,45,6011.614,39,30,149,149,1,20150930,20151119,0,4,23,male,9,20110925,1
20170318,13,2,3,3,47,67,13721.915,37,30,149,149,1,20151001,20151031,0,11,22,female,9,20110925,1
20170318,44,1,0,2,52,90,13119.719,41,30,149,0,1,20151001,20151009,1,5,29,male,9,20110926,1
20170318,3,0,0,2,16,21,4206.532,40,30,149,149,1,20151001,20151101,0,9,21,male,9,20110926,1
20170318,0,1,0,0,22,3,4194.039,40,30,149,149,1,20151001,20151101,0,9,22,male,9,20110926,1
20170318,2,1,0,0,84,59,19074.35,34,0,0,149,1,20150930,20151031,0,5,28,male,7,20110927,1
20170318,3,2,1,1,39,35,9707.448,34,0,0,149,1,20150930,20151031,0,5,27,male,9,20110927,1
20170318,6,0,0,2,22,28,5961.087,34,0,0,149,1,20150930,20151031,0,13,47,female,7,20110928,1
20170318,9,0,0,0,26,35,6617.381,34,0,0,149,1,20150930,20151031,0,4,0,,9,20110928,1
20170318,14,2,2,4,158,91,36704.541,34,0,0,149,1,20150930,20151031,0,5,34,male,9,20110928,1
20170317,4,3,2,3,43,48,11457.38,33,0,0,149,1,20150930,20151031,0,22,38,female,9,20110929,1
20170318,0,0,1,0,17,17,4656.92,34,0,0,149,1,20150930,20151031,0,14,30,male,9,20110930,1
20170318,67,12,3,1,13,83,5862.302,34,0,0,149,1,20150930,20151031,0,4,31,male,9,20111001,1
20170318,25,8,3,1,52,60,14053.858,33,0,0,149,1,20150930,20151031,0,5,36,female,9,20111001,1
20170318,34,13,7,6,83,74,24825.041,31,0,0,149,1,20150930,20151031,0,1,0,male,7,20111002,1
20170318,10,2,1,3,76,63,19977.625,31,0,0,149,1,20150930,20151031,0,6,26,female,9,20111004,1
20170319,12,2,7,3,51,23,14183.384,34,0,0,149,1,20150930,20151031,0,14,26,female,9,20111005,1
20170318,5,1,0,1,109,26,28023.897,33,0,0,149,1,20150930,20151031,0,4,58,male,9,20111006,1
20170319,0,1,3,0,11,12,2993.476,33,0,0,149,1,20150930,20151031,0,22,31,female,9,20111006,1
20170318,25,8,6,3,210,165,54995.604,33,0,0,149,1,20150930,20151031,0,4,43,female,9,20111006,1
20170319,5,6,0,5,135,143,34806.683,33,0,0,149,1,20150930,20151031,0,5,36,male,9,20111008,1
20170319,0,0,0,0,203,32,38807,31,0,0,149,1,20150930,20151031,0,13,0,,7,20111008,1
20170319,71,22,11,9,63,162,21403.807,23,0,0,149,1,20150930,20151031,0,4,33,female,9,20111008,1
20170319,6,1,1,5,31,31,9379.515,34,30,149,149,1,20151130,20151231,0,13,0,,9,20111008,1
20170319,80,16,4,7,64,157,17744.003,34,30,149,149,1,20151130,20151231,0,1,0,,9,20111009,1
20170319,7,2,1,0,19,29,5048.38,34,30,149,149,1,20151130,20151231,0,11,19,male,9,20111009,1
20170319,15,1,0,0,20,32,5010.391,34,30,149,149,1,20151130,20151231,0,15,21,male,9,20111010,1
20170319,17,6,2,5,34,22,10849.311,31,30,149,149,1,20151130,20151231,0,15,31,female,9,20111010,1
20170319,11,3,4,3,156,106,39534.973,33,30,149,149,1,20151130,20151231,0,4,28,male,9,20111012,1
20170319,3,1,4,1,133,136,34674.439,23,30,149,149,1,20151130,20160104,0,5,39,female,7,20111012,1
20170319,75,5,2,0,4,65,2730.028,34,30,149,149,1,20151130,20151231,0,12,31,male,9,20111012,1
20170319,3,1,1,2,135,125,33353.143,33,30,149,149,1,20151130,20160103,0,22,24,male,9,20111012,1
20170319,12,2,0,0,15,28,3989.711,41,30,149,149,1,20151130,20160101,0,1,0,,9,20111012,1
20170316,0,0,0,0,3,3,697.004,41,30,149,149,1,20151130,20151231,0,6,33,female,9,20111013,1
20170317,5,3,1,3,37,29,9581.754,41,30,149,149,1,20151130,20151231,0,6,26,male,9,20111013,1
20170316,2,1,2,1,78,59,20402.791,41,30,149,149,1,20151130,20151231,0,22,31,male,9,20111014,1
20170317,2,0,0,2,163,165,34967.484,41,30,149,149,1,20151130,20151231,0,5,24,male,9,20111014,1
20170317,1,1,0,0,60,58,15358.444,41,30,99,99,1,20151130,20151231,0,1,0,,7,20111015,1
20170317,107,30,4,0,26,157,9421.074,41,30,99,99,1,20151130,20151231,0,15,31,male,9,20111015,1
20170317,0,0,3,0,51,41,13164.59,41,30,99,99,1,20151130,20151231,0,6,24,female,9,20111015,1
20170311,17,3,1,5,74,73,18383.383,41,30,99,99,1,20151130,20151231,0,1,0,,9,20111015,1
20170311,5,2,0,1,207,141,47719.283,38,30,149,149,0,20151130,20151230,0,22,33,female,7,20111016,1
20170311,3,2,3,5,24,34,8520.619,37,30,149,149,1,20151201,20151231,0,13,27,male,9,20111016,1
20170311,6,1,1,0,10,15,3529.431,40,30,149,149,1,20151201,20160102,0,4,0,,9,20111019,1
20170317,17,1,1,0,55,51,12650.427,40,30,149,149,1,20151201,20160101,0,1,0,,7,20111020,1
20170317,7,2,1,1,23,33,6858.786,38,10,0,0,0,20151201,20151211,0,1,0,,7,20111020,1
20170310,13,0,2,4,140,78,33449,38,30,149,149,0,20151201,20151231,0,13,30,male,9,20111020,1
20170311,7,1,0,3,53,51,13119.246,34,30,149,149,1,20151130,20151231,0,5,27,male,9,20111020,1
20170311,8,4,0,4,59,50,16456.287,34,30,149,149,1,20151130,20160103,0,8,39,male,9,20111020,1
20170311,0,0,0,1,147,111,32397.046,34,30,149,149,1,20151130,20151231,0,13,26,male,9,20111021,1
20170312,13,9,0,2,21,41,6687.439,34,30,149,149,1,20151130,20151231,0,1,0,,7,20111022,1
20170312,11,1,1,0,66,45,15540.67,33,30,149,149,1,20151130,20151231,0,4,22,female,7,20141020,1
20170312,4,0,0,1,153,66,38913.002,33,30,149,149,1,20151130,20151231,0,5,29,male,3,20141020,1
20170312,172,62,21,15,27,185,22189.236,19,30,149,149,1,20151130,20151231,0,15,21,female,3,20141020,1
20170312,14,8,1,2,190,202,48657.87,31,30,149,149,1,20151130,20151231,0,1,0,,7,20141020,1
20170312,9,7,0,0,62,56,16453.587,41,30,129,129,1,20150228,20150401,0,1,20,female,3,20141020,1
20170326,4,3,0,2,127,105,33123.16,41,30,149,149,1,20150228,20150401,0,1,0,,9,20141020,1
20170326.1.0.1.1.18.20.5132.822.41.30.149.149.1.20150228.20150331.0.4.19.female.3.20141020.1
```

The original test data that we have used this project, looks like the following which is provided by KKBOX. From the above data which is normalized into binary data we will predict the Churn of a customer form the above data.

1. To get Started, We will import necessary packages and libraries that will be necessary to make the predictive analytics

“

```
# Dependencies of the Project
import pandas as pd
import numpy as np
import matplotlib.pyplot as pl

# Various tools that are used to process the K-NN
from sklearn import preprocessing,cross_validation,svm
from sklearn.cross_validation import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.externals.six import StringIO
from sklearn import tree
from matplotlib import style
from matplotlib.colors import ListedColormap
```

”

We have used scikit-learn and tools to develop this model. In the second step we got a error which is taken care of it is because of the dependencies but the package is necessary which is automatically updated.

2. Loading the dataset and preprocessing it

“

```
# Dataset that has to be imported
dataset = pd.read_csv('test.csv')
# Male and Female into Binary data
dataset = dataset.replace(['male','female'],
[0,1])
```

”

First we read the dataset which is in the same directory and then process the category data is converted to binary.

Then, we separated the features from the labels

“

```
# Separating value into two objects
#X = dataset.iloc[:, [2,3,4]].values
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 21].values
```

”

3. Now let us see some feature values,

“

```
from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values='NaN', strategy = 'median', axis = 0)
#X[:, [18,19]] = X[:,[18,19]].reshape(1, -1)
imputer = imputer.fit(X[:, [18,19]])
X[:, [18,19]] = imputer.transform(X[:, [18,19]])

print(X[:, 18])
```

”

The result we see is as follows,

```
[1.93352923e-06 1.93352923e-06 1.93352923e-06 ... 1.98310690e-06
 1.98310690e-06 1.98310690e-06]
```

Now time to normalize the data and then print the values of X

“

```
# Normalizing the data from (0 to 1)
X = ((X-X.min())/(X.max()-X.min()))
print(X)
```

”

We see the following

```
[[1.00000000e+00 2.33015061e-06 2.13183992e-06 ... 1.93352923e-06
  2.47888363e-06 9.97054095e-01]
 [9.99999950e-01 2.03268458e-06 2.03268458e-06 ... 1.93352923e-06
  2.28057294e-06 9.97054243e-01]
 [1.00000000e+00 4.51156821e-06 2.08226225e-06 ... 1.93352923e-06
  2.47888363e-06 9.97054293e-01]
 ...
 [9.99999306e-01 2.47888363e-06 2.08226225e-06 ... 1.98310690e-06
  2.08226225e-06 9.99032541e-01]
 [9.99999306e-01 2.42930596e-06 1.93352923e-06 ... 1.98310690e-06
  2.08226225e-06 9.99032541e-01]
 [9.99998959e-01 1.93352923e-06 1.93352923e-06 ... 1.98310690e-06
  2.08226225e-06 9.99032541e-01]]
```

4. Preparing the training and testing set

“

```
# Split the data into training and testing part
from sklearn.cross_validation import train_test_split
```

```
# Giving 20% of total dataset in testing part and remaining in training part (80%)

x_train, x_test, y_train, y_test = train_test_split(X,y,test_size=0.20, random_state = 0)
```

”

5. Applying K-NN and defined distance metrics, which is minkowski where p=2. It is also known as Euclidean Distance equation.

“

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(x_train, y_train)
```

”

We get the output as follows

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
        metric_params=None, n_jobs=1, n_neighbors=5, p=2,
        weights='uniform')
```

6. Implementation of KNN

“

```
# Giving X_test in predicting the result of y_pred
y_pred = classifier.predict(x_test)
print(y_pred[:])
np.savetxt("y_prediction.csv",y_pred,delimiter=',')
```

”

7. Calculating the Accuracy

“

```
# Calculating the accuracy of the algorithm by comparing the actual target feature and what we
received by our machine learning model
accuracy = classifier.score(x_test, y_test)
print("This is the accuracy of our algorithm")
print(accuracy)
```

”

This is the accuracy of our algorithm
0.934

8. Validating the result by Confusion Matrix

“

```
cm= confusion_matrix(y_test, y_pred)
print("The confusion matrix is described below.")
print(cm)
```

”

We get the result as follows
The confusion matrix is described below.
[[1815 46]
 [86 53]]

9. Testing
“

```
X_test2 = np.array([0.999999,2.08226e-06,1.93353e-06, 1.93353e-06,    1.93353e-06,
2.13184e-06,    2.231e-06,      4.77028e-05,    3.7679e-06,     3.42086e-06,    9.3206e-06,
9.3206e-06,     1.98311e-06,    0.999032,       0.999037,       1.93353e-06,    1.98311e-06,
1.93353e-06,    1.93353e-06,    2.13184e-06,    0.999538
])
X_test2 = X_test2.reshape(1, -1)
y_pred2 = classifier.predict(X_test2)

print("We inserted query into model and it predicted following decision")
print(y_pred2)
```

”

We inserted query into model and it predicted following decision
[0]

Therefore we can say that from the given query the customer will not churn.

Looking at matrix, we can say that the matrix (diagonally) the sum of (106+1479) = 1585, this is the accuracy of our model, which means that out of 2000 dataset which we have given in the testing phase, 1585 data was predicted correctly. And remaining dataset which is (69+346) = 415 dataset were predicted wrong by our model (which is the remaining i.e 1.0000 - 0.7925 = 0.2175%)

5.6 Conclusion

In this assignment, the team reported how we implemented prediction model for solving the predictive analytics solution defined for the real problem of KKBOX music stream service provider. We first reviewed the problem and explained the analytics solution for the real-world problem. Then, we reviewed all the features including descriptive and target features. We determined that which feature is considered as a categorical feature and which one is considered as continuous based on their nature. The prediction model is described and trained by data. In addition, the error of the model in prediction churn is provided. The accuracy of the model is close to 80% which is a good accuracy based on the fact that we used a portion of the data for the model. It can be expected that using the whole data increases the accuracy of the model. KKBOX can use this model to predict the churn of the subscribed users.

5.7 Individual Contributions

Arjun Aneja	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
William Clark	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
Sumati Kulkarni	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
Babak Maleki Shoja	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
Venkatesh Reedy Pala	Provided invaluable contributions to the completion of the tasks assigned to the group Drafted Report Reviewed Report
Vishwa Patel	Provided invaluable contributions to the completion of the tasks assigned to the group for this project Drafted report Reviewed Report

5.8 Team Summary

This phase of the project gave us the insight in implementing predictive models and evaluate them. Moreover, we understood we need to exclude some of the features to get the results. The importance of the evaluation for the model was investigated and it was a great experience to see how machine learning can solve real-world problems and challenges.

5.9 Implementation of Code is done as follows in the Jupyter Notebook

In [1]:
```
# Depedencies of the Project
import pandas as pd
import numpy as np
import matplotlib.pyplot as pl
```

In [2]:
```
# Various tools that are used to process the K-NN
from sklearn import preprocessing,cross_validation,svm
from sklearn.cross_validation import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.externals.six import StringIO
from sklearn import tree
from matplotlib import style
from matplotlib.colors import ListedColormap
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/cross_validation.py:41:
DeprecationWarning: This module was deprecated in version 0.18 in favor
of the model_selection module into which all the refactored classes and
functions are moved. Also note that the interface of the new CV iterato
rs are different from that of this module. This module will be removed
in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [3]:
```
style.use("ggplot")
```

In [4]:
```
# Dataset that has to be imported
dataset = pd.read_csv('test.csv')
# Male and Female into Binary data
dataset = dataset.replace(['male','female'],
[0,1])
```

In [ ]:
```
# Seperating value into two objects
#X = dataset.iloc[:, [2,3,4]].values
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 21].values
```

In [12]:
```
# Replacing nan value with median in gender column which is 18th column

from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values='NaN', strategy = 'median', axis = 0)
#X[:, [18,19]] = X[:,[18,19]].reshape(1, -1)
imputer = imputer.fit(X[:, [18,19]])
X[:, [18,19]] = imputer.transform(X[:, [18,19]])

print(X[:, 18])
```

```
[1.93352923e-06 1.93352923e-06 1.93352923e-06 ... 1.98310690e-06
 1.98310690e-06 1.98310690e-06]
```

In [13]:
```
# Normalizing the data from (0 to 1)
X = ((X-X.min())/(X.max()-X.min()))
print(X)
```

```
[[1.00000000e+00 2.33015061e-06 2.13183992e-06 ... 1.93352923e-06
  2.47888363e-06 9.97054095e-01]
 [9.99999950e-01 2.03268458e-06 2.03268458e-06 ... 1.93352923e-06
  2.28057294e-06 9.97054243e-01]
 [1.00000000e+00 4.51156821e-06 2.08226225e-06 ... 1.93352923e-06
  2.47888363e-06 9.97054293e-01]
 ...
 [9.99999306e-01 2.47888363e-06 2.08226225e-06 ... 1.98310690e-06
  2.08226225e-06 9.99032541e-01]
 [9.99999306e-01 2.42930596e-06 1.93352923e-06 ... 1.98310690e-06
  2.08226225e-06 9.99032541e-01]
 [9.99998959e-01 1.93352923e-06 1.93352923e-06 ... 1.98310690e-06
  2.08226225e-06 9.99032541e-01]]
```

In [14]:
```
# Split the data into training and testing part
from sklearn.cross_validation import train_test_split
```

In [15]:
```
# Giving 20% of total dataset in testing part and remaining in training
 part (80%)

x_train, x_test, y_train, y_test = train_test_split(X,y,test_size=0.20,
random_state = 0)
```

In [7]:
```
# Applying K-Neighbour classifier algoritham to our dataset with 5 surro
unding neighbors and default matrix
# and value of p=2 because of this it uses the Euclidean Distance algori
tham (2)
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski',
 p = 2)
classifier.fit(x_train, y_train)
```

Out[7]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

In [16]:
```
# Giving X_test in predicting the result of y_pred
y_pred = classifier.predict(x_test)
print(y_pred[:])
np.savetxt("y_prediction.csv",y_pred,delimiter=',')
```

```
[0 0 0 ... 0 0 0]
```

In [9]:
```
# Calculating the accuracy of the algoritham by comparing the actual tar
get feature and what we received by our machine learning model
accuracy = classifier.score(x_test, y_test)
print("This is the accuracy of our algoritham")
print(accuracy)
```

```
This is the accuracy of our algoritham
0.934
```

In [10]:
```
# Calculating confusion matrix for machine learning prediction and
cm= confusion_matrix(y_test, y_pred)
print("The confusion matrix is described below.")
print(cm)
```

```
The confusion matrix is described below.
[[1815   46]
 [  86   53]]
```

In [11]:
```
X_test2 = np.array([0.999999,2.08226e-06,1.93353e-06,   1.93353e-06,
1.93353e-06,    2.13184e-06,    2.231e-06,      4.77028e-05,    3.7679e-
06,     3.42086e-06,    9.3206e-06,     9.3206e-06,     1.98311e-06,
0.999032,       0.999037,       1.93353e-06,    1.98311e-06,    1.93353e
-06,    1.93353e-06,    2.13184e-06,    0.999538
])
X_test2 = X_test2.reshape(1, -1)
y_pred2 = classifier.predict(X_test2)

print("We inserted query into model and it predicted following decision"
)
print(y_pred2)
```

```
We inserted query into model and it predicted following decision
[0]
```

5.10 References

- Kelleher, John, Namee, Brian Mac, Arcy, Aoife D'. (2015). Fundamentals of Machine Learning for Predictive Data Analytics. The MIT Press Cambridge, Massachusetts London, England
- Kaggle, WSGM- KKBox's Churn Prediction Challenge https://www.kaggle.com/c/kkbox-churn-prediction-challenge#description 2018 Kaggle Inc

Microsoft Azure: Blood Donation & Energy Efficiency

6.1 Learning Goals

In this assignment, the team implemented **two** models of machine learning with different datasets and learning algorithms using Microsoft Azure. Microsoft Azure is a strong and easy to use tool for machine learning. In this assignment we learned how to select dataset, get insight about the problem by analyzing the dataset, selecting different algorithms to run, learned how to tune the parameters, and how to evaluate the results.

6.2 Introduction to Microsoft Azure

Microsoft Azure Machine Learning Studio is a collaborative, drag-and-drop tool you can use to build, test, and deploy predictive analytics solutions on your data. Machine Learning Studio publishes models as web services that can easily be consumed by custom apps or BI tools such as Excel.

Machine Learning Studio is where data science, predictive analytics, cloud resources, and your data meet.

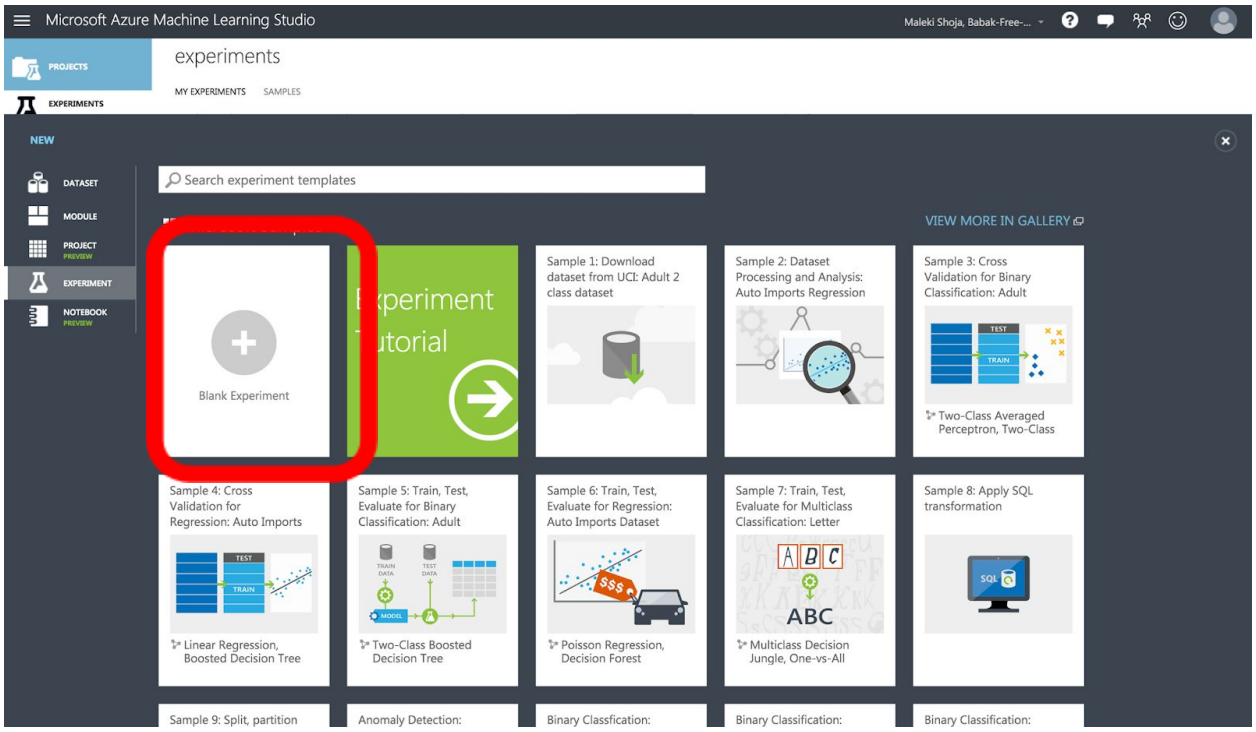
To develop a predictive analysis model, you typically use data from one or more sources, transform and analyze that data through various data manipulation and statistical functions, and generate a set of results. Developing a model like this is an iterative process. As you modify the various functions and their parameters, your results converge until you are satisfied that you have a trained, effective model.

Azure Machine Learning Studio gives you an interactive, visual workspace to easily build, test, and iterate on a predictive analysis model. You drag-and-drop ***datasets*** and analysis ***modules*** onto an interactive canvas, connecting them together to form an ***experiment***, which you run in Machine Learning Studio. To iterate on your model design, you edit the experiment, save a copy if desired, and run it again. When you're ready, you can convert your ***training experiment*** to a ***predictive experiment***, and then publish it as a ***web service*** so that your model can be accessed by others.

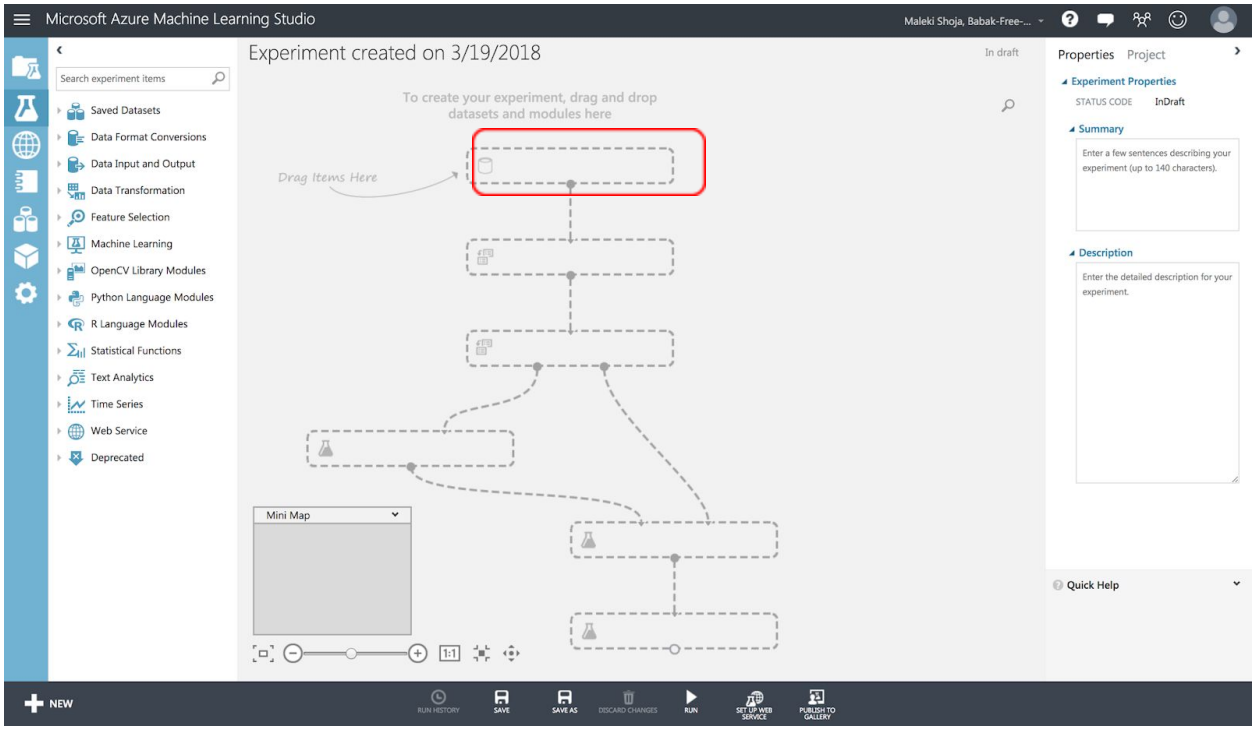
There is no programming required, just visually connecting datasets and modules to construct your predictive analysis model.

6.3. Starting An Experiment on Microsoft Azure

We open the Microsoft Azure and the first screen (after skipping the tour) is the following.



To start an Experiment, we click on “Blank Experiment” shown in the red box and a blank experiment will be opened as follows.



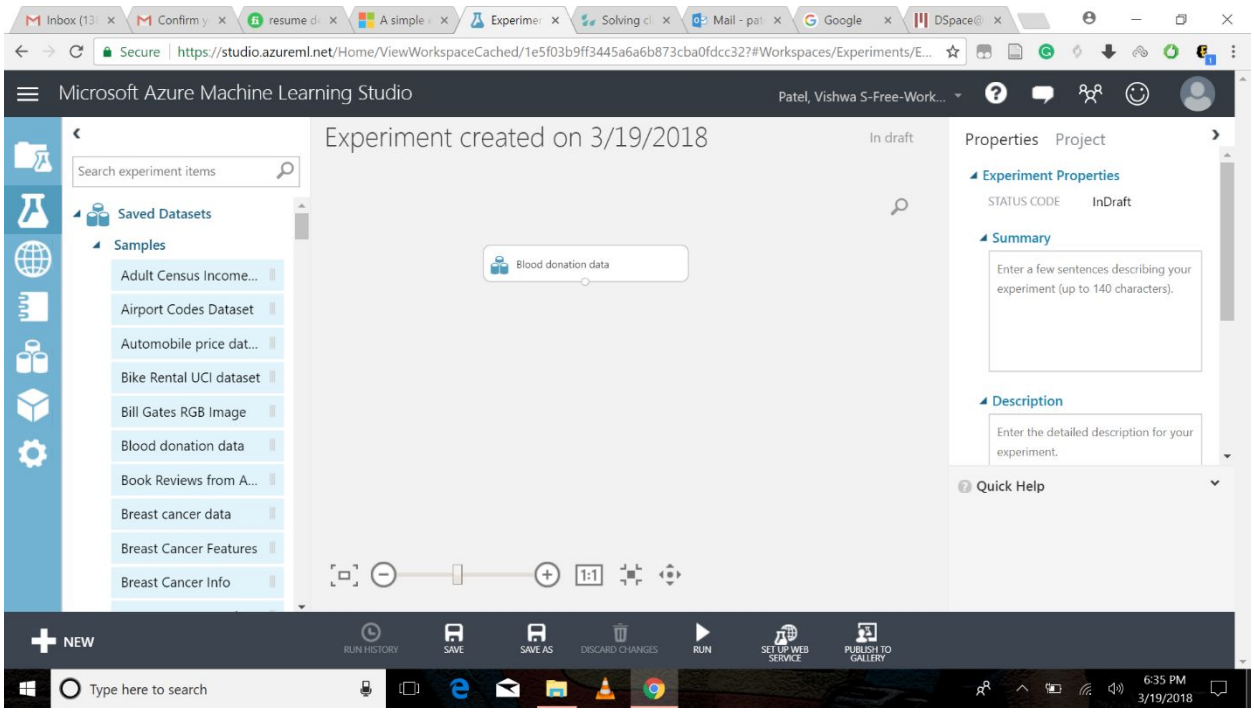
As shown in the image, the general schema for running a machine learning experiment is depicted. Now, step by step, we show how we should develop and implement a machine learning experiment.

It should be noted that in the right panel, there is a place to record the summary of the experiment and also description of the experiment.

6.4. Blood Donation Dataset Implementation Steps

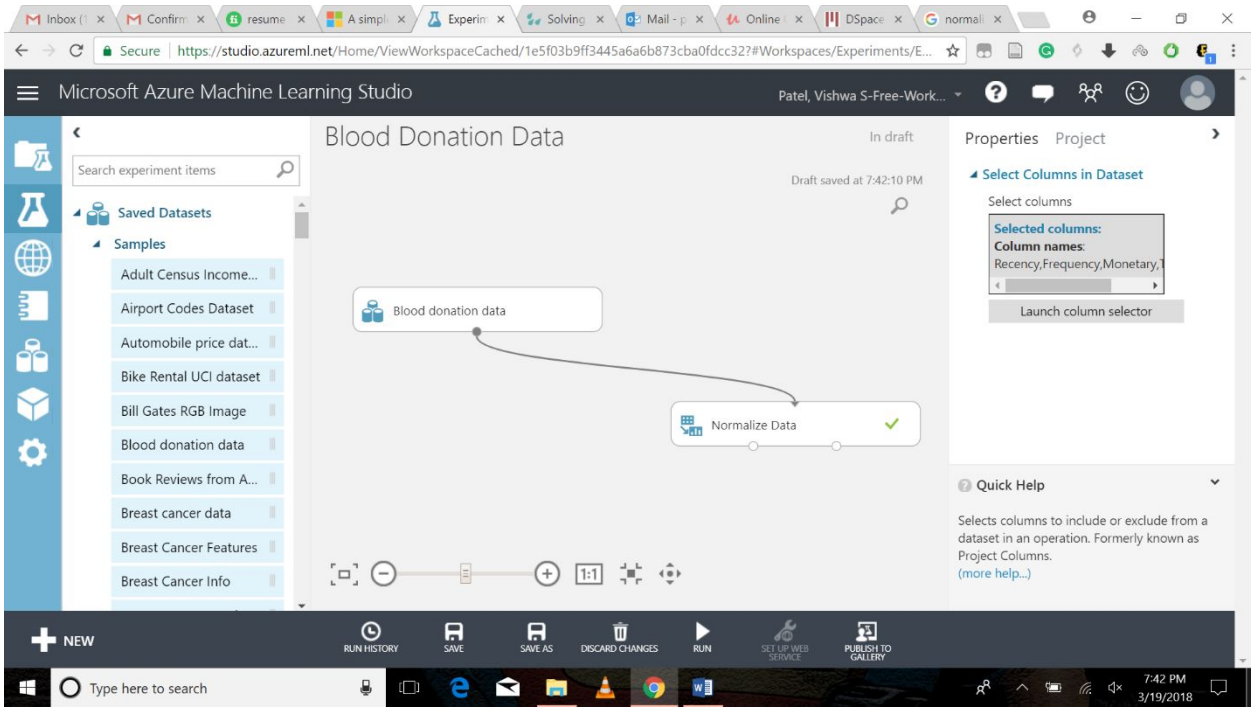
This dataset is regarded to predicting the class of donation (target feature) using descriptive features including Recency, Frequency, Monetary and Time for 748 data instances. The steps we took for this part are described as follows.

Step-1



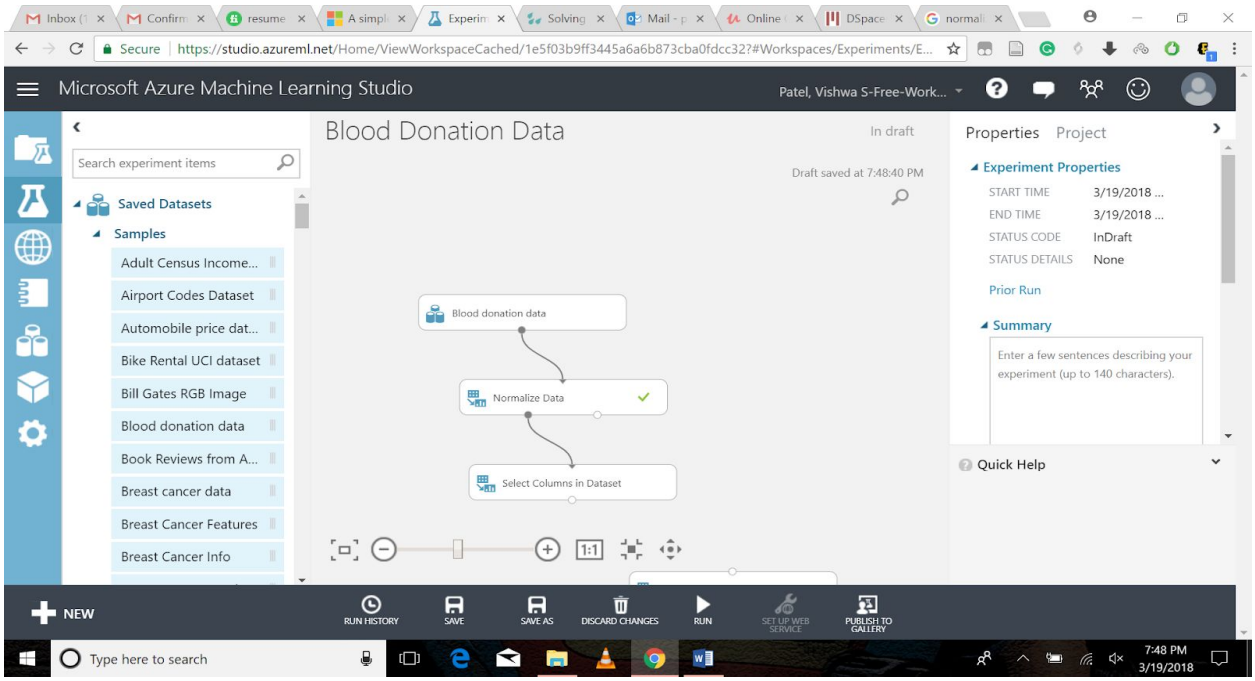
Here, for our group project we have selected **Blood Donation Data** dataset to perform Machine Learning algorithm on it as the first dataset. We can get the sample dataset of Blood Donation data from left panel under Saved Dataset section.

Step-2

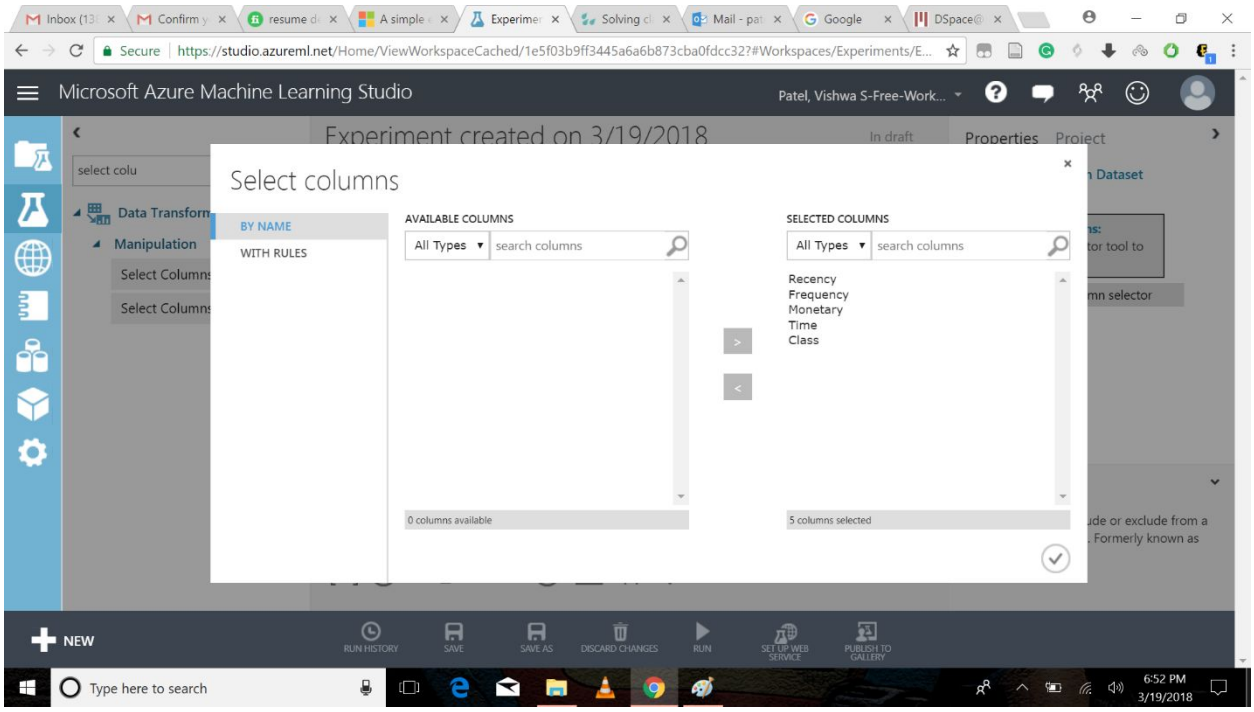


The next step is to normalize the 3 columns, given in the dataset. Hence we will use MINMAX function in order to perform Normalize the required columns in the given dataset.

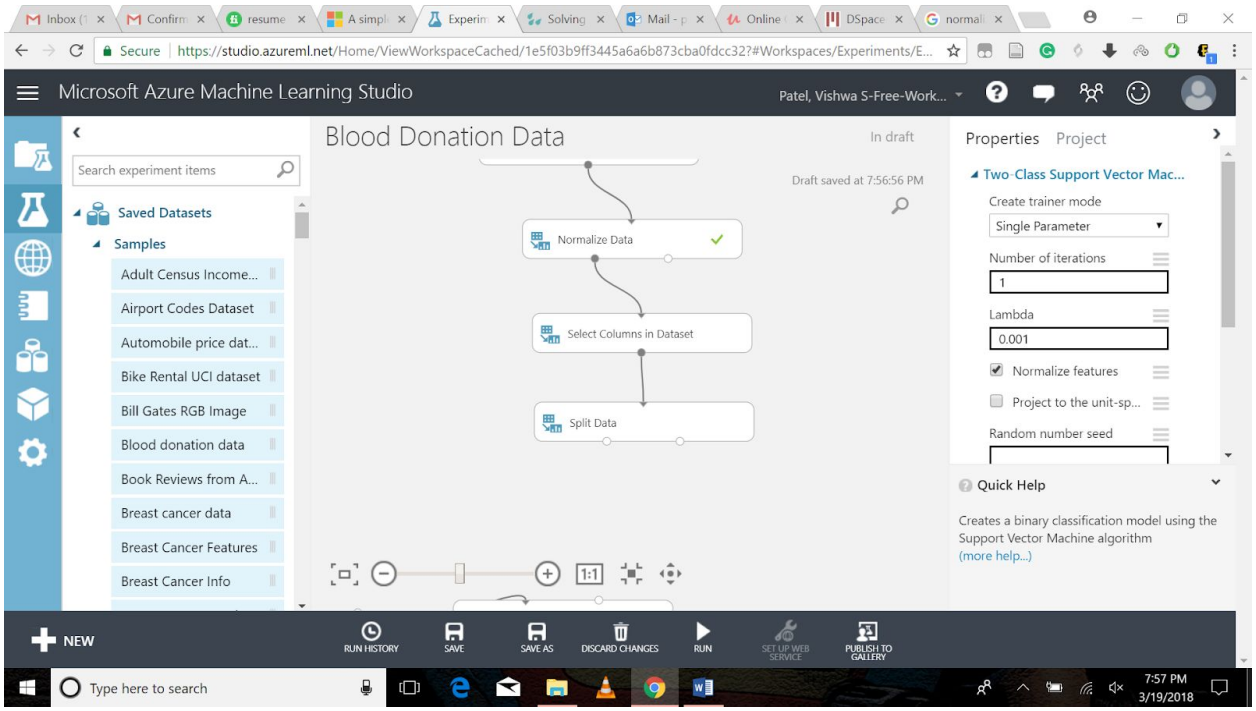
Step-3



Here in our dataset, there are no missing values present in the dataset. Hence, we will not use any missing values functions (i.e Mean, Median, Most frequent etc). Now, we will just select the columns from the earlier step, so that we can have a final dataset (with normalized dataset). In the next screenshot, we will select all the columns which are available in the dataset. We will do so, because in certain other datasets there will be many columns in dataset, and sometimes some columns are not much useful in the prediction analysis. Hence, if you wish to drop that particular column, then you can drop the column in this section.

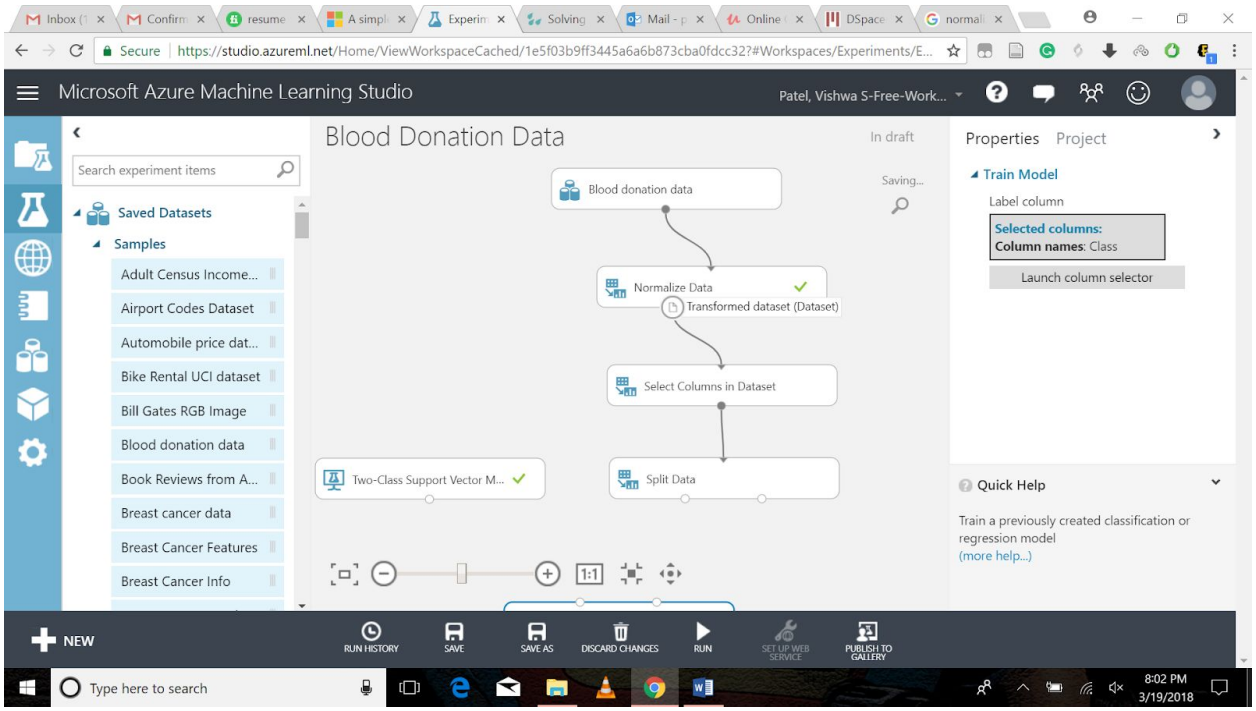


Step-4



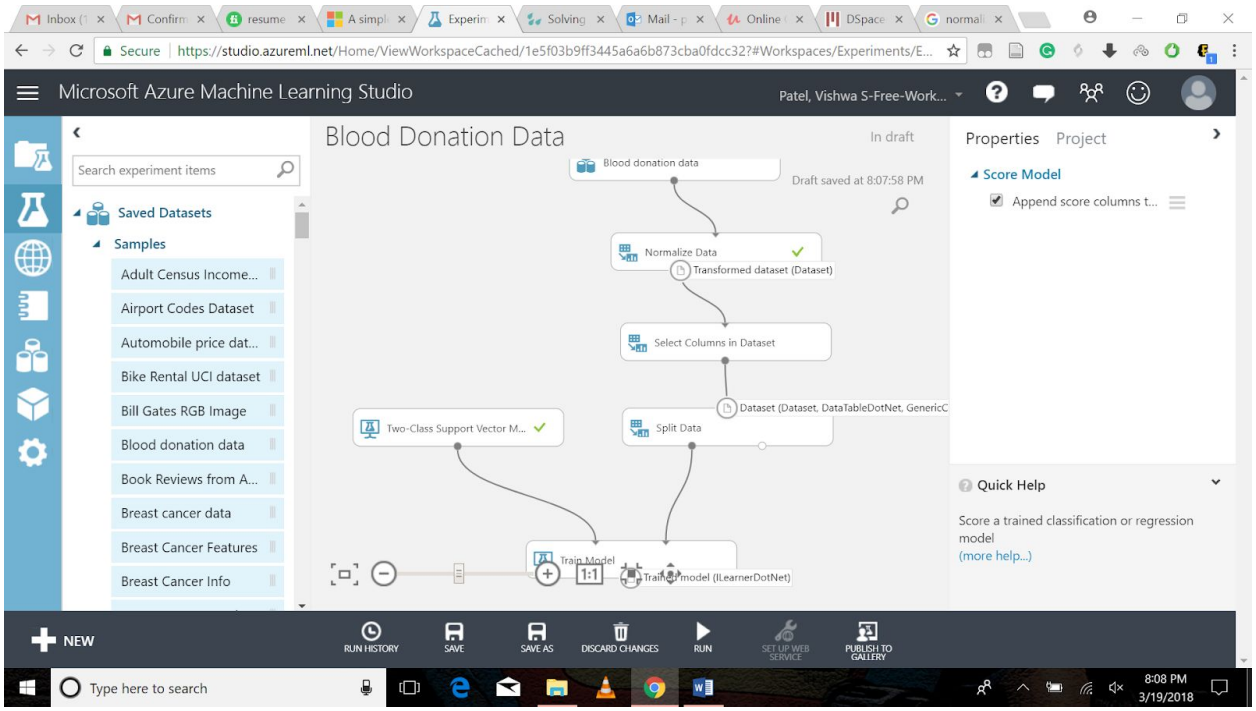
Now, the next step is to split the given dataset, into two different parts. Here, while selecting the split data, we will declare “Fraction of rows in first output database” equal to “0.75”. This you can adjust with any number between 0 to 1 because the more number you choose, that percentage of data (i.e 0.75 = 75%) will be given for training model and rest of it (0.25%) will be given for testing purpose.

Step-5



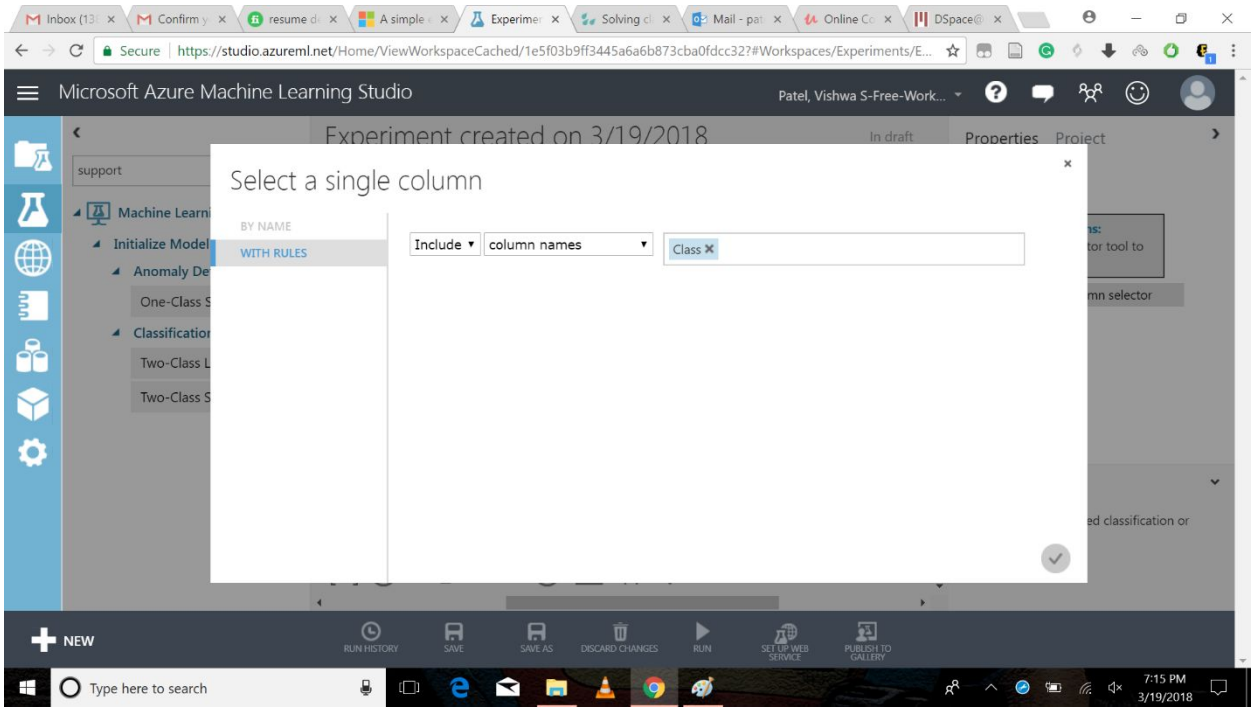
We have selected the “Two class Support Vector Machine” which is the classification algorithm. Because we are predicting classification of 0 or 1. This will create binary classification model using Support Vector Machine Algorithm.

Step-6



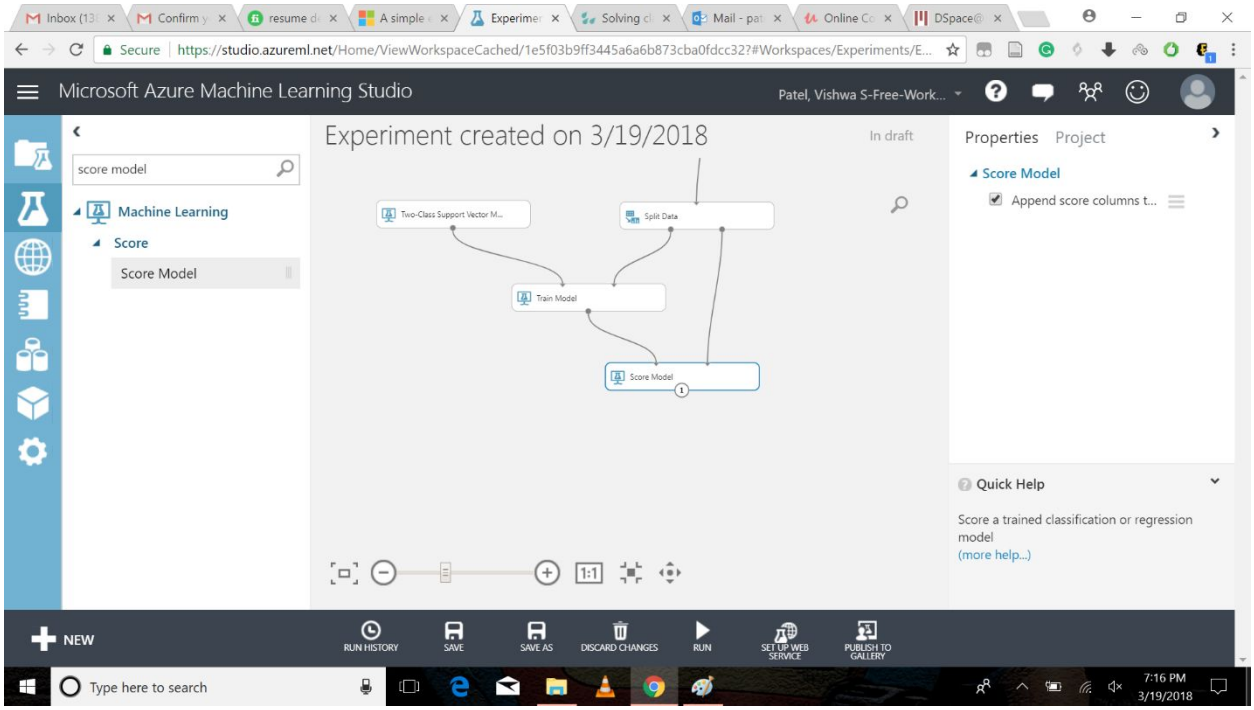
Next step is to train the model using the Two Class Support Vector Machine algorithm and Split Data. In this step, our model will train by using 75% of dataset.

Step-7



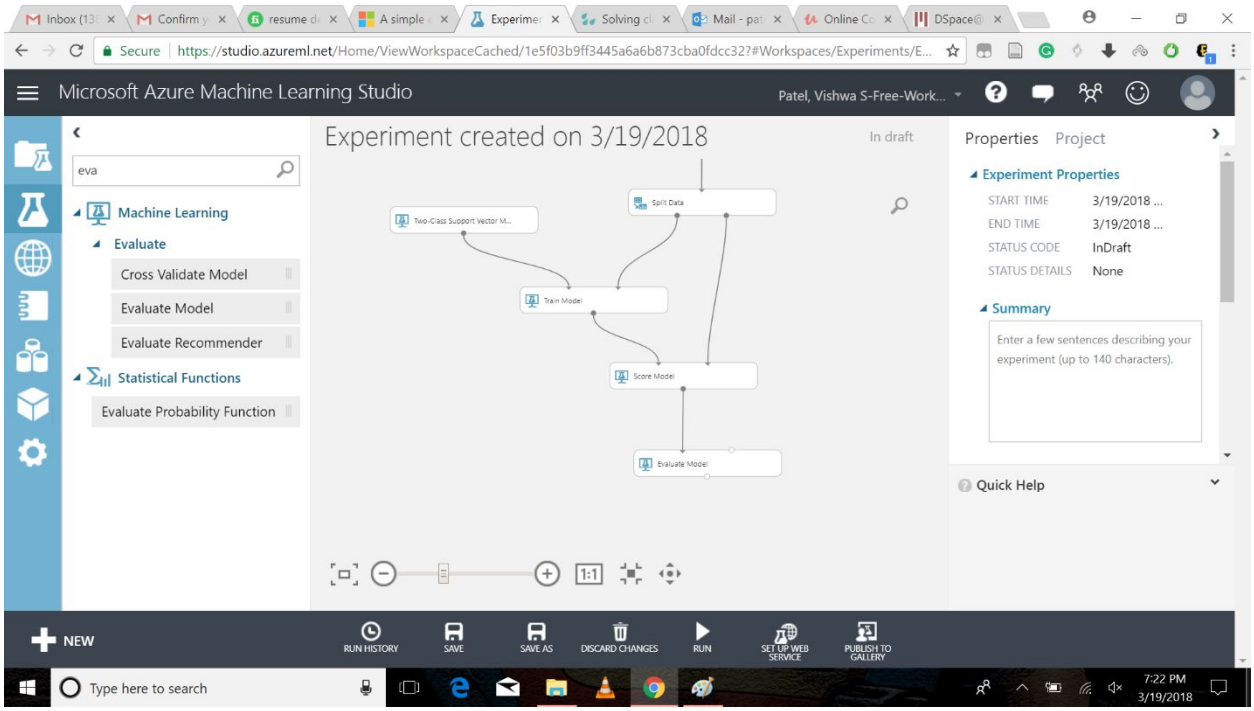
Now, click on the Train model, and launch column sector, select the Target Feature column in our case “Class”. Click ok and proceed.

Step-8



Now, We will score the model. This means that once our model is trained with 75% of dataset, its time to give 25% which is remaining data from dataset and the output of the training model. At the end of this, our model will predict the data based on what model learned till now.

Step-9



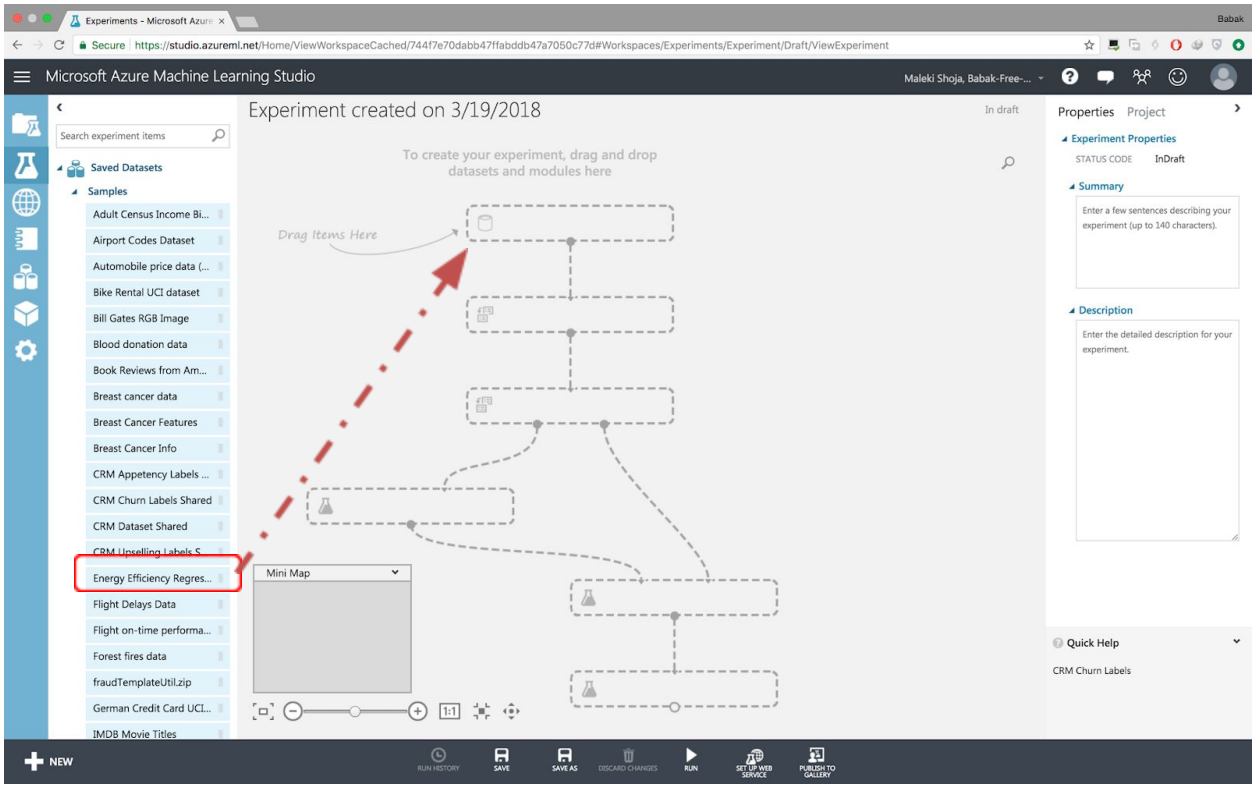
Next and final step is to use Evaluate Model. By using this, we are evaluating the model which predicted the accuracy of our dataset, and many other details associated with it. Attach Evaluate Model with Score Model. Once everything is done, then click **RUN** button at the bottom of page and it will show the results when all the steps are once processed.

6.5 Energy Efficiency Implementation Steps

This dataset is provided to predict either Heating Load or Cooling Load as the target feature and we selected Heating Load for our project assignment. Descriptive features are Relative Compactness, Surface Area, Wall Area, Roof Area, Overall Height, Orientation, Glazing Area, and Glazing Area Distribution. The next steps are described as follows.

Step 1: Add a dataset from samples

First step is to drag a dataset to the first node shown in red box in the picture. For this report, we selected "Energy Efficiency Regression dataset" from saved Datasets as shown in the following image. We drag it to the first node as demonstrated in the image.

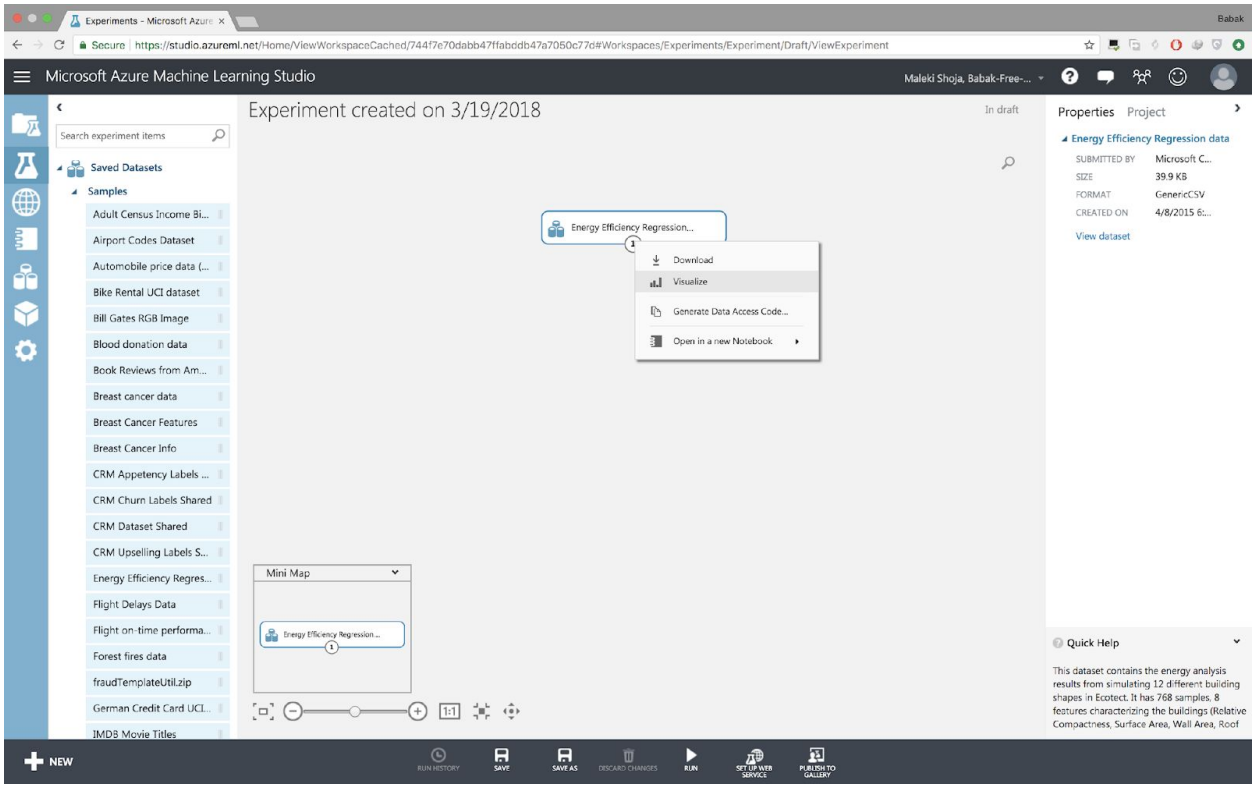


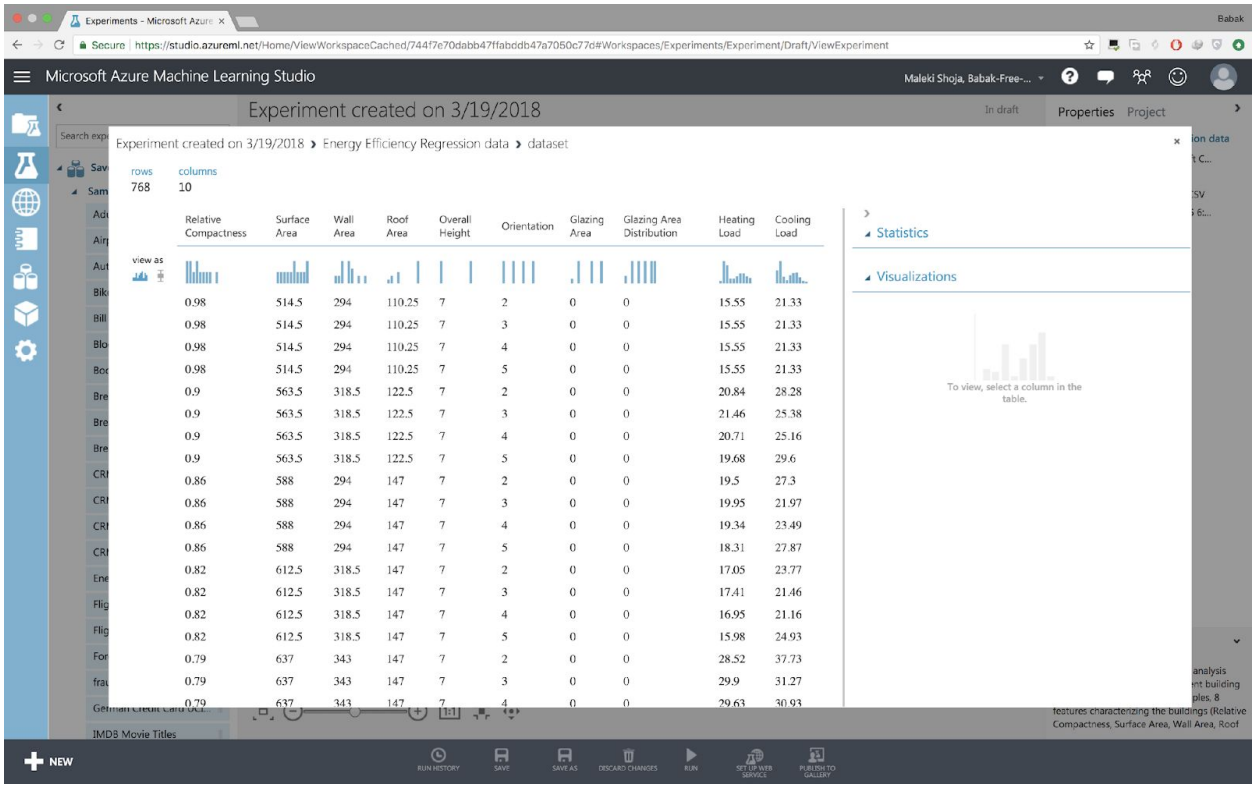
Here we can click on the dragged dataset, there is an option under "Visualize". By selecting this option, features and values in the dataset as well as statistics, charts and diagrams are demonstrated and by clicking on a feature, corresponding statistics and charts will be shown on the right panel. These are shown in the next three images. For instance, we selected the "Heating Load" column and the statistics are:

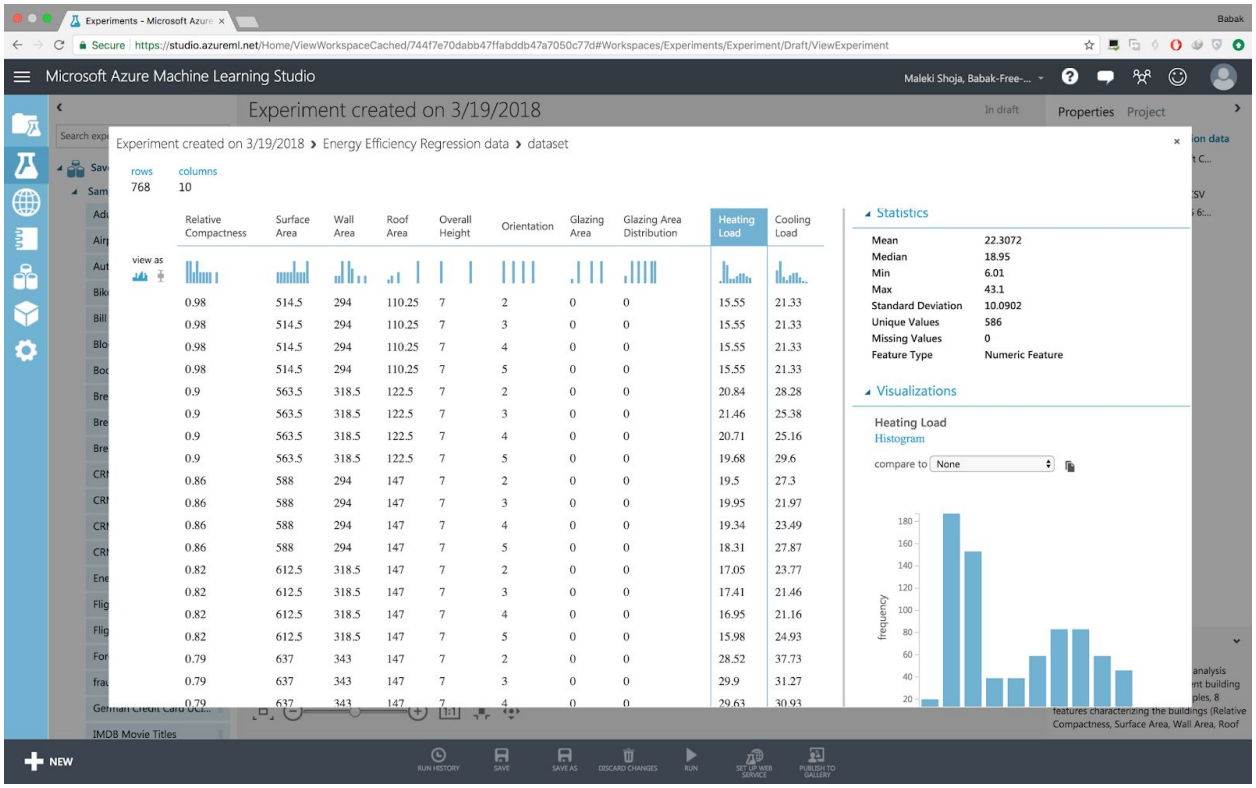
Mean	22.3072
Median	18.95
Min	6.01
Max	43.1
Standard Deviation	10.0902
Unique Values	586

Missing Values	0
Feature Type	Numeric Feature

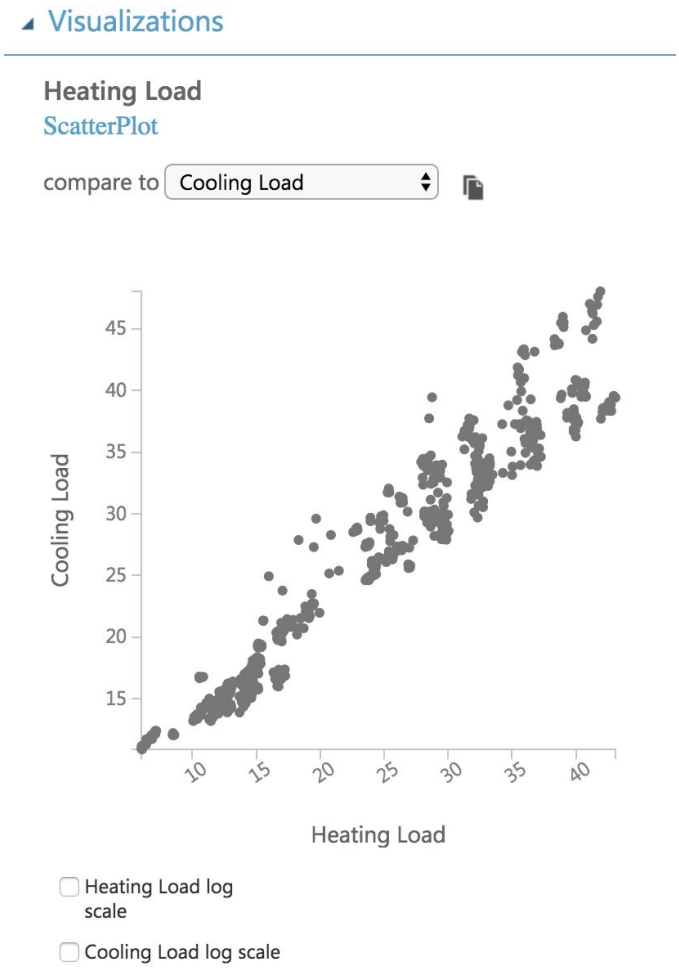
It also provides visualization which is demonstrated at the right panel.




Experiments - Microsoft Azure
Microsoft Azure Machine Learning Studio
Experiment created on 3/19/2018
Experiment created on 3/19/2018 › Energy Efficiency Regression data › dataset
rows 768
columns 10
Relative Compactness
Surface Area
Wall Area
Roof Area
Overall Height
Orientation
Glazing Area
Glazing Area Distribution
Heating Load
Cooling Load
Statistics
Visualizations
To view, select a column in the table.
NEW


Experiments - Microsoft Azure
Microsoft Azure Machine Learning Studio
Experiment created on 3/19/2018
Experiment created on 3/19/2018 › Energy Efficiency Regression data › dataset
rows 768
columns 10
Statistics
Mean 22.3072
Median 18.95
Min 6.01
Max 43.1
Standard Deviation 10.0902
Unique Values 586
Missing Values 0
Feature Type Numeric Feature
Visualizations
Heating Load
Histogram
compare to None
frequency
NEW

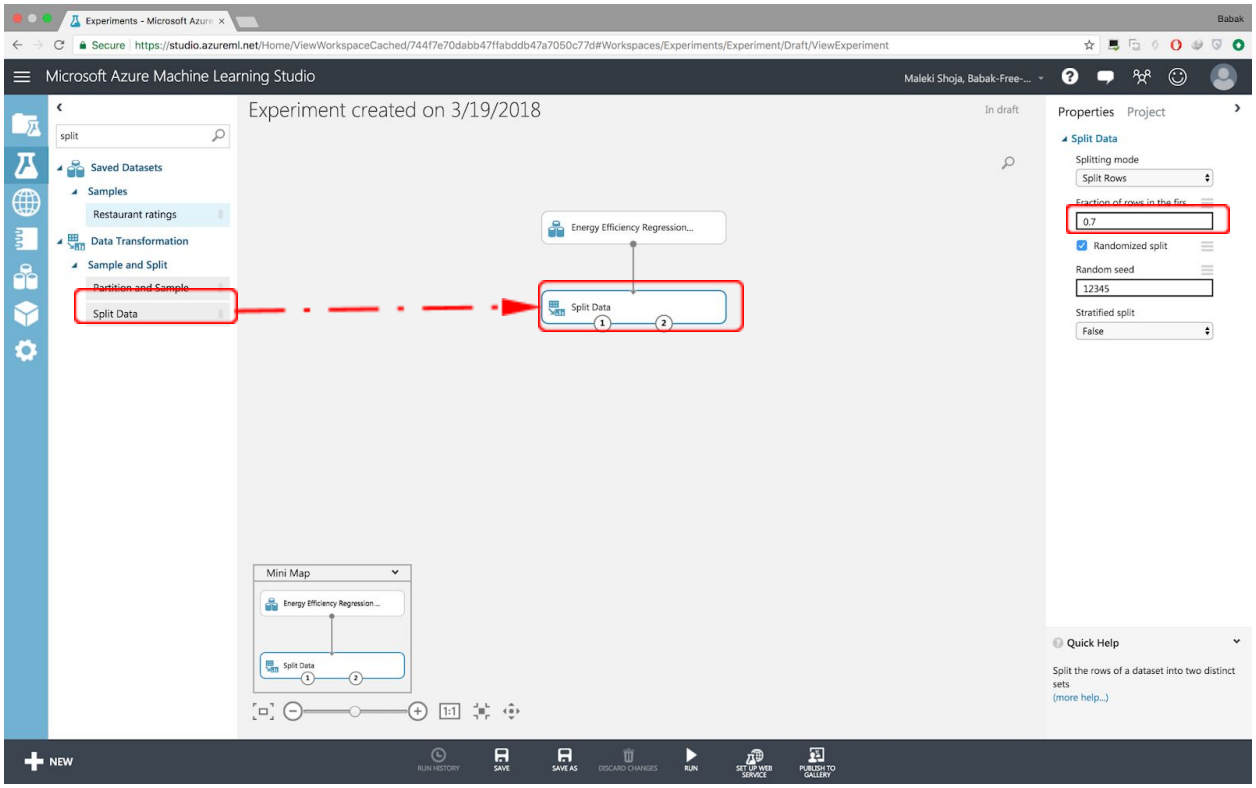
Under “Visualization” pane, you can select another feature and it provides a scatter plot or another proper chart to compare two features. We selected Heating Load and Cooling Load and the comparison is as follows.



This is a great tool to get insight about the data.

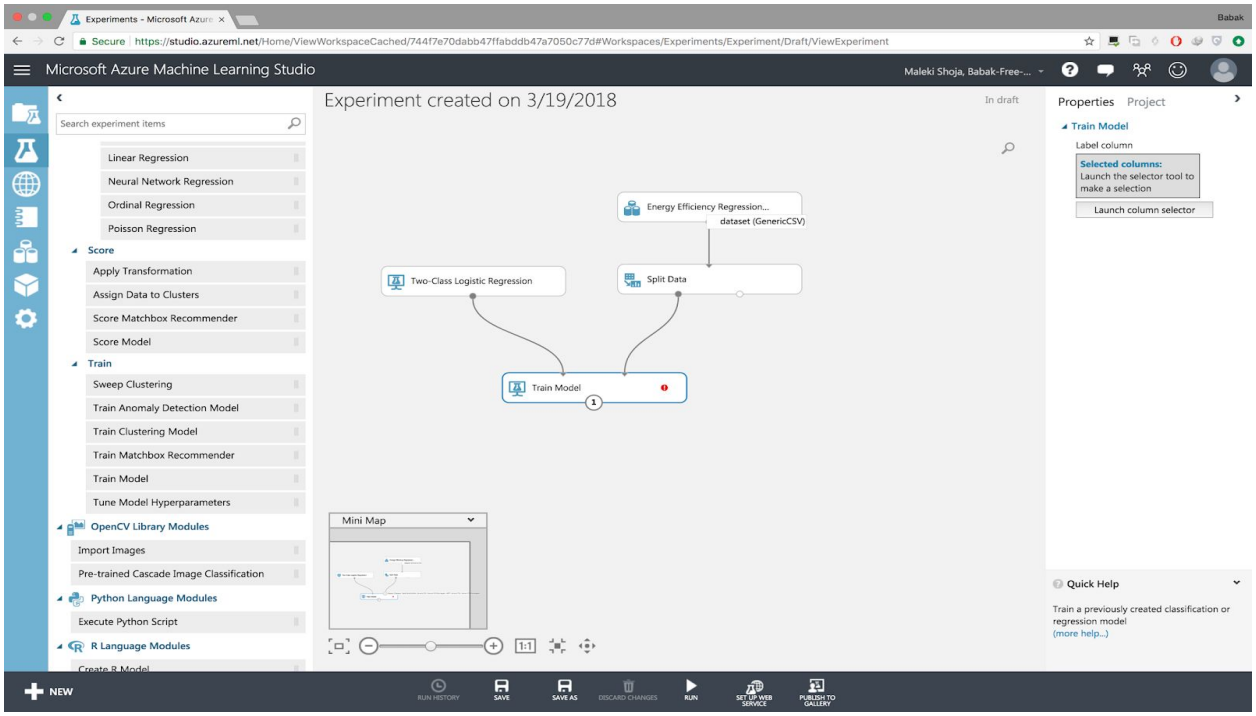
Step 2: Split dataset

In this step, we divide the data into two parts including training set and test set, respectively 70% and 30%. To do so, we can find “Split Data” under Data Transformation->Sample and Split at the left panel or simply by searching the word “Split”. Then we drag and drop it as the next step as shown below. Then we need to set “Fraction of rows in the first output dataset” to 0.7 (which demonstrates 70%. Selecting “Randomized Split”, the data will be split randomly. We also connect the first and second node.

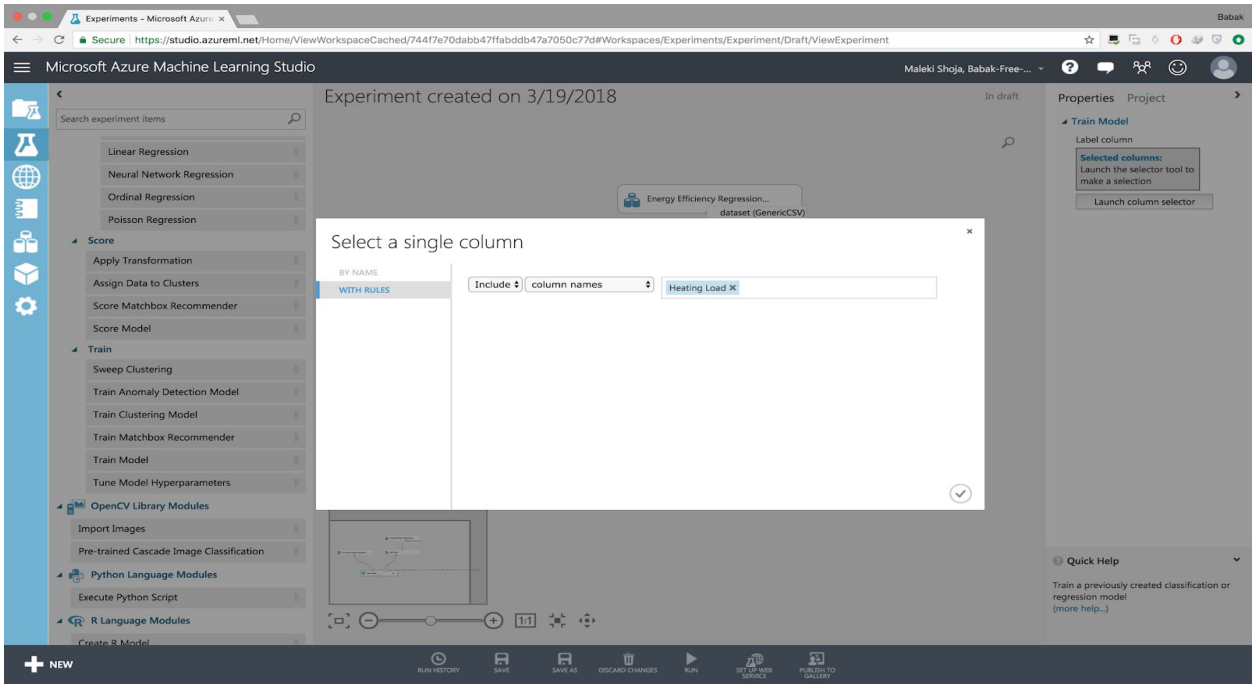


Step 3: Select an ML algorithm

In this step, we select an algorithm to train the model with the training dataset. First we need to select the algorithm under Machine Learning in the left panel. We selected Two-Class Logistic Regression algorithm and we drag and drop it to working space. Also, under Train, we need to drag and drop Train Model node.

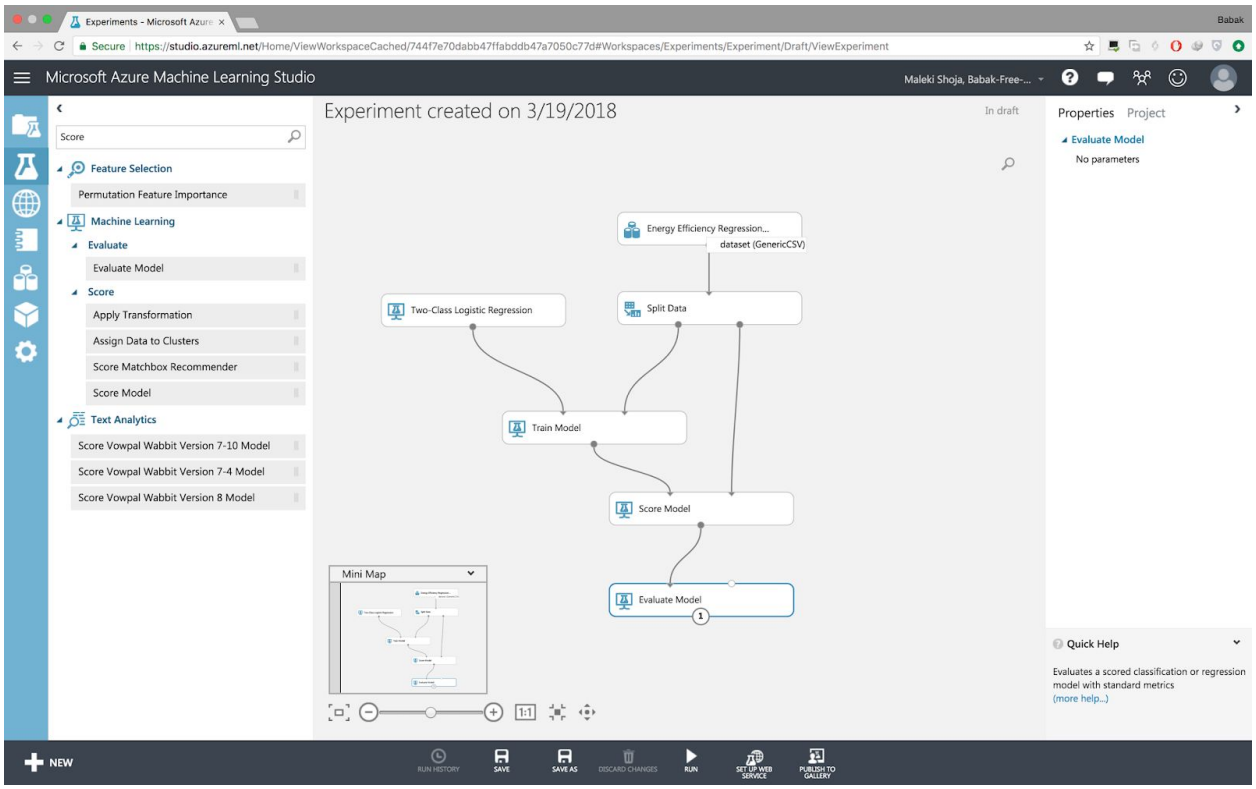


Now we need to click on Launch column selector at the right panel and select target feature which is Heating Load here.



Step 4: Make Predictions

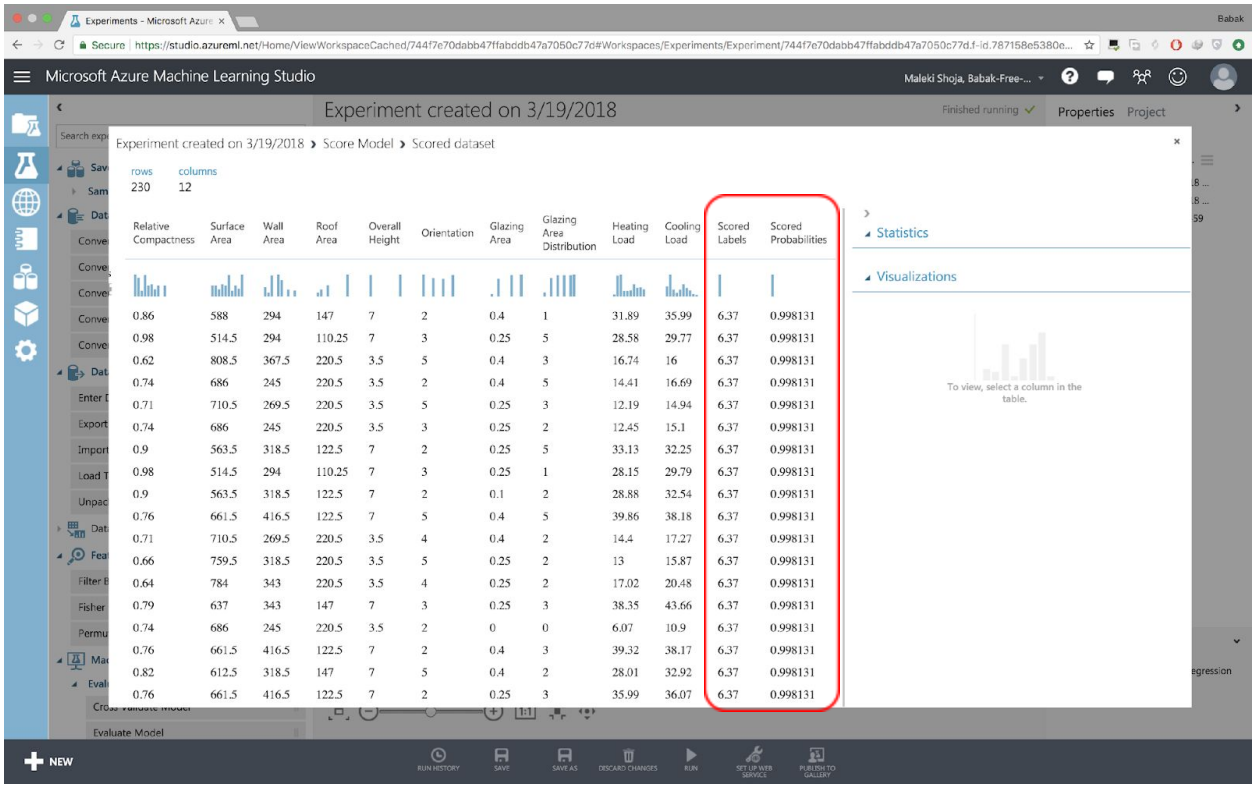
Now we are ready to run the experiment to get our predictions. This can take a minute or two. To do so, we need to add two nodes and drag and connect them as following image.



Step 5: Run the experiment

To do this, we hit the RUN button and wait for the model to run which may take one or two minutes.

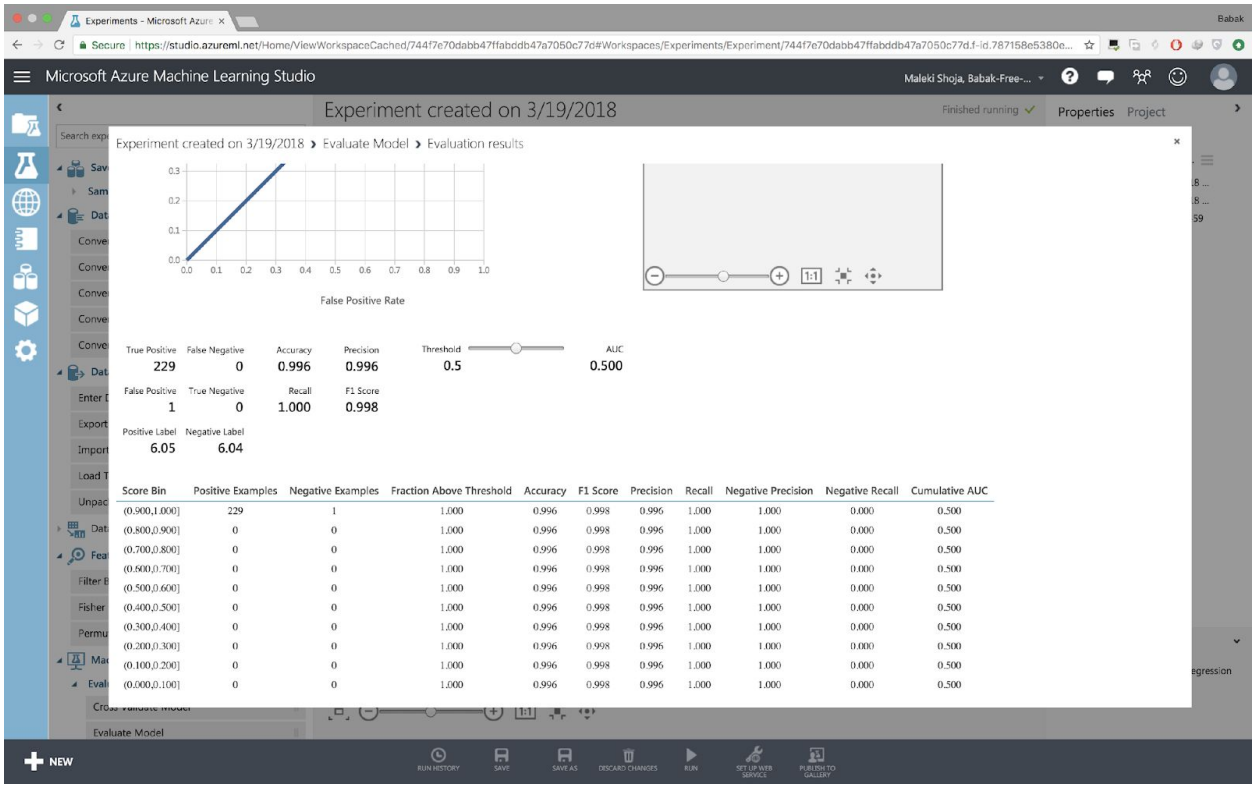
After the model is run, first we visualize the statistics in the Score Model similar to Step 1. The results is as follows.



Relative Compactness	Surface Area	Wall Area	Roof Area	Overall Height	Orientation	Glazing Area	Glazing Area Distribution	Heating Load	Cooling Load	Scored Labels	Scored Probabilities
0.86	588	294	147	7	2	0.4	1	31.89	35.99	6.37	0.998131
0.98	514.5	294	110.25	7	3	0.25	5	28.58	29.77	6.37	0.998131
0.62	808.5	367.5	220.5	3.5	5	0.4	3	16.74	16	6.37	0.998131
0.74	686	245	220.5	3.5	2	0.4	5	14.41	16.69	6.37	0.998131
0.71	710.5	269.5	220.5	3.5	5	0.25	3	12.19	14.94	6.37	0.998131
0.74	686	245	220.5	3.5	3	0.25	2	12.45	15.1	6.37	0.998131
0.9	563.5	318.5	122.5	7	2	0.25	5	33.13	32.25	6.37	0.998131
0.98	514.5	294	110.25	7	3	0.25	1	28.15	29.79	6.37	0.998131
0.9	563.5	318.5	122.5	7	2	0.1	2	28.88	32.54	6.37	0.998131
0.76	661.5	416.5	122.5	7	5	0.4	5	39.86	38.18	6.37	0.998131
0.71	710.5	269.5	220.5	3.5	4	0.4	2	14.4	17.27	6.37	0.998131
0.66	759.5	318.5	220.5	3.5	5	0.25	2	13	15.87	6.37	0.998131
0.64	784	343	220.5	3.5	4	0.25	2	17.02	20.48	6.37	0.998131
0.79	637	343	147	7	3	0.25	3	38.35	43.66	6.37	0.998131
0.74	686	245	220.5	3.5	2	0	0	6.07	10.9	6.37	0.998131
0.76	661.5	416.5	122.5	7	2	0.4	3	39.32	38.17	6.37	0.998131
0.82	612.5	318.5	147	7	5	0.4	2	28.01	32.92	6.37	0.998131
0.76	661.5	416.5	122.5	7	2	0.25	3	35.99	36.07	6.37	0.998131

As highlighted in the red box, Scored Labels shows the predicted values and Scored Probabilities shows the goodness of fit for the training model.

To see how good the model is on the test dataset, we visualize the results in Evaluate Model node and the results is shown as follows.



As we can see, this model works perfect using the algorithm we selected and all of the test data resulted in the True Positive. The accuracy is 0.996.

6.6 Conclusion

In this assignment, the team reported how we implemented machine learning on Microsoft Azure. We first introduced Microsoft Azure. Then using two different datasets and different learning algorithms for each dataset, we learned how to implement a machine learning algorithm for a dataset to predict the target feature. Meanwhile, we got familiar how we can use Microsoft Azure to get insight about the available dataset using statistics and charts. We are also capable of comparing different features with each other in this tool. We also understood how we should tune parameters to run the model such as splitting the dataset into training set and test set. Finally, we evaluated the model developed using the statistics provided after running the model. For instance, the accuracy of the results, number of True positives and false negatives are provided to decide about the model effectiveness and efficiency. We used two datasets and in each we discussed different aspects of using Microsoft Azure for machine learning purposes.

6.7 Individual Contributions

Arjun Aneja	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
William Clark	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
Sumati Kulkarni	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
Babak Maleki Shoja	Provided invaluable contributions to the completion of the tasks assigned to the group Implementation Drafted Report Reviewed Report
Venkatesh Reedy Pala	Provided invaluable contributions to the completion of the tasks assigned to the group Drafted Report Reviewed Report
Vishwa Patel	Provided invaluable contributions to the completion of the tasks assigned to the group for this project Implementation Drafted report Reviewed Report

6.8 Team Summary

This assignment provided an insight regarding a strong tool called Microsoft Azure for machine learning purposes. Using different datasets and learning algorithms, now we are familiar how to use the tool and we now understand capabilities of this tool. Also, it was a good experience in having a teamwork toward getting good results out of this assignment.

6.9 References

- Kelleher, John, Namee, Brian Mac, Arcy, Aoife D'. (2015). Fundamentals of Machine Learning for Predictive Data Analytics. The MIT Press Cambridge, Massachusetts London, England
- https://www.docs.microsoft.com

Naive Bayes Algorithm

7.1 Learning Goals

In this assignment, the team will develop a model for KKBOX Music Streaming Service Provider as our group project to illustrate how to create and use a naive Bayes model to predict churn of a subscribed user using descriptive features provided by KKBOX. To train a naive Bayes model using this data, we need to compute the prior probabilities of the target feature taking each level in its domain, and the conditional probability of each feature taking each level in its domain conditioned for each level that the target can take. The purpose of this assignment was to get familiar with applying a naive Bayes model to an existing problem, how we can evaluate the goodness of fit and to determine whether the user will churn or not.

7.1.1 Abstract

In this document, the model developed and used to predict churn of a subscriber based on descriptive features for KKBOX music streaming service provider is defined. First, we briefly review the business problem and descriptive and target features provided by the KKBOX for this problem. Then, we describe the model we used for predicting churn based on descriptive features. Next, the results of applying the naive Bayes model is provided. Finally, we draw conclusion regarding churn prediction problem.

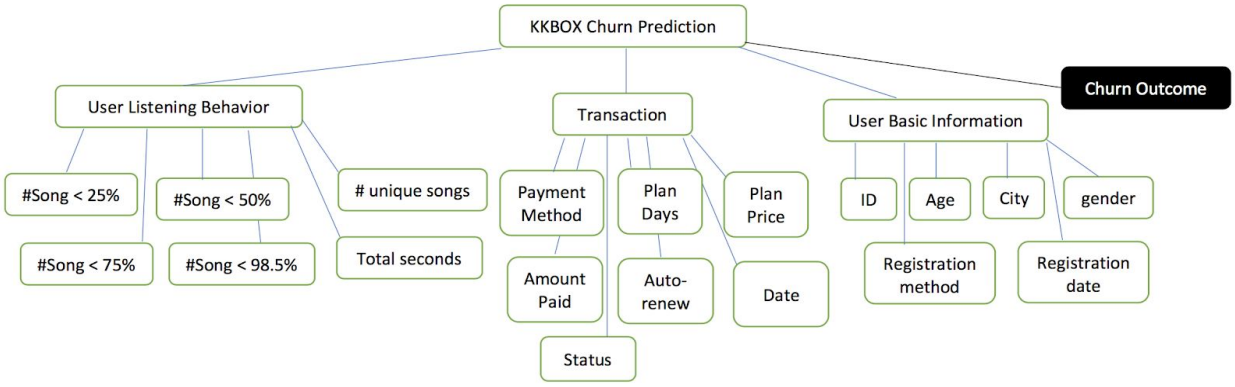
7.2 Business Problem

As provided in the project proposal, KKBOX, an Asian music streaming service provider, is facing the challenge of predicting whether a subscriber churn after his or her subscription expires or their decision to extend their subscription. This is a critical problem for such businesses and even a slight deviation from the predictions KKBOX has gathered a large amount of data from its users to resolve this problem.

The analytic solution to this problem defined as follows: We proposed to develop a model to predict the churn of a paid user after subscription expires. We take this into consideration to apply the prediction model and evaluate the results.

7.3 Descriptive and Target Features

Before describing the features, we should note that the prediction subject is defined as a paid subscriber. Here, we briefly review the features for the problem. The features of this problem and corresponding ABT developed as follows.



ABT for KKBOX Churn Prediction																			
User Basic Information						Transaction							User Listening Behavior						Target
ID	age	city	gender	Reg. Method	Reg. Date	Pay. Method	Plan days	Plan price	Amount paid	Auto-renew	date	status	# unq	# 25%	# 50%	# 75%	# 98.5%	Tot. Sec.	Churn

As explained in previous report, we need to predict whether a paid user churn when the subscription expires. We call this feature “Churn” which is going to be a

binary feature because churn will happen (Churn value equal to 1)" or the user renew his or her subscription (Churn value equal to 0).

Descriptive features are as follows.

- IDs: the ID of a user which is unique for each user;
- Age: the age of the user which is continuous;
- City: the location of the user which is a categorical feature;
- Gender: the gender of the user which is categorical with the cardinality of 2;
- Registration method: the method user utilized to register which is categorical;
- Registration date: the date of user registration which is considered as a continuous feature;
- Payment Method ID: the method user used to pay which is categorical;
- Payment plan days: the plan the user chose which is categorical with the cardinality of the available plans;
- Plan list price: the price of each plan which is categorical with the cardinality of the available plans;
- Actual amount paid: the amount the user actually paid which is a continuous feature;
- Auto-renew: the feature denotes that if a user has activated auto renew. It is categorical feature with cardinality of 2;
- Transaction date: similar to registration date, it is continuous and determines the date payment has paid;
- Status: denotes if a user is still active or canceled subscription which is categorical with the cardinality of 2;
- Number of songs played by user less than 25% or 50% or 75% or 98.5% which is continuous;
- Number of unique songs played by the user which is considered continuous;
- Total seconds of music played by the user which is considered continuous;

7.4 Naive Bayes Classifier Model

7.4.1 Introduction

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers. Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests.

An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.

7.4.2 Algorithm

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $x = (x_1, \ldots, x_n)$ representing some n features (independent variables), it assigns to this instance probabilities

$$p(C_k|x_1, \ldots, x\_n)$$

for each of K possible outcomes or classes C_k

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)}$$

In plain English, using Bayesian probability terminology, the above equation can be written as

$$posterior = \frac{prior \times likelihood}{evidence}$$

In practice, there is interest only in the numerator of that fraction because the denominator does not depend on C and the values of the features x_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model.

$$p(C_k, x_1, \ldots, x_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C_k, x_1, \ldots, x_n) &= p(x_1, \ldots, x_n, C_k) \\ &= p(x_1|x_2, \ldots, C_k)p(x_2, \ldots, x_n, C_k) \\ &= \ldots \\ &= p(x_1|x_2, \ldots, C_k)p(x_2, \ldots, x_n, C_k)\ldots p(x_{n-1}|x_n C_k)p(x_n|C_k)p(C_k) \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature x_i is conditionally independent of every other feature x_j for $j \neq i$ given the category C_k. This means that

$$p(x_i|x_{i+1}, \ldots, x_n, C_k) = p(x_i|C_k)$$

Thus, the joint model can be expressed as

$$\begin{aligned} p(C_k|x_1, \ldots, x_n) &\propto p(C_k, x_1, \ldots, x_n) \\ &\propto p(C_k)p(x_1|C_k)(x_2|C_k)(x_3|C_k)\ldots \\ &\propto p(C_k)\prod_{i=1}^{n} p(x_i|C_k) \end{aligned}$$

Where $\propto$ denotes proportionality.

This means that under the above independence assumptions, the conditional distribution over the class variable C is:

$$p(C_k|x_1, \ldots, x_n) = \tfrac{1}{Z}p(C_k)\prod_{i=1}^{n} p(x_i|C_k)$$

where the evidence $Z = p(x) = \sum_k p(C_k)p(x|C_k)$ is a scaling factor dependent only on $x_1, \ldots, x_n$ that is a constant if the values of the feature variables are known.

The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the maximum a posteriori or MAP decision rule. The corresponding classifier, a Bayes classifier, is the function that assigns a class label. $\hat{y} = C_k$ for some k as follows:

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{argmax}\, p(C_k)\prod_{i=1}^{n} p(x_i|C_k)$$

7.5 Implementation

We implemented above described algorithm on our dataset. We used Python and used following code to train and evaluate the model on our dataset.

```
# -*- coding: utf-8 -*-
"""
Created on Sat Feb 24 18:33:39 2018
@author: patel
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as pl
from sklearn import preprocessing,cross_validation,svm
from sklearn.cross_validation import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.externals.six import StringIO
from sklearn import tree
from matplotlib import style
from matplotlib.colors import ListedColormap
style.use("ggplot")

dataset = pd.read_csv('test.csv')
dataset = dataset.replace(['male','female'],
[0,1])

#print(dataset)

# Seperating value into two objects
#X = dataset.iloc[:, [2,3,4]].values
#A = dataset.iloc[:, 17].values
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 21].values
```

```
#dataset[X[:, 17]] = dataset[X[:, 17]].replace(0, dataset[X[:, 17]].mean())
#dataset[A] = dataset[A].replace(0, dataset[A].mean)

print("This is new age column replace with male and female into 0 and 1.")
#print(A)
print(X)
print(y)

print("This is the new value of gender column with missing values.")
print(X[:, 18])

# Replacing nan value with median in gender column which is 18th column

from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values='NaN', strategy = 'median', axis = 0)
#X[:, [18,19]] = X[:,[18,19]].reshape(1, -1)
imputer = imputer.fit(X[:, [18,19]])
X[:, [18,19]] = imputer.transform(X[:, [18,19]])
print("This is gender column after replacing missing values with median function")
print(X[:, 18])
print("\n")

# Split the data into training and testing part
from sklearn.cross_validation import train_test_split
# Giving 20% of total dataset in testing part and remaining in training part (80%)

x_train, x_test, y_train, y_test = train_test_split(X,y,test_size=0.20, random_state = 0)

# Implementing Naive-Bayes Classifier algorithm for our dataset
from sklearn.naive_bayes import GaussianNB

#Create a Gaussian Classifier
model = GaussianNB()
```

```
# Train the model using the training sets
model.fit(x_train,y_train)

#Predict Output
predicted= model.predict(x_test)
print("This is the predicted data for testing X which contains 2000 datasets")
print(predicted)
print("\n")

accuracy = model.score(x_test, y_test)
print("This is the accuracy of Naive Bayes Classifier algoritham")
print(accuracy)
print("\n")

# Calculating confusion matrix for machine learning prediction and
cm= confusion_matrix(y_test, predicted)
print("The confusion matrix is described below.")
print(cm)
```

We used total data to split into training and test. Some instances of training data are shown below:

date	num_25	num_50	num_75	num_985	num_100	num_unq	total_secs
20170331	8	4	0	1	21	18	6309.273
20170330	2	2	1	0	9	11	2390.699
20170331	52	3	5	3	84	110	23203.337
20170331	176	4	2	2	19	191	7100.454
20170331	2	1	0	1	112	93	28401.558
20170331	3	0	0	0	39	41	9786.842
20170330	9	1	0	0	18	26	4920.255
20170331	181	68	5	3	54	291	22433.105
20170331	3	0	1	1	181	150	46240.281
20170331	5	4	1	1	30	31	7881.618
20170331	7	1	2	7	69	74	18169.176
20170324	78	5	3	2	8	85	3490.02
20170325	7	1	0	0	45	21	8159.811
20170324	1	0	0	0	17	18	4012.814
20170325	5	3	2	1	34	43	9060.814
20170325	40	6	2	2	131	149	29400.889

payment_me	payment_pla	plan_list_pri	actual_amou	is_auto_rene	transaction_	membership
41	30	129	129	1	20150930	20151101
41	30	149	149	1	20150930	20151031
41	30	129	129	1	20150930	20160427
39	30	149	149	1	20150930	20151128
39	30	149	149	1	20150930	20151121
21	30	149	149	1	20150930	20151107
39	30	149	149	1	20150930	20151128
39	30	149	149	1	20150930	20151125
39	30	149	149	1	20150930	20151222
39	30	149	149	1	20150930	20151118
39	30	149	149	1	20150930	20151121
39	30	149	149	1	20150930	20151124
39	30	149	149	1	20150930	20151117
39	30	149	149	1	20150930	20151129
39	30	149	149	1	20150930	20151111
39	30	149	149	1	20150930	20151122

is_cancel	city	age	gender	registered_v	registration_	is_churn
0	1	0		11	20110911	1
0	1	0		7	20110914	1
0	1	0		11	20110915	1
0	1	0		11	20110915	1
0	6	32	female	9	20110915	1
0	4	30	male	9	20110916	1
0	1	0		7	20110916	1
0	5	34	male	9	20110916	1
0	5	19	male	9	20110917	1
0	13	63	male	9	20110918	1
0	1	0		7	20110918	1
0	22	18	male	9	20110919	1
0	4	34	female	9	20110919	1
0	4	28	female	9	20110920	1
0	12	29	female	9	20110922	1
0	1	0		9	20110922	1

We used our data set and the model is evaluated by a test dataset with 2000 instances. The results statistics are as follows.

Using this data and using python's inbuilt modeling tool we can build prediction model that would assign a conditional probability to each level of every feature and final prediction is based on the formula shown below:

$$M(q) = \underset{l \epsilon levels(t)}{argmax}((\prod_{i=1}^{m} P(q[i]t = l)) \times P(t = l))$$

7.6 Results

Table 1 Results of Bayes Classifier Algorithm

Results Statistics	
Number of instances	2000
Number of true predictions	1745
Number of false predictions	254
False prediction percentage	12.7

Model accuracy (%)	**87.3**

As demonstrated in the results and shown in Table 1, the model shows a good performance in predicting the Churn using available data. However, comparing to K-Neighbors Classifier we used before, K-Neighbors Classifier outperforms Bayes Classifier Algorithm. Consequently, the K-Neighbor Algorithm Classifier is a more suitable algorithm for our problem and available dataset.

7.7 Conclusion

The Naïve Bayes Model shows a good performance in predicting when the user will Churn. As it is not the best when compared to K-Neighbors Classifier, and also as K-Neighbors is more suitable algorithm for our type of dataset, we proceed with using K-Neighbors algorithm.

7.8 Individual Contributions

Arjun Aneja	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
William Clark	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report

Sumati Kulkarni	Provided invaluable contributions to the completion of the tasks assigned to the group Drafted Report Reviewed Report
Babak Maleki Shoja	Provided invaluable contributions to the completion of the tasks assigned to the group Drafted Report Reviewed Report
Venkatesh Reedy Pala	Provided invaluable contributions to the completion of the tasks assigned to the group Drafted Report Reviewed Report
Vishwa Patel	Provided invaluable contributions to the completion of the tasks assigned to the group for this project Drafted report Reviewed Report

7.9 Team Summary

This phase of the project gave us the insight in implementing predictive models and evaluate them. Moreover, we understood we need to exclude some of the features to get the results. The importance of the evaluation for the model was investigated and it was a great experience to see how machine learning can solve real-world problems and challenges.

7.10 References

- Kelleher, John, Namee, Brian Mac, Arcy, Aoife D'. (2015). Fundamentals of Machine Learning for Predictive Data Analytics. The MIT Press Cambridge, Massachusetts London, England

- Naive Bayes classifier. Wikipedia, the free encyclopedia https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Multiple Linear Regression

8.1 Learning Goals

In this assignment, the team developed a model for dataset which contains data of 50 companies in3 states. The descriptive features contains 1) R&D Spend by Company 2) Administration and 3) Marketing. Based on all of these descriptive features we then decide how much profit that company made. So profit is the Target Feature, The purpose of this assignment was to get familiar with applying a prediction model to an existing problem, how we can evaluate the goodness of fit and to determine features that can be excluded from the data.

8.2 Multiple Linear Regression

8.2.1 Introduction

Multiple linear regression (MLR) is an extension of simple linear regression. In the previous chapter we considered a single dependent variable, *y*, and a single independent variable *x*. MLR is used when there are two or more independent variables where the model using population information is

$$y_t = \beta_0 + \beta_1 x_{1t} + \beta_2 x_{2t} + \beta_3 x_{3t} + \ldots\ldots + \beta_k x_{kt}$$

It should not be surprising that most interesting phenomena are too complex to be modeled using just a single independent variable. MLR allows a much more comprehensive model than does simple LR and should provide superior predictions.

8.2.2 The assumptions of MLR

MLR uses some assumptions. These are necessary for the mathematics to work out properly. The assumptions used in MLR:

1. The error terms ε_i are normally distributed
2. The error terms are independent of past error terms, that is $E(\varepsilon_i | \varepsilon_{i-1}, \varepsilon_{i-2}..) = E(\varepsilon_i)$
3. The populations all have equal variances, $\sigma_1 = \sigma_2 = \sigma_3 = \ldots$
4. The independent variables are not correlated with each other.

In the least-squares model, the best-fitting line for the observed data is calculated by minimizing the sum of the squares of the vertical deviations from each data point to the line (if a point lies on the fitted line exactly, then its vertical deviation is 0). Because the deviations are first squared, then summed, there are no cancellations between positive and negative values. The least-squares estimates $b_o,\ldots,b_t$ are usually computed by statistical software.

8.2.3 Data Understanding

Dataset which is used in this report is "**50 Startup Companies Profit**". The dataset we have used for this project consists of 50 different companies located in 3 different states 1) California 2) New York 3) Florida. The other descriptive features are "R&D budget", "Administration" and "Marketing" which consists of amount of money which 50 companies spend on these different

fields. The target feature is to calculate the amount of Profit these companies made based on all of these descriptive features. Here is how the dataset looks like:

A	B	C	D	E
R&D Spend	Administration	Marketing Spend	State	Profit
165349.2	136897.8	471784.1	New York	192261.83
162597.7	151377.59	443898.53	California	191792.06
153441.51	101145.55	407934.54	Florida	191050.39
144372.41	118671.85	383199.62	New York	182901.99
142107.34	91391.77	366168.42	Florida	166187.94
131876.9	99814.71	362861.36	New York	156991.12
134615.46	147198.87	127716.82	California	156122.51
130298.13	145530.06	323876.68	Florida	155752.6
120542.52	148718.95	311613.29	New York	152211.77
123334.88	108679.17	304981.62	California	149759.96
101913.08	110594.11	229160.95	Florida	146121.95
100671.96	91790.61	249744.55	California	144259.4
93863.75	127320.38	249839.44	Florida	141585.52
91992.39	135495.07	252664.93	California	134307.35
119943.24	156547.42	256512.92	Florida	132602.65
114523.61	122616.84	261776.23	New York	129917.04
78013.11	121597.55	264346.06	California	126992.93
94657.16	145077.58	282574.31	New York	125370.37
91749.16	114175.79	294919.57	Florida	124266.9

50_Startups

8.2.4 What is Backward Elimination?

- The idea of Backward Elimination is to remove independent variables that are not statistically significant.
- If your dataset is huge, this could make a great difference, because your model can run with less data.
- Our goal here is to find a group of independent variables that all big impact to the dependent variable.

8.2.5 Mechanism of Backward Elimination

1. Select a significant level (i.e.: Significant Model = 0.05 ; If the P value is greater than this significant level, then we will remove it)
2. First fit ALL variables to the model.
3. Find the P values for ALL variables.
4. Remove the variable with the largest P value.
5. Fit the model with a variable removed from Step 4.
6. Repeat Step 4 & 5, until all P values are smaller than the significant level defined in Step 1.
7. Model is ready.

8.2.6 Steps of Backward Elimination

Step-1: Select a significant level to stay in the model (e.g. Significance Level = 0.05).

Step-2: Fit the full model with all possible predictors.

Step-3: Consider the predictor with the highest P-value. If P > SL, go to STEP-4 otherwise go to **FINISH.**

Step-4: Remove the predictor.

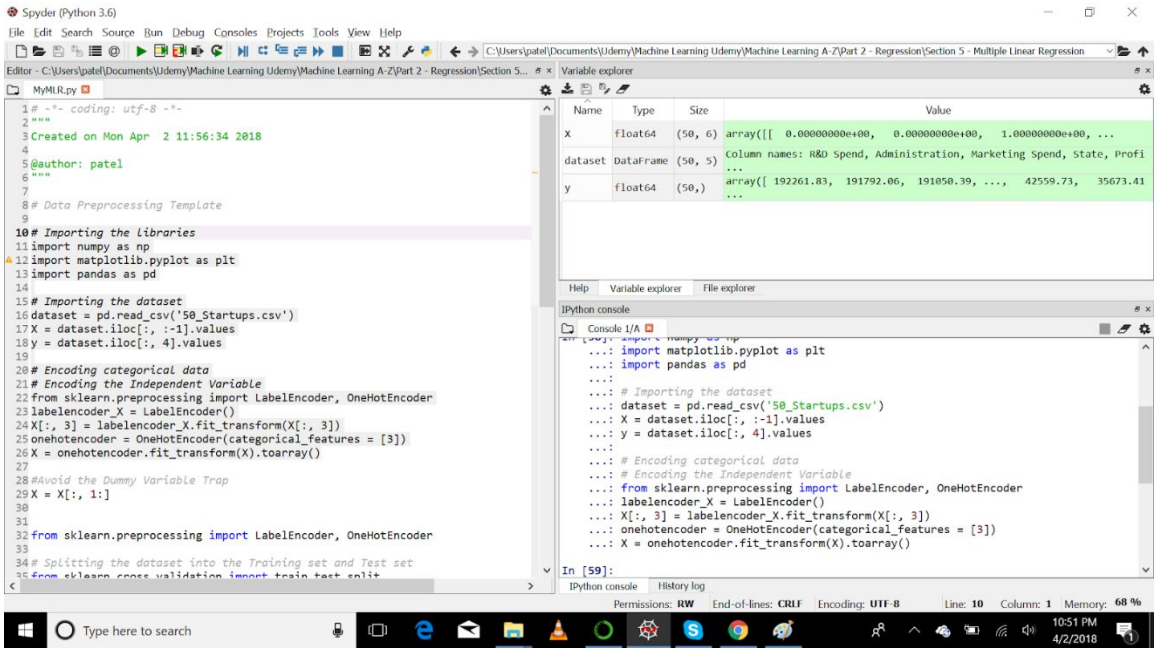
Step-5: Fit model without this variable and go back to **Step-3.**

FINISH: Your model is ready.

We can follow these steps to perform Backward Elimination for Multiple Linear Regression.

8.3 Implementation

Below mentioned are some screenshots of the code and the results we got while performing the Multiple Linear Regression with Backward Elimination.



In above screenshot, we have imported dataset from CSV file and stored into two different Matrices which are X(Descriptive Features) and y(Target Features). The variable values can be seen on right side of the photo.

X - NumPy array

	0	1	2	3	4	5
0	0	0	1	165349	136898	471784
1	1	0	0	162598	151378	443899
2	0	1	0	153442	101146	407935
3	0	0	1	144372	118672	383200
4	0	1	0	142107	91391.8	366168
5	0	0	1	131877	99814.7	362861
6	1	0	0	134615	147199	127717
7	0	1	0	130298	145530	323877
8	0	0	1	120543	148719	311613
9	1	0	0	123335	108679	304982
10	0	1	0	101913	110594	229161

X - NumPy array

	0	1	2	3	4
0	0	1	165349	136898	471784
1	0	0	162598	151378	443899
2	1	0	153442	101146	407935
3	0	1	144372	118672	383200
4	1	0	142107	91391.8	366168
5	0	1	131877	99814.7	362861
6	0	0	134615	147199	127717
7	1	0	130298	145530	323877
8	0	1	120543	148719	311613
9	0	0	123335	108679	304982
10	1	0	101913	110594	229161

Above mentioned photos, one contains 5 columns and other contains 4 columns. The reason is that in first picture we converted State column into numeric column and then created dummy variable for all 3 states 1) California 2) Florida 3) New York. But, in the second picture, we can see that after creating dummy variable, we have dropped one dummy column because it causes redundancy problem as for example if we have two dummy variables then if you take any one then its meaning will be same as other column for example, if one column has data like {0,1,0,1} and {1,0,1,0} for California state and Florida State respectively. Then when we look at first column {0,1,0,1} we can say that, if there is 0 then it is Florida State and 1 California State. Hence it will be lead to error in calculation not in syntax error. So it is wise to drop one dummy variable column. That is what we have done in our implementation.

X_train - NumPy array

	0	1	2	3	4
0	1	0	55493.9	103057	214635
1	0	1	46014	85047.4	205518
2	1	0	75328.9	144136	134050
3	0	0	46426.1	157694	210798
4	1	0	91749.2	114176	294920
5	1	0	130298	145530	323877
6	1	0	119943	156547	256513
7	0	1	1000.23	124153	1903.93
8	0	1	542.05	51743.2	0
9	0	1	65605.5	153032	107138
10	0	1	114524	122617	261776

y_train - NumPy array

	0
0	96778.9
1	96479.5
2	105734
3	96712.8
4	124267
5	155753
6	132603
7	64926.1
8	35673.4
9	101005
10	129917

y_test - NumPy array

	0
0	103282
1	144259
2	146122
3	77798.8
4	191050
5	105008
6	81229.1
7	97483.6
8	110352
9	166188

X_test - NumPy array

	0	1	2	3	4
0	1	0	66051.5	182646	118148
1	0	0	100672	91790.6	249745
2	1	0	101913	110594	229161
3	1	0	27892.9	84710.8	164471
4	1	0	153442	101146	407935
5	0	1	72107.6	127865	353184
6	0	1	20229.6	65947.9	185265
7	0	1	61136.4	152702	88218.2
8	1	0	73994.6	122783	303319
9	1	0	142107	91391.8	366168

We can see the train and test dataset for our project. We have divided the dataset, giving 20% of data in testing and remaining 80% of data in training the model.

```
In [7]: from sklearn.linear_model import LinearRegression
   ...: regressor = LinearRegression()
   ...: regressor.fit(X_train, y_train)
Out[7]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1,
normalize=False)
```

After splitting dataset, we can now train our model using X_train and y_train (descriptive features and target features) respectively. We are using here Linear Regression class which can also be used for Linear Regression model but in order to perform Linear Regression we just have to stop at next point where as for Multiple Linear Regression, we have to continue to work on our dataset.

y_pred - NumPy arra

	0
0	103015
1	132582
2	132448
3	71976.1
4	178537
5	116161
6	67851.7
7	98791.7
8	113969
9	167921

Above mentioned are predicted values for X_test matrix. This shows the values which are predicted by our model which was trained before by X_train data. The predicted values are stored in a matrix called “y_pred”.

X - NumPy array

	0	1	2	3	4	5
0	1	0	1	165349	136898	471784
1	1	0	0	162598	151378	443899
2	1	1	0	153442	101146	407935
3	1	0	1	144372	118672	383200
4	1	1	0	142107	91391.8	366168
5	1	0	1	131877	99814.7	362861
6	1	0	0	134615	147199	127717
7	1	1	0	130298	145530	323877
8	1	0	1	120543	148719	311613
9	1	0	0	123335	108679	304982
10	1	1	0	101913	110594	229161

OLS Regression Results

Dep. Variable:	y	R-squared:	0.951
Model:	OLS	Adj. R-squared:	0.945
Method:	Least Squares	F-statistic:	169.9
Date:	Mon, 23 Apr 2018	Prob (F-statistic):	1.34e-27
Time:	02:33:35	Log-Likelihood:	-525.38
No. Observations:	50	AIC:	1063.
Df Residuals:	44	BIC:	1074.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.013e+04	6884.820	7.281	0.000	3.62e+04	6.4e+04
x1	198.7888	3371.007	0.059	0.953	-6595.030	6992.607
x2	-41.8870	3256.039	-0.013	0.990	-6604.003	6520.229
x3	0.8060	0.046	17.369	0.000	0.712	0.900
x4	-0.0270	0.052	-0.517	0.608	-0.132	0.078
x5	0.0270	0.017	1.574	0.123	-0.008	0.062

Omnibus:	14.782	Durbin-Watson:	1.283
Prob(Omnibus):	0.001	Jarque-Bera (JB):	21.266
Skew:	-0.948	Prob(JB):	2.41e-05
Kurtosis:	5.572	Cond. No.	1.45e+06

Here, we have used “statsmodels” api in order to first add the whole column of “1”. This is necessary as in the equation of Multiple Linear Regression

$$y_t = \beta_0 + \beta_1 x_{1t} + \beta_2 x_{2t} + \cdots + \beta_k x_{kt}$$

B_0 is the constant. Hence if a feature is selected whose value is 0 then whole equation becomes 0 by default. Hence, in order to prevent that situation, we have to add a whole column of “1” which will be considered as constant value if the rest of the part of the equation becomes “0” then we will still have the default value.

Then we are getting a new array “X_opt” which consists of all columns existing in X matrix. Then we take regressor object and call method OLS and give “y” as endog which is the target feature and “X_opt” as the descriptive feature. Once this is done then call “summary()” method, so that we can know the p-value, t-value squared-R etc. All of these calculations are mentioned on right side of screenshot. The idea here is to calculate which column’s p-value is greater than Significant Value = 0.05 (for our project). If p-value is greater than 0.05 then in next step we have to remove that column from the “X_opt” and again give that array as input in next step. This way we are removing the columns which have really less impact on predicting target value. Hence, the name “Backward Elimination”.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.951
Model:	OLS	Adj. R-squared:	0.946
Method:	Least Squares	F-statistic:	217.2
Date:	Mon, 23 Apr 2018	Prob (F-statistic):	8.49e-29
Time:	02:37:46	Log-Likelihood:	-525.38
No. Observations:	50	AIC:	1061.
Df Residuals:	45	BIC:	1070.
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.011e+04	6647.870	7.537	0.000	3.67e+04	6.35e+04
x1	220.1585	2900.536	0.076	0.940	-5621.821	6062.138
x2	0.8060	0.046	17.606	0.000	0.714	0.898
x3	-0.0270	0.052	-0.523	0.604	-0.131	0.077
x4	0.0270	0.017	1.592	0.118	-0.007	0.061

Omnibus:	14.758	Durbin-Watson:	1.282
Prob(Omnibus):	0.001	Jarque-Bera (JB):	21.172
Skew:	-0.948	Prob(JB):	2.53e-05
Kurtosis:	5.563	Cond. No.	1.40e+06

Backward Elimination step is performed here by removing "2nd column". So we notice difference in above 2 photos, that column having 0.990 as its P value is not present in above mentioned set of results.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.951
Model:	OLS	Adj. R-squared:	0.948
Method:	Least Squares	F-statistic:	296.0
Date:	Mon, 23 Apr 2018	Prob (F-statistic):	4.53e-30
Time:	02:39:54	Log-Likelihood:	-525.39
No. Observations:	50	AIC:	1059.
Df Residuals:	46	BIC:	1066.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.012e+04	6572.353	7.626	0.000	3.69e+04	6.34e+04
x1	0.8057	0.045	17.846	0.000	0.715	0.897
x2	-0.0268	0.051	-0.526	0.602	-0.130	0.076
x3	0.0272	0.016	1.655	0.105	-0.006	0.060

Omnibus:	14.838	Durbin-Watson:	1.282
Prob(Omnibus):	0.001	Jarque-Bera (JB):	21.442
Skew:	-0.949	Prob(JB):	2.21e-05
Kurtosis:	5.586	Cond. No.	1.40e+06

Backward Elimination step is performed here by removing "1st column".

OLS Regression Results

Dep. Variable:	y	R-squared:	0.950
Model:	OLS	Adj. R-squared:	0.948
Method:	Least Squares	F-statistic:	450.8
Date:	Mon, 23 Apr 2018	Prob (F-statistic):	2.16e-31
Time:	02:47:33	Log-Likelihood:	-525.54
No. Observations:	50	AIC:	1057.
Df Residuals:	47	BIC:	1063.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.698e+04	2689.933	17.464	0.000	4.16e+04	5.24e+04
x1	0.7966	0.041	19.266	0.000	0.713	0.880
x2	0.0299	0.016	1.927	0.060	-0.001	0.061

Omnibus:	14.677	Durbin-Watson:	1.257
Prob(Omnibus):	0.001	Jarque-Bera (JB):	21.161
Skew:	-0.939	Prob(JB):	2.54e-05
Kurtosis:	5.575	Cond. No.	5.32e+05

Backward Elimination step is performed here by removing "4th column".

OLS Regression Results

Dep. Variable:	y	R-squared:	0.947
Model:	OLS	Adj. R-squared:	0.945
Method:	Least Squares	F-statistic:	849.8
Date:	Mon, 23 Apr 2018	Prob (F-statistic):	3.50e-32
Time:	02:48:43	Log-Likelihood:	-527.44
No. Observations:	50	AIC:	1059.
Df Residuals:	48	BIC:	1063.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.903e+04	2537.897	19.320	0.000	4.39e+04	5.41e+04
x1	0.8543	0.029	29.151	0.000	0.795	0.913

Omnibus:	13.727	Durbin-Watson:	1.116
Prob(Omnibus):	0.001	Jarque-Bera (JB):	18.536
Skew:	-0.911	Prob(JB):	9.44e-05
Kurtosis:	5.361	Cond. No.	1.65e+05

Backward Elimination step is performed here by removing "5th column".

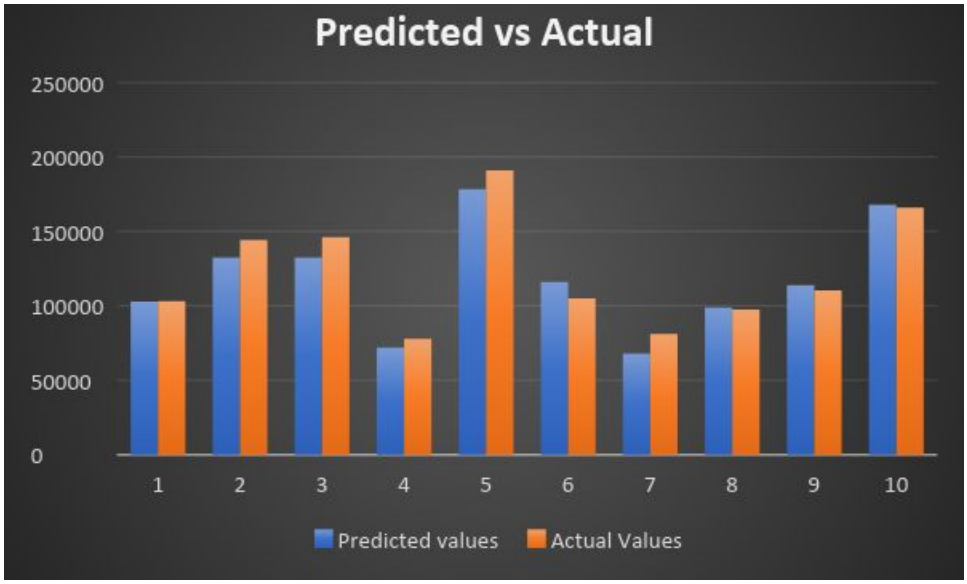
8.4 Results:

Hence, at the end of this step, we can see the p-value of 3rd column is less than significant value (note that the p-value is 0.000 that doesn't mean the value is 0 it means that the value is far less than 0 which cannot be described in 3 decimal points after 0). Hence, we can say that 3rd column has comparatively more impact on target feature (statistically) than the rest of the columns (descriptive features). As Multiple Linear Regression is the Error base learning, we can also see the error calculated for each set of columns in the above screenshots on right side.

Final step, for calculating the accuracy of the model, we have used R^2 technique. After applying, we got R^2 = 0.934706847328. We calculate that between two variables, y_test (which is the actual values for X_test matrix holding descriptive features) and y_pred (which holds the value predicted by model for X_test). From the R^2 we can say that our model is predicting values closest to the actual values. Usually, the value of R must be between 0-1.

8.5 Conclusion:

Here both Actual values and Predicted values are compared in chart. From the clustered chart we can see that the values are very close. So, model predicted values for test cases nearly as accurate as actual values. In below graph we can see that blue line is predicted values while orange is Actual values.



8.6 Individual Contributions

William Clark	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
Sumati Kulkarni	Provided invaluable contributions to the completion of the tasks assigned to the group Drafted report Reviewed Report
Babak Maleki Shoja	Provided invaluable contributions to the completion of the tasks assigned to the group Implementation Drafted Report Reviewed Report
Venkatesh Reedy Pala	Provided invaluable contributions to the completion of the tasks assigned to the group Drafted Report Reviewed Report
Vishwa Patel	Provided invaluable contributions to the completion of the tasks assigned to the group for this project Implementation Drafted report Reviewed Report

8.7 Team Summary

This assignment helped us get a better understanding of using Python to implement Multiple Linear Regression algorithm with Backward Elimination. As shown in the implementation section we were able to build Multiple Linear Regression algorithm with Backward Elimination in Python. Our team did a great job working and collaborating to successfully complete this project.

Support Vector Machine for Regression

9.1 Learning Goals

In this assignment, the team developed a model for dataset which contains data of a restaurants which are located in different Cities. Based on all of these descriptive features we then decide how much revenue that restaurant made. So revenue is the Target Feature, The purpose of this assignment was to get familiar with applying a prediction model to an existing problem, how we can evaluate the goodness of fit and to determine features that can be excluded from the data.

9.2 SVM for Regression

Support Vector Machine can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration. However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated.

The kernel we have selected is Radial Basis Function and Polynomial function.

The formula for Radial Basis Function is mentioned below:

Gaussian Radial Basis function

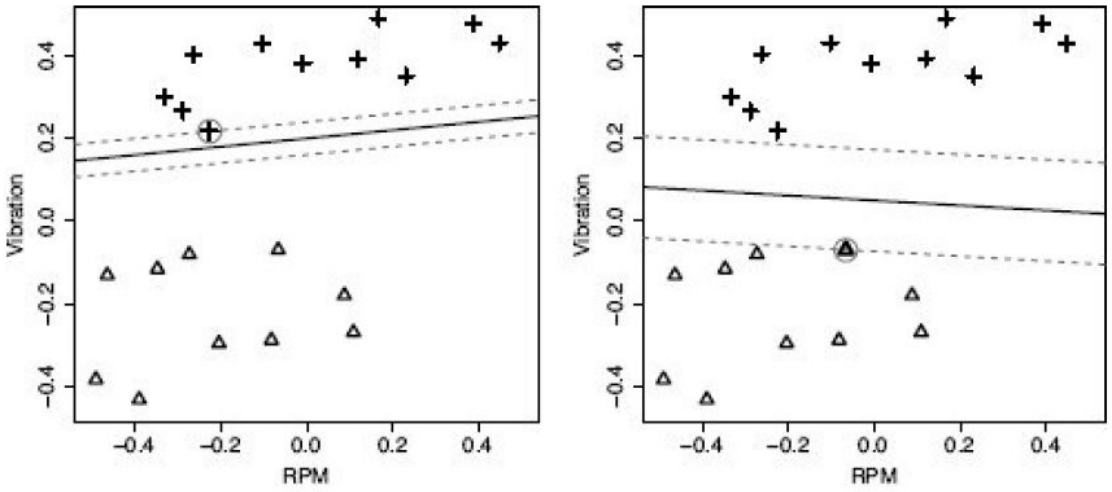
$$k(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\frac{\left\|\mathrm{x}_i - \mathrm{x}_j\right\|^2}{2\sigma^2}\right)$$

NOTE: Here we have used cropped image of formula of Gaussian Radial Basis Function because the formula mentioned above is difficult to write because of the mode between the both variables on right side.

The formula for polynomial function is:

$$k(xi, xj) = (xi.xj)d$$

For example, here an example from book is mentioned below. This diagram shows the clear boundary for 2 variables good and bad.



The above-mentioned figure shows a diagram with different decision boundary, which has a much larger margin. The intuition behind support vector machines is that this second decision boundary should distinguish between the two target levels much more reliably than the first. Training a support vector machine involves searching for the decision boundary, or **separating hyperplane**,20 that leads to the maximum margin as this will best separate the levels of the target feature. Although the goal of finding the best decision boundary is the same for algorithms that build support vector machines as it is for logistic regression models, the inductive bias encoded in the algorithms to select this boundary is different, which leads to different decision boundaries being found.
The instances in a training dataset that fall along the margin extents, and so define the margins, are known as the **support vectors**. These are the most important instances in the dataset because they define the decision boundary. There will always be at least one support vector for each level of the target feature, but there is no limit to how many

support vectors there can be in total.

9.3 Implementation with Explanation:

Step-1

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ
d	Open Date	City	City Group	Type	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25	P26	P27	P28	P29	P30	P31	P32	P33	P34	P35	P36	P37	revenue
0	7/17/1999	Ã°stanbul	Big Cities	IL	4	5	4	4	2	2	5	4	5	5	3	5	5	1	2	2	2	4	5	4	1	3	3	1	1	1	4	2	3	5	3	4	5	5	4	3	4	5653753
1	2/14/2008	Ankara	Big Cities	FC	4	5	4	4	1	2	5	5	5	5	1	5	5	0	0	0	0	0	3	2	1	3	2	0	0	0	0	3	3	0	0	0	0	0	0	0	0	6923131
2	3/9/2013	DiyarbakÃ	Other	IL	2	4	2	5	2	3	5	5	5	5	2	5	5	0	0	0	0	0	1	1	1	1	1	0	0	0	0	1	3	0	0	0	0	0	0	0	0	2055379
3	2/2/2012	Tokat	Other	IL	6	4.5	6	6	4	4	10	8	10	10	8	10	7.5	6	4	9	3	12	20	12	6	1	10	2	2	2.5	2.5	2.5	7.5	25	12	10	6	18	12	12	6	2675511
4	5/9/2009	Gaziantep	Other	IL	3	4	3	4	2	2	5	5	5	5	2	5	5	2	1	2	1	4	2	2	1	2	1	2	3	3	5	1	3	5	1	3	2	3	4	3	3	4316715
5	2/12/2010	Ankara	Big Cities	FC	6	6	4.5	7.5	8	10	10	8	8	8	10	8	6	0	0	0	0	0	5	6	3	1	5	0	0	0	0	7.5	5	0	0	0	0	0	0	0	0	5017319
6	10/11/2010	Ã°stanbul	Big Cities	IL	2	3	4	4	1	5	5	5	5	5	2	5	5	3	4	4	3	4	2	4	1	2	1	5	4	4	5	1	3	4	5	2	2	3	5	4	4	5166635
7	6/21/2011	Ã°stanbul	Big Cities	IL	4	5	4	5	2	3	5	4	4	4	4	3	4	0	0	0	0	0	3	5	2	4	2	0	0	0	0	3	2	0	0	0	0	0	0	0	0	4491607
8	8/28/2010	Afyonkara	Other	IL	1	1	4	4	1	2	1	5	5	5	1	5	5	1	1	2	1	4	1	1	1	1	1	4	4	4	2	2	3	4	5	5	3	4	5	4	5	4952497
9	11/16/2011	Edirne	Other	IL	6	4.5	6	7.5	6	4	10	10	10	10	2	10	7.5	0	0	0	0	0	25	3	3	1	10	0	0	0	0	5	2.5	0	0	0	0	0	0	0	0	5444227
10	8/9/2013	Kocaeli	Other	FC	9	6	6	6	4	4	10	8	10	10	8	10	7.5	0	0	0	0	0	25	15	15	3	20	0	0	0	0	10	2.5	0	0	0	0	0	0	0	0	3745135
11	5/22/2012	Ã°stanbul	Big Cities	IL	2	4	4	4	2	5	5	5	5	5	2	5	5	2	2	5	2	4	2	5	1	1	3	5	2	3	5	3	3	5	5	4	2	3	4	4	2	5161370
12	2/28/2013	Ankara	Big Cities	IL	2	2	4	4	2	1	5	4	5	5	3	5	5	0	0	0	0	0	3	4	1	2	2	0	0	0	0	1	3	0	0	0	0	0	0	0	0	1734634
13	10/16/2010	Ã°stanbul	Big Cities	FC	4	5	4	4	1	3	5	5	5	5	2	5	5	0	0	0	0	0	3	4	1	2	1	0	0	0	0	3	3	0	0	0	0	0	0	0	0	4807746
14	12/29/2011	Bursa	Other	FC	2	2	4	4	1	2	5	5	5	5	2	5	5	0	0	0	0	0	3	3	1	3	2	0	0	0	0	2	3	0	0	0	0	0	0	0	0	1999097
15	2/7/2012	Ã°stanbul	Big Cities	IL	12	7.5	6	6	2	10	10	10	10	10	4	10	7.5	3	10	15	3	12	10	9	3	2	5	8	8	10	2.5	7.5	7.5	5	15	20	2	12	3	16	4	3218918
16	1/7/2000	Ã°stanbul	Big Cities	FC	3	5	4	4	2	5	5	4	5	4	3	5	4	0	0	0	0	0	1	4	1	1	1	0	0	0	0	2	2	0	0	0	0	0	0	0	0	19696939
17	11/8/2009	Ã°stanbul	Big Cities	FC	2	4	4	5	1	3	5	4	5	5	3	5	5	0	0	0	0	0	1	1	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	8213524
18	4/21/2012	Ã°zmir	Big Cities	IL	4	5	4	3	1	2	5	5	5	5	2	5	5	3	2	3	2	3	4	5	1	4	2	5	3	3	3	2	3	3	5	5	4	4	4	3	2	5337526
19	8/16/2011	Sakarya	Other	IL	2	4	4	4	2	2	5	5	4	4	1	3	4	0	0	0	0	0	2	1	1	1	1	0	0	0	0	2	3	0	0	0	0	0	0	0	0	2021934
20	8/25/2010	ElazÃ±ÃŸ	Other	IL	3	4	4	4	2	2	5	4	5	5	3	5	5	3	5	5	4	4	5	5	1	1	5	2	4	1	5	1	3	5	1	2	2	4	5	5	4	5525735
21	1/25/2014	Ã°stanbul	Big Cities	FC	5	5	4	4	2	2	5	4	4	4	4	3	4	0	0	0	0	0	4	5	1	3	1	0	0	0	0	3	1	0	0	0	0	0	0	0	0	1149870

Our dataset is contained originally in single file, which is 'train.csv'. Train file is used to provide training and testing dataset to our SVR model.

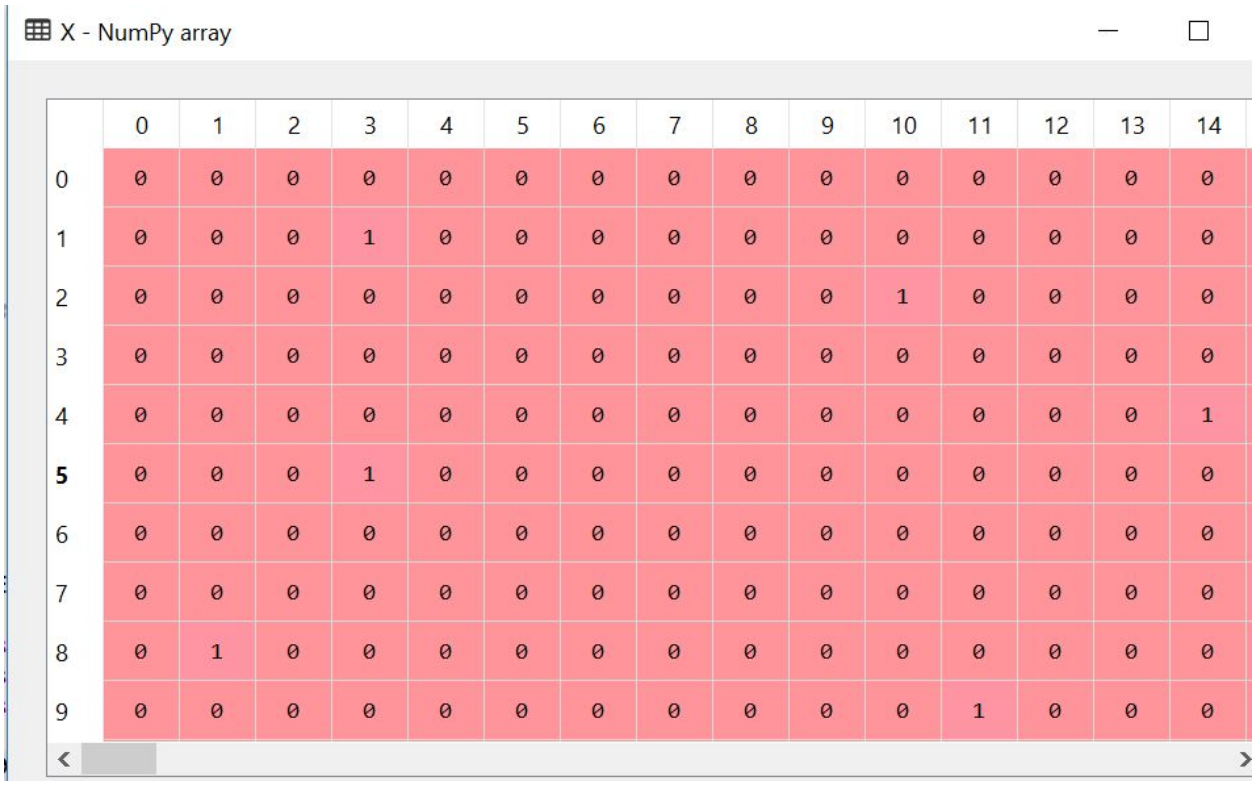
Our dataset is basically "Restaurant Revenue", it has several field combinations of both categorical and continuous features, but the target features is continuous. We have downloaded this dataset from 'Kaggle' website, which is free for people who want to learn Machine Learning. Target features is how much revenue that Restaurant in that city collected in a certain amount of time.

Step-2

Name	Type	Size	Value
x	object	(136, 39)	ndarray object of numpy module
dataset	DataFrame	(138, 43)	Column names: Id, Open Date, City, City Group, Type, P1, P2, P3, P4, P ...
y	float64	(136, 1)	array([[5653753.], [6923131.],

Now, we have divided the dataset into 2 matrices, one contains descriptive features(X) and other contain target feature(y). These both are selected from the ‘train.csv’ file.

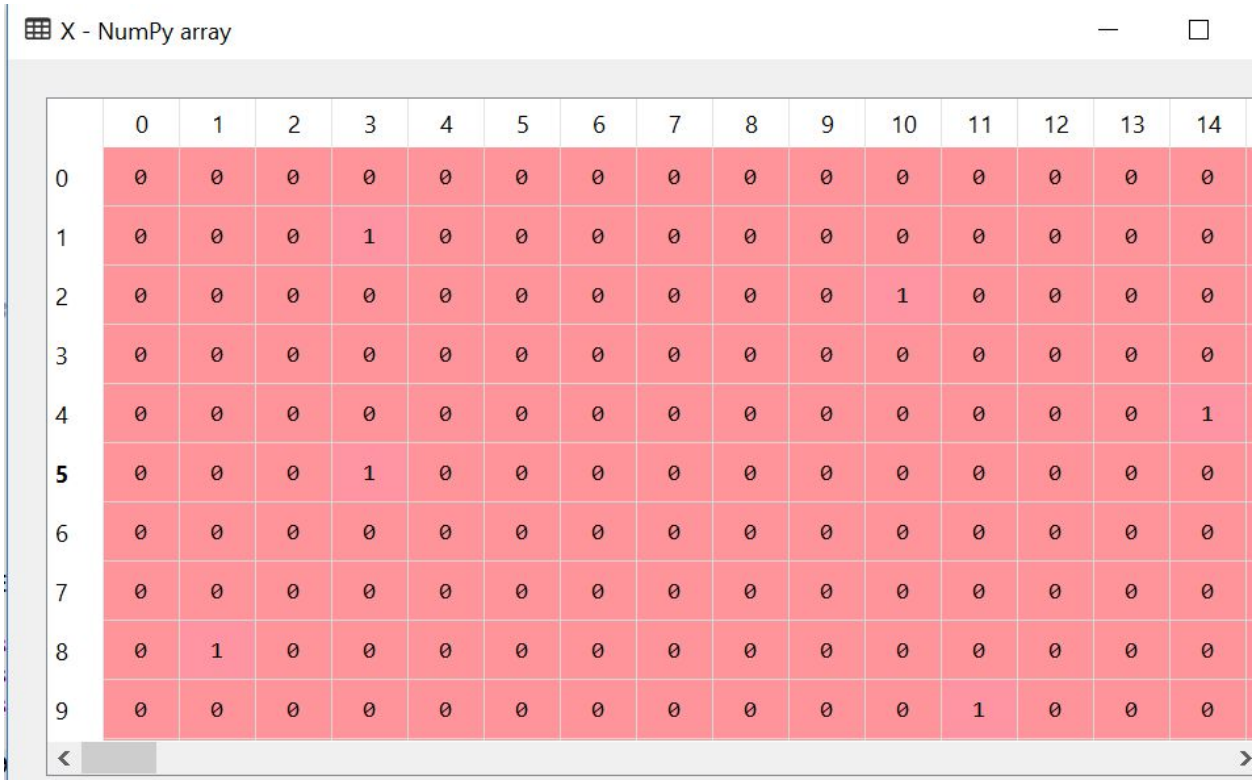
Step-3



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
5	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Now, the next step is to transform categorical values into continuous values by using LabelEncoder and OneHotEncoder these both libraries are used to transform the categorical data into continuous data, once the data is converted now with the help of OneHotEncoder we can create dummy variables for those categorical features. I have applied to first 3 columns for X matrix, since these 3 columns contains categorical values which are City, CityGroup and Type. After creating dummy variables, the number of columns have increased from 39 to 75, because there are more categorical variables.

Step-4



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
5	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Here, in the next step we need to remove one dummy variable because if we don't do that then it will be redundant for our model and to avoid that we remove one dummy variable, from X.

Step-5

X - NumPy array

	38	39	40	41	42	43
0	-0.00505289	0.393375	-0.318886	-0.362205	-0.00607872	-0.641776
1	-0.00505289	0.393375	-0.318886	-0.362205	-0.832784	-0.641776
2	-0.692246	-0.267107	-2.26803	0.622992	-0.00607872	-0.17252
3	0.68214	0.0631343	1.63026	1.60819	1.64733	0.296735
4	-0.348649	-0.267107	-1.29346	-0.362205	-0.00607872	-0.641776
5	0.68214	1.05386	0.1684	3.08598	4.95416	3.11227
6	-0.692246	-0.927589	-0.318886	-0.362205	-0.832784	0.765991
7	-0.00505289	0.393375	-0.318886	0.622992	-0.00607872	-0.17252
8	-1.03584	-2.24855	-0.318886	-0.362205	-0.832784	-0.641776

Now, the next step is to feature scale the dataset. We have scaled our dataset by using "Standardization Feature Scaling" which will transform the matrices X and y. But the trick here is that we don't have to Standardized the dummy variables, just the rest of descriptive features. For each matrix we need to create independent objects which will be used for that matrix respectively.

Step-6

Name	Type	Size	Value
X	float64	(136, 75)	array([[0. , 0. , 0. , ..., 0.56992382, ...
X_test	float64	(28, 75)	array([[0. , 0. , 1. , ..., 0.56992382, ...
X_train	float64	(108, 75)	array([[0. , 0. , 0. , ..., -0.59563467, ...
dataset	DataFrame	(138, 43)	Column names: Id, Open Date, City, City Group, Type, P1, P2, P3, P4, P ...
y	float64	(136, 1)	array([[5653753.], [6923131.],
y_test	float64	(28, 1)	array([[2267425.], [4952497.],
y_train	float64	(108, 1)	array([[3570392.], [3745135.],

In above image, we can see that 4 new matrices have been created namely: 1) X_train 2) X_test 3) y_train 4) y_test. Test matrices contain test data while Training variables contain training data. We have given 20% of data into testing part while 80% of data is given into training part. In above image we can see the number of rows and columns in each matrices and glimpse of values.

Step-7

```
In [28]: from sklearn.svm import SVR
    ...: regressor = SVR(kernel = 'rbf')
    ...: regressor.fit(X_train, y_train)
C:\Users\patel\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578:
DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
Out[28]:
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='auto',
  kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)

In [29]:
```

We have applied here the Support Vector Regression algorithm on our dataset and used ‘Radial Basis Function’ as our kernel. Once we fit the kernel, then we called ‘fit’ method which will take inputs X_train and y_train which are descriptive features and target features respectively. On right side of the screenshot, we can see that all features like size, coefficient, degree, epsilon etc.

Step-8

y_pred - NumPy array

	0
0	4005059
1	4003664
2	3338430
3	3397124
4	3115807
5	4083419
6	5378305
7	3995068
8	4958590
9	4817976
10	3802530

Here, we have given the X_test matrix as input of regressor object which is an object of SVR. It is noticeable that we have used an 'inverse_transform' method of StandardScalar library. The job of this function is to transform the standardized values into its original values. That is how we can have the values in money format (i.e. with standardized the data can look like 2.484939 but when we apply this function we can get the original revenue price i.e. 3204849). Here, the predicted values for test data is described, which is not in standardized form but in it's real form. These are the total number of revenue for our test dataset.

Step-9

```
In [32]: from sklearn.metrics import r2_score
    ...: r2score = r2_score(y_test, y_pred, sample_weight=None,
multioutput="uniform_average")
    ...:
    ...: print(r2score)
-9.98756707372e+12

In [33]:
```

Above mentioned value is R^2 measure score for SVM for regression with Radial Basis Function as kernel. Usually in all cases, the value of R^2 measure must be between 0-1. But as we can see here that the value obtained is negative and very less (in exponential of 12). This means that the model which we have used for this dataset is wrongly chosen. Hence, we can conclude that for this dataset, SVM for regression is not a good model.

Step-10

y_pred - NumPy array

	0
0	3914298
1	3900160
2	3798075
3	3827541
4	3831598
5	3929333
6	5095778
7	3889542
8	4247076
9	4003311
10	3934908

The above-mentioned screenshot is screenshot of the SVR algorithm, but the only change is kernel. In above photo, SVR algorithm is applied on test dataset by using kernel = 'poly'.R^2 measure for SVM with regression with "Polynomial" kernel is mentioned in the next image.

Step-11

```
In [62]: from sklearn.metrics import r2_score
    ...: r2score = r2_score(y_test, y_pred, sample_weight=None,
multioutput="uniform_average")
    ...:
    ...: print(r2score)
-9.62748626202e+12
```

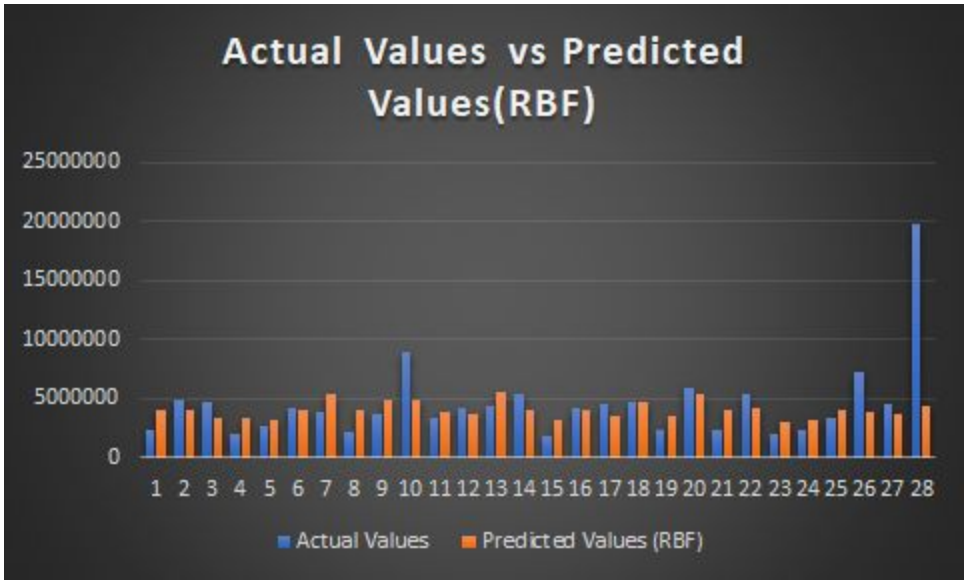
Here also the conclusion is the same, that SVM for regression is chosen by mistake. This model is not suitable for given dataset. Hence, it will be wise to use any other Regression model whoes R^2 measure comes between 0-1. The difference between results of both kernels is defined below briefly.

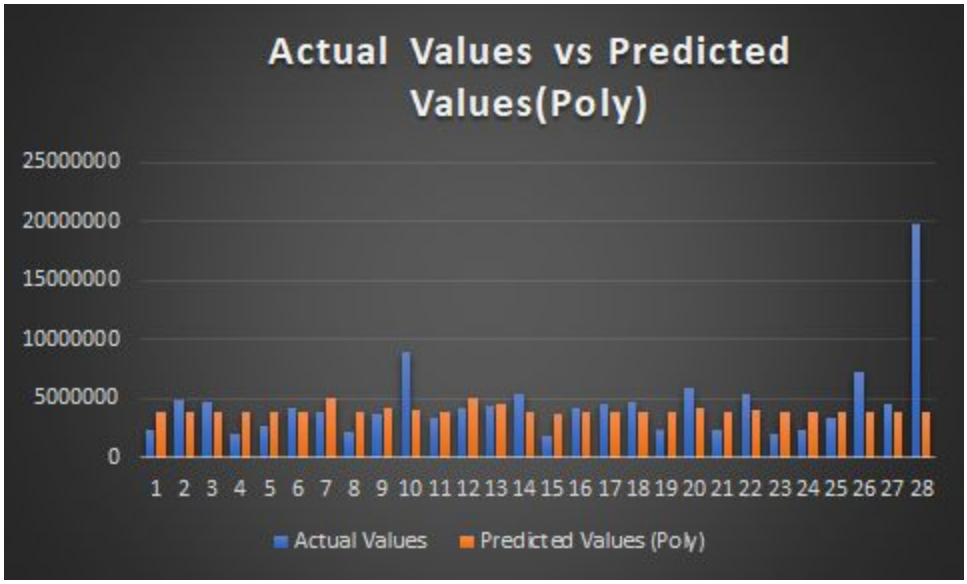
Step-12

Actual Values	Predicted Values (RBF)	Predicted Values (Poly)
2267425	4005059	3914298
4952497	4003664	3900160
4651866	3338430	3798075
2097022	3397124	3827541
2732645	3115807	3831598
4250553	4083419	3929333
3956086	5378305	5095778
2156098	3995068	3889542
3752885	4958590	4247076
8904084	4817976	4003311
3347767	3802530	3934908
4136425	3679909	5036062
4350573	5482412	4479908
5435276	4011316	3892872
1763231	3195832	3750267
4155435	4048682	3936488
4590423	3584322	3824447
4758476	4652873	3933932
2344689	3572247	3861869
5966193	5457470	4158067
2390534	3980198	3894360
5337526	4236699	3990202
2083447	3089577	3828483
2371202	3221194	3790427
3410878	4100493	3915313
7201784	3957044	3879404
4491607	3699564	3909354
19696939	4407066	3916315

The relation between these columns is described below. Firstly, we will compare the Actual Value column with predicted Values with RBF kernel. Then lastly we will compare Actual Values column with predicted values with Poly kernel.

9.4 Results





As, we can visualize from above mention graphs, that there is a major difference between the actual values and predicted values for both Radial Basis Function and Polynomial kernels of SVM for regression. And we can also see that both predicted values (RBF) and predicted values (POLY) are almost same.

9.5 References

1) http://www.saedsayad.com/support_vector_machine_reg.htm

2) Machine Learning for Predictive Data Analytics by: John D. Kelleher

Evaluation

10.1 Learning Goals

In this assignment, we applied different evaluation methods learned in the course to investigate the performance of different models we used during the course projects. Here, we learned which evaluation approach is appropriate for which model and how we should apply them in practice. In addition, we learned how to interpret the results of evaluation and evaluate the goodness of fit for each model.

10.1.1 Abstract

In this document, the team implemented confusion matrix for categorical target feature and calculated classification accuracy and misclassification rate, the combination of precision, recall and F1-measure for the project regarding Microsoft Azure, and RMSE for the model with continuous target feature. We provided the results of applying each evaluation criterion on corresponding models and discussed the models performances based on these results.

10.2 Introduction to Evaluation Approaches

In this section, we review different evaluation criteria used in performance examination of our models. Here, we present Confusion Matrix, RMSE, Precision, Recall and F1-Measure.

10.2.1 Confusion Matrix

One of the very useful analysis tools to capture what is happening in an evaluation test in a little more detail is confusion matrix which is the basis for many other performance measures. It calculates the frequency of different possible outcomes of the predictions made by the model for the test dataset. It shows the performance of the model. For a binary target feature, we can define four possible outcomes as follows.

- TP: Abbreviation for True Positive, when the model predicts an instance in the test set that had a positive target feature value and predicted positive by the model.
- TN: Abbreviation for True Negative, when the model predicts an instance in the test set that had a negative target feature value and predicted negative by the model.
- FP: Abbreviation for False Positive, when the model predicts an instance in the test set that had a negative target feature value and predicted positive by the model.
- FN: Abbreviation for False Negative, when the model predicts an instance in the test set that had a positive target feature value and predicted negative by the model.

The frequency of each of the above described outcomes are shown in a table called confusion matrix. Each cell in this matrix shows one of these outcomes and represents the frequency of outcome occurred when the test dataset is presented to the model. This structure is shown as follows. The column of the table is for prediction and the rows are related to true values of target feature. Top left of the matrix is TP, top right is FN, bottom left is FP and bottom right is TN which are all described above.

		Prediction	
		Positive	Negative
Target	Positive	TP	FN
	Negative	FP	TN

10.2.2 Misclassification Rate

From the confusion matrix it is obvious that values along the diagonal are related to correct model predictions and other ones are related to model's misclassification. In other words, we can use a measure called misclassification rate which is calculated based on confusion matrix and the formula is as follows.

$$misclassification\ rate = \frac{(FP+FN)}{(TP+TN+FP+FN)}$$

10.2.3 Classification Accuracy

In contrast to misclassification rate, there is another measure that completes our evaluation by showing how accurate our model is in predicting positive and negative outcomes. The formula for this measure is as follows.

$$classification\ accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)}$$

Later, we will show how we used these measures to evaluate our model with categorical target values after constructing confusion matrix for them.

10.2.4 Root Mean Squared Error (RMSE)

We have Mean Squared Error in the book to calculate error for continuous target features. However, there are some arguments regarding this criterion that they complain the actual mean squared error values themselves are not especially meaningful in relation to the scenario that a model is being used. This is because the use of the squared term and using root mean squared error solves its problem. RMSE is calculated using following formula.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (t_i - M(d_i))^2}{n}}$$

where t_i is the set of target values, $M(d_i)$ is corresponding model predictions.

10.2.5 Precision, Recall and F1-Measure

These measures are another way of evaluating the performance of the model. They are all based on confusion matrix. Precision shows the accuracy of the model in predicting positive predictions and recall shows the capability of the model on predicting positive outcomes out of all the positive outcomes of target feature. The formula for these two measures are as follows.

$$Precision = \frac{TP}{(TP+FP)},\ Recall = \frac{TP}{(TP+FN)}$$

These two measures need to be used together. In other words, we can take advantage of both and define a new measure. This measure is called F1-Measure which is a harmonic mean of precision and recall. The formula is as follows.

$$F_1 measure = 2\times\frac{(precision\times recall)}{(precision+recall)}$$

10.3 Implementation

10.3.1 Customer Churn Prediction Model

Here, we evaluate the models we applied on our customer churn prediction problem. We have done the estimation of the model fitting, the evaluation presents us with the righteousness of the decision we made with the data, it lets us check if it fits the solution that our initial problem is required for, it provides us with a chance to refine the model if not satisfied.

There are 2 major common type of errors in machine learning for this type of prediction. Firstly, False Negative (Type 1 error) which is to identify the customer who is likely to churn. Secondly, it is False Positive (Type 2 error) in which the customer is satisfied with the company yet decides to leave both leads to the revenue loss, the first case is why we are building the model, in the second the company is wasting the resources on the customer who churns despite the satisfaction.
Following is the code we used to find the confusion matrix and the results are presented.

Code for the Confusion matrix

```
# Calculating confusion matrix for machine learning prediction and
cm= confusion_matrix(y_test, y_pred)
print("The confusion matrix is described below.")
print(cm)
```

		Prediction	
		Positive	Negative
Target	Positive	106	346
	Negative	69	1479

Now, we can calculate misclassification rate and classification accuracy based on the formula we described before. We have:

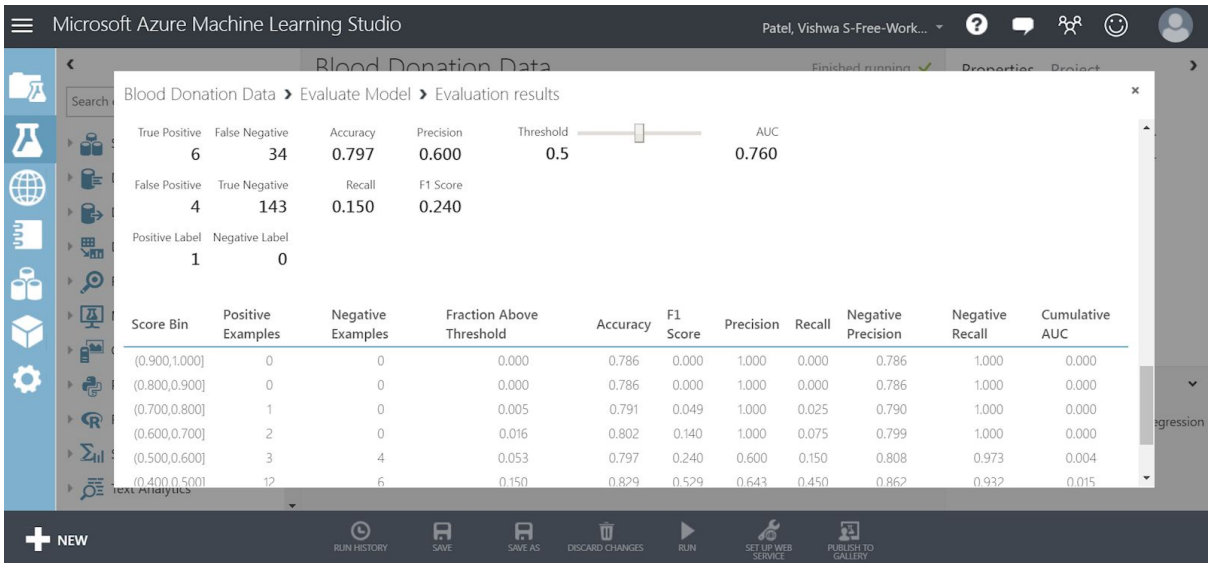
$$misclassification\ rate = \frac{(FP+FN)}{(TP+TN+FP+FN)} = \frac{415}{2000} = 0.2075$$

$$classification\ accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} = \frac{1585}{2000} = 0.7925$$

These measures show that our model has a reasonable performance and especially can predict most of the non-churn customers. The performance of the model is justified by the accuracy and the misclassification is around 20%, and the accuracy is around 79.25%.

10.3.2 Microsoft Azure for Customer Churn

We used our dataset and applied the predictive model using Microsoft Azure. In previous report, we showed how to implement a model in Microsoft Azure. Here, we show the results of implementing predictive model to Customer Churn problem.
Microsoft Azure provides confusion matrix for users. The results for confusion matrix for the model we implemented before is retrieved as follows.



$$Precision = \frac{TP}{(TP + FP)}$$

Precision = 0.600
Precision tells us how confident we can be that an instance predicted to have the positive target level (prediction that customer will stay) actually has the positive target level. Since precision in our case is 0.6, 40% of the time we might predict a customer to churn when they are likely to stay. This should be fine as this is the side we would wish to err on.

$$Recall = \frac{TP}{(TP + FN)}$$

Recall = 0.150
Recall tells us how confident we can be that all the instances with the positive target level (prediction that customer will stay) have been found by the model. Since we are more interested in the customers who might churn than who might stay, the lower score for recall may not be a problem to continue using this model.

$$F_1\ Measure = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} = 0.240$$

The low value of $F_1\ Measure$ indicates that the model is not performing well. Since we are interested in finding the customers churn we could possibly change the model such that a positive level indicates a customer who is likely to churn instead of a customer who is likely to stay. This could potentially improve the performance of the model.

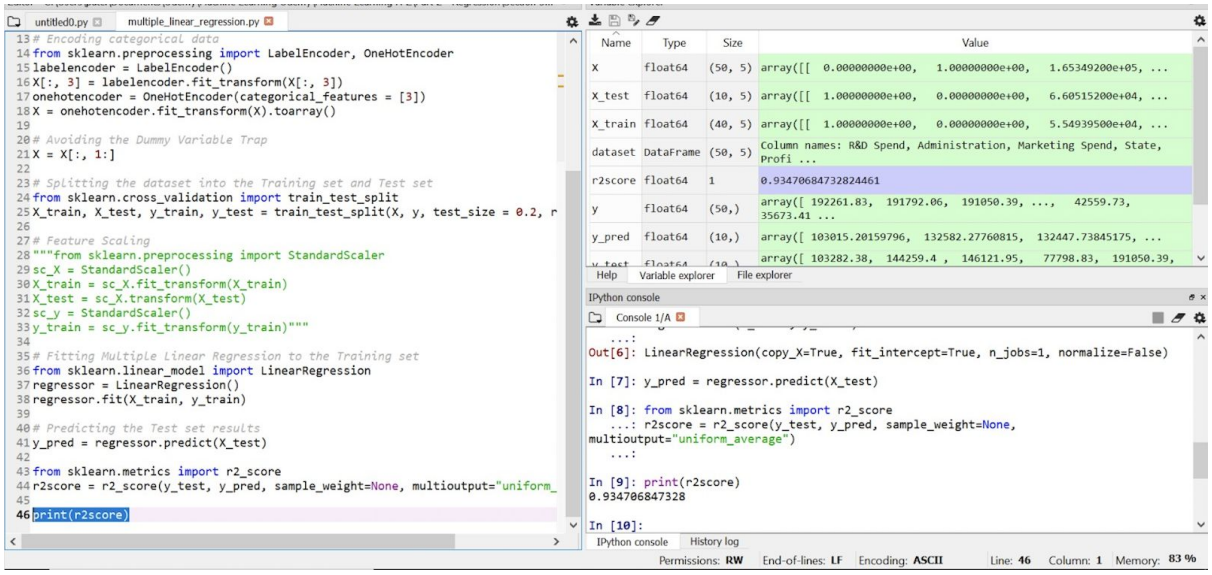
10.3.3 Model Evaluation with Continuous Target Feature

Here, we used a dataset in which there are four descriptive features including R&D expenditure, Administration Cost, Marketing expenditure and State to predict Profit as the continuous target feature. Following table shows a small portion of the data we used to implement the model.

R&D Spend	Administration	Marketing Spend	State	Profit
165349.20	136897.80	471784.10	New York	192261.83
162597.70	151377.59	443898.45	California	191792.06
153441.51	101145.55	407934.54	Florida	191050.39
142107.34	118671.85	383199.62	New York	182901.99
131876.90	91391.71	366168.42	Florida	166187.94
134615.46	99814.71	362861.36	New York	156991.12
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

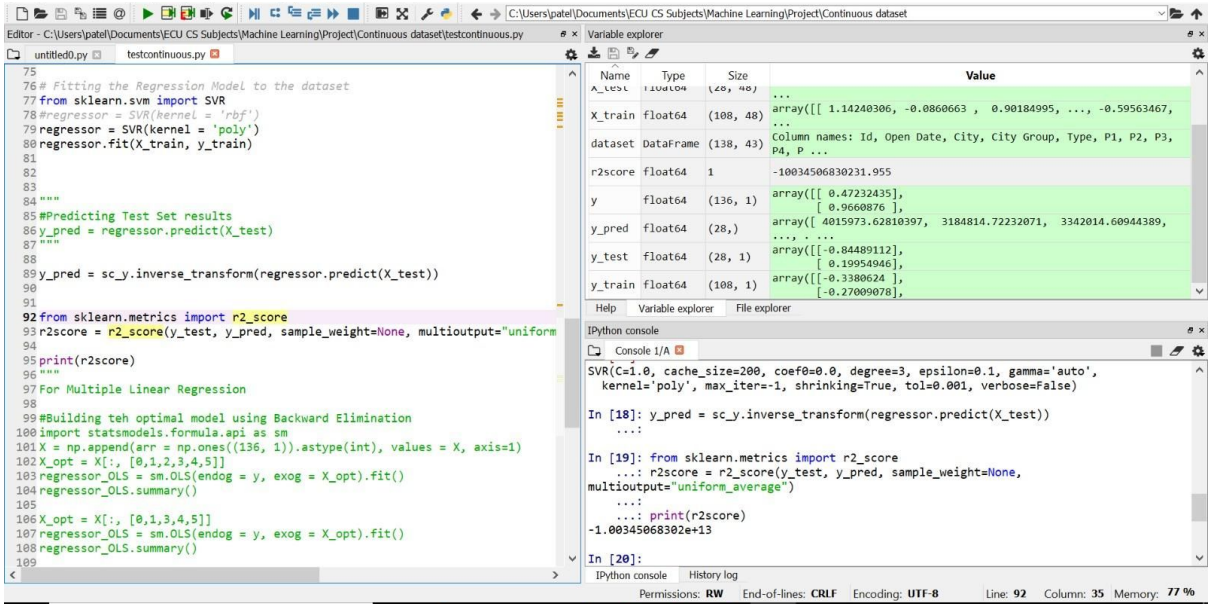
As the evaluation criteria, based on the discussion we had above, R^2 is selected as the evaluation criterion with the formula explained above. Following is the code we used to retrieve R^2 score for the model applied on the dataset. We here show the formula again.

$$R^2 = 1 - \frac{sum\ of\ squared\ errors}{total\ sum\ of\ squares}$$



The result, as shown in the figure, shows R^2 value of 0.934706847328. This value indicated a better value of the performance of the model and overall the model is efficient in terms of prediction capability.

We also performed the evaluation on the continuous dataset from a group of restaurants on which we have to predict the revenue.



As shown in the model the value of R^2 is -1.003. The model eventually failed during the evaluation which resulted in a negative of R^2. This is used to demonstrate not all the dataset will result the same evaluation, it is a measure of performance and evaluation. R^2 compares the fit of the chosen model with that of a horizontal straight line (the null hypothesis). If the chosen model fits worse than a horizontal line, then R^2 is negative. Note that R^2 is not always the

square of anything, so it can have a negative value without violating any rules of math. R^2 is negative only when the chosen model does not follow the trend of the data, so fits worse than a horizontal line. In our case, it means that the model applied is not performing well at all. It means that if we used a simple horizontal line instead of the prediction model, the error would be less. This was a great experience to understand not all models are suitable for a dataset.

10.4 Conclusion

In this assignment, we evaluate different models applied to different datasets during the project assignments for machine learning course. Since we had categorical and continuous target features for different models, we used different evaluation criteria including confusion-matrix-based criteria including misclassification rate and classification accuracy, RMSE and R^2, precision, recall and F1-measure. We used misclassification rate and classification accuracy for customer churn problem when we used our code to predict the results. The results for evaluation of this model shows we have a good performance regarding customer churn. When we used Microsoft Azure, we evaluate the model using precision, recall and F1-measure. The results showed poor performance of the model which we intentionally selected to see the bad performance of a model. Last but not least, we selected to datasets with continuous target features to show good and bad performance of a model using R^2 criterion. The model with profit prediction has R^2 value of 94% which shows high performance of the model. The other one, restaurant revenue prediction, the model resulted in a negative R^2 value which shows even a horizontal line can predict better than the model. This shows not all models are suitable for prediction of a dataset.

10.5 Individual Contributions

William Clark	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
Sumati Kulkarni	Provided invaluable contributions to the completion of the tasks assigned to the group Reviewed Report
Babak Maleki Shoja	Provided invaluable contributions to the completion of the tasks assigned to the group Implementation Drafted Report Reviewed Report
Venkatesh Reedy Pala	Provided invaluable contributions to the completion of the tasks assigned to the group Drafted Report Reviewed Report
Vishwa Patel	Provided invaluable contributions to the completion of the tasks assigned to the group for this project Implementation Drafted report Reviewed Report

10.6 Team Summary

This assignment provided an insight regarding evaluation process in CRISP-DM approach. Using different approaches for evaluation and applying them on different datasets, this was a great opportunity toward learning the evaluation part and how to apply it in practice.

10.7 References

- Kelleher, John, Namee, Brian Mac, Arcy, Aoife D'. (2015). Fundamentals of Machine Learning for Predictive Data Analytics. The MIT Press Cambridge, Massachusetts London, England

Code and Dataset Link

https://github.com/patelvishwa112/Machine-Learning-Project