Queen Mary
University of London

DATA ANALYTICS COURSEWORK

Submitted by Cara Evangeline Chandra Mohan - 190539421

Purchasing Intention of Online Shoppers

Group Size : 3

Student	Effort
Cara Evangeline Chandra Mohan	40
Sai Spandana Dadi	30
Sai Abhishek Samasani	30

Department of EECS

Msc Big Data Science 2019-2020

TABLE OF CONTENTS

1.Introduction and Motivation

Modern day e-Commerce is being used by everyone everywhere and the importance of retaining the online customers is necessary for a company to progress well. You can buy almost anything nowadays with just a few clicks, Clothes, appliances, gadgets, food, even cars – the list of what you can buy online is endless and all you need is an internet connection.

The increase in e-commerce usage by people over the past few years has made a great potential in the market, but the conversion rates [percentage of visitors that land on a website and complete a desired action] have not increased at the same rate and this leads to the necessity for solutions that present customized promotions to the online shoppers as suggested by ***Suchacka [1]***.

In physical merchandise, a salesperson offers a range of customized alternatives to customers based on the experience he/she has gained over time. This experience has an important influence on the effective use of time, purchase conversion rates, and sales figures as concluded by ***Moe et al.[2]***. Many e-commerce and IT companies have invested in early detection and behavioral prediction systems which imitate the behavior of a salesperson in a virtual shopping environment. In parallel along with these efforts, some academic studies addressing the problem from different perspectives using machine learning is done to achieve a good conversion rate using clickstream and session information data. Here we have worked on real time online shoppers purchasing behavior to predict the visitor's purchasing intention.

2.Project Aim

Our main interest and aim of choosing this project, is to deal with the interest and dis-interest of customers in buying a product online. In a crisis and such a time as this (COVID-19) the previous study made by the researchers helped the online market to pick up it's sale and overcome obstacles. Our aim is to:

- Visually analyse and explore the trends behind not proceeding to checkout and unsuccessfully skipping the page of interest.

- Identifying the important features that contribute towards classifying the purchaser’s intention.
- Classifying the e-customers based on their clicking trend whether they will successfully buy a product online.
- Creating a Directed Acyclic Graph to see the dependencies of features.

3.Paperwork

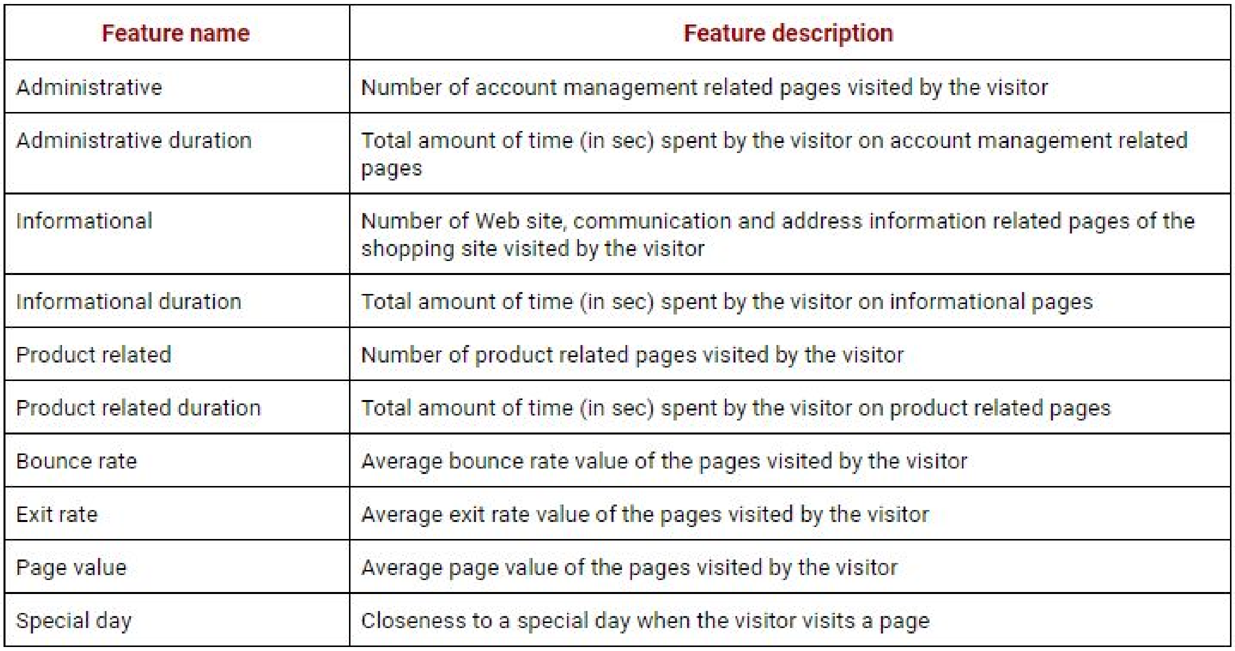
Many research studies have been done in this area where authors have suggested different methodologies to categorize the customers and to increase the conversion rate based on user click in a website.

- ***Okan et al [3]*** proposed multilayer perceptron algorithm to predict visitor’s purchasing intention and neural network algorithm involving LSTM-RNN to predict the likelihood to abandon the site.
- ***Moe [2]*** worked to categorize the visits of a user using data obtained from a given online store and the system developed will take customized actions based on the category of visit.
- In another study, ***Mobasher et al. [4]*** sets up two different clustering models which are based on transactions and pageviews of the user to derive useful aggregate usage profiles that can be effectively used by recommender systems to take specific actions in real time.
- ***K-nearest neighbor (KNN)*** classifier was used in [5] to categorize the user sessions either as browsing or buyer sessions based on historical data that was collected from an online bookstore.
- ***Budnikas [6]*** proposed to classify the visitor behavior patterns with the aim of determining the Web site component that has the highest impact on a fulfillment of business objectives.

4.Data Collection

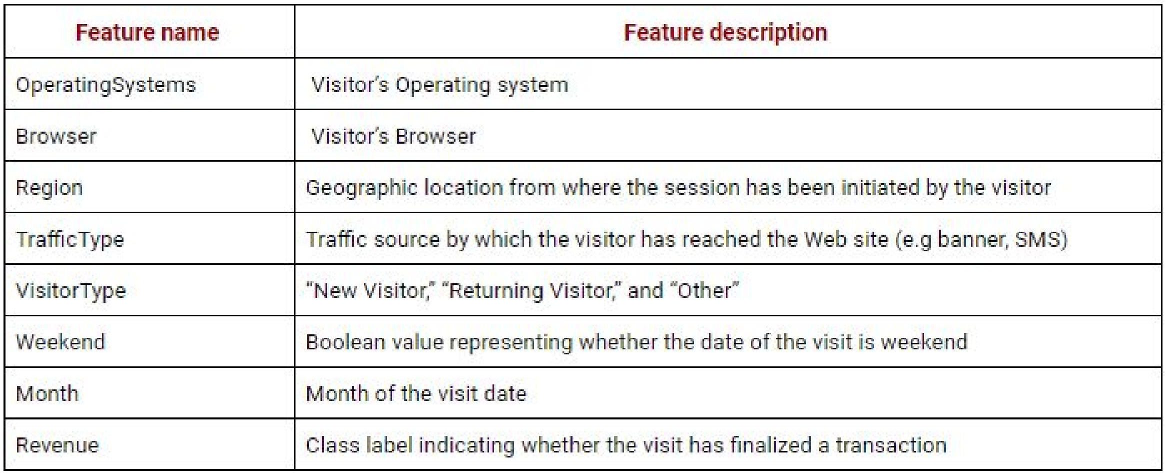
The dataset that we used in this Data analytics project is ‘Online Shoppers Purchasing Intention’ provided on the UC Irvine’s Machine Learning Repository. The dataset is in clean format with no missing values. For further analysis, the dataset is preprocessed according to the requirements.

There are 12,330 sessions (ie.tuples) in the dataset, with 18 attributes and table 1 and 2 describes the numerical and categorical features.



Feature name	Feature description
Administrative	Number of account management related pages visited by the visitor
Administrative duration	Total amount of time (in sec) spent by the visitor on account management related pages
Informational	Number of Web site, communication and address information related pages of the shopping site visited by the visitor
Informational duration	Total amount of time (in sec) spent by the visitor on informational pages
Product related	Number of product related pages visited by the visitor
Product related duration	Total amount of time (in sec) spent by the visitor on product related pages
Bounce rate	Average bounce rate value of the pages visited by the visitor
Exit rate	Average exit rate value of the pages visited by the visitor
Page value	Average page value of the pages visited by the visitor
Special day	Closeness to a special day when the visitor visits a page

Table 1: Description of numerical attributes



Feature name	Feature description
OperatingSystems	Visitor's Operating system
Browser	Visitor's Browser
Region	Geographic location from where the session has been initiated by the visitor
TrafficType	Traffic source by which the visitor has reached the Web site (e.g banner, SMS)
VisitorType	"New Visitor," "Returning Visitor," and "Other"
Weekend	Boolean value representing whether the date of the visit is weekend
Month	Month of the visit date
Revenue	Class label indicating whether the visit has finalized a transaction

Table 2: Description of categorical attributes

- ***Administrative, Administrative Duration, Informational, Informational Duration, Product Related and Product Related Duration***, the values of these features are derived from the URL information of the pages visited by the user and updated in real time when a user takes an action like moving from one page to another page.
- The ***Bounce Rate, Exit Rate and Page Value*** features shown in Table 1 represent the metrics measured by "Google Analytics" for each page in the e-commerce site. These values can be stored in the application database for all Web pages of the e-commerce site in the developed system and updated automatically at regular intervals.
- The value of ***Special day*** attribute is determined using the dynamics of e-commerce such as the duration between the order date and the delivery date.

5.Requirements

[1] The Language that we have used to perform analysis of our project is mainly ***Python in Google colab***. And following are the libraries used in python.

- ***Scikit-learn*** is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and

DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

- ***NumPy*** is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- ***Pandas*** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
- ***Seaborn*** is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- ***Matplotlib*** is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.

[2] ***Weka*** contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to these functions. We have used Weka software to perform data visualization and feature selection in a few places.

6.Data Preprocessing

There are a number of data preprocessing techniques.

- ***Data cleaning*** is applied to remove noise and inconsistencies in the data and also to fill missing values.
- ***Data integration*** combines data from multiple sources into one single coherent data store such as a data warehouse.
- ***Data transformations*** such as normalization, conversion to one-hot encode may be applied.
- ***Data reduction*** can reduce the data size by aggregating data cubes, eliminating redundant features or clustering.

The above techniques are not mutually exclusive; they sometimes may work together. For example, data cleaning may involve transformations to correct the inconsistent data by transforming all entries of a date field to a common format.

Data processing techniques when applied before mining of algorithms, can substantially improve the overall quality of the patterns mined and/or the time required for the actual mining.

6.1. Data Cleaning

Data cleaning routines work to clean the data by filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies. Dirty data can cause confusion for the mining procedure, resulting in unreliable output. Although most mining algorithms have some procedures for dealing with incomplete or noisy data, they are not always robust. Instead, they may concentrate on avoiding overfitting the data to the function being modeled. Therefore, useful pre-processing steps must be run through your data.

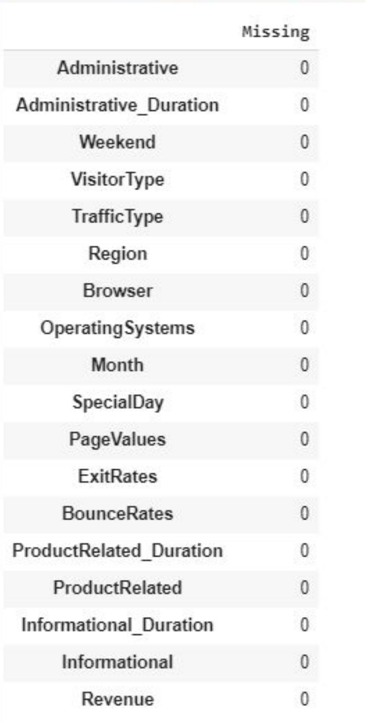
6.1.1. Missing Values

The concept of missing values is important to understand in order to successfully manage data. If the missing values are not handled properly by the data science researcher, then he/she may end up getting an inaccurate inference about the data. Because of improper handling, the result obtained by the researcher will differ from actual ones.

Code used to find missing tuples:

```
pd.DataFrame({'Missing':df.isnull().sum()}).sort_values
(by = ['Missing'], ascending=False).head(18)
```

From the given figure Fig 1, we come to a conclusion that there are no missing values in our dataset.



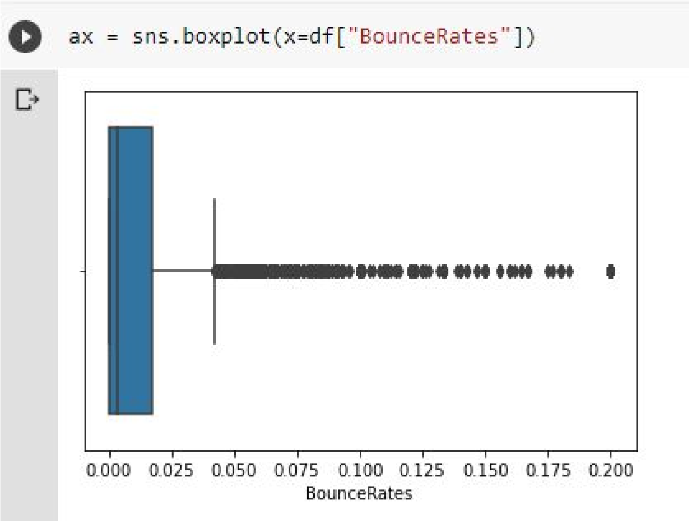
	Missing
Administrative	0
Administrative_Duration	0
Weekend	0
VisitorType	0
TrafficType	0
Region	0
Browser	0
OperatingSystems	0
Month	0
SpecialDay	0
PageValues	0
ExitRates	0
BounceRates	0
ProductRelated_Duration	0
ProductRelated	0
Informational_Duration	0
Informational	0
Revenue	0

Fig 1: Missing values

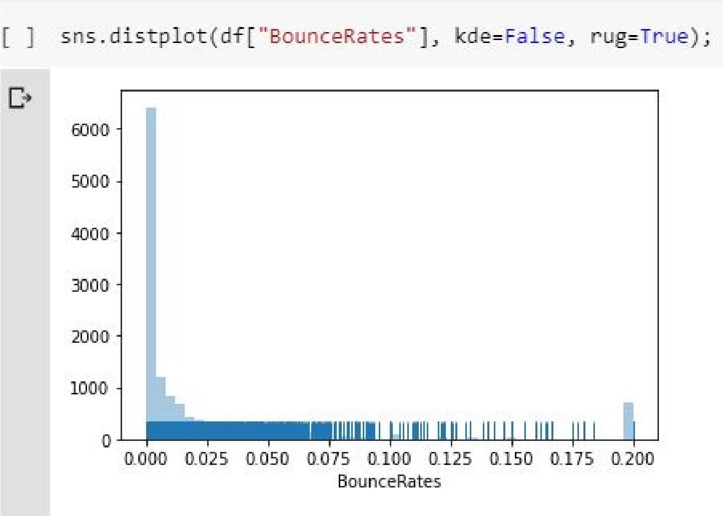
6.1.2. Noisy Data

Noisy data can be in the form of outliers. An outlier is a data point that differs significantly from other observations. An outlier may be due to variability in the measurement or experimental error, the latter is sometimes excluded from the data set. An outlier can cause serious problems in statistical analyses.

We can observe outliers using a boxplot where any data point that falls outside [-IQR*1.5, IQR*1.5] is considered as an outlier.



```
ax = sns.boxplot(x=df["BounceRates"])
```



```
[ ] sns.distplot(df["BounceRates"], kde=False, rug=True);
```

Fig 2: Boxplot of BounceRates

Fig 3: Distribution of BounceRates

Outliers can occur by chance in any distribution, but they also often indicate either measurement error or indicate that the population has a heavy-tailed distribution. In the former case one may discard them or use statistics/mining that are robust to outliers, whereas in the latter case they indicate that the distribution has high skewness and that one should be very cautious while using intuitions or tools that assume a normal distribution.

The above boxplot(Fig2) and histogram(Fig3) representation of the attribute 'BounceRates' summarizes the fact that out dataset distribution has high positive skewness and we don't need to process the data point that lie outside [-IQR*1.5, IQR*1.5] and even other attributes have produced a similar outcome indicating the skewness of the dataset.

6.2. Data Integration

During the stage of data integration, various data stores are used. This may possibly lead to the problem of redundancy in data. An attribute is called redundant if it can be derived from any other attribute or set of attributes. Inconsistencies in attribute naming can also lead to the redundancies in the data set.

Redundancies can be detected using following methods

- X^2 Test (For nominal Data or qualitative data or categorical data)
- Correlation coefficient and covariance (For numeric data)

We haven't combined any data to 'Online shoppers purchasing intention' dataset, yet before releasing the dataset in UCI repository, there would have been a probability of combining the attributes from various different sources, so testing for the correlation of attributes in this case is equally important.

From the correlation map in Fig 4, we can infer that 'ExitRates' and 'BounceRates' correlates the most. Now let us analyse the coefficient of correlation between every attribute.

From the feature correlation graph in Fig 5, the coefficient between the 2 attributes is 0.91.

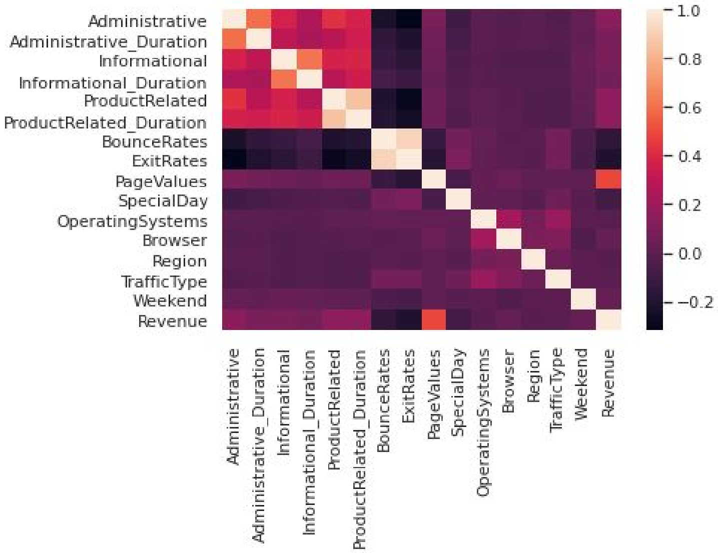
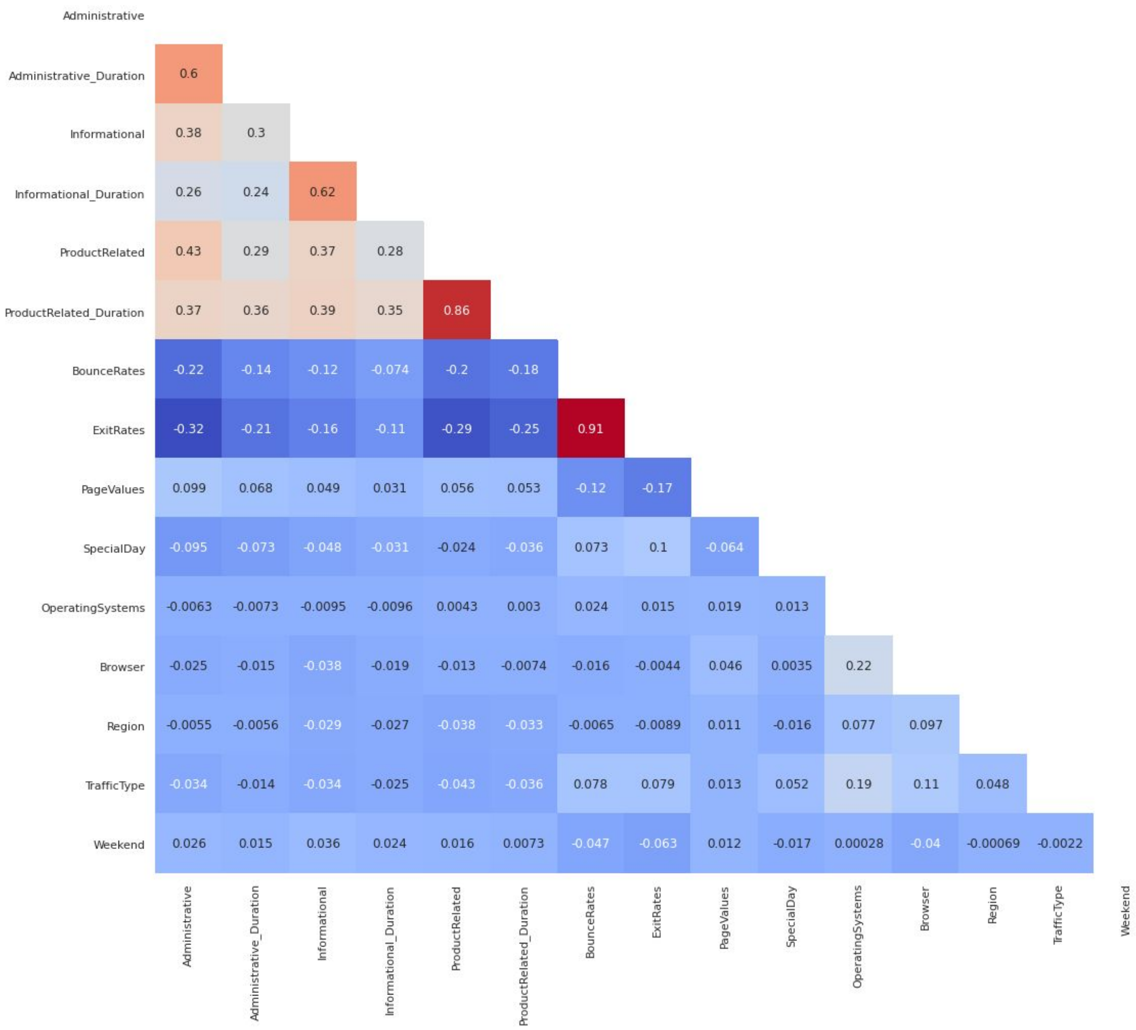
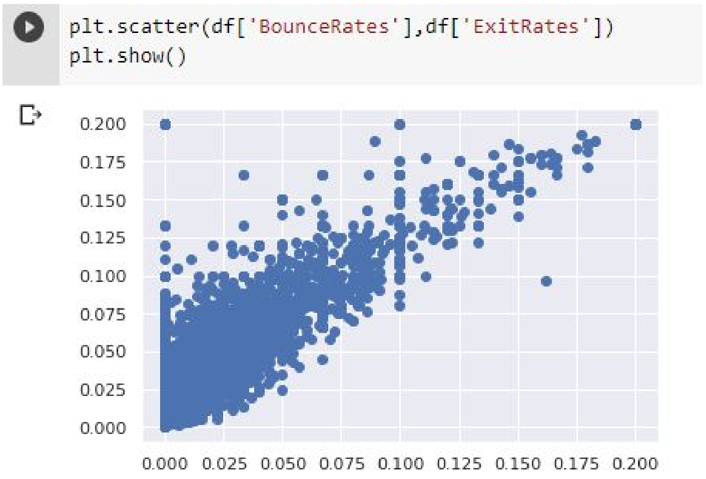


Fig 4: Correlation Map



	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	SpecialDay	OperatingSystems	Browser	Region	TrafficType	Weekend
Administrative															
Administrative_Duration	0.6														
Informational	0.38	0.3													
Informational_Duration	0.26	0.24	0.62												
ProductRelated	0.43	0.29	0.37	0.28											
ProductRelated_Duration	0.37	0.36	0.39	0.35	0.86										
BounceRates	-0.22	-0.14	-0.12	-0.074	-0.2	-0.18									
ExitRates	-0.32	-0.21	-0.16	-0.11	-0.29	-0.25	0.91								
PageValues	0.099	0.068	0.049	0.031	0.056	0.053	-0.12	-0.17							
SpecialDay	-0.095	-0.073	-0.048	-0.031	-0.024	-0.036	0.073	0.1	-0.064						
OperatingSystems	-0.0063	-0.0073	-0.0095	-0.0096	0.0043	0.003	0.024	0.015	0.019	0.013					
Browser	-0.025	-0.015	-0.038	-0.019	-0.013	-0.0074	-0.016	-0.0044	0.046	0.0035	0.22				
Region	-0.0055	-0.0056	-0.029	-0.027	-0.038	-0.033	-0.0065	-0.0089	0.011	-0.016	0.077	0.097			
TrafficType	-0.034	-0.014	-0.034	-0.025	-0.043	-0.036	0.078	0.079	0.013	0.052	0.19	0.11	0.048		
Weekend	0.026	0.015	0.036	0.024	0.016	0.0073	-0.047	-0.063	0.012	-0.017	0.00028	-0.04	-0.00069	-0.0022	

Fig-5:Feature Correlation



```
plt.scatter(df['BounceRates'],df['ExitRates'])
plt.show()
```

Here in Fig 6, as we can see, we don't need to remove either of the attributes since the correlation coefficient is not ~1 and BounceRates and ExitRates represent two different aspects.

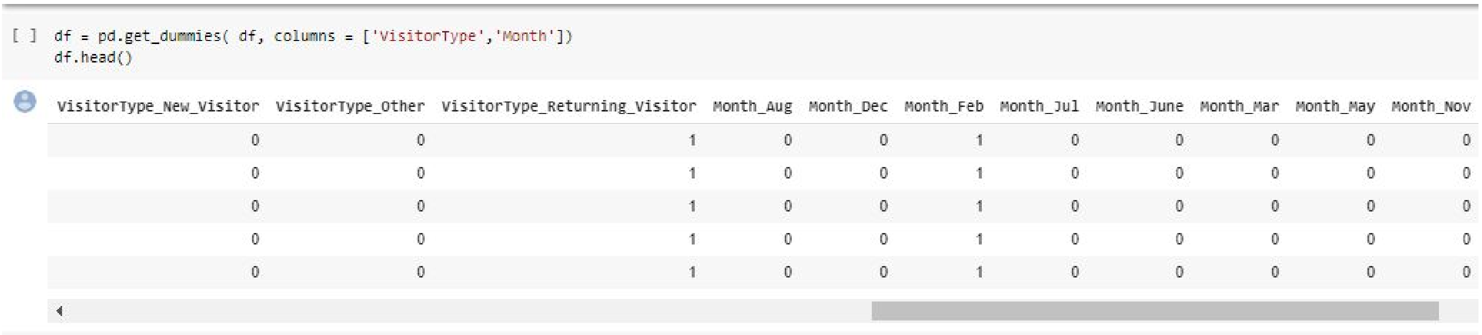
Fig 6: Correlation b/w BouceRates and ExitRates

6.3. Data Transformation

Data transformation is the process of changing the format or values of data. For data analytics projects, data can be transformed at two stages of the data pipeline either ETL (extract, transform, load) or ELT (extract, load, transform). Since our dataset is not of big volume we performed ELT.

In our dataset there are two categorical attributes which are non-numerical, so we transformed two attributes [‘VisitorType’ and ‘Month’] to one-hot encoded vectors as shown in Fig 7.

Fig 7: Dataset after transforming to one-hot encode



```
df = pd.get_dummies( df, columns = ['VisitorType','Month'])
df.head()
```

VisitorType_New_Visitor	VisitorType_Other	VisitorType_Returning_Visitor	Month_Aug	Month_Dec	Month_Feb	Month_Jul	Month_June	Month_Mar	Month_May	Month_Nov
0	0	1	0	0	1	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0

6.4. Data Reduction

Data reduction techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet almost closely maintains the integrity of the original data. The mining on the reduced data set should be more efficient yet should produce the same (or almost the same) analytical results. Methodologies for data reduction include the following:

1. Date cube aggregation
2. Data compression
3. Dimension reduction / feature selection
4. Numerosity reduction
5. Discretization and concept hierarchy generation

For our dataset we perform three of the data reduction method for analysis:

6.4.1. Data Compression

In Data Integration step we observed the following correlation coefficients for 6 attributes

- ‘Administrative’, ‘Administrative_Duration’ - 0.6
- ‘Informational’, ‘Informational_Duration’ - 0.62
- ‘ProductRelated’, ‘ProductRelated_Duration’ - 0.86

So, We converted these 6 attributes to 3 attributes

- AvgAdministrative = Administrative_Duration / Administrative
- AvgInformational = Informational_Duration / Informational
- AvgProductRelated = ProductRelated_Duration / ProductRelated

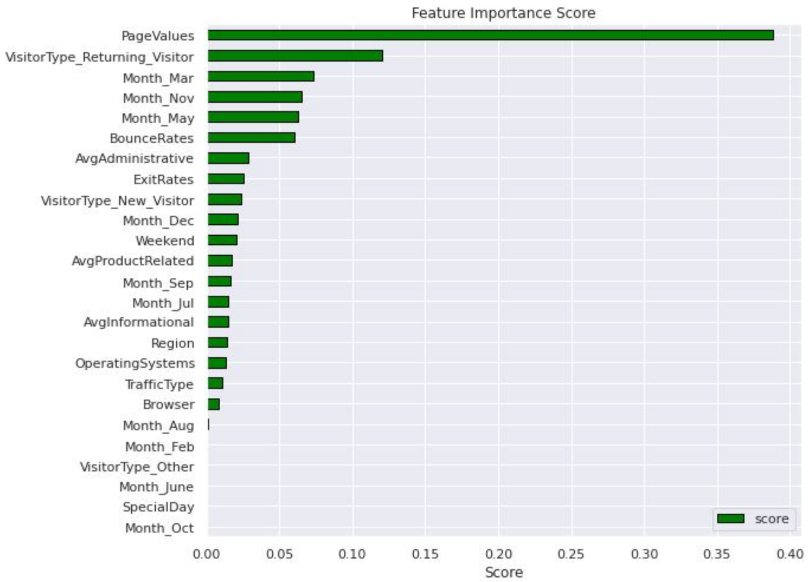
6.4.2. Feature Selection

Method 1: XGBoost

The more an attribute is used to make key decisions with decision trees, the higher its relative importance. This importance is calculated explicitly for each attribute in the dataset, allowing attributes to be ranked and compared to each other.

Importance is calculated for a single decision tree by the amount that each attribute split point improves the performance measure. The performance measure may be the purity (Gini index or Gain ratio) used to select the split points or another more specific error function. The feature importances are then averaged across all of the decision trees within the model.

Fig 8 : Feature importance score vs Features plot

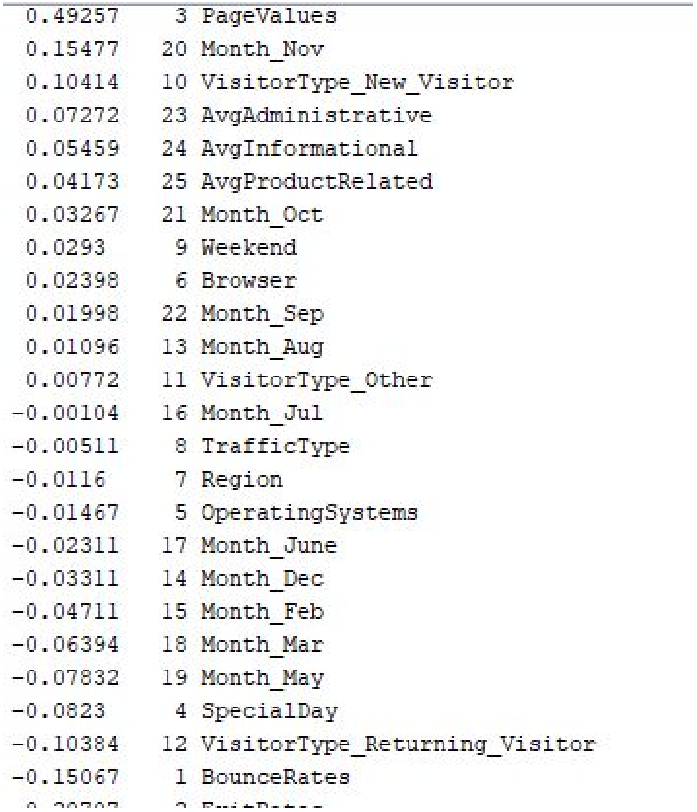


In our dataset it is observed that the feature ‘PageValues’ has the highest score. The ‘PageValues’ feature represents the average value for a page that a user visited before completing an e-commerce transaction, it can be seen as an important contribution to analyze a visitor’s transaction finalization decision.

Method 2:(Pearson correlation coefficient)

A popular technique for selecting the most relevant attributes in your dataset is to use correlation formally referred to as Pearson’s correlation coefficient. You can calculate the correlation between each attribute and the output variable and select only those attributes that have a moderate-to-high positive or negative correlation (close to -1 or 1) and drop those attributes with a low correlation (value close to zero). We used Weka to process the feature prioritization as shown in Fig 9.

Weka supports correlation based feature selection with the CorrelationAttributeEval technique that requires use of a Ranker search method.



```
 0.49257    3 PageValues
 0.15477   20 Month_Nov
 0.10414   10 VisitorType_New_Visitor
 0.07272   23 AvgAdministrative
 0.05459   24 AvgInformational
 0.04173   25 AvgProductRelated
 0.03267   21 Month_Oct
 0.0293     9 Weekend
 0.02398    6 Browser
 0.01998   22 Month_Sep
 0.01096   13 Month_Aug
 0.00772   11 VisitorType_Other
-0.00104   16 Month_Jul
-0.00511    8 TrafficType
-0.0116     7 Region
-0.01467    5 OperatingSystems
-0.02311   17 Month_June
-0.03311   14 Month_Dec
-0.04711   15 Month_Feb
-0.06394   18 Month_Mar
-0.07832   19 Month_May
-0.0823     4 SpecialDay
-0.10384   12 VisitorType_Returning_Visitor
-0.15067    1 BounceRates
-0.20707    2 ExitRates
```

Fig 9: Pearson correlation coefficient

Note: Discretization is Defined in Section II for structure Learning

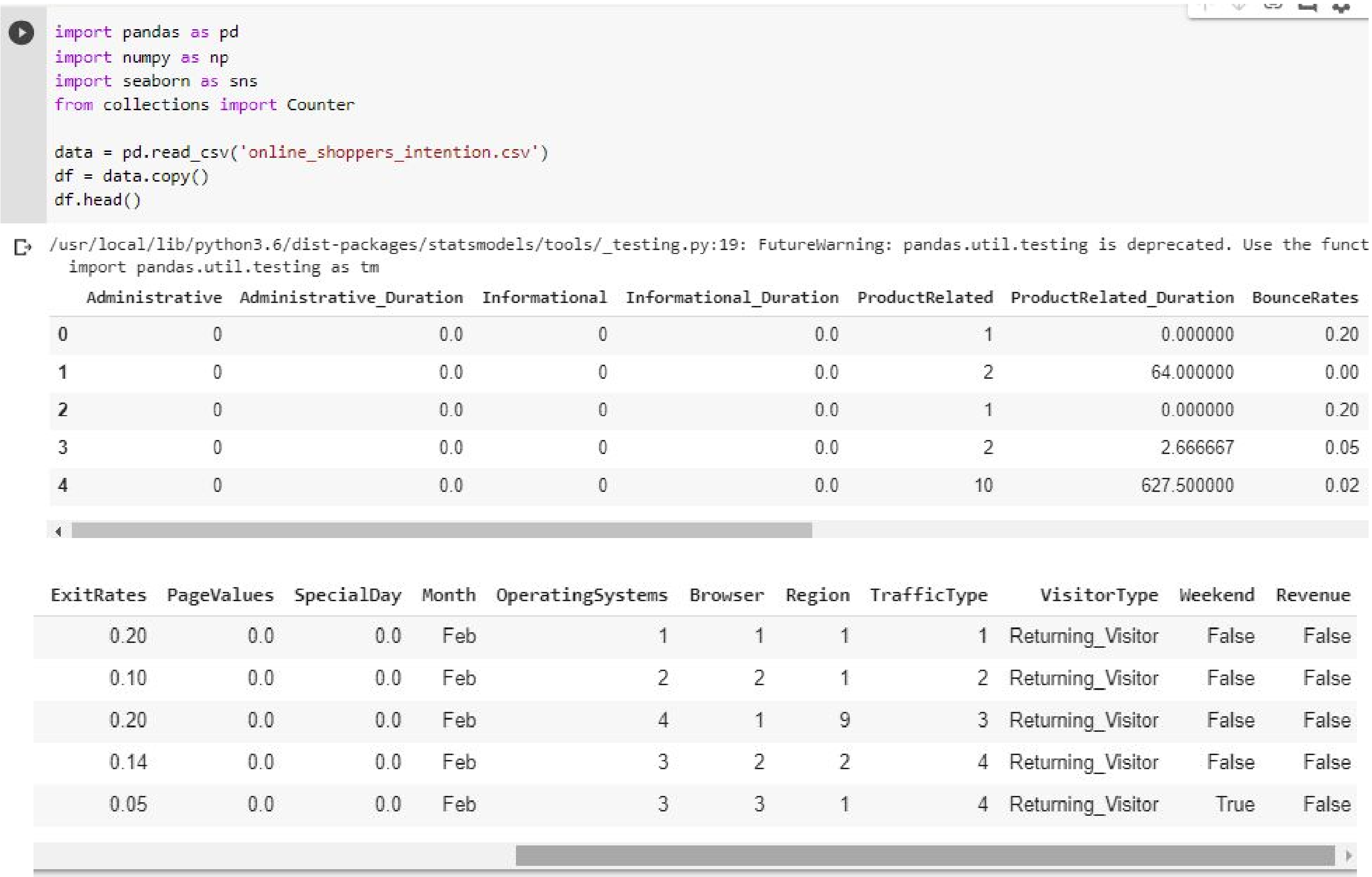
SECTION – I

DATA EXPLORATION AND ANALYSIS

Data Exploration and Visualization

Data visualization is the act of taking information (data) and placing it into a visual context, such as a map or graph. Data visualizations make big and small data easier for the human brain to understand, and visualization also makes it easier to detect patterns, trends, and outliers in groups of data. Good data visualizations should place meaning into complicated datasets so that the message is clear and concise. So before the analysis, to understand the data precisely we have explored the data by using various libraries to plot the graphs required.

7.1. How does the dataset look before pre-processing?



```
import pandas as pd
import numpy as np
import seaborn as sns
from collections import Counter

data = pd.read_csv('online_shoppers_intention.csv')
df = data.copy()
df.head()
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the funct
  import pandas.util.testing as tm
```

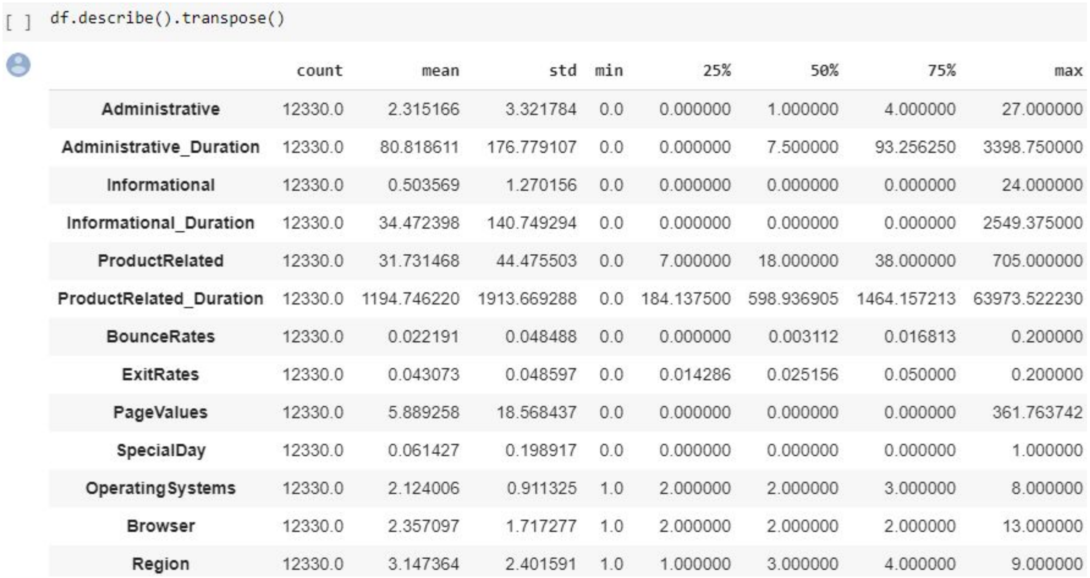
	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates
0	0	0.0	0	0.0	1	0.000000	0.20
1	0	0.0	0	0.0	2	64.000000	0.00
2	0	0.0	0	0.0	1	0.000000	0.20
3	0	0.0	0	0.0	2	2.666667	0.05
4	0	0.0	0	0.0	10	627.500000	0.02

ExitRates	PageValues	SpecialDay	Month	OperatingSystems	Browser	Region	TrafficType	VisitorType	Weekend	Revenue
0.20	0.0	0.0	Feb	1	1	1	1	Returning_Visitor	False	False
0.10	0.0	0.0	Feb	2	2	1	2	Returning_Visitor	False	False
0.20	0.0	0.0	Feb	4	1	9	3	Returning_Visitor	False	False
0.14	0.0	0.0	Feb	3	2	2	4	Returning_Visitor	False	False
0.05	0.0	0.0	Feb	3	3	1	4	Returning_Visitor	True	False

Fig 10: Dataset sample

The figure depicts the initial look of a sample of the dataset before checking for any redundancy, missing values or any form of pre-processing

7.2. Statistics of the Numerical attributes



```
[ ] df.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
Administrative	12330.0	2.315166	3.321784	0.0	0.000000	1.000000	4.000000	27.000000
Administrative_Duration	12330.0	80.818611	176.779107	0.0	0.000000	7.500000	93.256250	3398.750000
Informational	12330.0	0.503569	1.270156	0.0	0.000000	0.000000	0.000000	24.000000
Informational_Duration	12330.0	34.472398	140.749294	0.0	0.000000	0.000000	0.000000	2549.375000
ProductRelated	12330.0	31.731468	44.475503	0.0	7.000000	18.000000	38.000000	705.000000
ProductRelated_Duration	12330.0	1194.746220	1913.669288	0.0	184.137500	598.936905	1464.157213	63973.522230
BounceRates	12330.0	0.022191	0.048488	0.0	0.000000	0.003112	0.016813	0.200000
ExitRates	12330.0	0.043073	0.048597	0.0	0.014286	0.025156	0.050000	0.200000
PageValues	12330.0	5.889258	18.568437	0.0	0.000000	0.000000	0.000000	361.763742
SpecialDay	12330.0	0.061427	0.198917	0.0	0.000000	0.000000	0.000000	1.000000
OperatingSystems	12330.0	2.124006	0.911325	1.0	2.000000	2.000000	3.000000	8.000000
Browser	12330.0	2.357097	1.717277	1.0	2.000000	2.000000	2.000000	13.000000
Region	12330.0	3.147364	2.401591	1.0	1.000000	3.000000	4.000000	9.000000

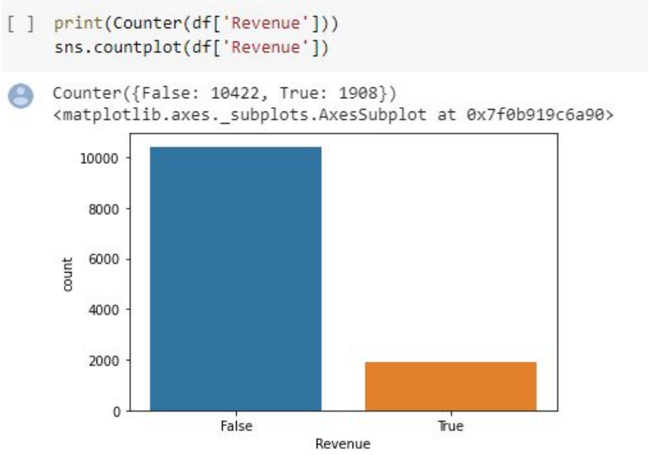
Fig 11: Statistics of numerical attributes

The above figure depicts the statistics such as mean , quartiles etc of the numerical attributes which help us analyze the numerical spread , minimum and maximum values of each of the attributes.

7.3. Distribution of classes

The class label of the dataset is 'Revenue'. As we can see from the histogram given, there are more negative classes as compared to positive classes which gives rise to class imbalance problems.

Fig 12: Barplot of class 'Revenue'



```
[ ] print(Counter(df['Revenue']))
    sns.countplot(df['Revenue'])
```

```
Counter({False: 10422, True: 1908})
<matplotlib.axes._subplots.AxesSubplot at 0x7f0b919c6a90>
```

The bar plot depicts that out of the 12330 sessions by different users, 84.5% (10,422) were negative class samples indicating revenue as ‘False’ and that the visitor did not end buying, and the rest (1908) were positive class samples indicating the revenue as ‘True’ and that the visitor ended buying the product.

7.4. Distribution of ‘Revenue’ over every ‘Month’ (Before one hot encoding)

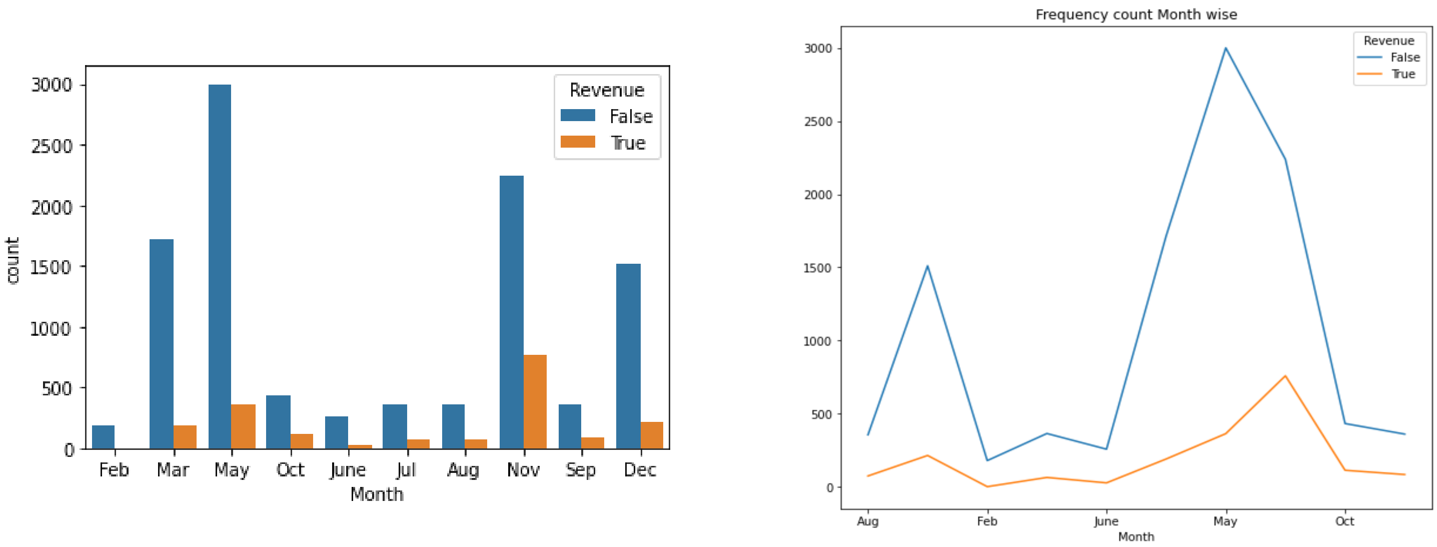
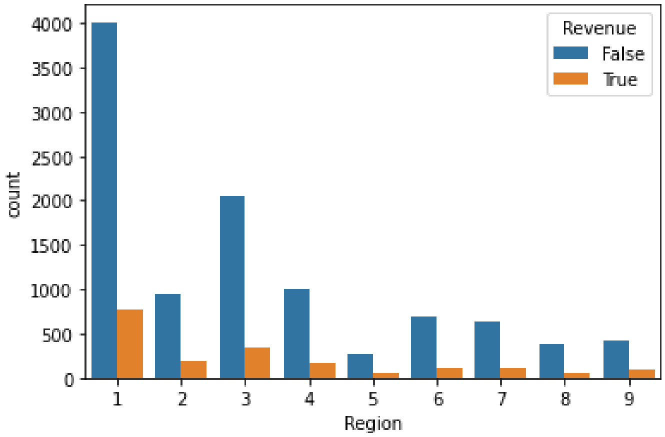


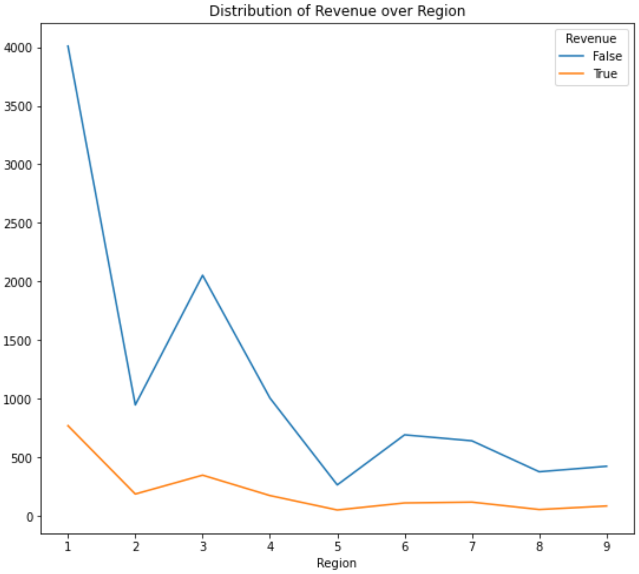
Fig 13: Countplot and Crosstab graph of ‘Month’

The above bar plot and graph depict how the revenue(visit resulting in a transaction or not) changes over months .It is observed from the plots that maximum transactions were made in the month of November possibly because of the season of advent and May had the highest traffic.

7.5. Distribution of ‘Revenue’ over ‘Region’

Fig 14: Countplot and Crosstab graph of ‘Region’





The above bar plot and graph depict how the revenue(visit resulting in a transaction or not) varies in different regions . The most visits to the pages and the most transactions have started in region 1 and the least have been initiated from region 5.

7.6. Distribution of 'Revenue' over 'Visitor Type' (Before one hot encoding)

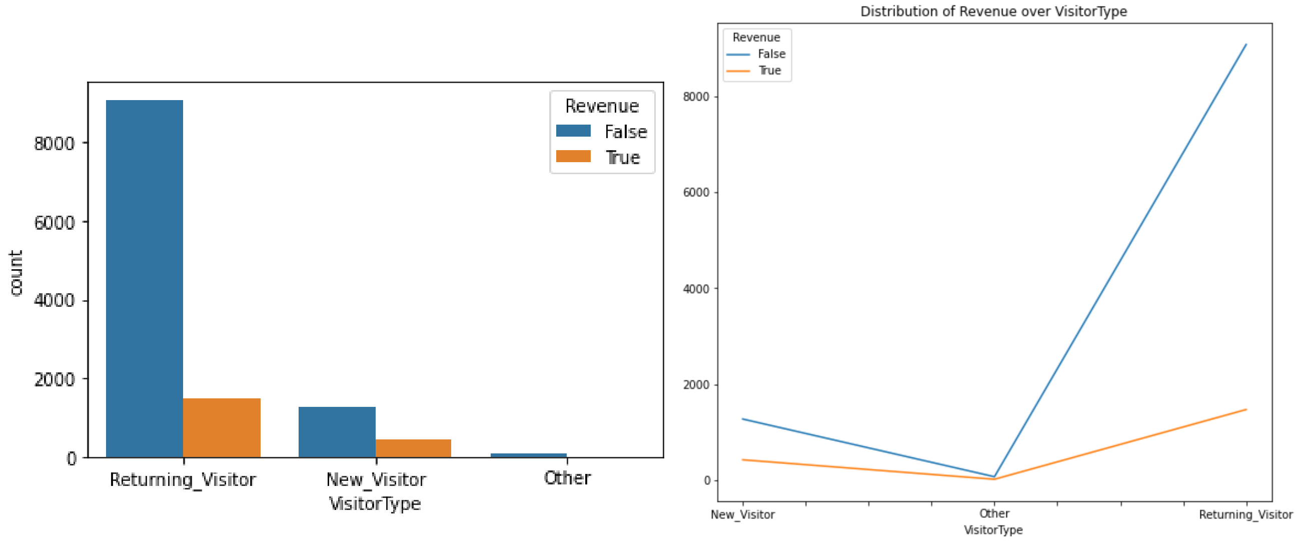
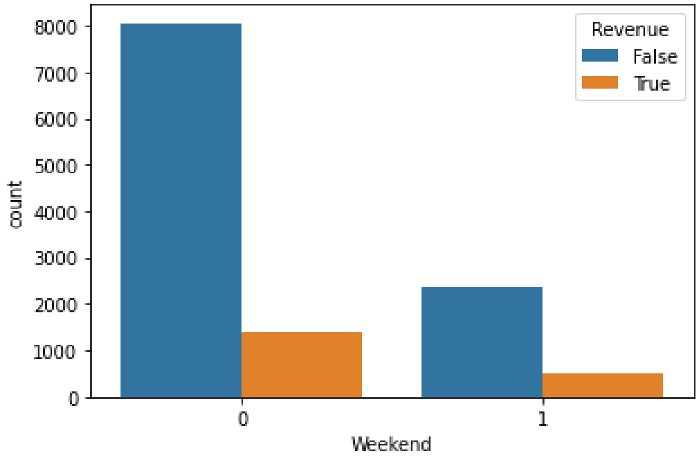


Fig 15: Countplot and Crosstab graph of 'VisitorType'

The above bar plot and graph depict that Returning visitors are more likely to make successful transactions and also visit the pages.

7. 7 Distribution of 'Revenue' over 'Weekend'



The bar plot indicates that more number of visits and transactions have taken place on weekdays.

Fig 16: Countplot of 'Weekend'

8.Data Analytical methods

For our dataset, Online Shoppers purchasing intention, we have applied the following classification algorithms.

- Logistic Regression
- K - Nearest Neighbors
- Naive Bayes
- Random Forest

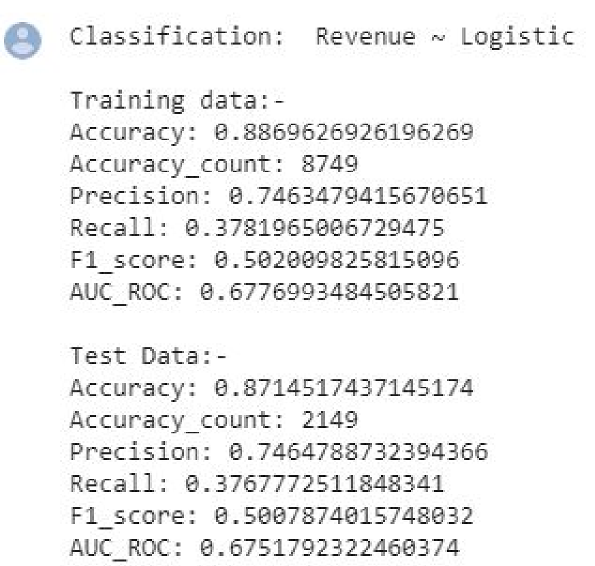
We used 25 attributes for all our classification models, we pre-processed our original dataset as mentioned in Data pre-processing step,

- Converted ‘Month’ and ‘VisitorType’ to one hot encode value.
- Converted (‘Administrative’, ‘Administrative_Duration’) to ‘AvgAdministrative’
- Converted (‘Informational’, ‘Informational_Duration’) to ‘AvgInformational’
- Converted (‘ProductRelated’, ‘ProductRelated_Duration’) to ‘AvgProductRelated’

8.1 Logistic Regression

It basically predicts the probability of occurrence of an event by fitting data to a *logit function*. Hence it is also known as logistic regression. The values obtained would always lie within 0 and 1 since it predicts the probability.

1. Split the dataset into training and testing in 8:2 ratio
2. For the features variables exclude the class ‘Revenue’ and initialize the label variable with class value ‘Revenue’
3. Train the model using **LogisticRegression(solver='liblinear',random_state=12)**
4. Internally it constructs a curve of separation until there is a proper fit of distribution where any data point falling on one side of the curve belongs to ‘False’ class and data point belonging to the other side is classified as ‘True’
5. Using the model parameters obtained, predict the values for test dataset and as well as the train dataset
6. Summarize the results
7. Construct the ROC curve



```
Classification:  Revenue ~ Logistic

Training data:-
Accuracy: 0.8869626926196269
Accuracy_count: 8749
Precision: 0.7463479415670651
Recall: 0.3781965006729475
F1_score: 0.502009825815096
AUC_ROC: 0.6776993484505821

Test Data:-
Accuracy: 0.8714517437145174
Accuracy_count: 2149
Precision: 0.7464788732394366
Recall: 0.3767772511848341
F1_score: 0.5007874015748032
AUC_ROC: 0.6751792322460374
```

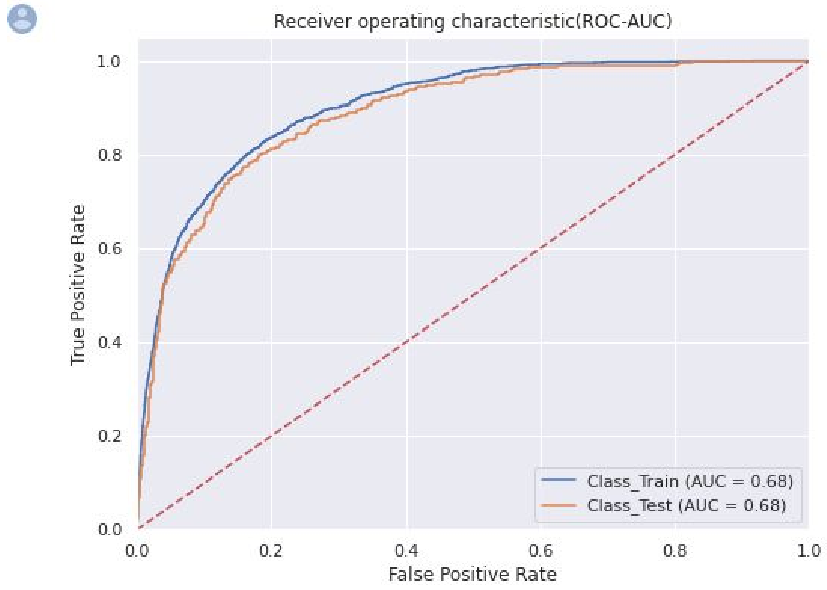


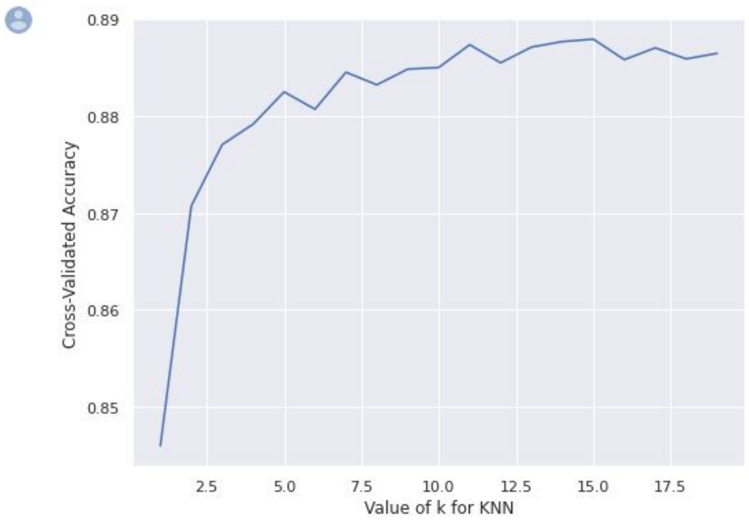
Fig 17: Summarized result and ROC graph of Logistics Regression

8.2. K-Nearest Neighbors:

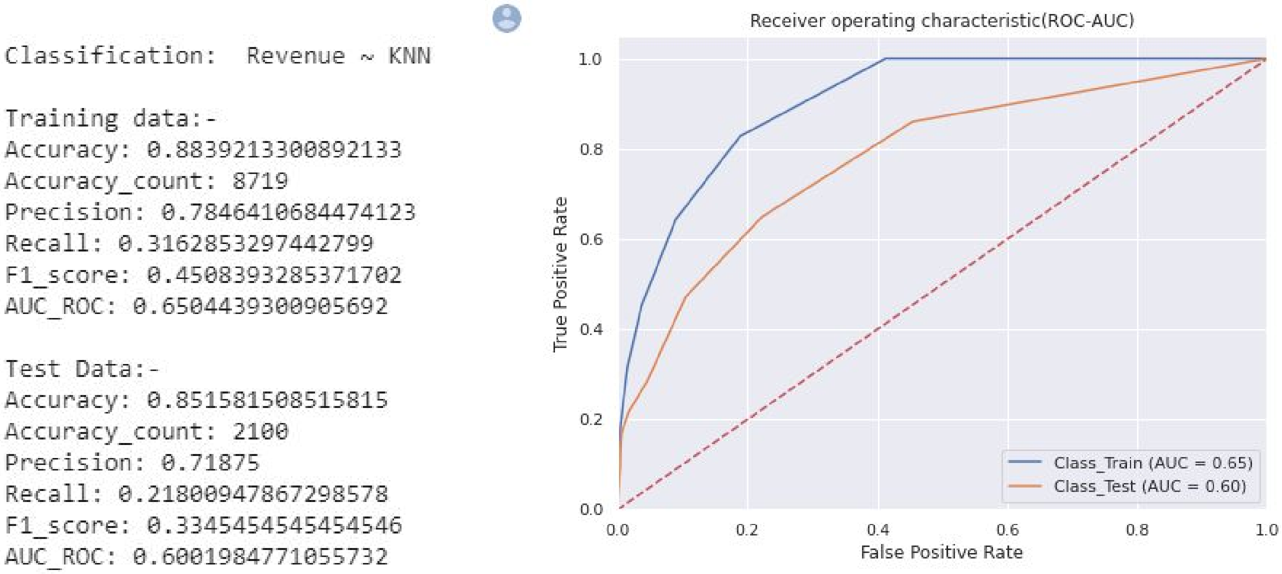
The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

- Before applying the algorithm on the train and test dataset, we found the best value of k.
- Using cross-validation, we ran the KNN algorithm for k values in the range(1,20) and found out the best value of k for which the algorithm works fine and we found out the value to be k=9. We consider the value of k at the elbow.

Fig 18: Best value of K in K-NN



1. Now, having found out the value of k, we apply **KNeighborsClassifier(n_neighbors=n_neighbors)** where n_neighbors = 9.
2. KNN algorithm finds k nearest neighbors for the incoming data point and allocates the class of majority of k neighborhoods.
3. Predict the class of test and train dataset
4. Summarize the accuracy, precision, recall, f1-score
5. Plot the ROC curve.



```
Classification:  Revenue ~ KNN

Training data:-
Accuracy: 0.8839213300892133
Accuracy_count: 8719
Precision: 0.7846410684474123
Recall: 0.3162853297442799
F1_score: 0.4508393285371702
AUC_ROC: 0.6504439300905692

Test Data:-
Accuracy: 0.851581508515815
Accuracy_count: 2100
Precision: 0.71875
Recall: 0.21800947867298578
F1_score: 0.3345454545454546
AUC_ROC: 0.6001984771055732
```

Fig 19: Summarized result and ROC graph of KNN

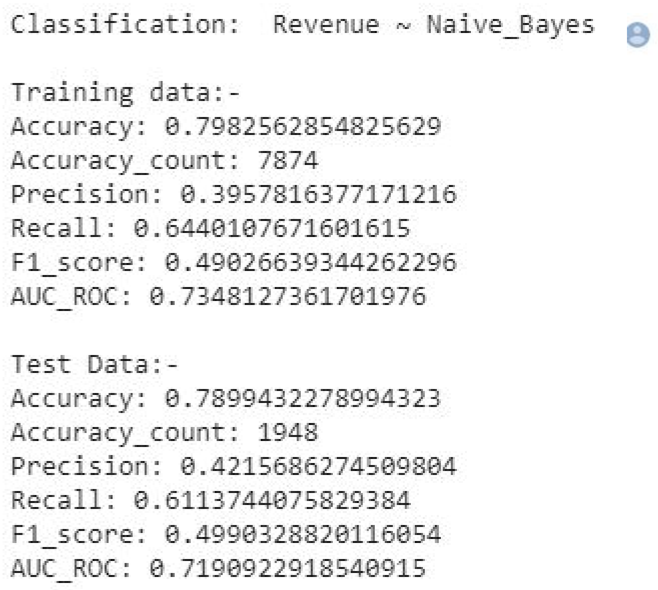
8.3. Naive Bayes:

A Naive Bayes classifier is a probabilistic machine learning model that is used for classification tasks. The basis of the classifier is based on the Bayes theorem. Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). The expression for Posterior Probability is as follows.

$$\underbrace{P(c\mid x)}_{\text{Posterior Probability}}=\frac{\overbrace{P(x\mid c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c\mid \mathrm{X}) = P(x_1\mid c)\times P(x_2\mid c)\times\cdots\times P(x_n\mid c)\times P(c)$$

1. **GaussianNB(priors=priors)** is used to model the Naive Bayes classifier
2. Split the data into train and test set
3. Based on the above equation given, the model predicts the class for every test and train tuple. We have 25 attributes, so P(c|x) will be found by multiplying the likelihood of every feature for that particular class c.
4. Summarize the result by finding the accuracy, recall, precision and F1 score and plot the ROC curve.



```
Classification:  Revenue ~ Naive_Bayes

Training data:-
Accuracy: 0.7982562854825629
Accuracy_count: 7874
Precision: 0.3957816377171216
Recall: 0.6440107671601615
F1_score: 0.49026639344262296
AUC_ROC: 0.7348127361701976

Test Data:-
Accuracy: 0.7899432278994323
Accuracy_count: 1948
Precision: 0.4215686274509804
Recall: 0.6113744075829384
F1_score: 0.4990328820116054
AUC_ROC: 0.7190922918540915
```

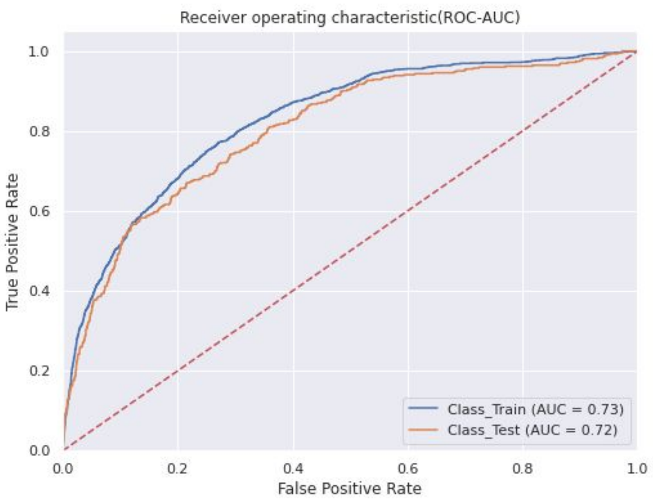


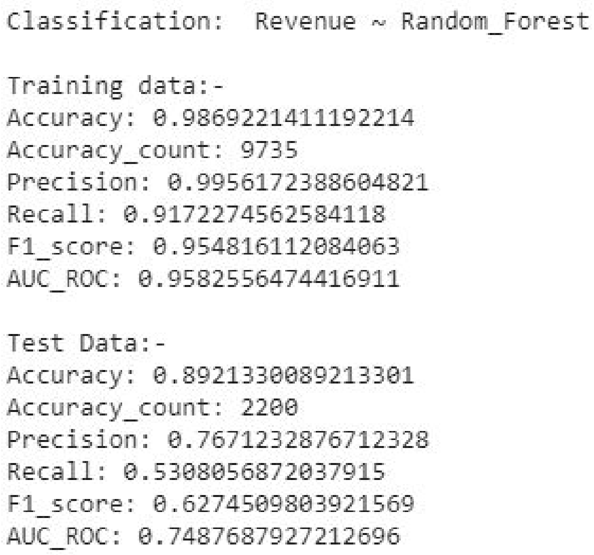
Fig 20: Summarized result and ROC graph of Naive Bayes

8.4. Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

1. Split the data into train and test set
2. Use **RandomForestClassifier(n_estimators= 50, max_depth = 15, random_state=12)** to construct 50 decision trees with a maximum depth of 12.
3. From the 50 decision trees, traverse and predict the class of every tuple. The mode of the output of 50 decision tree is considered to be the class of the tuple
4. Do Step 3 for both test and train dataset.

5. Summarize the results and plot the ROC curve.



```
Classification:  Revenue ~ Random_Forest

Training data:-
Accuracy: 0.9869221411192214
Accuracy_count: 9735
Precision: 0.9956172388604821
Recall: 0.9172274562584118
F1_score: 0.954816112084063
AUC_ROC: 0.9582556474416911

Test Data:-
Accuracy: 0.8921330089213301
Accuracy_count: 2200
Precision: 0.7671232876712328
Recall: 0.5308056872037915
F1_score: 0.6274509803921569
AUC_ROC: 0.7487687927212696
```

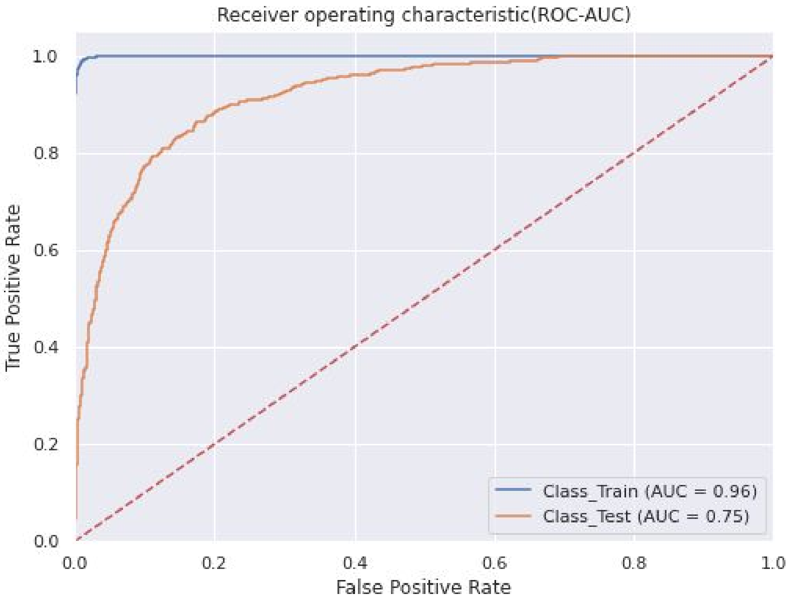


Fig 20: Summarized result and ROC graph of Random Forest

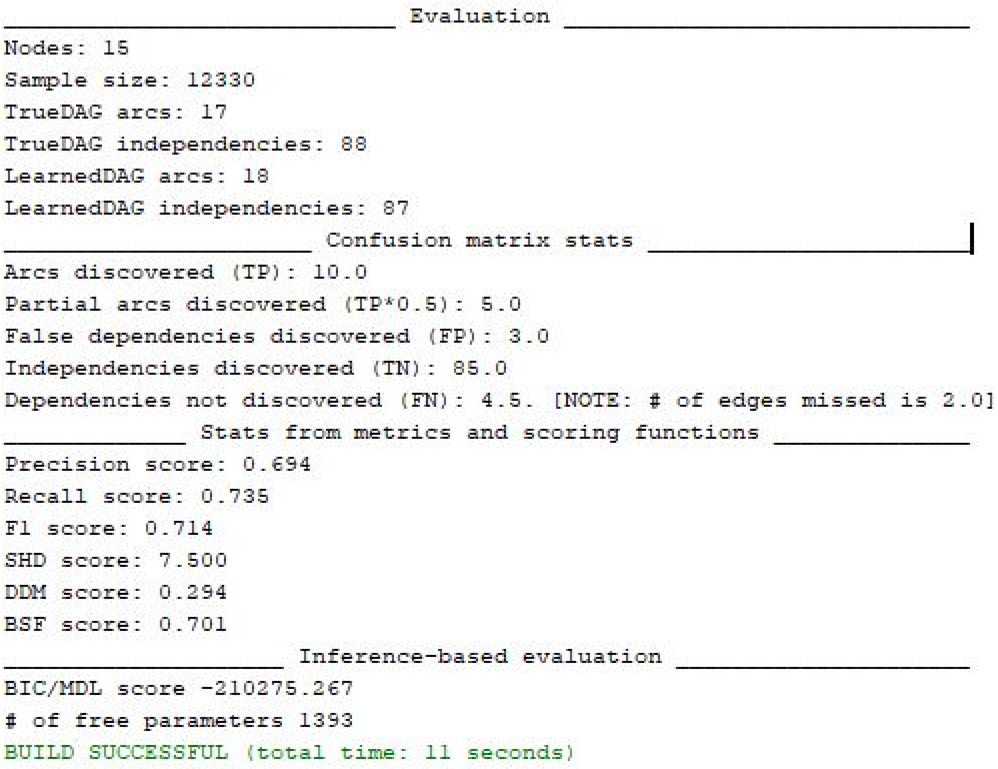
SECTION – II

STRUCTURE LEARNING USING BAYESIAN NETWORKS

Discretization

We converted 7 numerical continuous attributes to discrete values. In total 9 bins were considered according to [3, 7, 8] where four intervals of size σ to the right of σ+ μ/2 are converted to discrete levels from 1 to 4, and the four intervals of size σ to the left of σ - μ/2 are mapped to discrete levels from - 1 to - 4. While the feature values between σ - μ/2 and σ + μ/2 are converted to 0, very large positive or negative feature values are truncated and discretized to ± 4 appropriately. The attributes converted to discrete values are 'AvgAdministrative', 'BounceRates','AvgInformational', 'ExitRates','AvgProductRelated', 'PageValues','TrafficType' . The above listed attributes are converted to discrete values as the SaiyanH hybrid Bayesian Network(BN) structure learning algorithm assumes a unique discrete state for each unique variable value and becomes computationally extensive if there are continuous values.

Evaluation Metrics of the generated DAG



```
____________________________ Evaluation _____________________________
Nodes: 15
Sample size: 12330
TrueDAG arcs: 17
TrueDAG independencies: 88
LearnedDAG arcs: 18
LearnedDAG independencies: 87
______________________ Confusion matrix stats _______________________
Arcs discovered (TP): 10.0
Partial arcs discovered (TP*0.5): 5.0
False dependencies discovered (FP): 3.0
Independencies discovered (TN): 85.0
Dependencies not discovered (FN): 4.5. [NOTE: # of edges missed is 2.0]
_____________ Stats from metrics and scoring functions ______________
Precision score: 0.694
Recall score: 0.735
F1 score: 0.714
SHD score: 7.500
DDM score: 0.294
BSF score: 0.701
____________________ Inference-based evaluation _____________________
BIC/MDL score -210275.267
# of free parameters 1393
BUILD SUCCESSFUL (total time: 11 seconds)
```

Answer 1:

The steps followed to produce knowledge-based causal graph

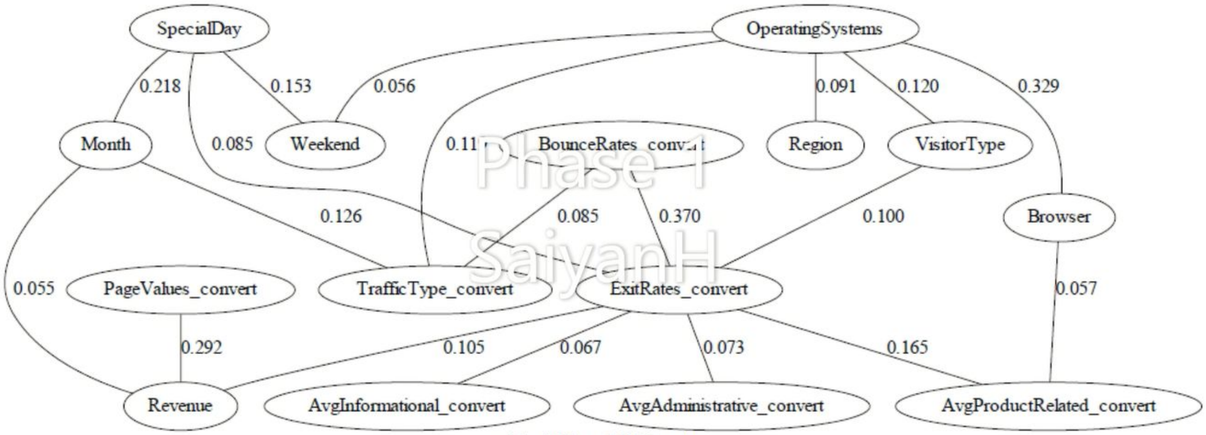
1. Constructed a decision tree using RandomTree in Weka environment to observe the dependencies of attributes

2. Based on domain knowledge and knowledge acquired from Weka Decision tree, we constructed the causal graph.

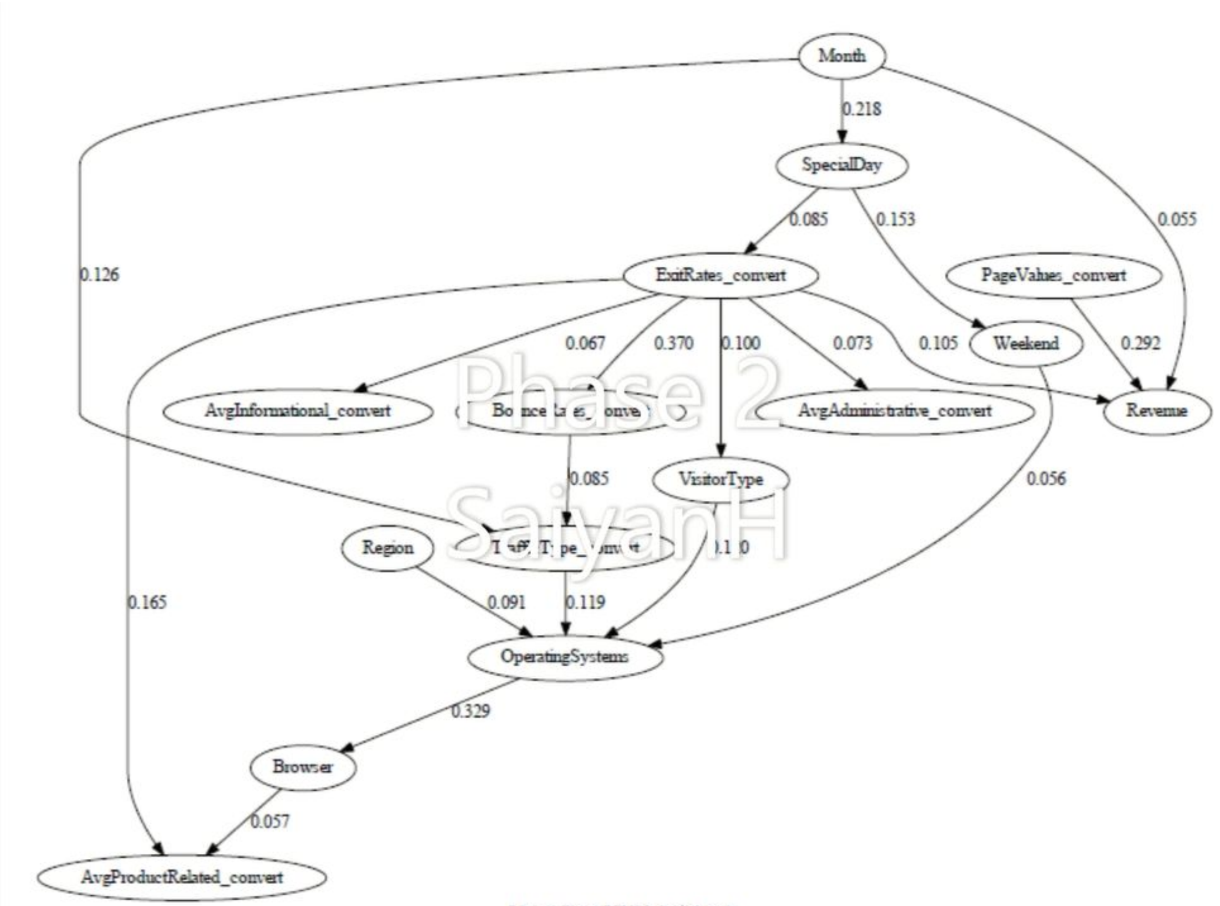
- 'Month' is the most decisive attribute
- 'SpecialDay' can be predicted from 'Month'
- 'Weekend' can be derived from 'SpecialDay'
- Based on 'Month' we can derive at the means of communication ('TrafficType_convert')
- 'TrafficType_convert' can help us to predict whether the user is returning/new user
- 'SpecialDay' probability helps in predicting 'ExitRate_convert' of the user
- 'TrafficType_convert' influences the 'OperatingSystems' (eg. SMS mode of communication makes the user use Android OS)
- 'OperatingSystems' helps in deriving the 'Browser' type
- Based on 'Region' we can predict the type of 'OperatingSystems' (eg. Research oriented areas involve Linux OS)
- 'VisitorType' influences 'ExitRates_convert'
- Avg number of pages(PageValues_convert) visited by the visitor can determine 'Revenue'
- Knowing the 'ExitRates_convert' can to some extent help in deciding the 'BounceRates_convert' and influences 'AvgAdministrative_convert', 'AvgInformational_convert', 'AvgProductRelated_convert'.
- The number of product related pages visited by the visitor('AvgProductRelated_convert') influences whether the user will confirm the product to checkout or not ('Revenue')

The knowledge elicited was handled by a single member of the group and so there was no disagreement.

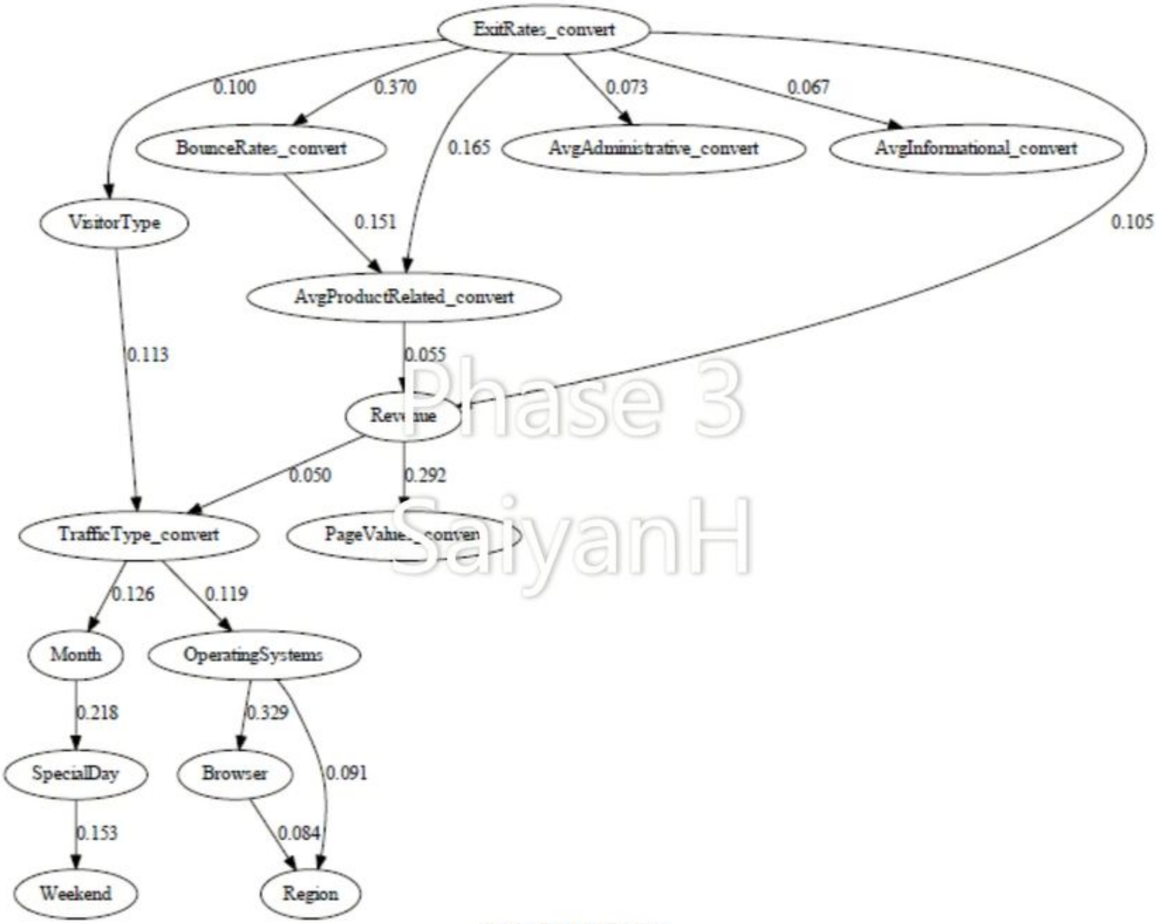
Answer 2:



SaiyanH_Phase_1 EMSG graph.
Total edges: 19



SaiyanH_Phase_2 EMSG directed graph.
Total arcs: 19



SaiyanH_Phase_3 graph (final).
Total arcs: 18

Phase 1: Initially we start with a complete graph(ie. edges between all pairs of nodes), then eventually we remove the edges between two nodes A and B if and only if A and B share

neighbour C where

$MMD(A \leftrightarrow C) > MMD(A \leftrightarrow B) < MMD(B \leftrightarrow C)$ [Here MMD(Marginal Discrepancy) score represents the discrepancy in marginal probabilities between prior and posterior distributions]

The order in which the edges are assessed for removal is from lowest to highest MMD score and thus we form EMSG(Extended Maximum Spanning Graph).

Phase 2: In this phase we determine the orientation of edges of EMSG from phase 1, the algorithm performs conditional independence tests across all pairs of nodes conditional on the remaining nodes and classifies each triple into conditional dependence, independence or insignificance. The order in which the edges are assessed for orientation is determined by node ordering, where nodes are ordered by the total MMD score they share with their neighbours.[In our dataset it starts with 'ExitRates_convert']. The result of this phase is a directed graph.

Answer 3:

Phase 2: From EMSG we construct a graph which has oriented edges based on a conditional independence test in conjunction with BIC and do-calculus. Phase 2 helps the algorithm in Phase 3 to reduce the search space. It takes best possible efforts to orient all edges so as to reduce complexity in Phase 3

Phase 3: This phase explores all the neighboring graphs based on BIC criterion and phase 2 graph is the start graph for phase 3. The search involves Hill-climbing algorithm where if the neighboring graph H (edge reversed/removed/added) has BIC greater than G is replaced with H. The end point is until there is no neighborhood increase in BIC.

If both graphs are identical, it simply means that there was no better neighborhood graph having a better BIC score

---

Answer 4:

Number of scores in:

- marginalDep.csv = 105
- conditionalInsignificance.csv = 859
- conditionalIndep.csv = 54
- conditionalDep.csv = 452

marginalDep.csv is the result of Phase 1 where we check the discrepancy score between every two variables.

Total number of variables/node/attribute = 15

Number of scores = (15*14)/2 = 105

I. **15 represents the total number of attributes**
II. **14 - one attribute less so that the same attribute that was used in (i) is not used**

III. (A,B) is same as (B,A) so we need to remove extra pair and hence divide by 2

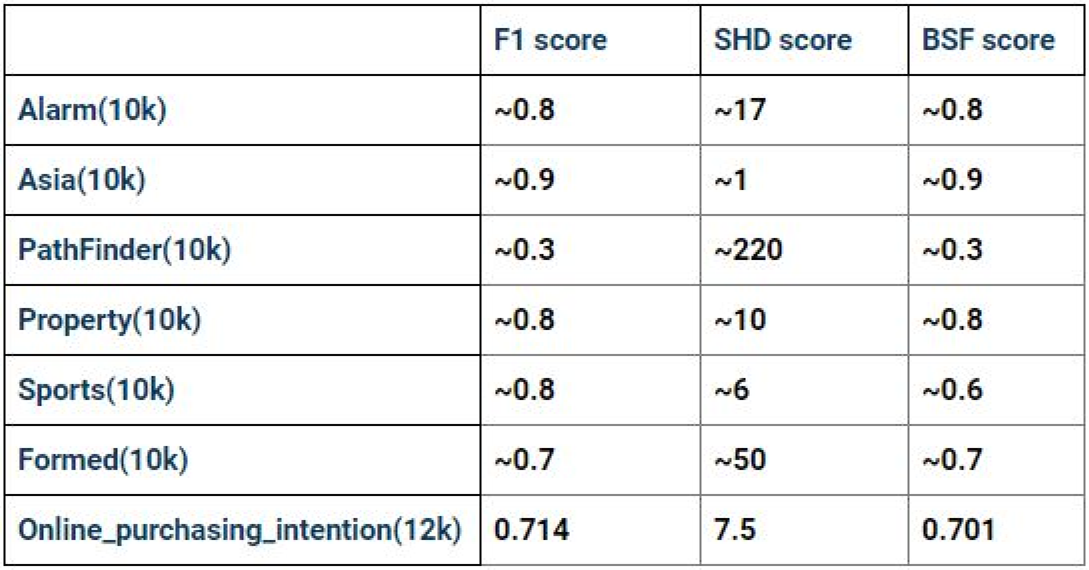
conditionalInsignificance.csv, conditionalIndep.csv, conditionalDep.csv are the results of Phase2.

(15*14*13)/2

I. 15 represents the total number of attributes
II. 14 - one attribute less so that the same attribute that was used in (i) is not used
III. 13 - two attributes lesser so that the attributes used in (i) and (ii) are not repeated.
IV. [(A,B)|C](A,B) is same as (B,A) so we need to remove extra pair and hence divide by 2

Number of scores totally [conditionalDep+conditionalInsignificance+ConditionalIndep] = (15*14*13)/2 = 1365. Based on the criterion used in Phase 2[9], conditionalDep has 452 scores and conditionalIndep has 54 scores, the rest falls in conditionalInsignificance (1365-452-54=859)

Answer 5:



	F1 score	SHD score	BSF score
Alarm(10k)	~0.8	~17	~0.8
Asia(10k)	~0.9	~1	~0.9
PathFinder(10k)	~0.3	~220	~0.3
Property(10k)	~0.8	~10	~0.8
Sports(10k)	~0.8	~6	~0.6
Formed(10k)	~0.7	~50	~0.7
Online_purchasing_intention(12k)	0.714	7.5	0.701

The results obtained are in par with other datasets,

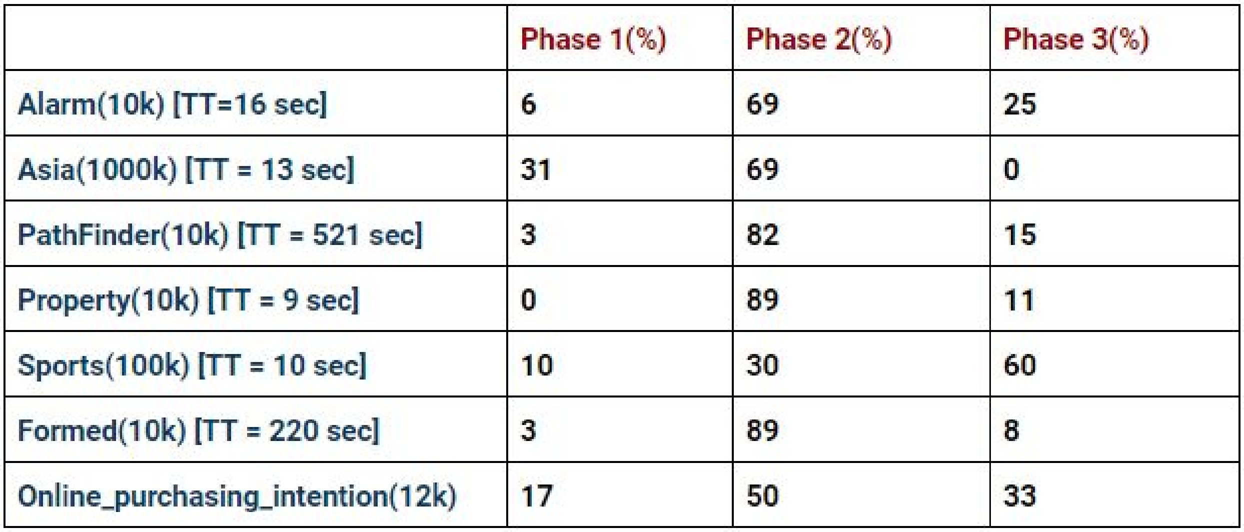
- F1 score is as expected, the higher the F1 score the more accurate the graph is to the ground truth, while drawing the knowledge based causal graph, there is a possibility of missing few arcs since humans can’t be accurate and the metrics used in algorithms[9] are more reliable.

- SHD is the number of steps to transform a discovered graph into a ground truth graph. Expected to have a higher value of SHD, since there is a possibility of removing/reversing/adding arcs to the transformed graph and as the number of steps to adopt might be more for 15 nodes.
- BSF is as expected, since we had the domain knowledge and considered other factors like decisive attribute and it can’t be more accurate than this because there is always a possibility of missing out in a complicated structure.

Answer 6:

Online_purchasing_intention : Structure learning elapsed time: 12 seconds total (Phase 1 = 2 secs, Phase 2 = 6 secs).

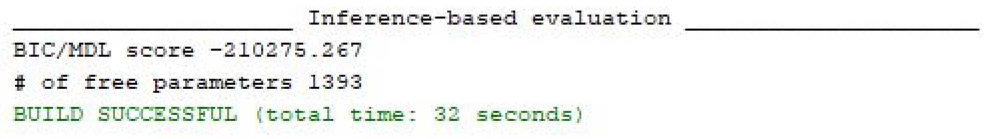
TT represents TotalTime



	Phase 1(%)	Phase 2(%)	Phase 3(%)
Alarm(10k) [TT=16 sec]	6	69	25
Asia(1000k) [TT = 13 sec]	31	69	0
PathFinder(10k) [TT = 521 sec]	3	82	15
Property(10k) [TT = 9 sec]	0	89	11
Sports(100k) [TT = 10 sec]	10	30	60
Formed(10k) [TT = 220 sec]	3	89	8
Online_purchasing_intention(12k)	17	50	33

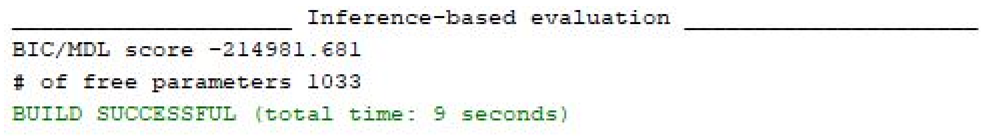
From Table 2, given in the quoted paper[9], we observe that as the dataset size increases, the evaluation time increases as well. For our dataset the elapsed time is as expected since the size of the data is less and usually phase 2 requires most of the processing time as the output says and the complexity in determining the graph is not so complicated because we have 15 nodes/attributes to evaluate which is comparatively simpler.

STEP 4 using Predicted graph



```
___________________ Inference-based evaluation ____________________
BIC/MDL score -210275.267
# of free parameters 1393
BUILD SUCCESSFUL (total time: 32 seconds)
```

STEP 3 using Truth Graph



```
___________________ Inference-based evaluation ____________________
BIC/MDL score -214981.681
# of free parameters 1033
BUILD SUCCESSFUL (total time: 9 seconds)
```

Answer 7:

BIC = -214981.681 is obtained from the truth graph (Step 3)

BIC = -210275.267 is obtained from the predicted graph(Step 4)

When the predicted and actual graph are the same, the BIC is lesser than expected because in the Hill-Climbing algorithm there is a possibility of missing out the global optima and getting stuck in the local optima and this is the case when BIC is calculated from a truth graph. And hence the two values are as we expected.

Answer 8:

No of free parameters: 1033 (Step 3)

No of free parameters: 1393 (Step 4)

For each variable V_i in G with parents P_i and S states

$$F = \sum_{i=1}^{V} \left((V_i^S - 1) P_i^S \right)$$

The number of parameters in both steps are as expected, it is quite obvious to have more parameters in the predicted graph since the number of states and parent nodes are higher than the actual ground truth graph.

10.Challenges faced

The Project aims mentioned in Part 2 were successfully implemented. Apart from that there were few challenges which we ruled out by extensive analysis.

- The Dataset faces the problem of class imbalance. We used various metrics like F1 score, precision, AUC_ROC to overcome the problem.
- We were in need to discretize the variables for the purpose of structure learning, so deciding the number of bins was a major issue, so we did research oriented analysis [3,7,8] and concluded the number of bins to be 9.
- For deciding the knowledge based causal graph, we read about the domain and decision tree to draw the ground truth graph accordingly.

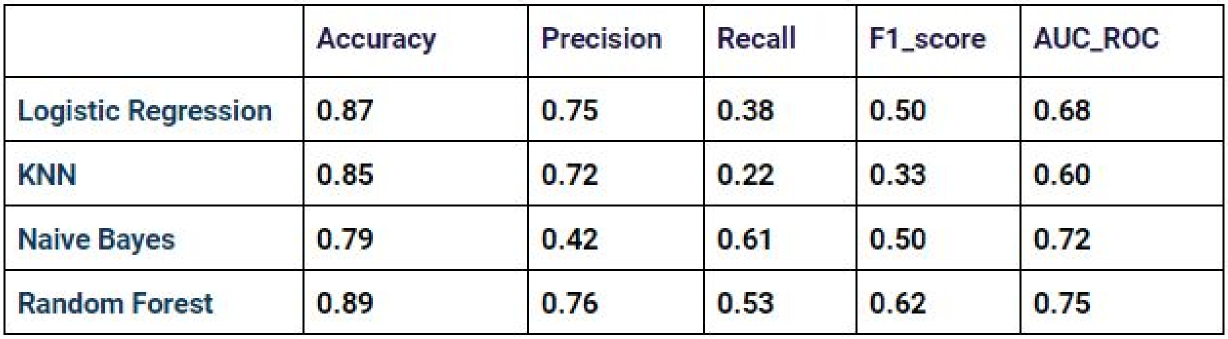
11. Business Applications

- To know the buying behaviour of visitors can have many implications in e-commerce helping the business to establish target ads.
- Based on the visitor clicking analysis, the business market can make dynamic offers and recommendations in real time
- Based on the user's region, the previously viewed web pages and few other important aspects markets can target customers.

12.Conclusion

In this project , we constructed different models to analyze the purchasing intentions of online shopping customers. We used the dataset “Online Shoppers Purchasing Intention ” from the UCI Machine Learning Repository for the analysis. We have also used the dataset to generate DAG(Directed acyclic Graphs) with the help of “SaiyanH hybrid Bayesian Network (BN) structure learning algorithm”.

In the first part of the report we have evaluated the performance of various models on the dataset and recorded the accuracy. The results are summarized in the table below:



	Accuracy	Precision	Recall	F1_score	AUC_ROC
Logistic Regression	0.87	0.75	0.38	0.50	0.68
KNN	0.85	0.72	0.22	0.33	0.60
Naive Bayes	0.79	0.42	0.61	0.50	0.72
Random Forest	0.89	0.76	0.53	0.62	0.75

Random Forest classifier performs the best compared to other classifiers because it has the highest accuracy and AUC_ROC value nearing 1. While considering the best outcome, we not only should look at accuracy because we have class imbalance problems here with this dataset and considering f1 score and AUC_ROC values will help with authenticity of getting the best possible outcome. Random forest performs the best here because it iterated over a set of 50 Random trees and got the most trustable output for all tuples.

In part II we were successfully able to visualize the causal graph of the dataset.

References

[1] Suchacka, Grażyna, and Grzegorz Chodak. "Using association rules to assess purchase probability in online stores." *Information Systems and e-Business Management* 15.3 (2017): 751-780.

[2] Moe, W. W. (2003). Buying, searching, or browsing: Differentiating between online shoppers using in-store navigational clickstream. *Journal of consumer psychology*, *13*(1-2), 29-39.

[3] Sakar, C. Okan, et al. "Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks." *Neural Computing and Applications* 31.10 (2019): 6893-6908.

[4] Mobasher, Bamshad, et al. "Discovery and evaluation of aggregate usage profiles for web personalization." *Data mining and knowledge discovery* 6.1 (2002): 61-82.

[5] Suchacka, Grażyna, Magdalena Skolimowska-Kulig, and Aneta Potempa. "A k-Nearest Neighbors method for classifying user sessions in e-commerce scenario." *Journal of Telecommunications and Information Technology* (2015).

[6] Budnikas, Germanas. "Computerised recommendations on e-transaction finalisation by means of machine learning." *Statistics in Transition. New Series* 16.2 (2015): 309-322.

[7] Peng, Hanchuan, Fuhui Long, and Chris Ding. "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy." *IEEE Transactions on pattern analysis and machine intelligence* 27.8 (2005): 1226-1238.

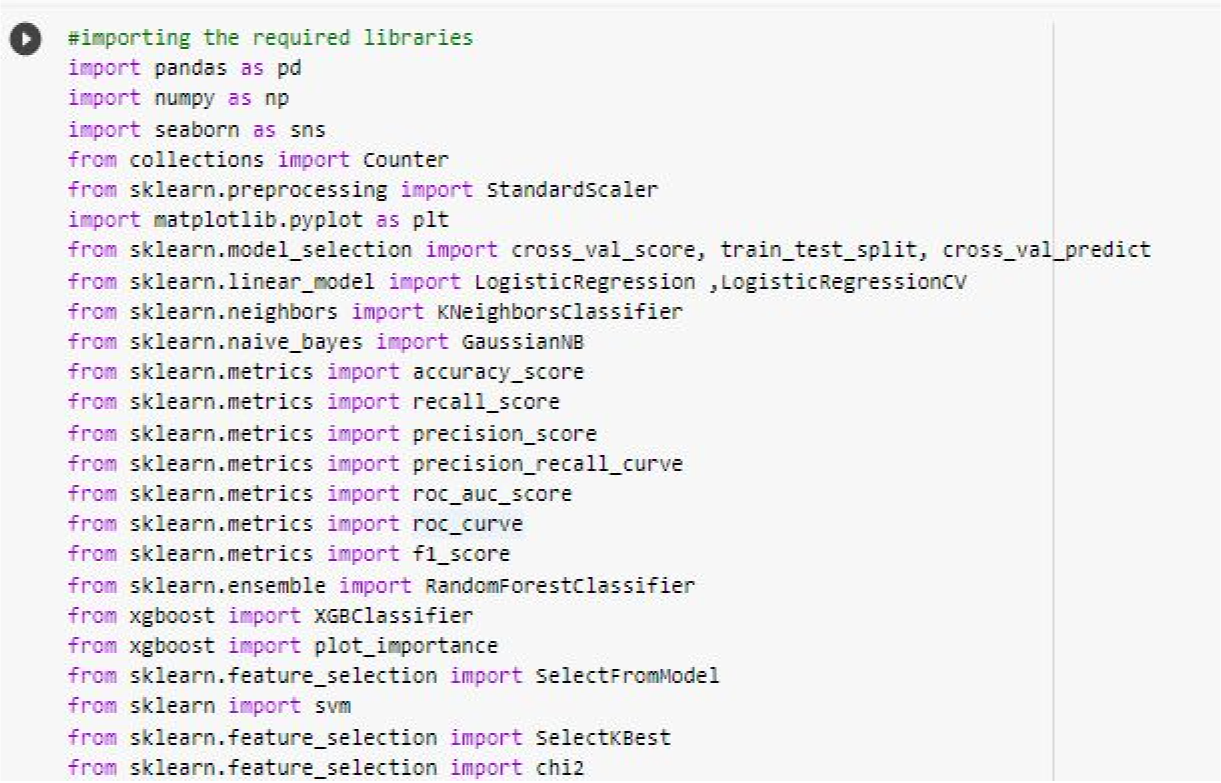
[8] Sakar, C. Okan, Olcay Kursun, and Fikret Gurgen. "A feature selection method based on kernel canonical correlation analysis and the minimum Redundancy–Maximum Relevance filter method." *Expert Systems with Applications* 39.3 (2012): 3432-3437.

[9] Constantinou, Anthony. "Learning Bayesian Networks that enable full propagation of evidence." *arXiv preprint arXiv:2004.04571* (2020).

Appendices

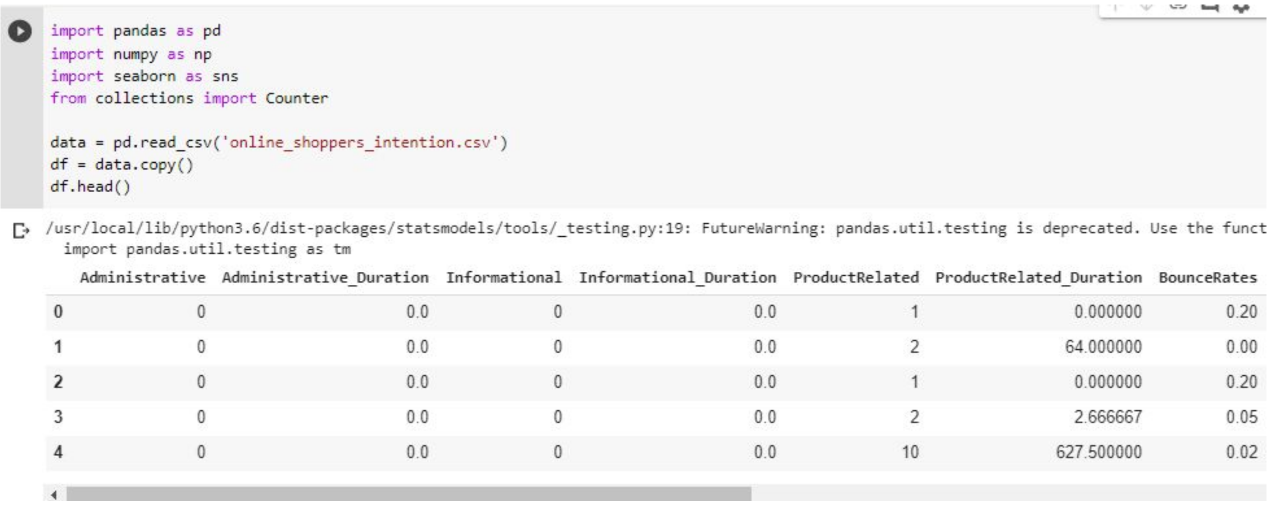
Code reference : https://github.com/caraevangeline/Data_Analytics_Project

Import the required libraries



```
#importing the required libraries
import pandas as pd
import numpy as np
import seaborn as sns
from collections import Counter
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score, train_test_split, cross_val_predict
from sklearn.linear_model import LogisticRegression ,LogisticRegressionCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import f1_score
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from xgboost import plot_importance
from sklearn.feature_selection import SelectFromModel
from sklearn import svm
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

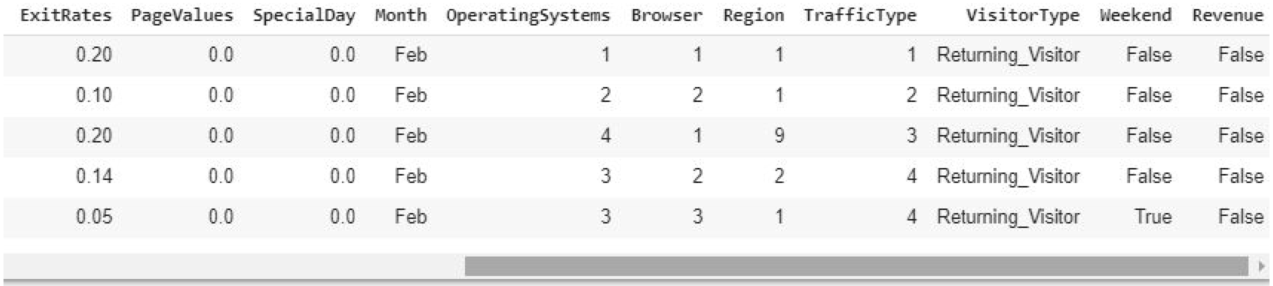
DataSet ‘Online shoppers purchasing Intention’



```
import pandas as pd
import numpy as np
import seaborn as sns
from collections import Counter

data = pd.read_csv('online_shoppers_intention.csv')
df = data.copy()
df.head()
```

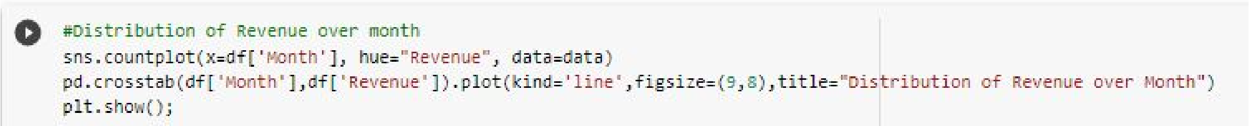
```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the funct
  import pandas.util.testing as tm
```

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates
0	0	0.0	0	0.0	1	0.000000	0.20
1	0	0.0	0	0.0	2	64.000000	0.00
2	0	0.0	0	0.0	1	0.000000	0.20
3	0	0.0	0	0.0	2	2.666667	0.05
4	0	0.0	0	0.0	10	627.500000	0.02



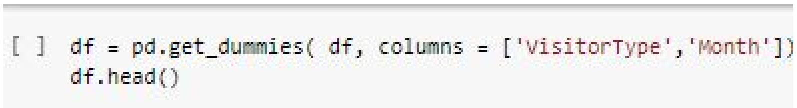
ExitRates	PageValues	SpecialDay	Month	OperatingSystems	Browser	Region	TrafficType	VisitorType	Weekend	Revenue
0.20	0.0	0.0	Feb	1	1	1	1	Returning_Visitor	False	False
0.10	0.0	0.0	Feb	2	2	1	2	Returning_Visitor	False	False
0.20	0.0	0.0	Feb	4	1	9	3	Returning_Visitor	False	False
0.14	0.0	0.0	Feb	3	2	2	4	Returning_Visitor	False	False
0.05	0.0	0.0	Feb	3	3	1	4	Returning_Visitor	True	False

Display countplot and crosstab of Month attribute wrt Revenue



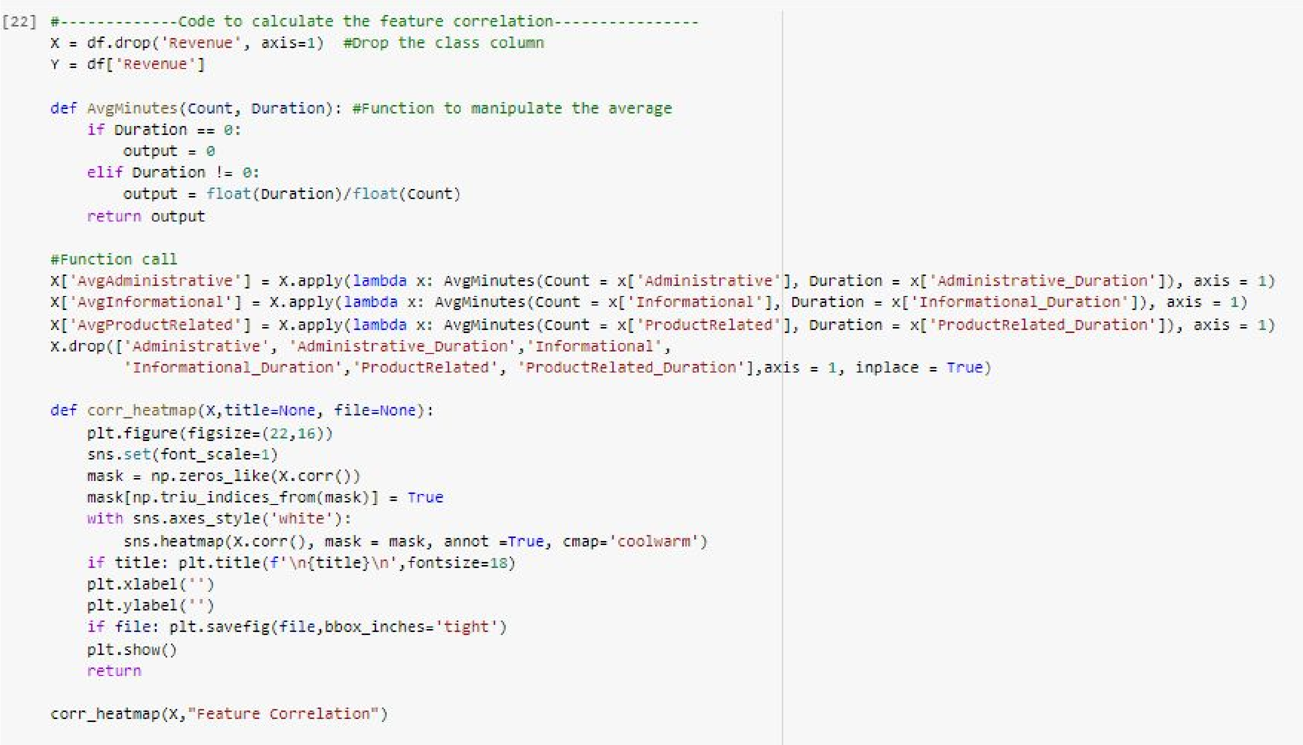
```
#Distribution of Revenue over month
sns.countplot(x=df['Month'], hue="Revenue", data=data)
pd.crosstab(df['Month'],df['Revenue']).plot(kind='line',figsize=(9,8),title="Distribution of Revenue over Month")
plt.show();
```

Convert Month and Weekend to one-hot encode



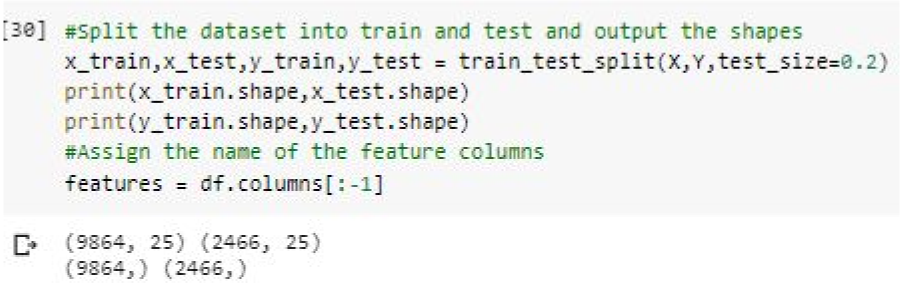
```
[ ]  df = pd.get_dummies( df, columns = ['VisitorType','Month'])
     df.head()
```

Feature Correlation



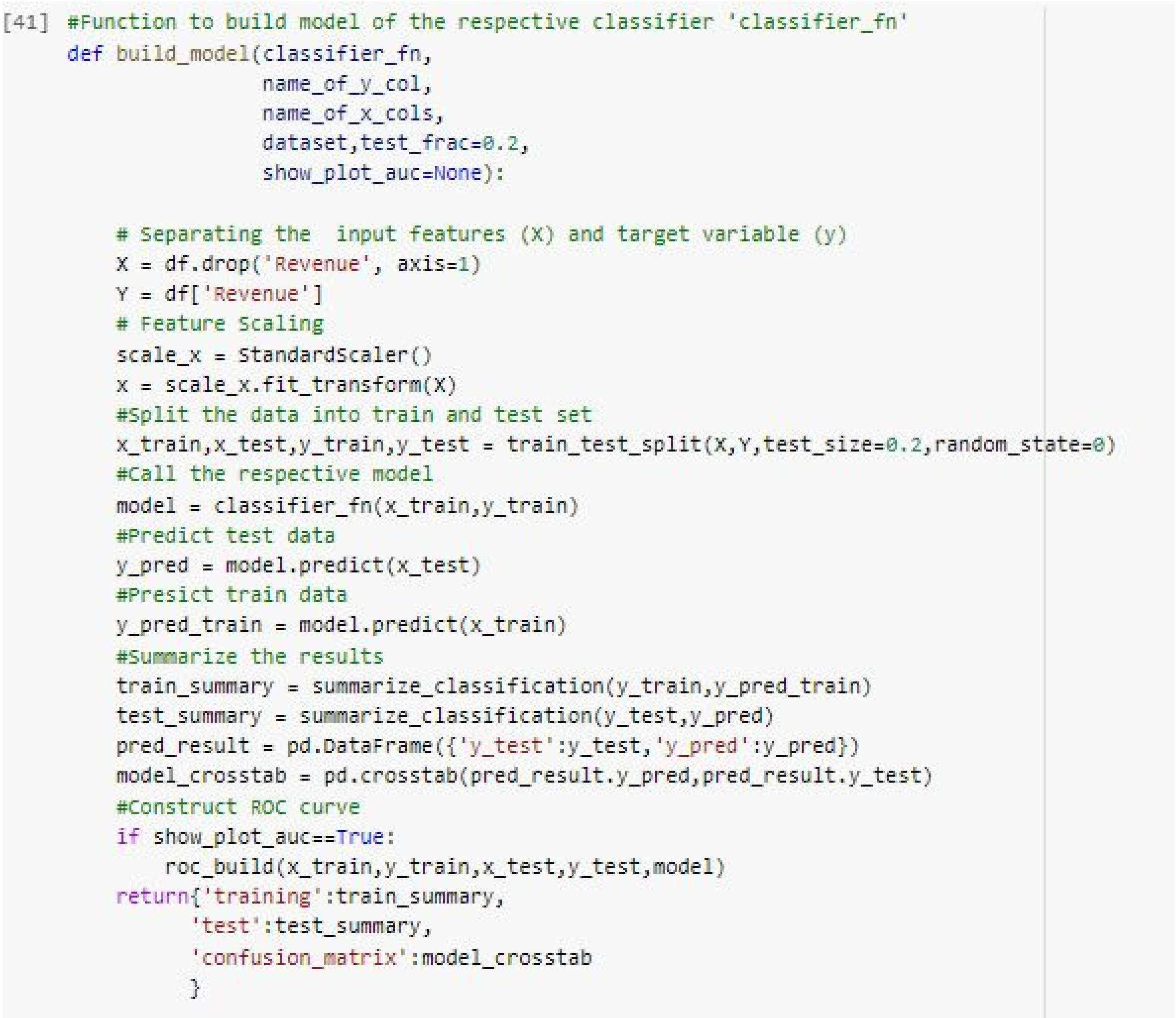
```
[22] #-------------Code to calculate the feature correlation---------------
     X = df.drop('Revenue', axis=1)  #Drop the class column
     Y = df['Revenue']

     def AvgMinutes(Count, Duration): #Function to manipulate the average
         if Duration == 0:
             output = 0
         elif Duration != 0:
             output = float(Duration)/float(Count)
         return output

     #Function call
     X['AvgAdministrative'] = X.apply(lambda x: AvgMinutes(Count = x['Administrative'], Duration = x['Administrative_Duration']), axis = 1)
     X['AvgInformational'] = X.apply(lambda x: AvgMinutes(Count = x['Informational'], Duration = x['Informational_Duration']), axis = 1)
     X['AvgProductRelated'] = X.apply(lambda x: AvgMinutes(Count = x['ProductRelated'], Duration = x['ProductRelated_Duration']), axis = 1)
     X.drop(['Administrative', 'Administrative_Duration','Informational',
             'Informational_Duration','ProductRelated', 'ProductRelated_Duration'],axis = 1, inplace = True)

     def corr_heatmap(X,title=None, file=None):
         plt.figure(figsize=(22,16))
         sns.set(font_scale=1)
         mask = np.zeros_like(X.corr())
         mask[np.triu_indices_from(mask)] = True
         with sns.axes_style('white'):
             sns.heatmap(X.corr(), mask = mask, annot =True, cmap='coolwarm')
         if title: plt.title(f'\n{title}\n',fontsize=18)
         plt.xlabel('')
         plt.ylabel('')
         if file: plt.savefig(file,bbox_inches='tight')
         plt.show()
         return

     corr_heatmap(X,"Feature Correlation")
```

Split the Dataset into train and test

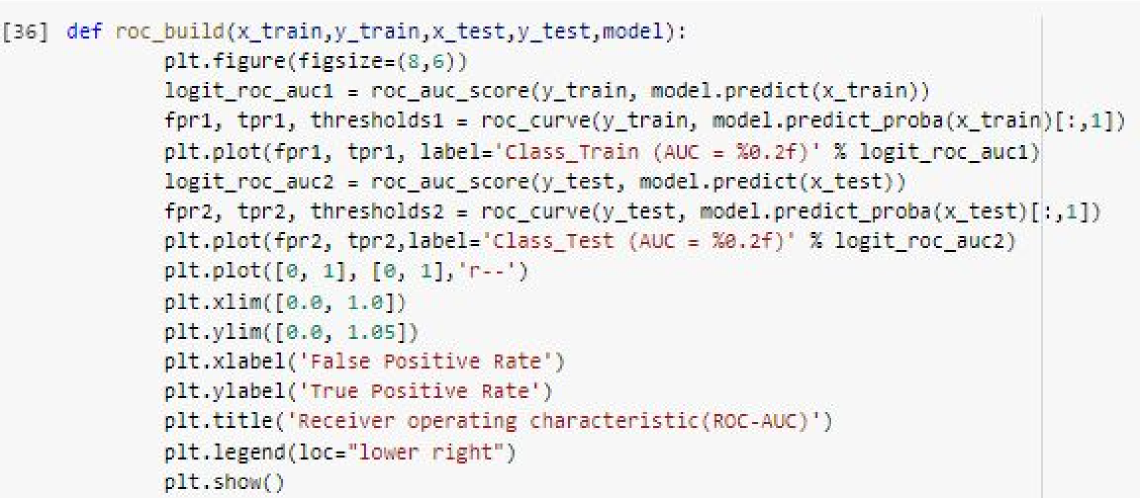


```
[30] #Split the dataset into train and test and output the shapes
     x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.2)
     print(x_train.shape,x_test.shape)
     print(y_train.shape,y_test.shape)
     #Assign the name of the feature columns
     features = df.columns[:-1]

     (9864, 25) (2466, 25)
     (9864,) (2466,)
```

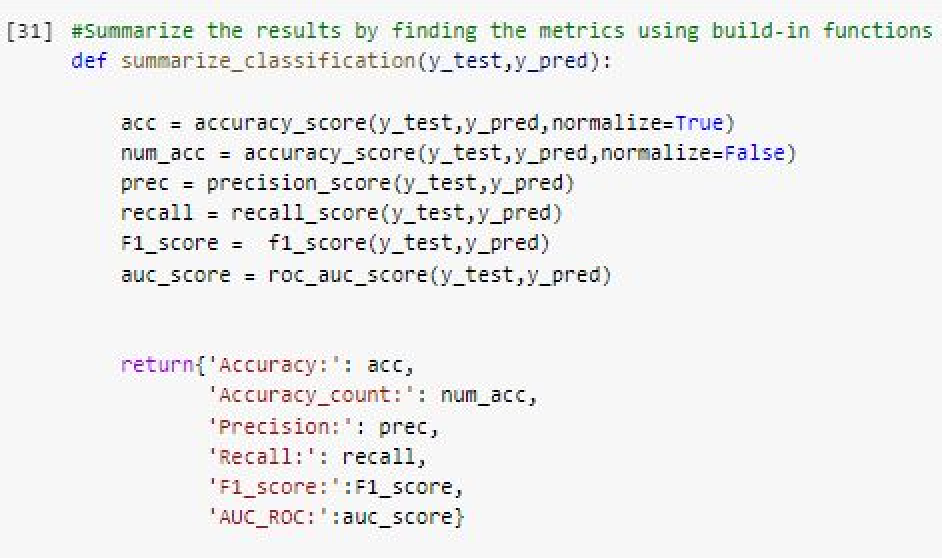
Function definitions to build model, display ROC plot and to summarize results



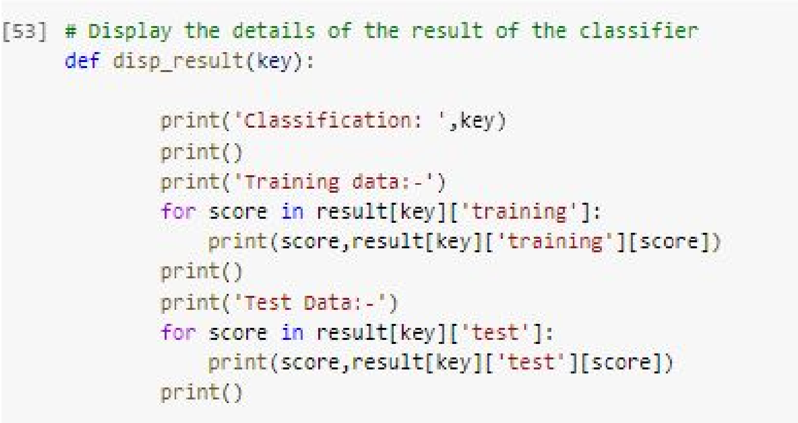
```
[41] #Function to build model of the respective classifier 'classifier_fn'
     def build_model(classifier_fn,
                     name_of_y_col,
                     name_of_x_cols,
                     dataset,test_frac=0.2,
                     show_plot_auc=None):

         # Separating the  input features (X) and target variable (y)
         X = df.drop('Revenue', axis=1)
         Y = df['Revenue']
         # Feature Scaling
         scale_x = StandardScaler()
         x = scale_x.fit_transform(X)
         #Split the data into train and test set
         x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,random_state=0)
         #Call the respective model
         model = classifier_fn(x_train,y_train)
         #Predict test data
         y_pred = model.predict(x_test)
         #Presict train data
         y_pred_train = model.predict(x_train)
         #Summarize the results
         train_summary = summarize_classification(y_train,y_pred_train)
         test_summary = summarize_classification(y_test,y_pred)
         pred_result = pd.DataFrame({'y_test':y_test,'y_pred':y_pred})
         model_crosstab = pd.crosstab(pred_result.y_pred,pred_result.y_test)
         #Construct ROC curve
         if show_plot_auc==True:
             roc_build(x_train,y_train,x_test,y_test,model)
         return{'training':train_summary,
               'test':test_summary,
               'confusion_matrix':model_crosstab
               }
```



```
[36] def roc_build(x_train,y_train,x_test,y_test,model):
             plt.figure(figsize=(8,6))
             logit_roc_auc1 = roc_auc_score(y_train, model.predict(x_train))
             fpr1, tpr1, thresholds1 = roc_curve(y_train, model.predict_proba(x_train)[:,1])
             plt.plot(fpr1, tpr1, label='Class_Train (AUC = %0.2f)' % logit_roc_auc1)
             logit_roc_auc2 = roc_auc_score(y_test, model.predict(x_test))
             fpr2, tpr2, thresholds2 = roc_curve(y_test, model.predict_proba(x_test)[:,1])
             plt.plot(fpr2, tpr2,label='Class_Test (AUC = %0.2f)' % logit_roc_auc2)
             plt.plot([0, 1], [0, 1],'r--')
             plt.xlim([0.0, 1.0])
             plt.ylim([0.0, 1.05])
             plt.xlabel('False Positive Rate')
             plt.ylabel('True Positive Rate')
             plt.title('Receiver operating characteristic(ROC-AUC)')
             plt.legend(loc="lower right")
             plt.show()
```

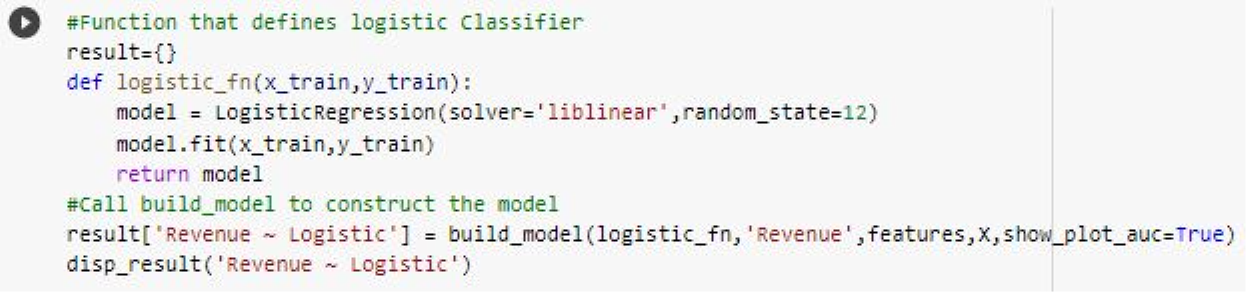


```
[31] #Summarize the results by finding the metrics using build-in functions
     def summarize_classification(y_test,y_pred):

         acc = accuracy_score(y_test,y_pred,normalize=True)
         num_acc = accuracy_score(y_test,y_pred,normalize=False)
         prec = precision_score(y_test,y_pred)
         recall = recall_score(y_test,y_pred)
         F1_score =  f1_score(y_test,y_pred)
         auc_score = roc_auc_score(y_test,y_pred)


         return{'Accuracy:': acc,
                'Accuracy_count:': num_acc,
                'Precision:': prec,
                'Recall:': recall,
                'F1_score:':F1_score,
                'AUC_ROC:':auc_score}
```



```
[53] # Display the details of the result of the classifier
     def disp_result(key):

             print('Classification: ',key)
             print()
             print('Training data:-')
             for score in result[key]['training']:
                 print(score,result[key]['training'][score])
             print()
             print('Test Data:-')
             for score in result[key]['test']:
                 print(score,result[key]['test'][score])
             print()
```

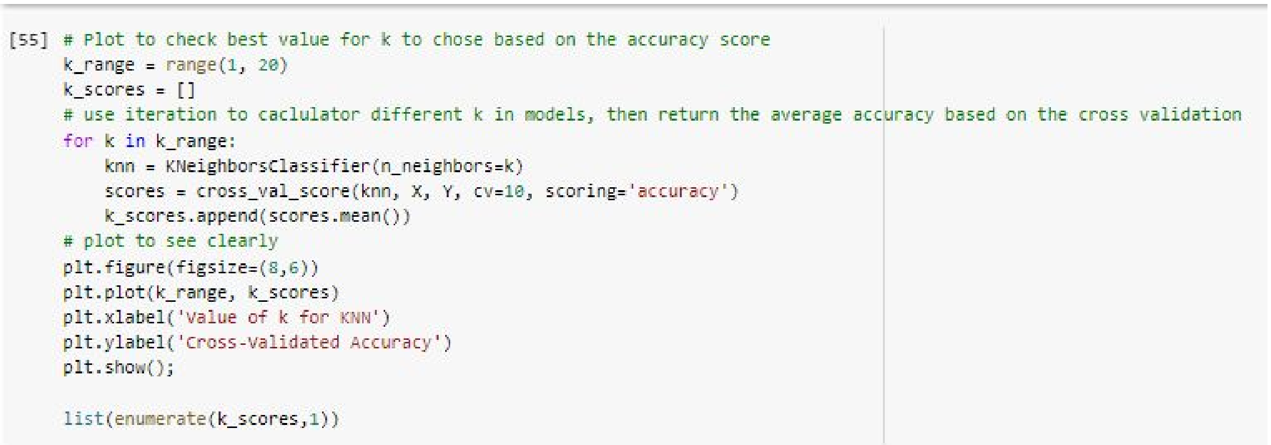
Logistic Classifier



```
#Function that defines logistic Classifier
result={}
def logistic_fn(x_train,y_train):
    model = LogisticRegression(solver='liblinear',random_state=12)
    model.fit(x_train,y_train)
    return model
#Call build_model to construct the model
result['Revenue ~ Logistic'] = build_model(logistic_fn,'Revenue',features,X,show_plot_auc=True)
disp_result('Revenue ~ Logistic')
```

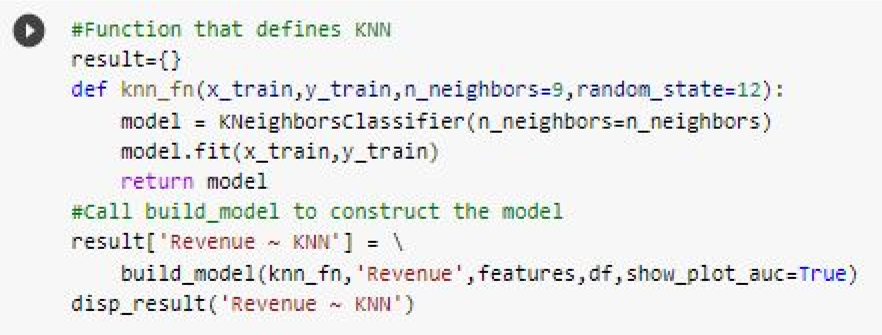
KNN Classifier

1. To evaluate the value of k



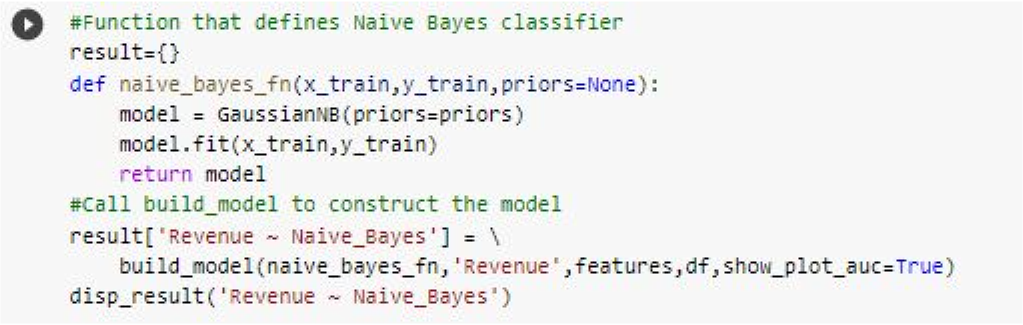
```
[55] # Plot to check best value for k to chose based on the accuracy score
     k_range = range(1, 20)
     k_scores = []
     # use iteration to caclulator different k in models, then return the average accuracy based on the cross validation
     for k in k_range:
         knn = KNeighborsClassifier(n_neighbors=k)
         scores = cross_val_score(knn, X, Y, cv=10, scoring='accuracy')
         k_scores.append(scores.mean())
     # plot to see clearly
     plt.figure(figsize=(8,6))
     plt.plot(k_range, k_scores)
     plt.xlabel('Value of k for KNN')
     plt.ylabel('Cross-Validated Accuracy')
     plt.show();

     list(enumerate(k_scores,1))
```

2. Define KNN Classifier



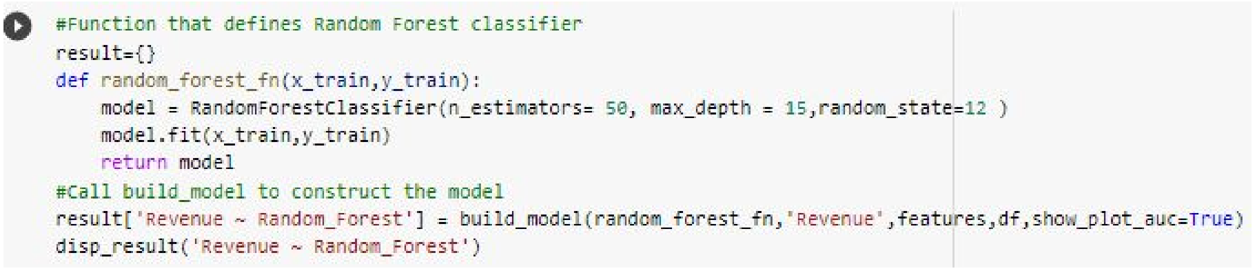
```
#Function that defines KNN
result={}
def knn_fn(x_train,y_train,n_neighbors=9,random_state=12):
    model = KNeighborsClassifier(n_neighbors=n_neighbors)
    model.fit(x_train,y_train)
    return model
#Call build_model to construct the model
result['Revenue ~ KNN'] = \
    build_model(knn_fn,'Revenue',features,df,show_plot_auc=True)
disp_result('Revenue ~ KNN')
```

Naive Bayes Classifier



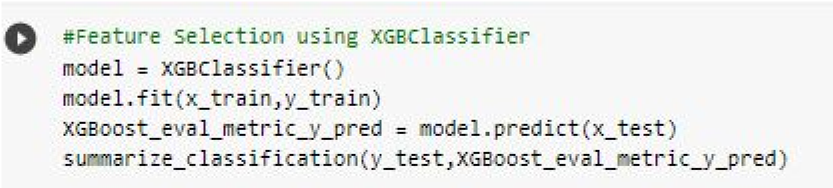
```
#Function that defines Naive Bayes classifier
result={}
def naive_bayes_fn(x_train,y_train,priors=None):
    model = GaussianNB(priors=priors)
    model.fit(x_train,y_train)
    return model
#Call build_model to construct the model
result['Revenue ~ Naive_Bayes'] = \
    build_model(naive_bayes_fn,'Revenue',features,df,show_plot_auc=True)
disp_result('Revenue ~ Naive_Bayes')
```

Random Forest Classifier

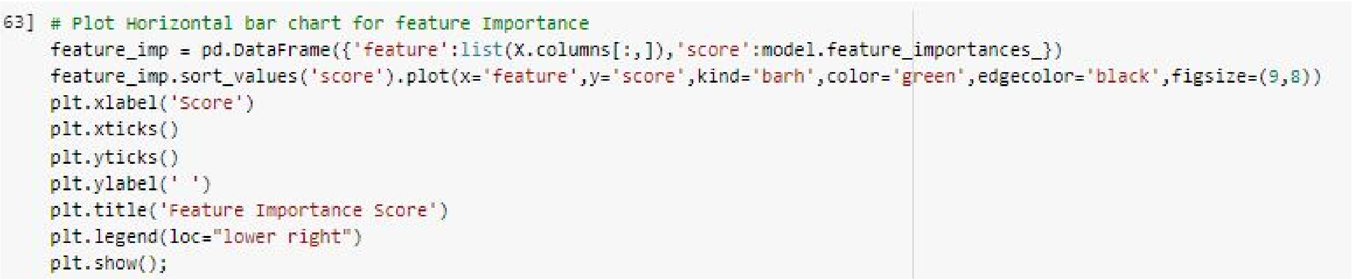


```
#Function that defines Random Forest classifier
result={}
def random_forest_fn(x_train,y_train):
    model = RandomForestClassifier(n_estimators= 50, max_depth = 15,random_state=12 )
    model.fit(x_train,y_train)
    return model
#Call build_model to construct the model
result['Revenue ~ Random_Forest'] = build_model(random_forest_fn,'Revenue',features,df,show_plot_auc=True)
disp_result('Revenue ~ Random_Forest')
```

Feature selection using XGBClassifier

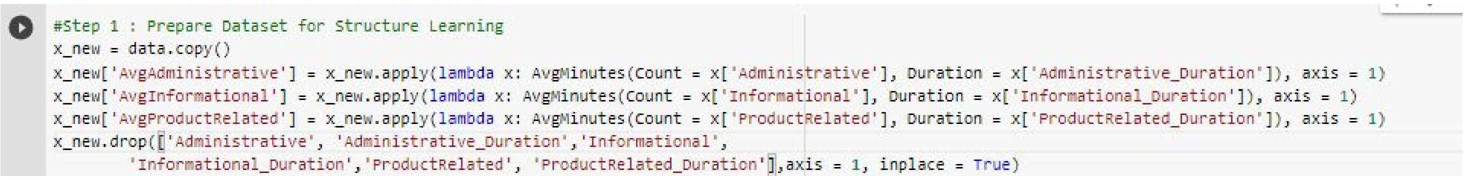


```
#Feature Selection using XGBClassifier
model = XGBClassifier()
model.fit(x_train,y_train)
XGBoost_eval_metric_y_pred = model.predict(x_test)
summarize_classification(y_test,XGBoost_eval_metric_y_pred)
```

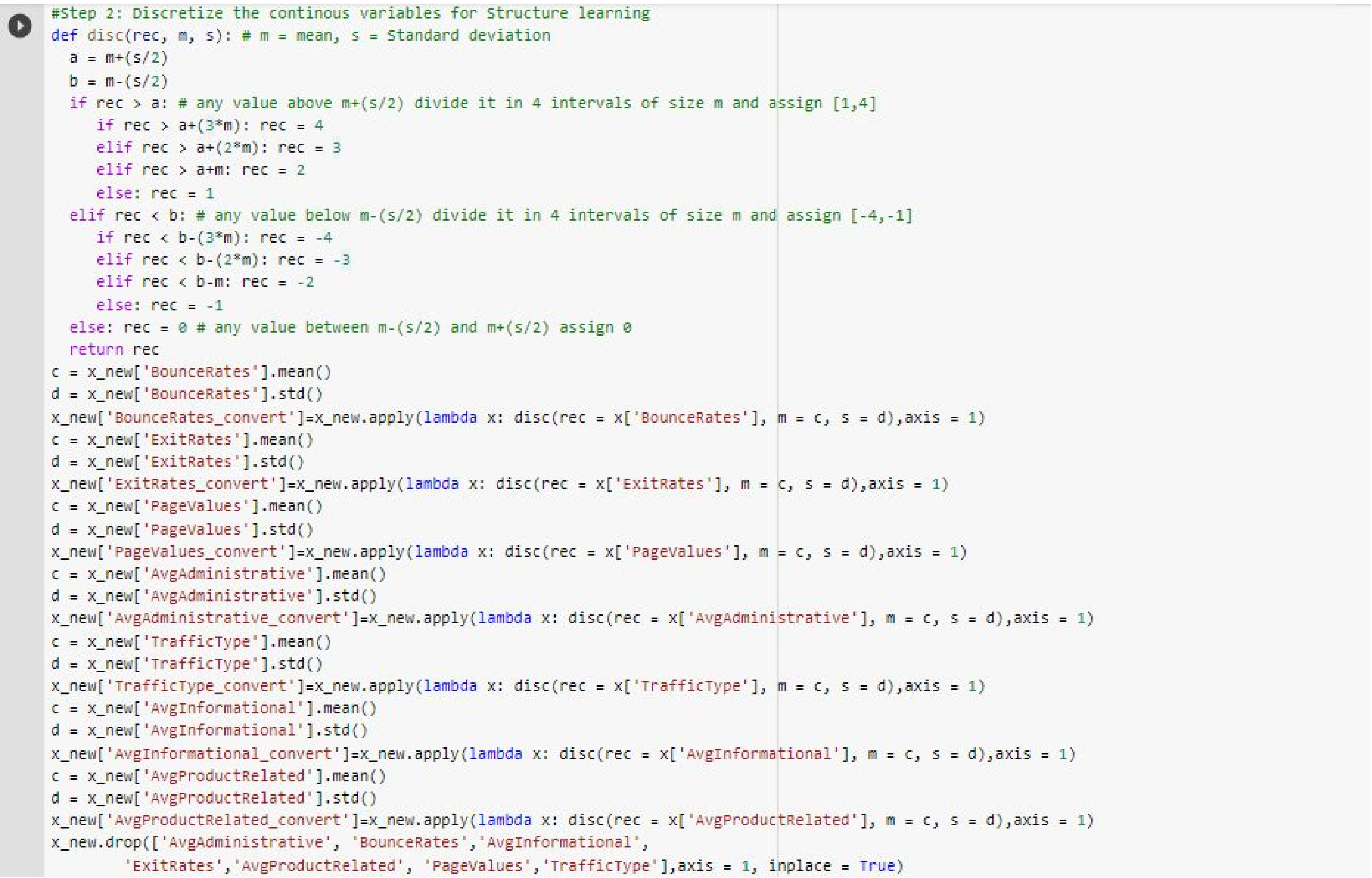


```
63] # Plot Horizontal bar chart for feature Importance
    feature_imp = pd.DataFrame({'feature':list(X.columns[:,]),'score':model.feature_importances_})
    feature_imp.sort_values('score').plot(x='feature',y='score',kind='barh',color='green',edgecolor='black',figsize=(9,8))
    plt.xlabel('Score')
    plt.xticks()
    plt.yticks()
    plt.ylabel(' ')
    plt.title('Feature Importance Score')
    plt.legend(loc="lower right")
    plt.show();
```

Discretization of attributes



```
#Step 1 : Prepare Dataset for Structure Learning
x_new = data.copy()
x_new['AvgAdministrative'] = x_new.apply(lambda x: AvgMinutes(Count = x['Administrative'], Duration = x['Administrative_Duration']), axis = 1)
x_new['AvgInformational'] = x_new.apply(lambda x: AvgMinutes(Count = x['Informational'], Duration = x['Informational_Duration']), axis = 1)
x_new['AvgProductRelated'] = x_new.apply(lambda x: AvgMinutes(Count = x['ProductRelated'], Duration = x['ProductRelated_Duration']), axis = 1)
x_new.drop(['Administrative', 'Administrative_Duration','Informational',
        'Informational_Duration','ProductRelated', 'ProductRelated_Duration'],axis = 1, inplace = True)
```



```
#Step 2: Discretize the continous variables for Structure learning
def disc(rec, m, s): # m = mean, s = Standard deviation
  a = m+(s/2)
  b = m-(s/2)
  if rec > a: # any value above m+(s/2) divide it in 4 intervals of size m and assign [1,4]
     if rec > a+(3*m): rec = 4
     elif rec > a+(2*m): rec = 3
     elif rec > a+m: rec = 2
     else: rec = 1
  elif rec < b: # any value below m-(s/2) divide it in 4 intervals of size m and assign [-4,-1]
     if rec < b-(3*m): rec = -4
     elif rec < b-(2*m): rec = -3
     elif rec < b-m: rec = -2
     else: rec = -1
  else: rec = 0 # any value between m-(s/2) and m+(s/2) assign 0
  return rec
c = x_new['BounceRates'].mean()
d = x_new['BounceRates'].std()
x_new['BounceRates_convert']=x_new.apply(lambda x: disc(rec = x['BounceRates'], m = c, s = d),axis = 1)
c = x_new['ExitRates'].mean()
d = x_new['ExitRates'].std()
x_new['ExitRates_convert']=x_new.apply(lambda x: disc(rec = x['ExitRates'], m = c, s = d),axis = 1)
c = x_new['PageValues'].mean()
d = x_new['PageValues'].std()
x_new['PageValues_convert']=x_new.apply(lambda x: disc(rec = x['PageValues'], m = c, s = d),axis = 1)
c = x_new['AvgAdministrative'].mean()
d = x_new['AvgAdministrative'].std()
x_new['AvgAdministrative_convert']=x_new.apply(lambda x: disc(rec = x['AvgAdministrative'], m = c, s = d),axis = 1)
c = x_new['TrafficType'].mean()
d = x_new['TrafficType'].std()
x_new['TrafficType_convert']=x_new.apply(lambda x: disc(rec = x['TrafficType'], m = c, s = d),axis = 1)
c = x_new['AvgInformational'].mean()
d = x_new['AvgInformational'].std()
x_new['AvgInformational_convert']=x_new.apply(lambda x: disc(rec = x['AvgInformational'], m = c, s = d),axis = 1)
c = x_new['AvgProductRelated'].mean()
d = x_new['AvgProductRelated'].std()
x_new['AvgProductRelated_convert']=x_new.apply(lambda x: disc(rec = x['AvgProductRelated'], m = c, s = d),axis = 1)
x_new.drop(['AvgAdministrative', 'BounceRates','AvgInformational',
        'ExitRates','AvgProductRelated', 'PageValues','TrafficType'],axis = 1, inplace = True)
```