MSCI 623 Final Project

Retail Analytics and Customer Loyalty using Machine Learning methods

Khalid Shaikh Qamar Ali Hasan
20801907

Nilesh Vishwani
20861160

August 04, 2020

University of Waterloo
200 University Ave W, Waterloo, ON, N2L 3G1

Abstract

In this project, we have used the retail transactional dataset from the UCI Machine Learning Repository. Based on the UCI website, the data contains the transactions of a UK-based and registered non-store online retail carried out between 01/12/2010 and 09/12/2011. The company is the main seller of unique all-occasion gifts and the customers of the company are mainly wholesalers. Our aim in this project is to perform exploratory, descriptive, predictive, and prescriptive analysis to help organizations to improve their sale and revenue using transactional data. We have extended our research of the analysis conducted on this dataset, and performed exploratory data analysis, determined product performance based on the transactions using association rules, devised a Loyalty program based on customer segmentation using clustering algorithms, and strategy recommendations for customer retention using five comparative classification models. We have carried out our analysis and implemented the algorithms using Python programming. The preliminary results of our analysis are adequate for retail businesses to dive deep into the nuances of transactional models in retail and e-commerce industry.

Table of Contents

List of Figures

List of Table

1. Introduction

Modern businesses require data-driven decision making for continuous improvement and commercial growth. Be it retail or the e-commerce industry, transactions play a key role to understand customer's purchase patterns, products' performance, and organizational augmentation. Online retail business has seen a significant growth due to digital transformation and ease of conducting businesses because of globalization. Since maintaining transactions have never been so easy because of enhanced computer systems, retailers leave no stone unturned to unleash business insights using analytical methods to drive market growth and improve business strategies. In our online retail data, we have implemented Association rules using Apriori algorithm to determine product relationships, and discover top performing products using support, confidence, lift and leverage metrics. We have performed Recency-Frequency-Monetary (RFM) analysis by tuning the feature variables to segment the customers based on their transactional behavior and used Principal Component Analysis with K-means clustering to discover customer clusters using RFM variables. Based on clustering, we have used a Loyalty program model to segment customers into different classes which could be used for cluster-based incentives and rewards for brand loyalty. To proffer Loyalty classes to the new customers based on initial transactions, we have experimented with classification algorithms namely, Logistic Regression, Decision Trees, Naïve Bayes, K-Nearest Neighbour and Random Forest. We have performed model evaluation and model comparison based on supervised and unsupervised algorithms to formulate strategies for business growth.

2. Literature Review

Retail analytics have played a crucial role in modern business. In general, on-premises and online retailers anticipated customers based on customer referrals and advertisements. Despite retailers focusing customer retention, they were not able to understand customer's behaviour which resulted in churning of customers. With the advent of advanced algorithms and systems, data-specific insights generation has revolutionised the way in which businesses are conducted. As per the UCI Machine Learning Repository website, several research works have been carried out on transactional data. In the published papers [1], [2] and [3], we learn how researchers have remarkably deployed conceptual methods to understand numerous techniques to model a successful retail industry analysis. Customer segmentation, pricing optimization, product improvement and demand forecasting are some of the areas in which the works have been conducted. [4] discusses about predicting customer churn using Pareto/NBD model, developing recommender systems based on semi-supervised RFM analysis and classification using Support Vector Machine and Stochastic Gradient Descent. Research paper [5] applies association rules to understanding customer behaviour and product relationships. Paper [6] applies data mining techniques for identifying high-value low-risk customers. It also demonstrates the importance of Pareto Principle in marketing by explaining how 20% of the customers generate 80% of the profit. [7] provides a thorough work to forecast future sales using time series analysis based on transactional data. [8] discusses about customer segmentation method based on K-means clustering using cosine similarity as the similarity measure. Research papers [9], [10] and [11] explain about customer segmentation based on hierarchical clustering, self-organizing maps method (SOM) to determine the best number of clusters, and other classification techniques.

Our goal is to understand product performance using market basket analysis, strategy recommendation for Loyalty program using RFM analysis, and check the results against Principal Component Analysis and K-means clustering. We also aim at developing classification models to classify new customers into different Loyalty classes.

3. Exploratory Data Analysis

The Online Retail transactional data was downloaded from https://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx as a CSV file. It consists of 541909 instances with 8 feature variables as shown below:

Table 1: Data description

Feature Name	Feature Description	Data Type
Invoice No.	Invoice Number	Categorical
Stock Code	Item Code/Product Code	Categorical
Description	Item Name/Product Name	Categorical
Quantity	Quantity/Transaction	Numeric
Invoice Date	Invoice Date and time	Datetime
Unit Price	Price of product per unit	Numeric
Customer ID	Customer Number	Numeric
Country	Name of the country	Categorical

To perform our analysis in python, we loaded the file into a DataFrame using Google Colaboratory Python Notebook.

3.1. Data Cleansing

We check if there are null values in the DataFrame because these values hinder the analysis and result in improper model generation. After removing the rows containing NULL values from the DataFrame, we get 3,97,884 instances for our analysis. This implies that 24.93% of customer ID is missing because it has null values for 1,35,080 instances.

As the value of Quantity cannot be negative, we check for the value of Quantity that is negative. We also check for the unit price variable as it cannot be negative or 0 because every product has a positive value of price.

For the ease of our analysis, we consider data for the year 2011 despite data being available from December 2010 to December 2011. This is because it would be a biased analysis if we perform exploratory analysis by considering merely a month of the year 2010. So, we consider the data from January 2011 to December 2011, to get the unbiased insights about customers' purchase pattern for one particular year for our analysis.

Further, Stock code variable consists of different types of levels like 'Bank Charges', 'Postal Charges', etc., which are not based on generating revenue but on expenditure. So, we remove these instances because they do not relate to customer transactions.

Consequently, for our exploratory analysis, we consider data with 370284 instances.

3.2. Data insights

To get an idea about the dataset, and to understand how we can solve the business problem based on our analytical methods, we perform insights analytics on the data. To get an overview about the explanatory variables that will be used for our analysis, we perform various data mining techniques to keep up with the essential and main patterns in the data.

3.2.1. Distinct value of Numerical Features

From our analysis, we observe that we have a total of 17008 unique transactions, 3591 unique items in our data, and 4214 unique customers from different regions across the globe.

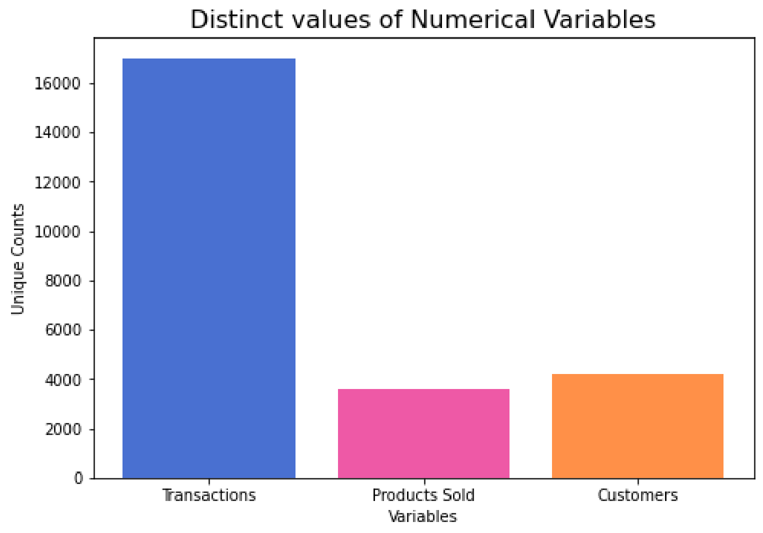


Figure 1: Distinct values of Numerical Features

3.2.2. Top-5 countries based on Customer ID

Following are the countries having maximum number of customers purchasing online:

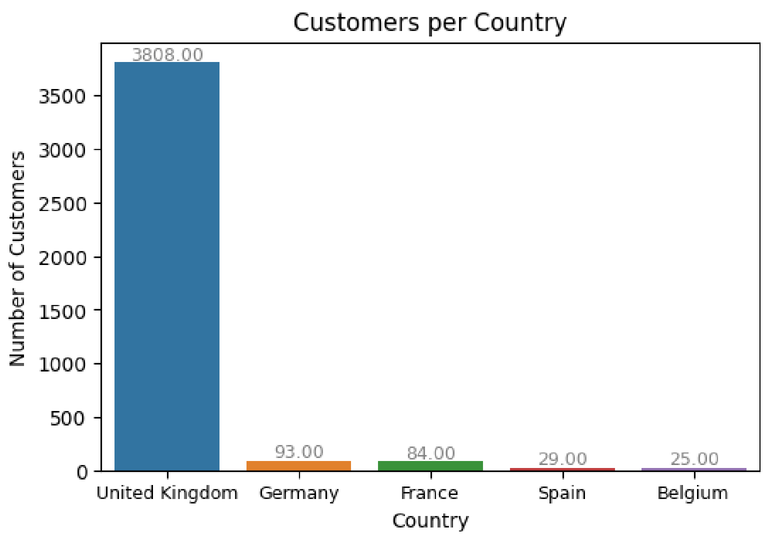


Figure 2: Top-5 countries based on Customer ID count

As more than 90% of the customers are from United Kingdom (UK), so our aim is to analyze the Customers from United Kingdom. Overall, after performing all kinds of data cleansing and preprocessing, the DataFrame consists of 330379 instances for UK.

3.2.3. Top 5 Products (Variable: StockCode)

The reason for choosing the StockCode variable and not the Description variable is that it is readable and easy to classify. So, Top-5 products are classified based on StockCode. Count of all the Top-5 products is greater than 1000, so these 5 products have relatively high demand. In contrast to this, the price of these items would be relatively lower because demand is high. From a business perspective, increasing the price for these highly demanding products might have a significant impact on sales and revenue of the business.

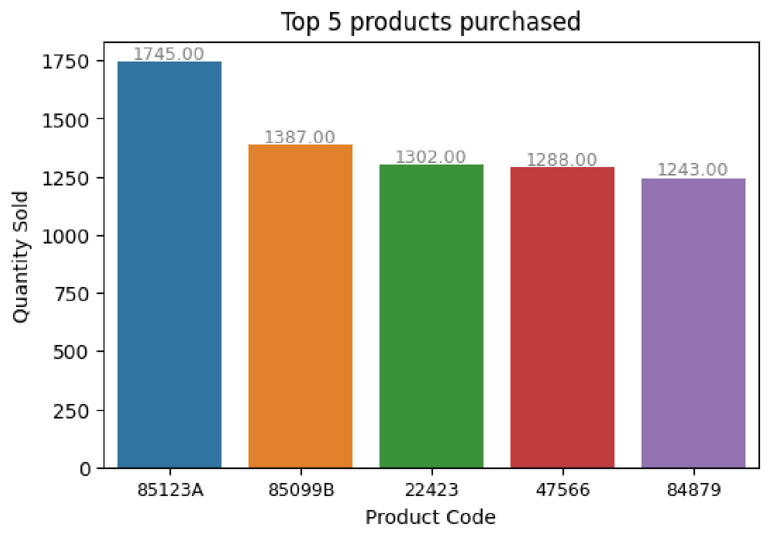


Figure 3: Top-5 high demanding products

Top five item code descriptions are as follows:

85123A	: WHITE HANGING HEART T-LIGHT HOLDER
85099B	: JUMBO BAG RED RETROSPOT
22423	: REGENCY CAKESTAND 3 TIER
47566	: PARTY BUNTING
84879	: ASSORTED COLOUR BIRD ORNAMENT

3.2.4. Time Series Plot

Based on the time series plot of the Total Revenue (Calculated by multiplying quantity and unit price), plot indicates that the data is non-stationary for the year 2011 i.e., there are trends and seasonality which need to be removed to predict future demand.

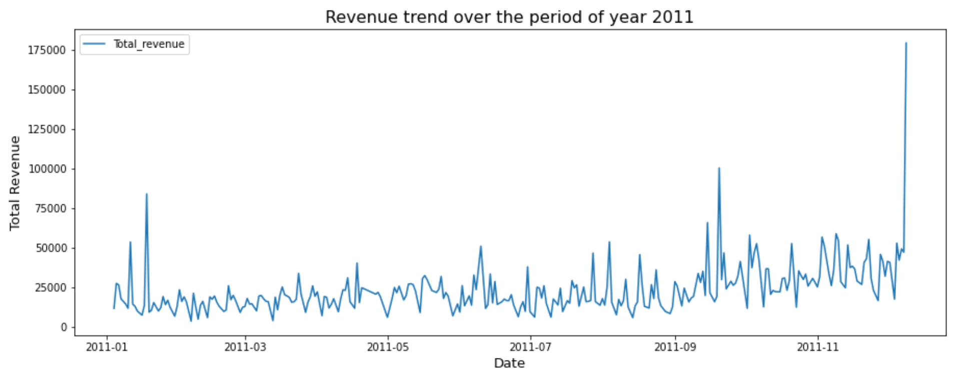


Figure 4: Revenue trend for the year 2011 (Time Series plot)

3.2.5. Month-wise purchase pattern

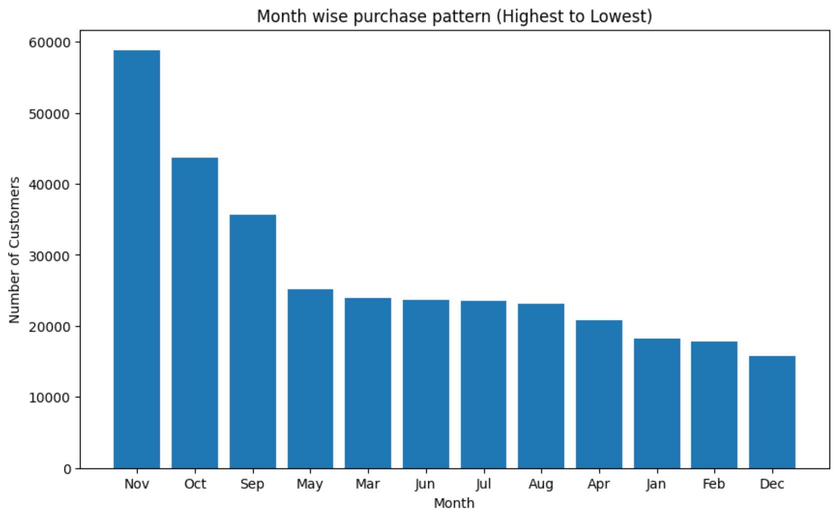


Figure 5: Month-wise purchase pattern (Sorted: Highest to Lowest)

From the above graph, the decreasing order of customers purchase pattern is shown. This trend is observed because during October and November, people buy gift items for the end of the year, Christmas eve and long winter vacation, which implies that the purchase in December is low. This is the main reason that there are festive sales during December to increase the purchase for the customers to complement with rest of the months.

3.2.6. Day-wise purchase pattern

Based on this plot, it is evident that there is highest traffic of order deliveries on Thursday, which is followed by Wednesday and Tuesday. These details help in preplanning regarding procurement and delivery of products by the vendors accordingly. There are minimum orders on Sunday which indicates that Customers prefer to spend their time on weekends with family.

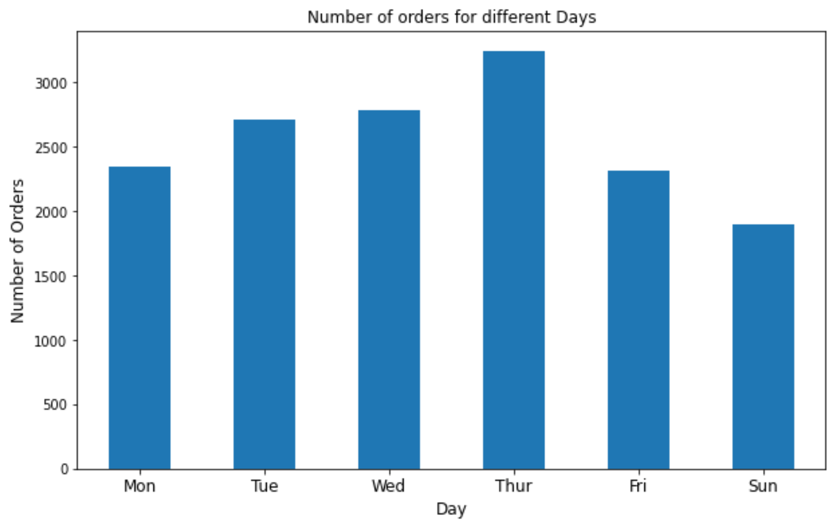


Figure 6: Day-wise purchase pattern (Sorted: Monday to Sunday)

3.2.7. Hourly purchase pattern

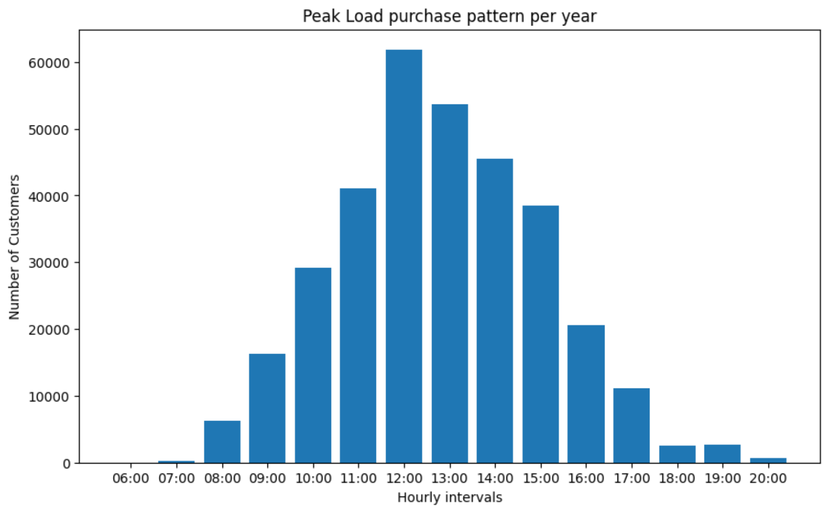


Figure 7: Hourly purchase pattern (Sorted: Clockwise)

From the graph, it is evident that peak time for order purchases and deliveres is between 12 pm to 1 pm. Whereas, during the nighttime (from 9 pm to 6 am), there's no minimal order purchases and deliveries.

4. Machine Learning Methods

To represent data in the context of our business problem, we use unsupervised and supervised machine learning algorithms for a deep understanding of data insights and analytics.

4.1. Association Rule Mining[13]

Association rule learning is a rule-based machine learning method to discover interesting relations between variables. It uses some measures of interestingness such as Support, Confidence, Lift and Leverage. One of the key applications of association rule learning is market basket analysis, which is used to unravel the purchase pattern of the customers.

4.1.1. Apriori Algorithm

This algorithm is used to derive the correlation between frequently purchased itemset. In our analysis, the data is featured in a way that the items are grouped based on individual invoices i.e., grouped based on 'InvoiceNo' variable and 'Description' variable. This is to stack itemset of each transaction together.

Here, StockCode (or Description) is a categorical data, which is converted to binary using label encoding i.e. one hot encoding and tabulated against each InvoiceNo.

4.1.2. Support

Frequency of items in the data set is an indication of Support. Using Apriori algorithm, Support of each itemset is calculated and sorted in descending order. Based on the experimental support values, we set minimum support = 0.01. This implies the list of all itemset which appear in transaction at least 1% of the time. Since our dataset is diverse and our categorical variable has more than 3000 distinct products, so we set minimum threshold to a least value to get an overall idea of the itemset support values.

Table 2: Frequent itemset (Top 5 Support values)

Itemset	Support
WHITE HANGING HEART T-LIGHT HOLDER	0.110385
JUMBO BAG RED RETROSPOT	0.089655
REGENCY CAKESTAND 3 TIER	0.084292
PARTY BUNTING	0.083115
ASSORTED COLOUR BIRD ORNAMENT	0.079323

As observed, 'WHITE HANGING HEART T-LIGHT HOLDER' has maximum support of ~0.11 in the entire transaction, which is followed by 'JUMBO BAG RED RETROSPOT' having support of ~0.09, and 'REGENCY CAKESTAND 3 TIER' having support of ~0.0843. For instance, ~0.11 support value indicates that for 11% of the transactions, 'WHITE HANGING HEART T-LIGHT HOLDER' item is purchased.

4.1.3. Confidence, Lift, and Leverage

Confidence indicates how often the rule (if-then statements) has been found to be true. We used other metrics such as Lift and Leverage to evaluate our product relationships. Lift determined the percentage increase in expectation that someone will purchase an item, given that some other item has been purchased. Leverage measures the difference of two different items appearing together in the data set and what would be expected if they were statistically dependent.

We sorted our data based on confidence values, and we observe the following:

Table 3: Association rules between the items (Sorted by: Confidence metrics)

Antecedents	Consequents	Antecedent Support	Consequent Support	Support	Confidence	Lift	Leverage
(HERB MARKER THYME)	(HERB MARKER ROSEMARY)	0.010986	0.011248	0.010463	0.952381	84.673311	0.010339
(HERB MARKER PARSLEY)	(HERB MARKER ROSEMARY)	0.010921	0.011248	0.010267	0.940120	83.583206	0.010144
(REGENCY TEA PLATE PINK, REGENCY TEA PLATE ROSES)	(REGENCY TEA PLATE GREEN)	0.010725	0.014844	0.010005	0.932927	62.847212	0.009846
(HERB MARKER ROSEMARY)	(HERB MARKER THYME)	0.011248	0.010986	0.010463	0.930233	84.673311	0.010339
(REGENCY TEA PLATE GREEN, REGENCY TEA PLATE PINK)	(REGENCY TEA PLATE ROSES)	0.010790	0.017395	0.010005	0.927273	53.307724	0.009818

Here, we observe that {HERB MARKER ROSEMARY} is purchased whenever {HERB MARKER THYME} is purchased with a confidence of 0.954 (i.e., 95.4%). This is followed by the {HERB MARKER PARSLEY} → {HERB MARKER ROSEMARY} having 94% confidence. Thus, these products are important in our business, and there are several strategies that could be recommended based on these relationships. For example, one of the strategies is to recommend HERB MARKER ROSEMARY as a suggestion whenever HERB MARKER THYME is purchased. In case of an on-premises retail shop, these two products are kept in similar shelves for ease of accessibility. In contrast to this, itemset with high confidence are placed in two faraway shelves. This is because the retail shop owners expect the customers to pass through different isles, and thus, engaging them to look at other products and promotions, which in turn, persuades them to try new items.

In an e-commerce model, the itemset with highest confidence values are not recommended instantly. Rather, itemset with confidence values less than the maximum confidence values are displayed as recommendation to the customers, along with their consequents to improve the item performance based on customer behavior.

4.2. Clustering and Segmentation[14]

Clustering algorithms are unsupervised techniques to discover hidden clusters based on different variables, which could be identified differently based on the analytical problem.

4.2.1. RFM (Recency, Frequency, Monetary) analysis

RFM (Recency-Frequency-Monetary) analysis is a clustering technique used to examine how recently the purchase has been carried out by the customer (Recency), how frequently a customer purchases (Frequency), and amount spent by the customer (Monetary). This analysis is carried out to determine quantitatively the best customers who would retain, and the others who have the propensity to churn.

Calculating RFM variables (Feature Engineering):

The overall data is grouped by the Customer ID. Recency values are calculated by subtracting the individual purchase dates from the last date of the year in the data. Frequency values are calculated by counting transactions conducted by the customer over the period of 1 year. Monetary values are calculated based on aggregating the Customer ID by taking sum of their expenditure in the transactions. Following is an example of new feature generation grouped by Customer IDs.

Table 4: Example of RFM variables (Recency, Frequency and Monetary)

CustomerID	Recency	Frequency	Monetary
12346.0	325	1	77183.60
12747.0	2	9	3489.74
12748.0	0	171	28131.42
12749.0	3	5	4040.88
12820.0	3	4	942.34

Using these newly generated feature variables, we create quantiles to distribute the variables and assign them values from 1 to 4. We then group these variables together based on a specific calculation pattern to identify different clusters by assigning the scores. These scores are calculated using the group values based on the business problem. Here in our data, we aim at implementing the Loyalty program model, where the customers are provided with individual promotions based on the above variables. Loyalty Program is the reward provided to the customers based on their loyalty, and, provided to the customers who have the propensity to churn after a few transactions depending on individual experiences too. For the latter customers, special promotions could be provided to lure them to purchase from our business. For the former customers, special price discrimination is provided to retain them. Thus, based on the above example, we get the updated DataFrame as shown in the figure below:

Table 5: Group and Score variable generation using RFM variables

CustomerID	Recency	Frequency	Monetary	RQuartile	FQuartile	MQuartile	Group	Score
12346	325	1	77183.60	1	1	4	114	6
12747	2	9	3489.74	4	4	4	444	12
12748	0	171	28131.42	4	4	4	444	12
12749	3	5	4040.88	4	4	4	444	12
12820	3	4	942.34	4	3	3	433	10

Dividing into Loyalty groups:

Based on our strategy recommendation, we aim at providing 4 classes of Loyalty cards[12]. This is like the airline reward programs where individual customers are provided with different loyalty policies based on their travel history. Here, we provide the following Loyalty cards to the customers:

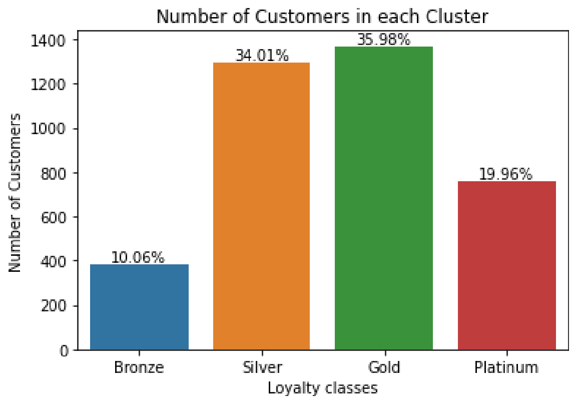
- **Platinum:** The class of customers who are extremely loyal to our Online Retail business. They are to be awarded special incentives and rewards based on their purchase pattern
- **Gold:** The class of customers who are loyal to our Online Retail business and purchase frequently but also purchase from other vendors. They require discounts on majority of the products.
- **Silver:** The class of customers who are casual visitors and are not as regular as the Gold class customers. They require promotions to decide on purchases and are mostly small-scale wholesalers
- **Bronze:** The class of customers who have higher propensity to churn. They are provided with least benefits but lured through various promotions and special prices on distinct products.

Using the above example, we cluster the scores into 4 intervals and rename each interval with our proposed loyalty classes. Thus, we get updated DataFrame as shown in the figure below:

Table 6: Interval and Loyalty class variable generation using RFM scores

CustomerID	Recency	Frequency	Monetary	RQuartile	FQuartile	MQuartile	Group	Score	Interval	LoyaltyClass
12346	325	1	77183.60	1	1	4	114	6	(3,6]	Silver
12747	2	9	3489.74	4	4	4	444	12	(10,12]	Platinum
12748	0	171	28131.42	4	4	4	444	12	(10,12]	Platinum
12749	3	5	4040.88	4	4	4	444	12	(10,12]	Platinum
12820	3	4	942.34	4	3	3	433	10	(6,10]	Gold

We plot the following count plot to visualize the number of customers in each cluster. As observed, 19.96% (approximately 20%) of the customers are the most loyal customers. This explains the fact that 80% of the business profit comes from 20% of the customers. Thus, it proves the 80-20 rule (Pareto principle)



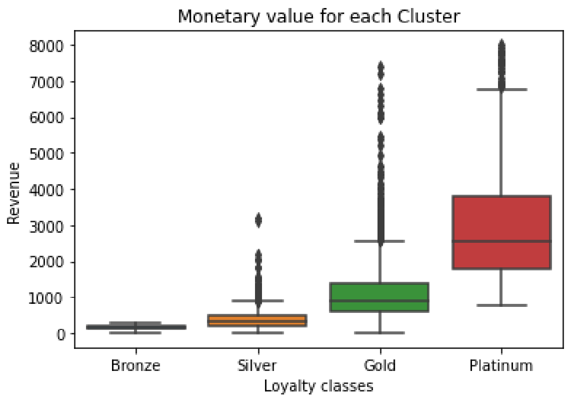


Figure 8: Number of customers in each cluster *Figure 9: Revenue generated by each cluster*

The revenue plot indicates that about 80% of the revenue is generated from Platinum customers which aligns with the 80-20 Pareto Principle i.e. ~20% customers fall in Platinum Class, who are the most precious for the business.

4.2.2. Principal Component Analysis and K-means Clustering[15]

To improve the performance of the model and discover relationships between the RFM variables, we have combined PCA and K-means clustering which helps in dimensionality reduction of RFM variables.

Distribution of RFM variables:

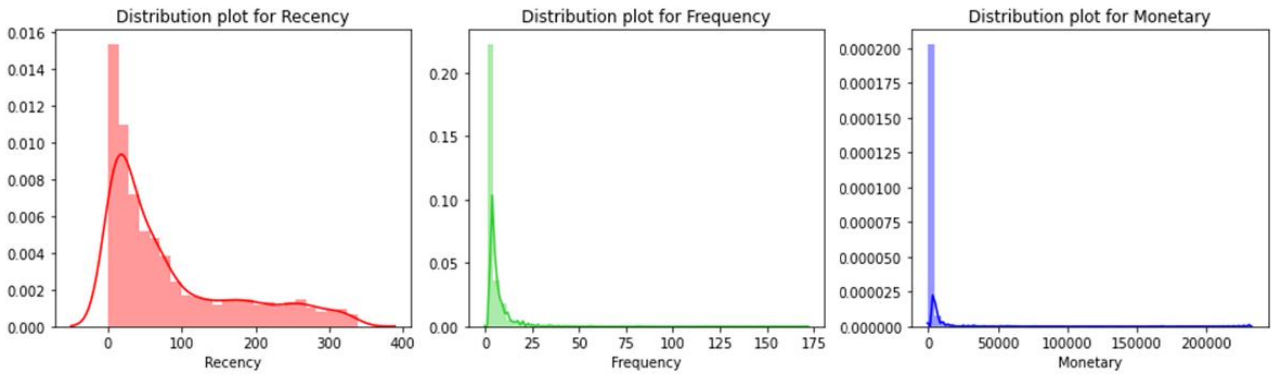


Figure 10: Distribution plots for RFM variables

In the above distribution plots, we observed that Recency, Frequency and Monetary values are rightly skewed. So, we performed log transform to normalize the data.

Calculating Log Values for RFM variables:

We calculate logarithmic values for RFM values as they are not distributed normally across their respective mean. This is also because while performing Principal Component Analysis, we reduce the dimensions from three to two in our analysis. For zero and negative values, we add a delta component to handle infinite numbers during log transformation.

Distribution of log values of RFM variables:

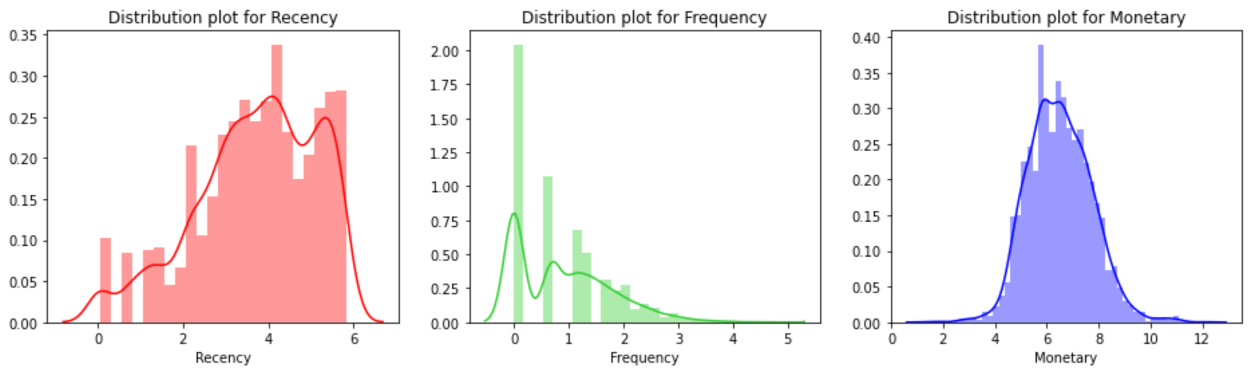


Figure 11: Distribution plots for Log transformed RFM variables

As observed in above distribution plots, Recency and Monetary are approximately normal, whereas, Frequency is rightly skewed. Thus, Principal Component Analysis will be carried out on Log transformed data.

Correlation plots for RFM variables:

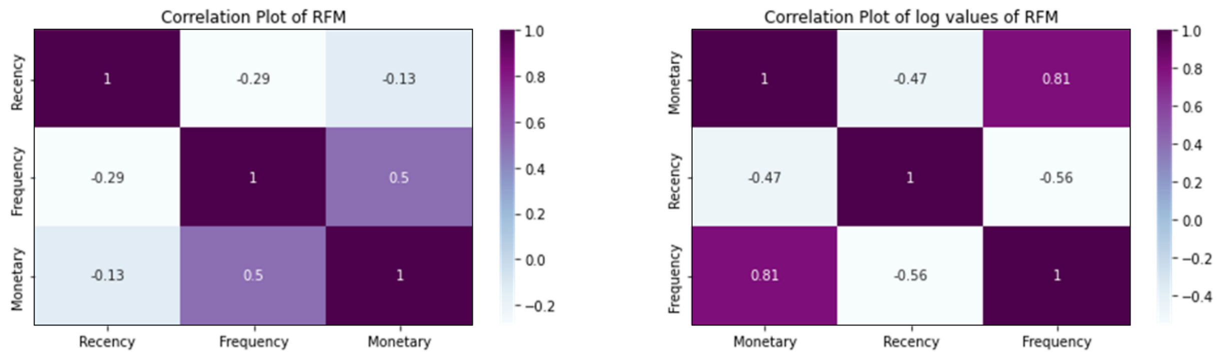


Figure 12: Correlation plots for RFM variables and Log transformed RFM variables

The correlation plot of RFM variables demonstrates the independence of the variables, whereas the correlation plot of log values of RFM variables shows correlation between the Frequency and Monetary values.

Combining PCA and K-means:

PCA is performed on the Log transformed data so that result obtained align with the goals of the project. This is because the data for RFM variables are skewed if performed without log transformation.

The explained variance ratio in Recency is 0.745, Frequency is 0.194, Monetary is 0.06, which indicates that most of variance in our data is explained by “Recency” variable and least variance is explained by “Monetary” variable. So, the first principal component takes into consideration for the highest variance i.e. Recency while second principal component takes into consideration the second highest variance i.e. Frequency into account while there is no principal component required for Monetary because explained variance for that variable is much lower.

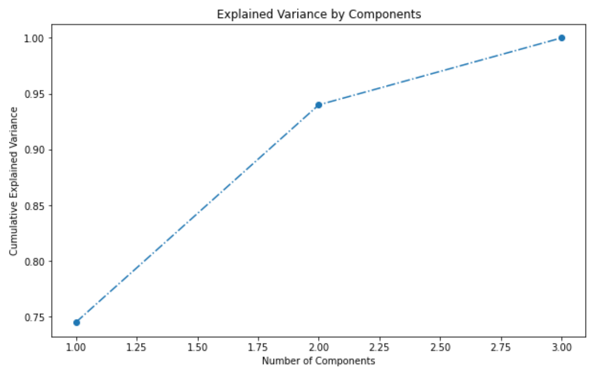


Figure 13: Identifying components using the Explained variance ratio

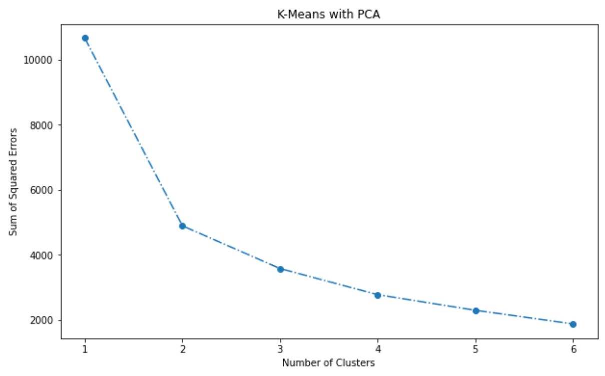


Figure 14: Elbow method to determine the optimum number of clusters

PCA is combined with K-means by fitting the scores obtained from PCA into K-means algorithm and within cluster sum of squares is taken to find the number of clusters.

From the 'Explained Variance Plot' it is evident that there are 2 principal components because the rule says that about 80% of variance should be preserved. Here, approximately 94% of variance is preserved, which depicts the accuracy of clusters for 2 components. This is determined using the silhouette score which has a maximum value when 2 clusters are formed. This is also evident from the Elbow Curve that there are 2 clusters for K-means algorithm because the graph steeply declines after 2 clusters.

Thus, we plot these components as shown below:

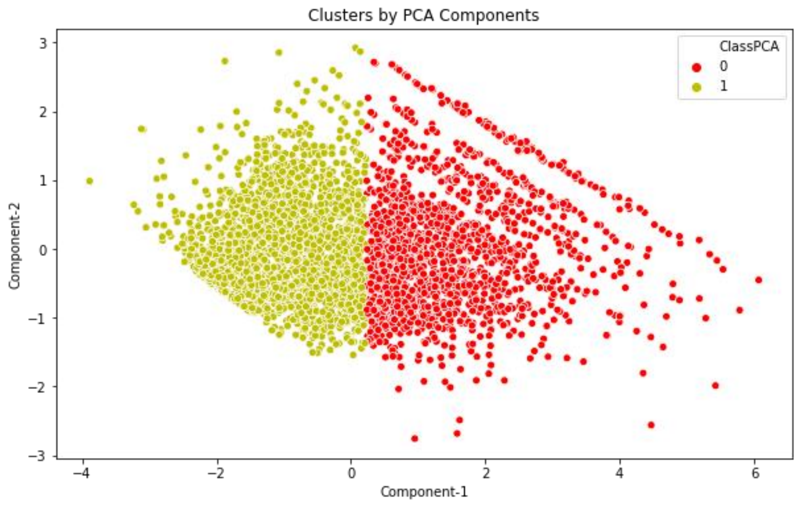


Figure 15: Distribution of clusters based on PCA

This indicates that there are two groups formation based on the RFM variables. For the above graph, it is evident that Component-1 explains higher variance in the range (-4,6) as compared to Component-2, which explains the variance in the range (-3,3).

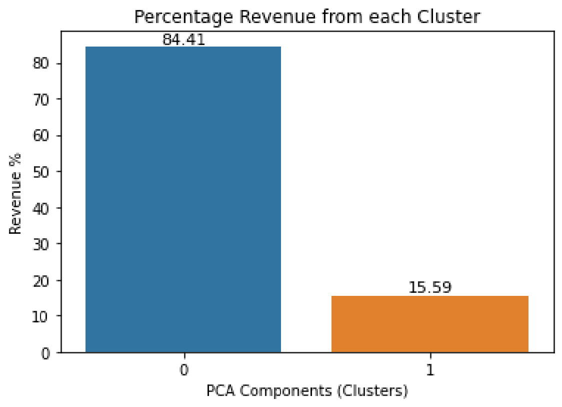


Figure 16: Percentage Revenue from each Component

Analyzing both clusters generated using PCA depicts that the revenue generated in Cluster 0 is above 84.4%, while the rest of the revenue is generated by Cluster 1. Thus, this demonstrates that there are mainly two types of customers. The one who are extremely loyal (contribute approximately 80% of revenue) and the rest who could churn if not provided with relevant pricing, discounts, and promotions.

4.3. Classification Algorithms[16]

We aim at building classification models to predict the Loyalty class that will assigned to new customers based on their RFM scores determined through their behaviour and purchase patterns. As we have 4 different classes i.e., Platinum, Gold, Silver and Bronze, we have compared 5 different classification models to come up with the effective classification model based on their evaluation metrics.

To perform our analysis, we have split our RFM data into 70% train data and 30% test data. The model is fitted on train data and predicted on test data.

4.3.1. Logistic Regression

As we want to predict 4 different classes of customers, logistic regression is used because these are ordinal classes which can be linearly separable into 4 groups.

Accuracy: 78.74%
Balanced Accuracy: 80.42%

Confusion Matrix

Predicted Class

True Class	Platinum	Gold	Silver	Bronze
Platinum	98	0	0	21
Gold	1	281	67	54
Silver	0	27	214	0
Bronze	25	48	0	307

4.3.2. Decision Trees

One of the fastest methods to classify the customers into 4 different classes and easy to visualize. Also, it considers all the possible outcomes into analysis by analyzing comprehensively across each branch. Then, the decision node is identified accordingly.

Accuracy: 95.1 %
Balanced Accuracy: 96.45 %

Confusion Matrix

Predicted Class

True Class	Platinum	Gold	Silver	Bronze
Platinum	119	0	0	0
Gold	0	367	4	32
Silver	0	0	241	0
Bronze	3	17	0	360

4.3.3. Naïve Bayes

This classifier is particularly used when there is assumption of independence into analysis and it performs better under categorical input variables. But it predicts the class data way faster.

Accuracy: 76.38 %
Balanced Accuracy: 74.95 %

Confusion Matrix

		Predicted Class			
		Platinum	**Gold**	**Silver**	**Bronze**
True Class	**Platinum**	117	0	0	2
	Gold	1	304	49	49
	Silver	0	34	207	0
	Bronze	83	52	0	245

4.3.4. KNN

KNN is particularly used when the dataset is small, so it is also known as Lazy learning algorithm. Thus, based on distance function i.e. similarity measure it classifies any new case in the dataset.

Accuracy: 82.76 %
Balanced Accuracy: 83.21 %

Confusion Matrix

		Predicted Class			
		Platinum	**Gold**	**Silver**	**Bronze**
True Class	**Platinum**	109	0	0	10
	Gold	0	309	38	56
	Silver	0	33	208	0
	Bronze	20	40	0	320

4.3.5. Random Forest

This algorithm uses ensemble of decision trees. It is mainly used because it does not allow overfitting in the model and achieve higher accuracy without tuning the hyper-parameters.

Accuracy: 93.79 %
Balanced Accuracy: 95.61 %

Confusion Matrix

		Predicted Class			
		Platinum	**Gold**	**Silver**	**Bronze**
True Class	**Platinum**	117	0	0	2
	Gold	0	383	1	19
	Silver	0	11	230	0
	Bronze	0	38	0	342

4.3.6. Model Comparison

Using confusion matrices from the above fitted models, it is evident that Decision Tree and Random Forest models have much better accuracy as compared to Logistic Regression and Naïve Bayes. In contrast to this, KNN perform relatively better than the latter two models, but not as good as the former two models.

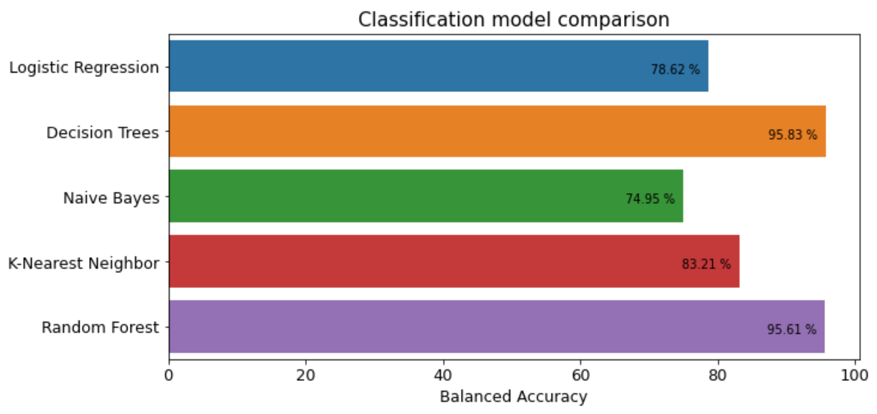


Figure 17: Model comparison based on balanced accuracy

5. Conclusion

Retail analytics is mainly focused on customer retention, product performance and revenue optimization. Based on our analysis using the transactional data, we conclude the following:

- Top 5 items based on Support value were obtained by performing association rule mining using Apriori algorithm. As an instance, item with description ‘White Hanging Heart T-Light Holder’ was observed having maximum support of 0.110385, which implied that this product has highest importance in the Online Retail transactions.
- Product relationships were determined and identified based on Confidence, Lift, and Leverage metrics. As observed for the maximum confidence, customers purchased (Herb Marker Rosemary) whenever they purchase (Herb Marker Thyme). This association was observed with 95.24% confidence and 84.67% expectation that these item set will be purchased. Similar product relationships were determined for all the products in the transactions which gave an idea of high and low performing products.
- Customers were segmented into 4 groups using Recency-Frequency-Monetary analysis. Based on these groups, we proposed a Loyalty program model having 4 loyalty classes i.e., Platinum, Gold, Silver and Bronze, to provide customers with pricing and promotions based on how recently the purchase has been carried out by the customer (Recency), how frequently a customer purchases (Frequency), and amount spent by the customer (Monetary)
- Clusters were formed using Principal Component Analysis and K-means clustering to understand two major clusters which commensurate with 80-20 Pareto Principle by explained variance ratio. This also suggested the fact that there are mainly two types of customers i.e., loyal customers who contribute towards 80% of the revenue, and the others, who have infrequent purchase behaviour.
- To propose suitable loyalty class to the new customers, multiple classification models were performed using train and test data, having accuracies in the following order:
  Decision Tree ≈ Random Forest > K-nearest Neighbour > Logistic Regression > Naïve Bayes
- Based on the above supervised and unsupervised machine learning algorithms, we performed a complete retail analytics solution for different aspects of transactional data i.e., customers, products, and transactions, which can be used for revenue optimization, improving product performance and increasing customer retention rate.

6. Future Scope

- To investigate further on business problems having millions of transactional data points, other advanced algorithms such as Neural Network based classification technique could be used. This will learn small patterns and features in the data and improve model performance.
- For customer segmentation, other methods such as K-prototype clustering, density based clustering and hierarchical clustering could be implemented based on the number of categorical and numerical features which could be developed through feature engineering.
- To forecast future revenue based on day-to-day customer transactions, various time series forecasting model such as ARIMA, Exponential smoothening, and Holt-Winter smoothening could be used on stationary data depending upon the trend and seasonality components.

7. References

[1] Chen, Daqing & Sain, Sai & Guo, Kun. (2012). Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining. Journal of Database Marketing & Customer Strategy Management. 19. 10.1057/dbm.2012.17.

[2] Bandara, Kasun & Shi, Peibei & Bergmeir, Christoph & Hewamalage, Hansika & Tran, Quoc & Seaman, Brian. (2019). Sales Demand Forecast in E-commerce using a Long Short-Term Memory Neural Network Methodology.

[3] Likhith D, Vamsi A, Rajashree Shettar. (2020). Customer Attrition and Retention. International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-9 Issue-7.

[4] Punya P Shetty, Varsha C M, Varsha D Vadone, Shalini Sarode, Pradeep Kumar D. (2019). Customers Churn Prediction with Rfm Model and Building a Recommendation System using Semi-Supervised Learning in Retail Sector. International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8, Issue-1.

[5] Anthony O. Otiko, John A. Odey, Gabriel A. Inyang. (2019). Conceptualisation Of Market Segmentation and Patterns For pre-Christmas Sales In An Online Retail Store. Journal of Science, Engineering and Technology, Vol. 6 (1), pages 51-59.

[6] Dr. Sankar Rajagopal. (2011). Customer Data Clustering Using Data Mining Technique. International Journal of Database Management Systems (IJDMS) Vol.3.

[7] Chen, Daqing & Guo, Kun & Li, Bo. (2019). Predicting Customer Profitability Dynamically over Time: An Experimental Comparative Study.

[8] A. Aziz. (2017). Customer segmentation based on behavioral data in emarketplace.

[9] Balmeet Kaur, Pankaj Kumar Sharma. (2019). Implementation of Customer Segmentation using Integrated Approach. International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-6S

[10] Golmah, Vahid. (2014). A Case Study of Applying SOM in Market Segmentation of Automobile Insurance Customers. International Journal of Database Theory and Application. 7. 25-36. 10.14257/ijdta.2014.7.1.03.

[11] Hasan Ziafat, Majid Shakeri (2014), Using Data Mining Techniques in Customer Segmentation, Hasan Ziafat Int. Journal of Engineering Research and Applications www.ijera.com ISSN: 2248-9622, Vol. 4, Issue 9 (Version 3), 70-79

[12] Hederstierna, Anders & Sällberg, Henrik. (2008). Bronze, Silver and Gold: Effective Membership Design in Customer Rewards Programs. 2nd European Conference on Information Management and Evaluation, ECIME 2008.

[13] https://towardsdatascience.com/learning-associations-74a8c27cf142

[14] https://towardsdatascience.com/customer-segmentation-data-science-modeling-210fb36c90bb

[15] https://scikit-learn.org/stable/tutorial/statistical_inference/unsupervised_learning.html

[16] https://scikit-learn.org/stable/supervised_learning.html

8. Appendix

Original file is located at
https://colab.research.google.com/drive/1FDC6nVq-B50PqpGOGaoTzE-DlR5LcDHu

```
# Importing Libraries
import pandas as pd
import numpy as np
import warnings
import datetime
import datetime as dt
import seaborn as sns
import matplotlib.pyplot as plt

# Importing csv file into the system.
dataframe = pd.read_excel('http://archive.ics.uci.edu/ml/machine-learning-
databases/00352/Online%20Retail.xlsx')

df = pd.DataFrame(dataframe)

# Understanding the data.
df.head()
df.info()
df.shape

#Checking number of NULL values in the dataframe
np.sum(df.isnull())

print("Customers missing: ", round(df['CustomerID'].isnull().sum() * 100 / len(df),3),"%" )

# Removing instances where Customer ID = NULL
df.dropna(subset=['CustomerID'],how='all',inplace=True)

# Removing negative values of Quantity in the dataframe if present.
df = df.drop(df[(df["Quantity"] < 0)].index)

# Dropping zero values of Unit Price
df = df.drop(df[(df["UnitPrice"] == 0)].index)

# Considering data for the year 2011 because for the year 2010, only December data is available.
df = df.query("InvoiceDate.dt.year != 2010")

# Removing all other values from Stock Code except Product Codes.
df = df.drop(df[df['StockCode'].astype('str') == 'M'].index)
df = df.drop(df[df['StockCode'].astype('str') == 'PAD'].index)
df = df.drop(df[df['StockCode'].astype('str') == 'C2'].index)
df = df.drop(df[df['StockCode'].astype('str') == 'BANK CHARGES'].index)
df = df.drop(df[df['StockCode'].astype('str') == 'POST'].index)
df = df.drop(df[df['StockCode'].astype('str') == 'DOT'].index)
df.shape
# Number of Unique Transactions
print("Number of transactions: ", df['InvoiceNo'].nunique())
print("Number of products bought: ", df['StockCode'].nunique())
print("Number of customers:", df['CustomerID'].nunique())
print('Number of countries: ', df['Country'].nunique())
```

```
ux=["Transactions", "Products Sold", "Customers"]
uy = [df['InvoiceNo'].nunique(), df['StockCode'].nunique(), df['CustomerID'].nunique()]
unique = pd.DataFrame(np.column_stack([ux, uy]), columns=['Variables', 'Unique Count'])
unique.head()

fig = plt.figure()
a = fig.add_axes([0,0,1,1])
a.set_ylabel('Unique Counts')
a.set_xlabel('Variables')
a.bar(ux,uy, color=['#4970D1', '#EE59A6', '#FF9048'])
plt.title('Distinct values of Numerical Variables', fontsize = 16)
plt.show()

#Exploring Top 5 Countries on the basis of Customers per Country:-
top_5_countries = pd.DataFrame(df.groupby('Country')['CustomerID'].nunique())
top_5_countries.columns = ['Customers_by_Country']
top_5_countries.sort_values('Customers_by_Country', inplace=True, ascending=False)
top_5_countries = top_5_countries[:5][:]

figure, axis = plt.subplots(figsize=(6,4),dpi=100)
axis=sns.barplot(x=top_5_countries.index, y=top_5_countries['Customers_by_Country'])
axis.set_xticklabels(axis.get_xticklabels(), rotation=0, fontsize=9, ha="center")
for i in axis.patches: axis.annotate("%.2f" % i.get_height(), (i.get_x() + i.get_width() / 2., i.get_height()),
                ha='center', va='center', fontsize=9, color='gray', xytext=(0, 4), textcoords='offset points')
plt.xlabel('Country')
plt.ylabel('Number of Customers')
plt.title('Customers per Country')
plt.show()

# Considering data for United Kingdom because it has maximum customers.
df_uk = df[df['Country'] =="United Kingdom"]

# Considering Top 5 StockCode on the basis of Count of Description
pro =
df_uk[['StockCode','Description']].groupby(['StockCode'])['Description'].size().nlargest(5).reset_index(name='Cou
nt')

fig, axis = plt.subplots(figsize=(6,4),dpi=100)
axis=sns.barplot(x=pro['StockCode'], y=pro['Count'])
axis.set_xticklabels(axis.get_xticklabels(), rotation=0, fontsize=9, ha="center")
plt.title("Top 5 products purchased")
plt.xlabel("Product Code")
plt.ylabel("Quantity Sold")

for i in axis.patches:
            axis.annotate("%.2f" % i.get_height(), (i.get_x() + i.get_width() / 2., i.get_height()),
                ha='center', va='center', fontsize=9, color='gray', xytext=(0, 4), textcoords='offset points')
plt.show()

df_uk['Date'], df_uk['Time'] = df_uk['InvoiceDate'].dt.normalize(), df_uk['InvoiceDate'].dt.time
# Adding 1 because we want to start days from Monday rather than Sunday.
df_uk.insert(loc=10, column='Day', value=(df_uk.InvoiceDate.dt.dayofweek)+1)
# Generating new feature Revenue by multiplying Quantity and Unit Price.
df_uk['Total_revenue'] = df_uk['Quantity'] * df_uk['UnitPrice']

# Aggregating Total Revenue generated each day.
```

```
df_revenue = df_uk.groupby(['Date'],as_index=False).agg({'Total_revenue': 'sum'})

# Converting into date format just to ensure that each instance is in date format.
df_revenue['Date'] = pd.to_datetime(df_revenue['Date'])
df_revenue['Date'] = pd.to_datetime(df_revenue['Date'], format="%Y-%m-%d")
df_revenue.set_index('Date', inplace=True)

# Time Series Plot.
df_revenue.plot(figsize = (15,6))
plt.xlabel('Date',fontsize = 13)
plt.ylabel('Total Revenue', fontsize = 13)
plt.xticks(rotation=0, ha='center')
plt.title('Revenue trend over the period of year 2011', fontsize = 16)
plt.show()

# Month wise purchase pattern of Customers.
dtc = (df_uk['Date'].dt.strftime("%b")).value_counts(sort=True)
plt.subplots(figsize=(10,6),dpi=100)
plt.bar(dtc.index, dtc)
plt.title("Month wise purchase pattern (Highest to Lowest)")
plt.ylabel("Number of Customers")
plt.xlabel("Month")
plt.show()

# Daywise purchase pattern of Customers.
axis = df_uk.groupby('InvoiceNo')['Day'].unique().value_counts().sort_index().plot(kind='bar', figsize=(10,6))
axis.set_xlabel('Day',fontsize=12)
axis.set_ylabel('Number of Orders',fontsize=12)
axis.set_title('Number of orders for different Days',fontsize=12)
axis.set_xticklabels(('Mon','Tue','Wed','Thur','Fri','Sun'), rotation='horizontal', fontsize=12)
plt.show()

# Converting Time Feature into 24 intervals of 1 hour each.
s = df_uk['InvoiceDate'].dt.floor('1H')
df_uk['Time'] = s.dt.strftime('%H:%M')
# Hourly Purchase pattern of Customers.
dtc = (df_uk['Time']).value_counts(sort=False).sort_index(ascending=True)
plt.subplots(figsize=(10,6),dpi=100)
plt.bar(dtc.index, dtc)
plt.title("Peak Load purchase pattern per year")
plt.ylabel("Number of Customers")
plt.xlabel("Hourly intervals")
plt.show()

# Importing libraries for Association Rules.
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules

# Dividing each Invoice on the basis of Description.
case_UK = (df_uk.groupby(['InvoiceNo',
'Description'])['Quantity'].sum().unstack().reset_index().fillna(0).set_index('InvoiceNo'))

# One Hot Encoding.
def OHE(x):
    while(x<=0): return 0
    while(x>=1): return 1
```

```
case_encoded = case_UK.applymap(OHE)
case_UK = case_encoded

# Building the asociation model with Minimum Support.
support_items = apriori(case_UK, min_support = 0.01, use_colnames = True)

# Itemsets in the descending order based on their support.
support_items.sort_values(by=['support'], ascending=False)

# Applying Association Rule and sorting in descending order on the basis of "Confidence" and "Lift".
associationRule = association_rules(support_items, metric ="lift", min_threshold = 1)
associationRule = associationRule.sort_values(['confidence', 'lift'], ascending =[False, False])

# Importing libraries for KMeans and PCA.
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

# Performing RFM analysis for Customer Segmentation.
dfRecency = df_uk.groupby('CustomerID', as_index=False)['Date'].max()
dfRecency.columns = ['CustomerID', 'RecentBuyDate']
dfRecency.head()

# Calculating Recency by Feature Engineering using Recent Buy Date.
dfRecency['RecentBuyDate'] = pd.to_datetime(dfRecency['RecentBuyDate']).dt.date
dfRecency['Recency'] = dfRecency['RecentBuyDate'].apply(lambda current: (dt.date(2011,12,9) - current).days)
dfRecency.drop('RecentBuyDate', axis=1, inplace=True)

# Sorting Recency in Descending Order.
dfRecency.sort_values(by='Recency', ascending=False).head()

# Check and drop any dupicates for "Invoice No." and "Customer ID" if present.
dfFreqUk = pd.DataFrame(df_uk)
dfFreqUk.drop_duplicates(subset=['InvoiceNo', 'CustomerID'], inplace=True)

# Calculating Frequency by Feature Engineering using Customer ID.
dfFrequency = dfFreqUk.groupby('CustomerID', as_index=False)['InvoiceNo'].count()
dfFrequency.columns = ['CustomerID','Frequency']

# Calculating Monetary by Feature Engineering using Total Revenue.
dfMonetary = df_uk.groupby('CustomerID', as_index=False).agg({'Total_revenue': 'sum'})
dfMonetary.columns = ['CustomerID', 'Monetary']

# Grouping "Recency", "Frequency", "Monetary"
dfMerge1 = dfRecency.merge(dfFrequency, on='CustomerID')
dfMerge = dfMerge1.merge(dfMonetary,on='CustomerID')
dfMerge.set_index('CustomerID',inplace=True)

# Making Quantiles
classOfValues = dfMerge.quantile(q=[0.25,0.5,0.75])
classOfValues

# Generating RQuartile, FQuartile and MQuartile and assigning values- (1,2,3,4)
# Value is denoted by a, recency denoted by r, quartiles denoted by x
```

```
def RecencyQuartile(a,r,x):
    if a <= x[r][0.25]: return 4
    elif a <= x[r][0.50]: return 3
    elif a <= x[r][0.75]: return 2
    else: return 1
# Value is denoted by a, Frequency-Monetary denoted by m, quartiles denoted by x
def FrequencyMonetaryQuartile(a,m,x):
    if a <= x[m][0.25]: return 1
    elif a <= x[m][0.50]: return 2
    elif a <= x[m][0.75]: return 3
    else: return 4

# Build RFM Segmentation Table.
dfSeg = pd.DataFrame(dfMerge)
dfSeg['RQuartile'] = dfSeg['Recency'].apply(RecencyQuartile, args=('Recency',classOfValues))
dfSeg['FQuartile'] = dfSeg['Frequency'].apply(FrequencyMonetaryQuartile, args=('Frequency',classOfValues))
dfSeg['MQuartile'] = dfSeg['Monetary'].apply(FrequencyMonetaryQuartile, args=('Monetary',classOfValues))

# Grouping RQuartile, FQuartile and MQuartile to generate new feature Group which provides insights
regarding customer behaviour.
dfSeg['Group'] = dfSeg.RQuartile.map(str) + dfSeg.FQuartile.map(str) + dfSeg.MQuartile.map(str)

# Converting Group Variable into Score so that Intervals can be formed.
dfSeg['Score'] = dfSeg[['RQuartile', 'FQuartile', 'MQuartile']].sum(axis = 1)

# Distribution plot for Recency, Frequency and Monetary to visualize each generated feature.
plt.figure(figsize=(16,4))
plt.subplot(1, 3, 1)
#Distribution plot for Recency
axis = sns.distplot(dfSeg['Recency'], color = 'red')
plt.title("Distribution plot for Recency")
plt.subplot(1, 3, 2)
#Distribution plot for Frequency
axis = sns.distplot(dfSeg['Frequency'], color='limegreen')
plt.title("Distribution plot for Frequency")
plt.subplot(1, 3, 3)
#Distribution plot for Monetary
axis = sns.distplot(dfSeg['Monetary'], color = 'blue')
plt.title("Distribution plot for Monetary")
plt.show()

# Logarithmic transformation for Recency, Frequency and Monetary features.
# Adding delta(0.05) component for zero and negative values to handle infinite numbers generated during log
transformation as Log(0)=undefined
rfm_r_log = np.log(dfSeg['Recency']+0.05)
rfm_f_log = np.log(dfSeg['Frequency']+0.001)
rfm_m_log = np.log(dfSeg['Monetary']+0.05)
dfLog = pd.DataFrame({'Monetary': rfm_m_log, 'Recency': rfm_r_log, 'Frequency': rfm_f_log})

# As recency can never be negative so ensuring that condition.
dfLog = dfLog.drop(dfLog[(dfLog["Recency"] < 0)].index)

# Correlation Plot for RFM values without and with log transformation.
plt.figure(figsize=(16,4))
plt.subplot(1, 2, 1)
sns.heatmap(dfSeg[['Recency', 'Frequency', 'Monetary']].corr(), annot = True, cmap="BuPu")
```

```
plt.yticks(va="center")
plt.title("Correlation Plot of RFM")
plt.subplot(1, 2, 2)
sns.heatmap(dfLog.corr(), annot = True, cmap="BuPu")
plt.yticks(va="center")
plt.title("Correlation Plot of log values of RFM")
plt.show()


# Segmenting customers on the basis of Loyalty Class by generating intervals from Score variable.
dfSeg['Interval'] = pd.cut(x=dfSeg['Score'], bins=[0, 3, 6, 10, 12])
dfSeg['LoyaltyClass'] = pd.cut(x=dfSeg['Score'], bins=[0, 3, 6, 10, 12], labels=['Bronze', 'Silver', 'Gold', 
'Platinum'])


dfSegB = pd.DataFrame(dfSeg)
dfSegB.reset_index(drop=True, inplace=True)
dfSegBox = dfMerge1.merge(dfMonetary,on='CustomerID')
dfSegBox.set_index('CustomerID',inplace=False)
dfSegBox.reset_index(drop=True, inplace=True)
dfSegBoxCon = pd.concat([dfSegBox, dfSeg.iloc[:,-1]], axis=1)

# Number of customers in each Loyalty class.
axis = sns.countplot(x="LoyaltyClass", data=dfSegBoxCon)

for i in axis.patches:
    h = i.get_height()
    axis.text(i.get_x()+i.get_width()/2.,h+10,
            '{:1.2f}%'.format(h/len(dfSegBoxCon)*100),ha="center",fontsize=10)
plt.xlabel('Loyalty classes')
plt.ylabel('Number of Customers')
plt.title('Number of Customers in each Cluster')
plt.show()

# Revenue generated from each cluster , and taking revenue less than 7000 for visualization purpose.
z = dfSegBoxCon[dfSegBoxCon["Monetary"]<7000]
sns.boxplot(x="LoyaltyClass", y="Monetary", data=z)
plt.xlabel('Loyalty classes')
plt.ylabel('Revenue')
plt.title('Monetary value for each Cluster')
plt.show()

len(dfSegBoxCon)

# Using values of revenue less than 7000, instances decreases by 117 so just for visualization in boxplot it won't 
make much difference.
len(z)


# Distribution plot for log transformed values of Recency, Frequency and Monetary.
plt.figure(figsize=(16,4))
plt.subplot(1, 3, 1)
#Distribution plot for Recency
axis = sns.distplot(dfLog['Recency'], color = 'red')
plt.title("Distribution plot for Recency")
plt.subplot(1, 3, 2)
#Distribution plot for Frequency
axis = sns.distplot(dfLog['Frequency'], color='limegreen')
plt.title("Distribution plot for Frequency")
```

```python
plt.subplot(1, 3, 3)
#Distribution plot for Monetary
axis = sns.distplot(dfLog['Monetary'], color = 'blue')
plt.title("Distribution plot for Monetary")
plt.show()

# Scatter plot for Frequency and Recency of Customers.
plt.figure(figsize=(10,6))
plt.scatter(dfLog["Recency"],dfLog["Frequency"])
plt.xlabel("Recency")
plt.ylabel("Frequency")
plt.title("Relationship between Recency and Frequency of the Customers")
plt.show()

# Scaling the data to achieve normalized values.
s = StandardScaler()
segm_std = s.fit_transform(dfLog)

pca = PCA()
pca.fit(segm_std)

# Obtaining Explained variance for 3 factors.
pca.explained_variance_ratio_

# Plotting the explained variance by components to get number of components.
plt.figure(figsize = (10,6))
plt.plot(range(1,4),pca.explained_variance_ratio_.cumsum(), marker = 'o', linestyle = "dashdot")
plt.title("Explained Variance by Components")
plt.xlabel("Number of Components")
plt.ylabel("Cumulative Explained Variance")
plt.show()

pca = PCA(n_components=2)

pca.fit(segm_std)

# Obtaining PCA scores.
scoresPCA = pca.transform(segm_std)

# Using PCA scores in K Means algorithm.
SS = []
for i in range(1,7):
  model = KMeans(n_clusters=i,init='k-means++',random_state=30)
  model.fit(scoresPCA)
  SS.append(model.inertia_)

# Obtaining Silhoutte_Score to know number of clusters.
from sklearn.metrics import silhouette_score
for n_clusters in range(2,7):
    km = KMeans(init='k-means++', n_clusters = n_clusters, n_init=100)
    km.fit(dfLog)
    clusters = km.predict(dfLog)
    silhouette_avg = silhouette_score(dfLog, clusters)
    print("For n_clusters =", n_clusters, "The average silhouette_score is :", silhouette_avg)
```

```
# Using Elbow method to find number of clusters.
plt.figure(figsize = (10,6))
plt.plot(range(1,7), SS, marker ='o', linestyle ='dashdot')
plt.xlabel("Number of Clusters")
plt.ylabel("Sum of Squared Errors")
plt.title("K-Means with PCA")
plt.show()

model = KMeans(n_clusters=2, init='k-means++', random_state=30)

# Fitting Kmeans model on the basis of PCA scores.
model.fit(scoresPCA)

dfPCA = pd.concat([dfLog.reset_index(drop=True),pd.DataFrame(scoresPCA)],axis=1)
dfPCA.columns.values[-2:] = ["Component-1","Component-2"]
dfPCA["ClassPCA"] = model.labels_

# Plotting 2 clusters formed by PCA.
plt.figure(figsize = (10,6))
sns.scatterplot(dfPCA["Component-1"], dfPCA["Component-2"], hue = dfPCA["ClassPCA"], palette=['r','y'])
plt.title("Clusters by PCA Components")
plt.show()

# Generating new dataframe to divide revenue generated by each class.
dfPCA_ = pd.DataFrame(dfPCA)

# Converting logarithmic values of monetary to numeric values.
dfPCA_['Monetary'] = np.exp(dfPCA_["Monetary"])

# Obtaining sum of revenue and percentage of revenue for each class.
dfPCA_ = dfPCA_.groupby('ClassPCA',as_index=False).agg({'Monetary': 'sum'})
dfPCA_['Percentage'] = round(dfPCA_['Monetary']/sum(dfPCA_['Monetary'])*100,2)

# Plotting revenue for each PCA component using barplot.
axis = sns.barplot(x="ClassPCA", y="Percentage", data=dfPCA_)
for i in axis.patches:
        axis.annotate("%.2f" % i.get_height(), (i.get_x() + i.get_width() / 2., i.get_height()),
           ha='center', va='center', fontsize=11, color='black', xytext=(0, 5),
           textcoords='offset points')
plt.xlabel('PCA Components (Clusters)')
plt.ylabel('Revenue %')
plt.title('Percentage Revenue from each Cluster')
plt.show()

# Importing libraries for Classification.
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import balanced_accuracy_score
import warnings
warnings.filterwarnings("ignore")

# Assigning "Recency", "Frequency", "Monetary" in X , while "Loyalty Class" in y.
X = dfSeg[["Recency", "Frequency", "Monetary"]]
```

```
y = dfSeg["LoyaltyClass"]

# Splitting into train and test data.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0, shuffle=False)

# Model 1 :- Logistic Regression
model_1 = LogisticRegression()
model_1.fit(X_train, y_train)

y_pred_1 = model_1.predict(X_test)

# Confusion Matrix for Logistic Regression
from sklearn.metrics import confusion_matrix, classification_report
confusionMatrix = confusion_matrix(y_test, y_pred_1)
print(confusionMatrix)

# Plotting Confusion Matrix for logistic regression.
from mlxtend.plotting import plot_confusion_matrix
matrix = np.array(confusionMatrix)

classNames = [{'Platinum' : 0, 'Gold': 1, 'Silver': 2, 'Bronze': 3}]

fig, ax = plot_confusion_matrix(conf_mat = matrix,
                                colorbar = True,
                                show_absolute = False,
                                show_normed = True,
                                cmap="YlGnBu") #class_names = classNames

plt.show()

# Accuracy and Balanced Accuracy for Logistics Regression.
print('Accuracy         : {}%'.format(round(metrics.accuracy_score(y_test, y_pred_1)*100,2)))
print('Balanced Accuracy: {}%'.format(round(balanced_accuracy_score(y_test, y_pred_1)*100,2)))
print(classification_report(y_test, y_pred_1))

# Importing libraries for Decision Trees
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import metrics

# Model 2 :- Decision Trees
from sklearn.metrics import accuracy_score, classification_report
model_2 = DecisionTreeClassifier(min_samples_split=100)
model_2.fit(X_train, y_train)
y_pred_2 = model_2.predict(X_test)

# Confusion Matrix for Decision Tree
from sklearn.metrics import confusion_matrix, classification_report
confusionMatrix = confusion_matrix(y_test, y_pred_2)
print(confusionMatrix)

# Plotting Decision Tree.
tree.plot_tree(model_2)
import graphviz
treeData = tree.export_graphviz(model_2, out_file=None)
network = graphviz.Source(treeData)
```

```
network.render("Retail")

# Accuracy and Balanced Accuracy for Decision Tree.
print('Accuracy          : ',round(accuracy_score(y_test, y_pred_2)*100,2),'%')
print('Balanced Accuracy: ',round(balanced_accuracy_score(y_test, y_pred_2)*100,2),'%')
print(classification_report(y_test, y_pred_2))


# Importing Libraries for Naive Bayes.
from sklearn.naive_bayes import MultinomialNB
# Model 3 :- Naive Bayes
model_3 = MultinomialNB()
model_3.fit(X_train,y_train)
y_pred_3 = model_3.predict(X_test)
# Accuracy and Balanced Accuracy for Naive Bayes.
print('Accuracy          : ',round(accuracy_score(y_pred_3,y_test)*100,2),'%')
print('Balanced Accuracy: ',round(balanced_accuracy_score(y_pred_3, y_test)*100,2),'%')


# Confusion Matrix for Decision Tree.
from sklearn.metrics import confusion_matrix, classification_report
confusionMatrix = confusion_matrix(y_test, y_pred_3)
print(confusionMatrix)
print(classification_report(y_test, y_pred_3))


# Importing Libraries for KNN.
from sklearn.neighbors import KNeighborsClassifier
# Model 4 :- K Nearest Neighbour.
model_4 = KNeighborsClassifier(n_neighbors=3)
model_4.fit(X_train,y_train)
y_pred_4 = model_4.predict(X_test)
# Accuracy and Balanced Accuracy for KNN.
print('Accuracy          : ',round(accuracy_score(y_pred_4,y_test)*100,2),'%')
print('Balanced Accuracy: ',round(balanced_accuracy_score(y_pred_4,y_test)*100,2),'%')


# Confusion Matrix for KNN.
from sklearn.metrics import confusion_matrix, classification_report
confusionMatrix = confusion_matrix(y_test, y_pred_4)
print(confusionMatrix)

print(classification_report(y_test, y_pred_4))

# Importing Libraries for Random Forest.
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
# Model 5 :- Random Forest
model_5 = RandomForestClassifier(max_depth = 3, random_state=0)
model_5.fit(X_train, y_train)
y_pred_5 = model_5.predict(X_test)
# Accuracy and Balanced Accuracy for Random Forest.
print('Accuracy          : ',round(accuracy_score(y_pred_5, y_test)*100,2),'%')
print('Balanced Accuracy: ',round(balanced_accuracy_score(y_pred_5, y_test)*100,2),'%')


# Confusion Matrix for Random Forest.
from sklearn.metrics import confusion_matrix, classification_report
confusionMatrix = confusion_matrix(y_test, y_pred_5)
print(confusionMatrix)
print(classification_report(y_test, y_pred_5))

# Model Comparison on the basis of Accuracy and Balanced Accuracy of each model.
models = {'Classification Model': ['Logistic Regression','Decision Trees','Naive Bayes','K-Nearest Neighbor',
'Random Forest'],
        'Accuracy': [round(accuracy_score(y_pred_1,y_test)*100,2),
                    round(accuracy_score(y_pred_2,y_test)*100,2),
                    round(accuracy_score(y_pred_3,y_test)*100,2),
                    round(accuracy_score(y_pred_4,y_test)*100,2),
                    round(accuracy_score(y_pred_5,y_test)*100,2)],
        'Balanced Accuracy': [round(balanced_accuracy_score(y_pred_1,y_test)*100,2),
                    round(balanced_accuracy_score(y_pred_2,y_test)*100,2),
                    round(balanced_accuracy_score(y_pred_3,y_test)*100,2),
                    round(balanced_accuracy_score(y_pred_4,y_test)*100,2),
                    round(balanced_accuracy_score(y_pred_5,y_test)*100,2)]
        }
models_df = pd.DataFrame(models, columns = ['Classification Model', 'Accuracy', 'Balanced Accuracy'])

models_df

# Model Comparison Plot based on Balanced Accuracy
plt.figure(figsize=(10,6))
axis = sns.barplot(x=models_df['Balanced Accuracy'], y=models_df['Classification Model'], data=models_df,
orient='h', saturation=0.8)
axis.axes.set_title("Classification model comparison", fontsize=16)
axis.set_yticklabels(models_df['Classification Model'], fontsize=13)
axis.set_xticklabels([0, 20, 40, 60, 80, 100], fontsize=13)
plt.xlabel("Balanced Accuracy", fontsize = 13)
plt.ylabel("", fontsize = 10)
plt.subplots_adjust(left=1, right=1.8, top=2.4, bottom=1.8)

for i, p in enumerate(axis.patches):
    axis.annotate("%.2f %%" % (p.get_width()),
            (p.get_x() + p.get_width(), p.get_y() + 0.7),
            xytext=(-48, 10), textcoords='offset points')

plt.show()

# End of File
```