DATA 228 – GROUP PROJECT

CUSTOMER LIFETIME VALUE PREDICTION

Group 4

Gayathri Sundareshwar

Keerthana Gopikrishnan

Razan Dababo

Acknowledgement

Working on research like this helps to prepare us and give us the skills to thrive in the work industry. Hence, this work can be considered as an excellent opportunity. We would like to start by acknowledging our gratitude to Professor Ming-Hwa Wang who gave us this opportunity.

In addition, we would like to extend our heartfelt thanks to all the contributors listed in the bibliography. Last but certainly not the least, we would like to thank our parents who gave us the wings to fly and also the members of this team for their timely contribution and continued support.

Abstract

Customer lifetime value (CLV) for a business is the net profit or loss to the business from an individual customer over the entire lifetime of transactions of that customer with the business. Hence, CLV is the net of the revenues generated from that customer over their lifetime of transactions minus the cost of acquiring, selling, and servicing that customer, taking into consideration the time value of money. CLV is becoming recognized as a key measure of the worth of a customer, as it considers not only the customer's present value but also the expected value over their projected lifetime with the business. CLV models allow companies to make data-driven decisions in areas such as customer segmentation to sustain long-term relationships with clients, resource allocation and formulation of marketing strategies and investments, evaluation of marketing campaigns and strategies, retention and acquisition of customers, and estimation of company value. In this research, we will be implementing and evaluating supervised and unsupervised machine learning techniques to predict CLV using a dataset of customer purchases belonging to an online retailer based in the United Kingdom (UK).

List of Tables & Figures

Table of Contents

2 Introduction

2.1 Objective

The objective of this research is to build a model to predict customer lifetime value (CLV) and evaluate its reliability. Our research can be useful for businesses looking for strategies to improve future profits from a given customer.

2.2 What is the Problem

The business world is more data driven than ever. Companies are constantly coming up with new strategies to retain customers. It is a proven statistic that repeat (or loyal) customers spend 67% more than new customers. Retaining customers by building a strong relationship between the customer and firm is a crucial factor in developing a successful business. Since competition is high and there are a wide variety of options available for customers for each product, companies need to stay on top of market trends and rapidly come up with new strategies to retain their customers.

Companies often spend a lot to acquire and retain customers. Companies need to know if the expenses for acquiring and retaining such customers are justified, and also, they must identify which customers are most valuable to the firm in terms of profit. Firms also must analyze customer loyalty, because loyalty plays a major role in repeat customers' purchase decisions; otherwise, companies could be wasting money on relatively unprofitable customers. Without a proper method to predict the worth of a customer to the company over a period of time, it is difficult to compute the profit acquired through that customer. In order to have a successful business model, predicting the future profits made through a customer is vital.

2.3 How Our Research Is Related to Big Data

This research is related to big data in several ways. Firstly, we aim to predict CLV using machine learning models and Application Programing Interface (API) that are suitable for very large datasets. Secondly, we will be using a large dataset to identify patterns and anomalies within the data, which in turn will help us make accurate predictions. The core objective of this research is to gain advanced knowledge of machine learning through implementation. Accordingly, we will use classification and regression to predict CLV using an online retail purchase history dataset containing more than a million datapoints.

2.4 Why Other Approaches Are Not Ideal

Some of the existing CLV models makes use of any one selected machine learning algorithm. Sometimes that algorithm might hold true but other times it may not. Among all CLV classification techniques, a number of them have restrictions and preference in attribute value types and the size of attributes. Further, simply applying one learning model on a particular dataset usually will not yield the best results.

2.5 Why Our Approach Is Better

In our research, instead of tackling the problem using a single approach, we will build machine learning model to increase the accuracy of the CLV prediction using supervised and unsupervised algorithms (see Section 3.2) on the dataset in order to attain better accuracy in the predicting CLV. After comparing the results among all applied models under each approach (supervised and unsupervised), the best performing one will be chosen as the final model. This approach enables us to add more weight to the features that are important in the dataset, so that the model can be better trained better for prediction.

2.6 Statement of the Problem

To achieve utmost accuracy in predicting CLV, an improved algorithm is required. The models and the algorithms used must be capable of addressing typical problems involved with both prediction and reliability of CLV. Therefore, it is important for the model to be consistent in order to attain higher accuracy.

2.7 Area of Investigation

In our research, the area of investigations is the intersection of big data technology and retail, with attention given to the following aspects:

(a) The features used on our model are based on customer and sales data, which can be easily obtained from a company's CRM (Customer Relationship Management) or ERP (Enterprise Resource Planning) System.

(b) Apart from CLV prediction, our research can be applied to similar problems such as forecasting the number of purchases.

(c) Classification and regression models in the field of machine learning and statistics are a powerful tool for gathering actionable insights in the world of retail.

3 Theoretical Basis and Literature Review

3.1 Definition of the Problem

Measuring customer lifetime value is a highly complex and difficult task. A lot of different factors play a role in converting input data into quantifiable – or machine-readable – format. Information from customer reviews, demographics, search logs of products and interaction to related pages are some of the key performance indicators that can offer a first gateway to understanding CLV. Modern machine learning algorithms offer tools where the data can be used to predict customers' purchase decisions and in turn can be helpful in the improving CLV predictions. In this research, we are building a machine learning model using supervised and unsupervised algorithms to predict CLV. The novelty of the work is that this approach leverages both supervised and unsupervised learning results. This in turn could work as a blueprint for similar tasks in this domain.

3.1.1 What is CLV and Why is it important?

An important aspect of any successful business is its ability to retain existing customers and attract new ones. Repeat customers tend to spend 67% more than new customers. CLV is a business metric that is extremely useful to businesses in helping them understand and quantify the worth of an individual customer. CLV can be summarized as the profit a business can expect to make from an individual customer over the lifetime of the business-customer relationship. Predicting this can help a company allocate resources and strategize marketing campaigns tailored to different types of customers. Figure 1 depicts the process involved in prediction of CLV through the traditional approach.

Predicting CLV helps businesses achieve the following, to name a few:

1. Increase overall profit.
2. Retain customers and increase customer loyalty.
3. Maintain steady cash flow.
4. Target the right customer base.

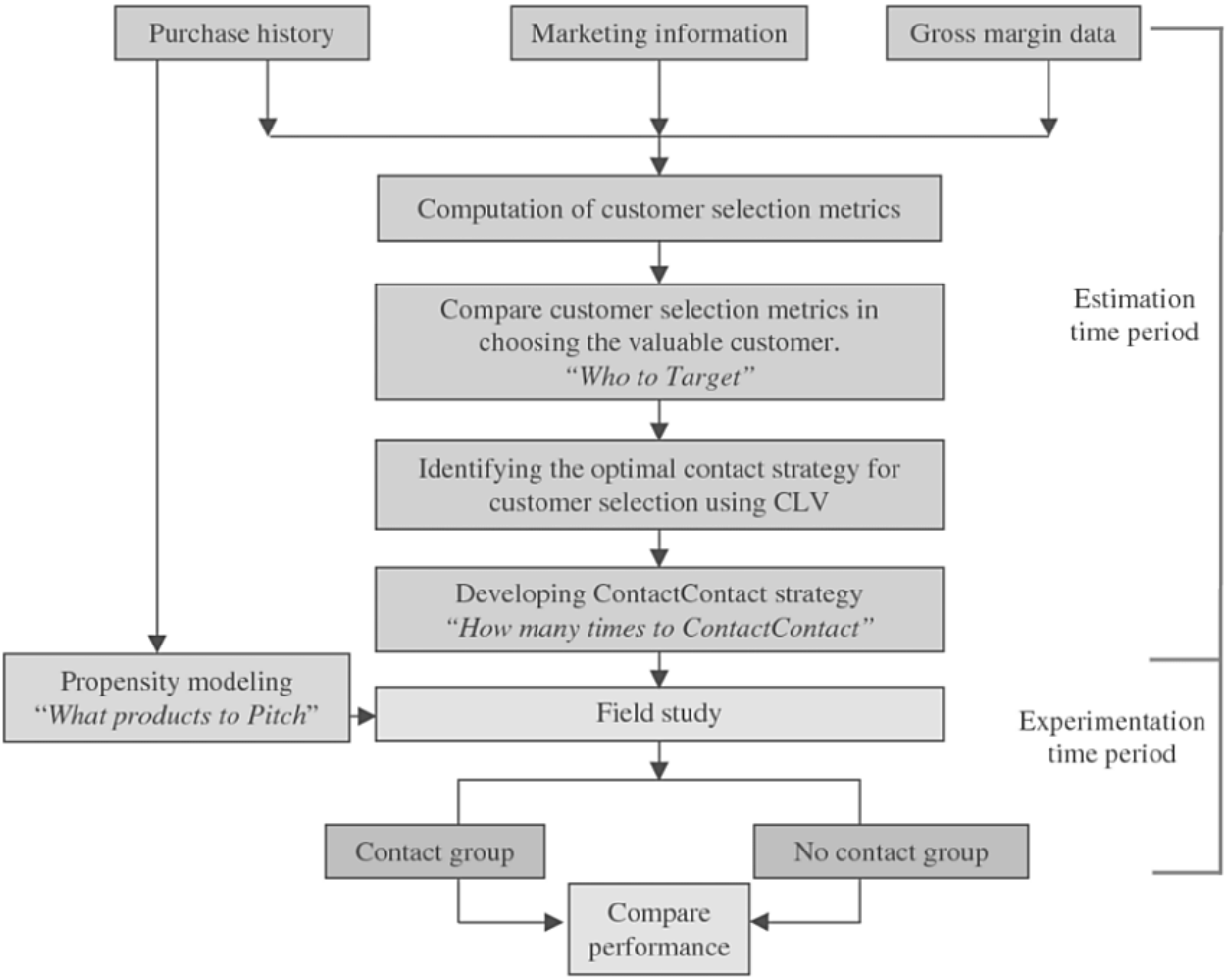

Fig 1 Theoretical approach of CLV

3.2 Theoretical Background of the Problem

Every customer is unique in a way. No two customers and their purchase decisions are the same. Profitability from customers varies based on the preference of customers, their loyalty, and their financial strength. Without this basic information about their customers, businesses cannot make reliable decisions.

The most important benefit that can be attained from predicting CLV is that it can take full advantage of the prevalent information and identify the real sources of profit. To attain more accuracy, precise and sufficient information about customers is indispensable. It has been claimed by many researchers that, traditional marketing metrics, such as brand awareness/attitude and market share, are not adequate to evaluate returns on marketing investments. In this case, the assessment of CLV makes it possible to link long-term financial returns explicitly to marketing actions.

Machine learning algorithms can be used to arrive at a solution for this. Machine learning differentiates between two classification problems. In some cases, the data set contains unstructured data and machine learning is used to find hidden patterns in the data by grouping into clusters or classes. This is called unsupervised learning. In other cases, the classes or labels are predefined, and the task is to search for relationships or rules to distinguish one class from another. This is known as supervised learning.

3.2.1 Calculating Customer Lifetime Value

CLV is a measure of the profit streams generated by a customer across the entire customer life cycle. On the surface, it may seem that measuring customer profitability is a simple and

straightforward process; however, it is quite complex and involves many variables. CLV calculation involves the following:

(a) Historical data tracking a customer's buying pattern;

(b) Historical data log of customer searches and related browser click log;

(c) Knowledge of the statistical techniques to best forecast and model customer behavior;

(d) Ability to identify spending frequency, monetary value, and average time period that the customer stays loyal to the company;

(e) Ability to fully understand the limitations and assumptions built into the model.

Multiple models have been developed for determining CLV in the past. Each of those models has different assumptions and basis. From a theoretical standpoint, these models can be classified into three broad categories:

(a) Scoring model

(b) Econometric model

(c) Probability model

In the Scoring model, simple scores are created based on consumers' purchase characteristics (e.g., the recency, frequency, and monetary value – or RFM – model). In RFM prediction model , the customers are categorized into groups based on the quantiles of recency, frequency, and monetary benefits. These attributes are then used to predict the final profit for each customer by using the past average profits of the other customers in the same group.

In the probability model, consumer behavior is viewed as the expression of an underlying stochastic process that is determined by individual characteristics (e.g., the negative binomial

distribution, or NBD, model). NBD-based approaches have the advantage of being principled and intuitive. They are particularly accurate when the specific distributional assumptions are correct or approximately satisfied and when the CLV is not influenced by other hidden variables to a large extent.

In econometric models, consumer behavior is explained as a function of a set of covariates. However, this model differs in that it calculates CLV by projecting the profit to be made from a customer over a period by the company, then calculating the present value of that stream of cash flows.

A general formula for calculating customer lifetime value is as follows:

$$CPi = \sum_{t=1}^{T} \frac{\sum_{j=1}^{Ji} (pijt - cijt) - \sum_{k=1}^{Ki} mcikt}{(1 + r)t}$$

Fig 2. CLV Formula

The variables in the formula depicted in fig 2 are defined below:

(a) CPi is the profit of customer i to a firm;

(b) pijt is the price of purchase j made by customer i in period t;

(c) cijt is the unit cost of purchase j made by customer i in period t;

(d) mcikt is variable marketing cost, k, for customer j in period t; and

(e) r is the discount rate for money established to reflect the riskiness of cash flow.

3.3 Related Research to Solve the Problem

Multiple papers published by the Institution of Electrical and Electronics Engineers (IEEE) and Association for Computer Machinery (ACM) were reviewed while trying to come up with the ideal solution for predicting CLV. The most prominent papers are listed below:

(a) *"Hybrid machine learning models for predicting types of Human T-cell Lymphotropic Virus"* by Gaurav Sharma, Prashant Singh Rana and Seema Bawa provided insights on hybrid machine learning and how it could be used to the advantage of this project.

(b) "*Predicting Subscriber Dissatisfaction and Improving Retention in the Wireless Telecommunications Industry*" by Michael C. Mozer, Richard Wolniewicz, David B. Grimes, Eric Johnson, and Howard Kaushansky helped in understanding the real-world problem of retaining customers in wireless telecommunication industries and the approaches used to increase profit.

(c) "*Improved Customer Lifetime Value Prediction With Sequence-To-Sequence Learning and Feature-Based Models*" by Josef Bauer and Dietmar Jannach provided insights on different existing CLV approaches along with their advantages and disadvantages.

3.4 Advantages and Disadvantages of Previous Research

Researching and reading through multiple existing papers provided clarity on the approach to be taken to attain higher CLV prediction accuracy. A common theme among the research papers reviewed was they tended to focus more on implementation of machine learning models and their

performance. The papers helped us in understanding and differentiating between the feasible and non-feasible approaches

Some of the advantages noted from referring to the listed papers as well as other additional papers are summarized below:

(a) The hybrid model helped us in learning more about the hybrid machine learning and how it can be useful in increasing the accuracy of prediction.

(b) Though the cost of acquiring customers has gone down, it is still more beneficial to retain existing customers since the statistics prove that firms can earn more profit through loyal repeat customers compared to newly acquired ones.

(c) Extensive knowledge on different existing models and their drawbacks was gathered from these papers. This in turn, helped in paving the way to arrive at the best way to implement our research.

Some of the disadvantages of previous research are summarized below:

(a) Most of the previous research rely on one specific model without taking other possible/optimal models into consideration.

(b) Outliers in customer buying patterns were not taken in as a special factor when computing the customer lifetime value.

(c) Most of the assumptions are made based on the prices offered by the firm itself; competitor prices were not taken into consideration, even though they play a major role in customer's purchase behavior.

(d) Most of the models led to overfitting when the input was a smaller set of historical data.

3.5 Our Proposed Solution to Solve the Problem

Our solution is to implement supervised and unsupervised algorithms with an online retail dataset as input. We will then evaluate the performance of both models using the CLV formula depicted in Figure 2. In addition to these models, other analyses will be conducted to assess the relationship between all the features and how they can be used to improve efficiency and accuracy.

3.6 How Our Solution Differs from Others

Unlike the existing methods, where a single approach is taken to predict CLV, our approach involves using two different models to predict CLV then choosing the best-performing model among the two.

3.7 Why Our Solution Is Better

Using our approach, both structured and unstructured data can be used as input factors for predicting CLV. The first model, which is supervised learning, will use structured data to predict CLV, whereas the unstructured data will be passed through the unsupervised learning model to make predictions. Hence, the utmost usage of gathered historical data is guaranteed.

4 Hypothesis

The hypothesis for this project is to develop models based on supervised algorithms (such as support vector machine, logistic regression, decision tree classifier and random forest) and also unsupervised algorithms (such as Gaussian Mixture model and k-means clustering) to achieve better CLV prediction than the traditional models.

We will test and evaluate each model then select the best-performing model as the final model.

In the case that our models do not achieve optimal performance, we would like to evaluate what decisions led to the suboptimal results and further explore any improvements or enhancements that could be made to the models.

5 Methodology

5.1 How to Generate and Collect Input Data

To implement our research, we used a large dataset, "Online Retail Data Set," provided by UCI Machine Learning Repository (Center for Machine Learning and Intelligent Systems). We downloaded the dataset folders and description. Dataset contains all customer purchase transactions occurring between 01/12/2010 and 09/12/2011 for a United Kingdom-based online retailer.

The data consists of information on purchases made by each customer, such as price, quantity, description of product and country they live in. Table 1 shows the input features we will be using to obtain different metrics that are used in calculation of CLV.

S.No	Attributes	Description
1	InvoiceNO	Helps us to count the number of time transaction performed(frequency)
2	StockCode	Unique Code given to identify the products
3	Description	Explanation of the product
4	Quantity	Purchased item units in each transaction
5	InvoiceDate	Help us to calculate numbers of days customer stayed with the product
6	UnitPrice	Each unit purchased by the customer will help us to calculate the total purchased amount
7	CustomerID	Uniquely define your customers
8	Country	The country from which the order was placed.

Table 1: Input features description

5.2 How to Solve the Problem

There are many factors that influence a customer's retention, such as demographics (age, geography), behavior (Recency, Purchase Frequency, Monetary Contribution).

We can perform operations to obtain such metrics:

(a) Calculate total price of purchase for each customer.

(b) Calculate the number of orders for each customer.

(c) Calculate the number of days between the present date and the date of last purchase for each customer.

(d) Calculate the number of repeat purchases for each customer.

We can then use these metrics to estimate CLV for each customer. The traditional methods to predict CLV are effective, but there are always new ways to improve prediction.

5.2.1 Algorithm Design

Some of the basic steps involved in Algorithm Design are listed below: The most optimal approach will be decided after a thorough inspection of data and gaining better insights.

Data Preprocessing

Data preprocessing, also called feature engineering (defining the features such as recency , frequency, monetary benefit) , is required in order to build a better model with good prediction.

Data Wrangling

The data set is cleaned so that the model can yield good predictions. We will either remove the missing records or fill it with 0 in case if the column is atomic and no other values

are influenced by this column– the method is chosen based on the feature type and how important the feature is in the model. We can also use the method of removing missing values in 2 cases, first if the proportion of a feature's missing values exceeds 97%, since there would not be enough information to determine the influence of the prediction. And second, if we deem the records to be untouched through manipulation. It is always advised to remove records in such cases instead of tampering with it.

We also need to remove outliers that are extreme values of the attributes which might arise due to special reasons.

Removing Outliers

We also need to remove abnormal data points such as extreme value of the house due to special reasons.

5.2.2 Building Models

To improve the prediction of CLV, we propose using Supervised Algorithm and Unsupervised Algorithms.

Supervised Algorithm

Supervised machine learning algorithm uses labeled datasets which are passed as input to train algorithm that classify data and predict outcomes accurately. This algorithm measures accuracy through the loss function, adjusting until the error has been sufficiently minimized.

We will be using several supervised algorithms such as:

1.) Support Vector Machine (SVM)

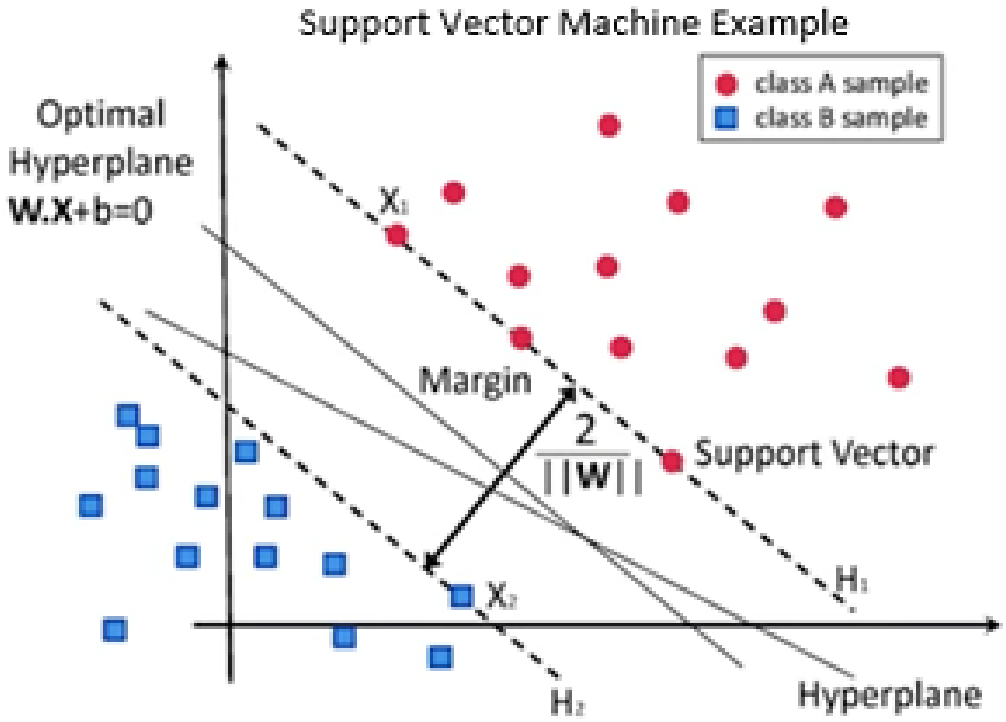

Fig 3. SVM Example

We will be implementing the principles of statistical learning theory. The complexity of the data will account to using linear and non-linear implementation. SVM tends to have high accuracy and works well in complicated domains with clear margin separation.

A hyper-plan is constructed to differentiate data points. To get better result, testing for linear SVM uses the penalty parameter. Figure 3 illustrates the basic algorithmic construct of SVM.

2.) Logistic Regression

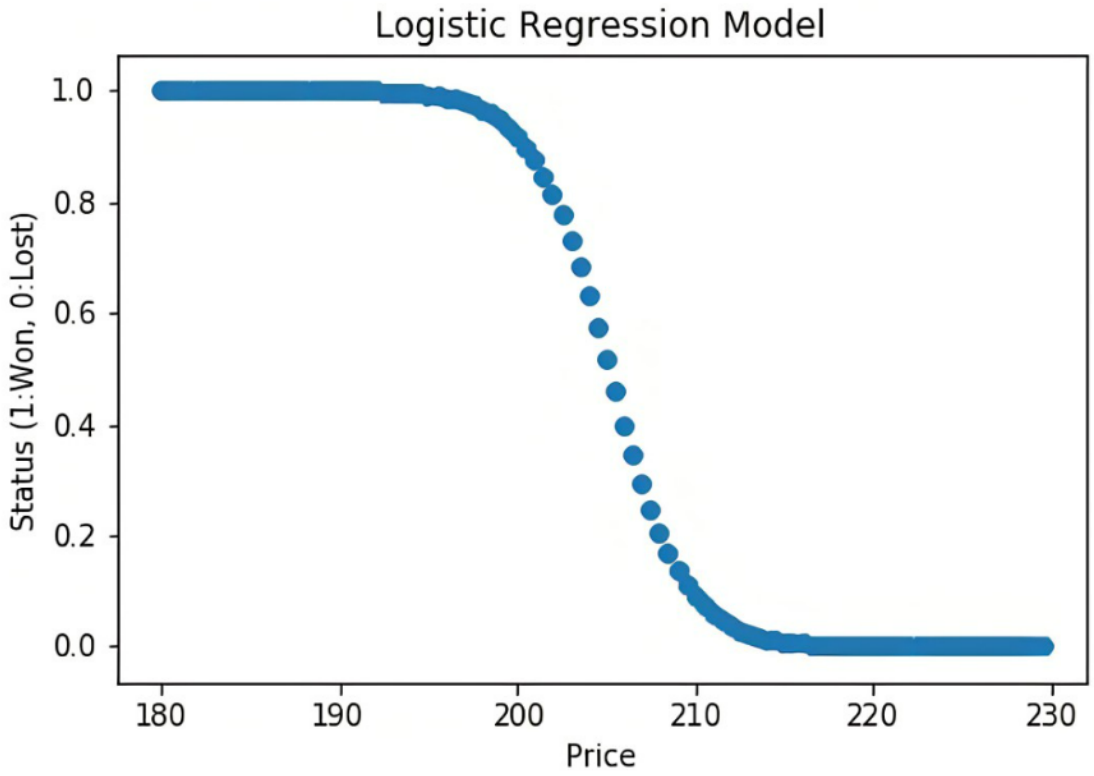

Fig 4. Logistic Regression Example

Logistic regression models are basic linear models for classification, used to predict binary or discrete dependent variables. Regression also involves penalty parameter to minimize the error.

The new data can be added, and the model can continue to update in the ongoing process and model performs well even if the correlation of features are not regular. Figure 4 illustrates the basic construct of logistic regression.

3.) Decision Tree Classifier

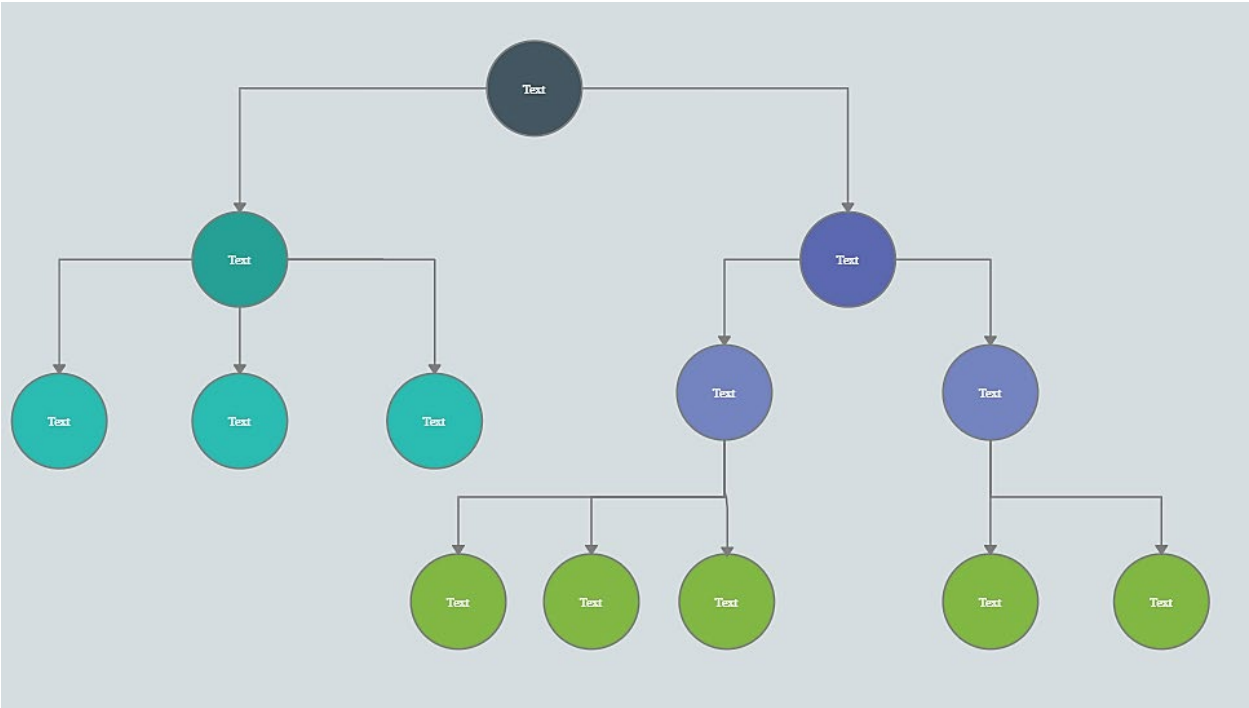

Fig 5. Decision Tree Classifier Example

Decision Tree Classifier follows a rule-based approach to classification and regression problems. This type of classifier uses the values in each feature to split the dataset such that all datapoints that belong to the same class are grouped together.

The class discriminator recursively partitions the training data until each partition consists of one dominant class. An accurate turnover prediction model works around historical and recent data. Figure 5 illustrates the basic construct of Decision Tree Classifier.

4.) Random Forest

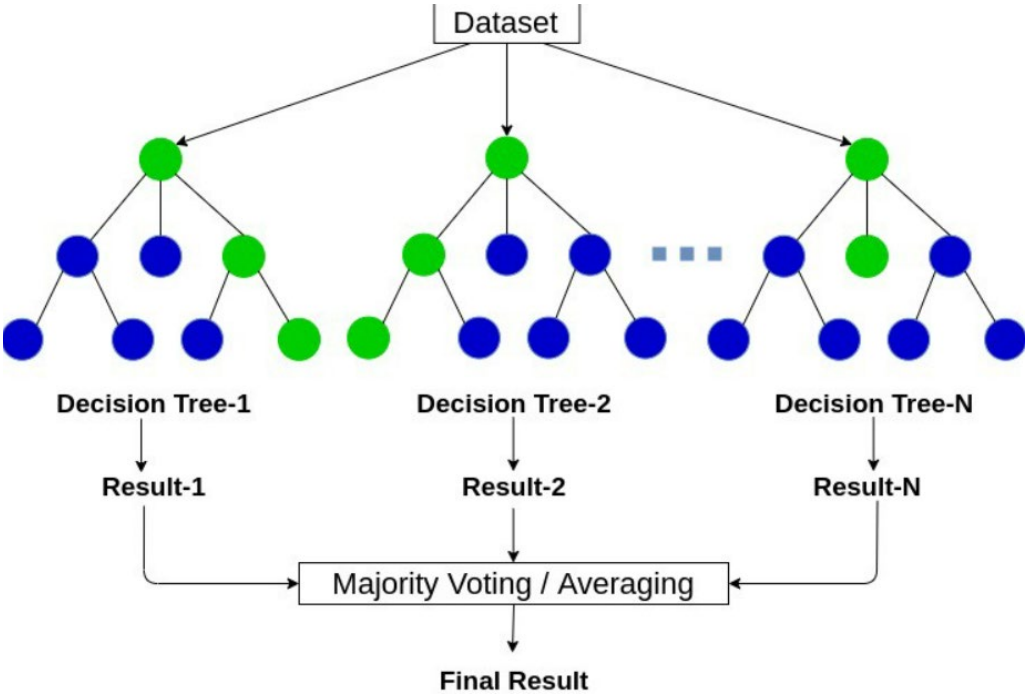

Fig 6. Random Forest Example

The random forest model is set of individual decision trees and has the ability to accept non-linear features. It is suited for high dimensional spaces and large number of training data. The best working solution is to measure the splitting performance, the ability to bootstrap samples and calculate maximum features per tree. This algorithm has the ability to perform a warm start.

Unsupervised Algorithms

Unsupervised machine learning algorithms are used to analyze and cluster unlabeled datasets. They discover hidden patterns or groups without need for human intervention. The are ideal solutions for exploratory data analysis, cross-selling strategies, customer segmentation.

We will be using several unsupervised algorithms, featured below:

1.) Gaussian Mixture Model (GMM)

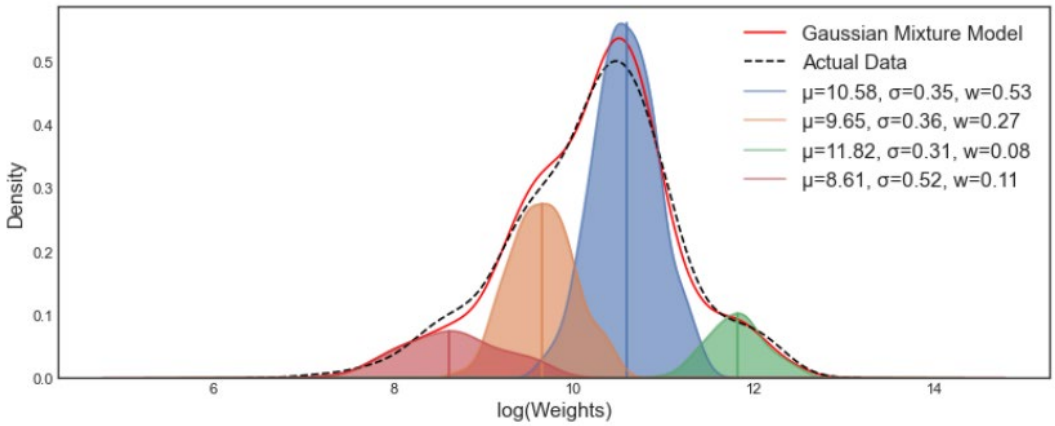

Fig 7. Gaussian Mixture Model Example

GMM is a probabilistic model to represent normally distributed subpopulations with an overall population. Data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. The number of features can be selected in mixture model in an efficient way. It calculates the probability of the data point belonging to a certain center. Figure 7 illustrates the basic construct of GMM.

2.) K-means Clustering

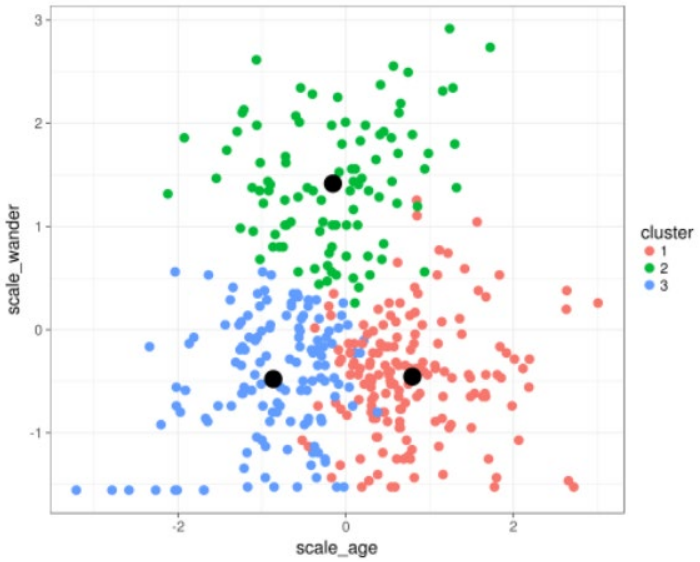

Fig 8. K-means Clustering

K-means model is a special case of GMM in which each cluster's covariance along all dimensions approaches 0. The data points will get assigned to exactly one cluster. This approach is robust and easy to understand. It is computationally efficient, and results are good when data points are distinct and well separated. Figure 8 illustrates the basic construct of k-means.

5.2.3 Performance Evaluation

The performance of all the models is evaluated using the accuracy score. The performance of these models with different algorithms are tested in order to determine the optimal approach. The dataset is put into test using all the above listed models and insights derived from each of the results as well as the ROC Curve Analysis will determine the best suited ones.

5.2.4 Languages Used

For developing and implementing our models, we use Python programming language along with Pyspark API, which exposes the Spark programming model to Python. This decision was influenced by the following factors:

Machine learning and Artificial Intelligence (AI) make it possible to create innovative solutions to common problems, such as customer value prediction, frequency purchase, recommendation systems and search engines. The data set consists of a large number of records. Python is an easy-to-use programming language, and Pyspark API is designed to utilize distributed, in-memory data structures, thus improving data processing speeds when handling large amounts of data.

5.2.5 Tools/Packages Used

We have used following tools/packages in our project:

(a) Numpy

(b) Matplotlib

(c) Jupyter Notebook

(d) Pandas

(e) Skilearn

(f) Plotly

(g) Xgboost

(h) Lifetimes

5.3 How to Generate Output

Our well-trained algorithm will make the predictions from input data which is split into training and testing, whereby 80% of the dataset is used for training and the remaining 20% for testing. With training data, the proposed models are developed and tested with the test dataset, which predicts CLV. Both models are then compared to identify the best model.

5.4 How to Test Against Hypothesis

The goal of our research is to find whether supervised or unsupervised algorithms perform better in predicting CLV. Both models will be evaluated based on various parameters, and the best-performing model will be chosen.

6 Implementation

6.1 Code

Importing Packages:

Install these packages if it is not present in the system

pip install xgboost

pip install lifetimes

pip install plotly

pip install pyspark==3.2.0

IMPORTING BASIC NECESSDARY PACKAGES

In [ ]:

```
import pyspark

from pyspark import *
from pyspark.sql import *
from pyspark import SparkContext
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
import sys
import re
import warnings
sc=SparkContext.getOrCreate()
```

In [ ]:

```
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import xlrd
import lifetimes
import datetime
warnings.filterwarnings("ignore")
np.random.seed(42)
from datetime import datetime, timedelta,date
```

Fig 9. Importing the necessary packages and libraries

Calculating Recency Score:

In [ ]:
```
#order cluster method
def order_cluster(cluster_field_name, target_field_name,df,ascending):
    new_cluster_field_name = 'new_' + cluster_field_name
    df_new = df.groupby(cluster_field_name)[target_field_name].mean().reset_index()
    df_new = df_new.sort_values(by=target_field_name,ascending=ascending).reset_index(drop=True)
    df_new['index'] = df_new.index
    df_final = pd.merge(df,df_new[[cluster_field_name,'index']], on=cluster_field_name)
    df_final = df_final.drop([cluster_field_name],axis=1)
    df_final = df_final.rename(columns={"index":cluster_field_name})
    return df_final
```

In [ ]:
```
tx_user.count()
```

Type *Markdown* and LaTeX: α^2

Calculating recency score and generating recency cluster through kmeans

In [ ]:
```
#calculate recency score
tx_max_purchase = tx_3m.groupby('Customer ID').InvoiceDate.max().reset_index()
tx_max_purchase.columns = ['Customer ID','MaxPurchaseDate']
tx_max_purchase['Recency'] = (tx_max_purchase['MaxPurchaseDate'].max() - tx_max_purchase['MaxPurchaseDate']).dt.days
tx_user = pd.merge(tx_user, tx_max_purchase[['Customer ID','Recency']], on='Customer ID')
```

In [ ]:
```
# Finding the best number of clusters
# Within Cluster Sum Of Squares (WCSS) measures sum of distances of observations from their cluster centroids which is g
# By plotting the WCSS against the the number of clusters (K Value)
# Then we can identify the optimal number of clusters value.


#Transformation process: normalizing our values
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler((0,1))
x_scaled = min_max_scaler.fit_transform(tx_user[['Recency']])
data_scaled = pd.DataFrame(x_scaled)


plt.figure(figsize=(8,6))
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',n_init=10, max_iter = 300)
    kmeans.fit(data_scaled)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method', fontsize=20)
plt.xlabel('Number of Clusters',fontsize=16)
plt.ylabel('WCSS',fontsize=16)
plt.show()
```

In [ ]:
```
kmeans = KMeans(n_clusters=3)
kmeans.fit(tx_user[['Recency']])
tx_user['RecencyCluster'] = kmeans.predict(tx_user[['Recency']])

tx_user = order_cluster('RecencyCluster', 'Recency',tx_user,False)
```

In [ ]:
```
tx_user.count()
```

Fig 10. Calculating the recency score

Calculating frequency Score:

Calculating frequency score and generating frequency cluster through kmeans

In [ ]:
```
#calcuate frequency score
tx_frequency = tx_3m.groupby('Customer ID').InvoiceDate.count().reset_index()
tx_frequency.columns = ['Customer ID','Frequency']
tx_user = pd.merge(tx_user, tx_frequency, on='Customer ID')
```

In [ ]:
```
#Transformation process: normalizing our values
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler((0,1))
x_scaled = min_max_scaler.fit_transform(tx_user[['Frequency']])
data_scaled = pd.DataFrame(x_scaled)


plt.figure(figsize=(8,6))
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',n_init=10, max_iter = 300)
    kmeans.fit(data_scaled)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method', fontsize=20)
plt.xlabel('Number of Clusters',fontsize=16)
plt.ylabel('WCSS',fontsize=16)
plt.show()
```

In [ ]:
```
kmeans = KMeans(n_clusters=5)
kmeans.fit(tx_user[['Frequency']])
tx_user['FrequencyCluster'] = kmeans.predict(tx_user[['Frequency']])
tx_user = order_cluster('FrequencyCluster', 'Frequency',tx_user,True)
```

Fig 11. Calculating the frequency score

Calculating revenue score:

Calculating revenue score and generating revenue cluster through kmeans

In [ ]:
```
#calcuate revenue score
tx_3m['Revenue'] = tx_3m['Price'] * tx_3m['Quantity']
tx_revenue = tx_3m.groupby('Customer ID').Revenue.sum().reset_index()
tx_user = pd.merge(tx_user, tx_revenue, on='Customer ID')
```

In [ ]:
```
#Transformation process: normalizing our values
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler((0,1))
x_scaled = min_max_scaler.fit_transform(tx_user[['Revenue']])
data_scaled = pd.DataFrame(x_scaled)


plt.figure(figsize=(8,6))
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',n_init=10, max_iter = 300)
    kmeans.fit(data_scaled)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method', fontsize=20)
plt.xlabel('Number of Clusters',fontsize=16)
plt.ylabel('WCSS',fontsize=16)
plt.show()
```

In [ ]:
```
kmeans = KMeans(n_clusters=5)
kmeans.fit(tx_user[['Revenue']])
tx_user['RevenueCluster'] = kmeans.predict(tx_user[['Revenue']])
tx_user = order_cluster('RevenueCluster', 'Revenue',tx_user,True)
```

Fig 12. Calculating the revenue score

Calculating overall pricing, and revenue:

Overall scoring

```
In [ ]:
#overall scoring
tx_user['OverallScore'] = tx_user['RecencyCluster'] + tx_user['FrequencyCluster'] + tx_user['RevenueCluster']
tx_user['Segment'] = 'Low-Value'
tx_user.loc[tx_user['OverallScore']>3,'Segment'] = 'High-Value'
```

```
In [ ]:
tx_user.count()
```

```
In [ ]:
tx_user.head(5)
```

Calulating revenue by multiplying price and quantity

```
In [ ]:
#calculate revenue and create a new dataframe for it
tx_6m['Revenue'] = tx_6m['Price'] * tx_6m['Quantity']
tx_user_6m = tx_6m.groupby('Customer ID')['Revenue'].sum().reset_index()
tx_user_6m.columns = ['Customer ID','m6_Revenue']

```

```
In [ ]:
tx_user_6m
```

```
In [ ]:
tx_user.count()

```

```
In [ ]:
tx_user_6m.count()
```

Merging together

```
In [ ]:

tx_merge = pd.merge(tx_user, tx_user_6m, on='Customer ID', how='left')

tx_graph = tx_merge.query("m6_Revenue < 30000")
```

```
In [ ]:
tx_merge
```

Fig 13. Overall Scoring, Calculating revenue and merging the values together

Correlation calculation:

Correlations calculation

```
In [ ]:
#calculate and show correlations
corr_matrix = tx_class.corr()
corr_matrix['LTVCluster'].sort_values(ascending=False)
```

```
In [ ]:
print('\033[1mCorrelation Matrix'.center(100))
plt.figure(figsize=[10,8])
sns.heatmap(tx_class.corr(), annot=True, vmin=-1, vmax=1, center=0) #cmap='BuGn'
plt.show()
```

```
In [ ]:
df_sample_data=tx_class
```

```
In [ ]:
tx_class
```

Fig 14. Correlation Calculation

Test – Train Split, utilizing training data and running through models:

TRAINING AND TESTING MODELS

In [ ]:
```
#create X and y, X will be feature set and y is the label - LTV
X_all_base = tx_class.drop(['LTVCluster','m6_Revenue'],axis=1)
y_all_base = tx_class['LTVCluster']
```

Creating train and test splits from the data

In [ ]:
```
# Let's keep 20 % of the data for testing purposes
test_size = .20
random_state = 45
# Use for testing later and don't touch: X_test_base / y_test_base
X_train_base, X_test_base, y_train_base, y_test_base = cross_validation.train_test_split(
    X_all_base, y_all_base, test_size = test_size, random_state = random_state)
clf_dict_base = {}
clf_report_base = []
clf_feature_relevance_base = []
```

Using training data

In [ ]:
```
clf_dict_base = {}
clf_report_base = []
clf_feature_relevance_base = []
for clf in [LinearSVC(random_state = random_state),
            LogisticRegression(random_state = random_state),
            DecisionTreeClassifier(random_state = random_state),
            SVC(random_state = random_state),
            RandomForestClassifier(random_state = random_state)]:
    # Extract name of estimator
    clf_name = clf.__class__.__name__
    print ("Training", clf_name, "...")
    # Fit model on training data
    clf_dict_base[clf_name] = clf.fit(X_train_base, y_train_base)
    # Predict based on it
    # y_pred = clf.predict(X_train)

    # Perform cross validation
    start = time()
    X_s, y_s = shuffle(X_train_base, y_train_base)
    scores = cross_validation.cross_val_score(clf, X_s, y_s, cv=5, scoring='accuracy')
    end = time()
    duration = end - start
    print ("Average CV performance for {}: {:.6} (in {:.6} seconds)".format(clf_name, scores.mean(), duration))
    clf_report_base.append([clf_name, scores.mean(), duration])

    # Sample Store feature relevance information
    if clf_name in ["RandomForestClassifier", "DecisionTreeClassifier"]:
        clf_feature_relevance_base.append(clf.feature_importances_.tolist())
    elif clf_name == "LinearSVC":
        clf_feature_relevance_base.append(clf.coef_[0].tolist())
# Store information in list for better visibility
clf_report_base = pd.DataFrame(clf_report_base, columns=['classifier', 'Train_accuracy', 'Train_time'])
```

In [ ]:
```
pd.DataFrame(clf_feature_relevance_base, columns=X_train_base.columns, index=['LinearSVC',
                                                                              'DecisionTreeClassifier',
                                                                              'RandomForestClassifier'])
```

Fig 15. Training and testing data split, passing training data through models

Accuracy score of different models (Training data):

Accuracy and time taken by different models

```
In [ ]:
clf_report_base.sort_values(by=['Train_accuracy', 'Train_time'], ascending=False)
```

```
In [ ]:
tx_class.head(5)
```

```
In [ ]:
predictor_list_base = []
for relevance in clf_dict_base['RandomForestClassifier'].feature_importances_:
    predictor_list_base.append(relevance)
new_base = pd.DataFrame(predictor_list_base, columns=['importance'], index=X_all_base.columns.values.tolist())
new_base.sort_values(by='importance', ascending=False, inplace=True)
new_base['features'] = new_base.index


```

Fig 16. Accuracy score of different models – training data

Passing testing data through models: and calculating accuracy score:

Using the Testing data `

```
In [ ]:
clf_dict_base_test = {}
clf_report_base_test = []
clf_feature_relevance_base_test = []

for clf in [LinearSVC(random_state = random_state),
            LogisticRegression(random_state = random_state),
            DecisionTreeClassifier(random_state = random_state),
            SVC(random_state = random_state),
            RandomForestClassifier(random_state = random_state)]:
    # Extract name of estimator
    clf_name = clf.__class__.__name__
    print ("Testing", clf_name, "...")
    # Fit model on training data
    clf_dict_base[clf_name] = clf.fit(X_train_base, y_train_base)
    # Predict based on it
    # y_pred = clf.predict(X_train)

    # Perform cross validation
    start = time()
    X_t, y_t = shuffle(X_test_base, y_test_base)
    scores = cross_validation.cross_val_score(clf, X_t, y_t, cv=5, scoring='accuracy')
    end = time()
    duration = end - start
    print ("Average CV performance for {}: {:.6} (in {:.6} seconds)".format(clf_name, scores.mean(), duration))
    clf_report_base_test.append([clf_name, scores.mean(), duration])

    # Store feature relevance information
    if clf_name in ["RandomForestClassifier", "DecisionTreeClassifier"]:
        clf_feature_relevance_base_test.append(clf.feature_importances_.tolist())
    elif clf_name == "LinearSVC":
        clf_feature_relevance_base_test.append(clf.coef_[0].tolist())
# Store information in list for better visibility

clf_report_base_test = pd.DataFrame(clf_report_base_test, columns=['classifier', 'Test_accuracy', 'Test_time'])
```

Accuracy and time taken by different models

```
In [ ]:
clf_report_base_test.sort_values(by=['Test_accuracy', 'Test_time'], ascending=False)
```

Fig 17. Passing testing data through models and calculating accuracy score

Principal Component Analysis

Principal Component Analysis

In [ ]:

```
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA as RandomizedPCA

def pca_results(good_data, pca):
    '''
    Create a DataFrame of the PCA results
    Includes dimension feature weights and explained variance
    Visualizes the PCA results
    '''

    # Dimension indexing
    dimensions = dimensions = ['Dimension {}'.format(i) for i in range(1,len(pca.components_)+1)]

    # PCA components
    components = pd.DataFrame(np.round(pca.components_, 4), columns = good_data.keys())
    components.index = dimensions

    # PCA explained variance
    ratios = pca.explained_variance_ratio_.reshape(len(pca.components_), 1)
    variance_ratios = pd.DataFrame(np.round(ratios, 4), columns = ['Explained Variance'])
    variance_ratios.index = dimensions

    # Create a bar plot visualization
    fig, ax = plt.subplots(figsize = (25,8))

    # Plot the feature weights as a function of the components
    components.plot(ax = ax, kind = 'bar');
    ax.set_ylabel("Feature Weights")
    ax.set_xticklabels(dimensions, rotation=0)


    # Display the explained variance ratios
    for i, ev in enumerate(pca.explained_variance_ratio_):
        ax.text(i-0.40, ax.get_ylim()[1] + 0.05, "Explained Variance\n          %.4f"%(ev))

    # Return a concatenated DataFrame
    return pd.concat([variance_ratios, components], axis = 1)
```

In [ ]:

```
tx_class.head(5)
```

In [ ]:

```
from sklearn.decomposition import PCA

pca_data = df_sample_data.iloc[:,1:6]
pca = PCA(n_components=3)
pca = pca.fit(pca_data)

# Generate PCA results plot
pca_results = pca_results(pca_data, pca)
```

In [ ]:

```
pd.DataFrame(data=[np.cumsum(pca.explained_variance_ratio_)], columns="Add " +
             pca_results.index.values, index=['Combined Explained Variance'])
```

In [ ]:

```
tx_class.head(5)
```

Fig 18. Principal Component Analysis

Calculating Sillhoutte Score

Calculating Sillhoutte Score

In [ ]:

```
from sklearn import mixture
from sklearn.metrics import silhouette_score

score_list = []
score_columns = []
preds = {}
centers = {}
sample_preds = {}

for n in range(4,1,-1):
    print ("Calculating clusters with {} dimensions.".format(n))
    clusterer = mixture.GaussianMixture(n_components=n)
    # Future
    # clusterer = mixture.GaussianMixture(n_components=n)
    clusterer.fit(reduced_data)

    preds[n] = clusterer.predict(reduced_data)
    centers[n] = clusterer.means_
    score = silhouette_score(reduced_data, preds[n], metric='euclidean')
    score_list.append(score)
    score_columns.append(str(n) + " components")

score_list = pd.DataFrame(data=[score_list],columns=score_columns, index=['Silhouette Score'])
score_list
```

In [ ]:

```
tx_class.head(5)
```

Fig 19. Calculating Sillhoutte Score

Function for plotting 2D visualization

In [ ]:

```
def cluster_results(reduced_data, preds, centers):
    predictions = pd.DataFrame(preds, columns = ['Cluster'])
    plot_data = pd.concat([predictions, reduced_data], axis = 1)

    # Generate the cluster plot
    fig, ax = plt.subplots(figsize = (10,5))

    # Color map
    cmap = cm.get_cmap('gist_rainbow')

    # Color the points based on assigned cluster
    for i, cluster in plot_data.groupby('Cluster'):
        cluster.plot(ax = ax, kind = 'scatter', x = 'Dimension 1', y = 'Dimension 2', \
                     color = cmap((i)*1.0/(len(centers)-1)), label = 'Cluster %i'%(i), s=30);

    # Plot centers with indicators
    for i, c in enumerate(centers):
        ax.scatter(x = c[0], y = c[1], color = 'white', edgecolors = 'black', \
                   alpha = 1, linewidth = 2, marker = 'o', s=200);
        ax.scatter(x = c[0], y = c[1], marker='$%d$'%(i), alpha = 1, s=100);

    # Set plot title
    #plt.legend(loc='upper left', numpoints=1, ncol=4, fontsize=12, bbox_to_anchor=(0, 0))
    ax.set_title("Cluster Learning on PCA-Reduced Data - Centroids Marked by Number",
                 fontsize = 14)
```

Fig 20. Function to plot 2d Visualization

Function for plotting 3D Visualization

In [ ]:

```
def cluster_results_3d(reduced_data, preds, centers):
    from mpl_toolkits.mplot3d import Axes3D

    predictions = pd.DataFrame(preds, columns = ['Cluster'])
    plot_data = pd.concat([predictions, reduced_data], axis = 1)
    cmap = cm.get_cmap('gist_rainbow')

    fig = plt.figure(figsize = (20,10))
    ax = fig.add_subplot(111, projection='3d')
    ax2 = fig.add_subplot(111, projection='3d')

    fig = fig.gca(projection='3d')

    for i, c in enumerate(centers):
        ax2.scatter(c[0], c[1], c[2], color = 'white', edgecolors = 'black', \
                    alpha = 1, linewidth = 2, marker = 'o', s=200,
                    zorder=1);
        ax2.scatter(c[0], c[1], c[2], marker='$%d$'%(i), alpha = 1, s=100,
                    zorder=1);

    for i, cluster in plot_data.groupby('Cluster'):
        ax2.scatter(cluster['Dimension 1'], cluster['Dimension 2'], cluster['Dimension 3'],
                    c = cmap((i)*1.0/(len(centers)-1)), alpha=0.2,
                    label = 'Cluster %i'%(i), s=20,
                    zorder=.5)

    fig.set_xlabel('Dimension 1')
    fig.set_ylabel('Dimension 2')
    fig.set_zlabel('Dimension 3')
    plt.legend(loc='upper left', numpoints=1, ncol=4, fontsize=12, bbox_to_anchor=(0, 0))
    ax2.set_title("Cluster Learning on PCA-Reduced Data 3D Plot",
                  fontsize = 14);
    plt.show()
```

Fig 21. Function for plotting 3D visualization

Visualizing 2D and 3D Clusters:

Visualizing cluster results

In [ ]:

```
from matplotlib.axes._axes import _log as matplotlib_axes_logger
matplotlib_axes_logger.setLevel('ERROR')
```

In [ ]:

```
# with preds[n] and centers[n] where n is the number of Dimensions
no_clusters = 3
cluster_results(reduced_data, preds[no_clusters], centers[no_clusters])

```

In [ ]:

```
cluster_results_3d(reduced_data, preds[no_clusters], centers[no_clusters])
```

In [ ]:

```
predictions = pd.DataFrame(preds[no_clusters], columns = ['Cluster'])
df_sample_data['customer_cluster'] = predictions
```

In [ ]:

```
df_sample_data.head()
```

In [ ]:

```
print (df_sample_data.columns)
```

In [ ]:

```
df_sample_data.head(5)
```

Fig 22. Visualizing the 2D and 3D Clusters

Passing the training data generated after GMM Clustering using the models:

TRAINING AND TESTING MODELS - AFTER CLUSTERING USING GMM

Using Training Data

In [ ]:

```
# Let's start the prediction
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC, SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score
from sklearn import model_selection
from time import time
# Let's keep 20 % of the data for testing purposes
test_size = .20
random_state = 45
#create X and y, X will be feature set and y is the label - LTV
X_all_clus = df_sample_data.drop(['LTVCluster','m6_Revenue'],axis=1)
y_all_clus = df_sample_data['LTVCluster']
# Use for testing later and don't touch: X_test_clus / y_test_clus
X_train_clus, X_test_clus, y_train_clus, y_test_clus = cross_validation.train_test_split(
    X_all_clus, y_all_clus, test_size = test_size, random_state = random_state, stratify=y_all_clus)
clf_dict_clus = {}
clf_report_clus = []
clf_feature_relevance_clus = []
for clf_clus in [LinearSVC(random_state = random_state),
                 LogisticRegression(random_state = random_state),
                 DecisionTreeClassifier(random_state = random_state),
                 SVC(random_state = random_state),
                 RandomForestClassifier(random_state = random_state)]:
    # Extract name of estimator
    clf_name = clf_clus.__class__.__name__
    print ("Training", clf_name, "...")
    # Fit model on training data
    clf_dict_clus[clf_name] = clf_clus.fit(X_train_clus, y_train_clus)
    # Predict based on it
    # y_pred = clf.predict(X_train)
    # Perform cross validation
    start = time()
    scores = cross_validation.cross_val_score(clf_clus, X_train_clus, y_train_clus, cv=5, scoring='accuracy')
    end = time()
    duration = end - start
    print ("Average CV performance for {}: {:.6} (in {:.6} seconds)".format(clf_name, scores.mean(), duration))
    clf_report_clus.append([clf_name, scores.mean(), duration])
    # Store feature relevance information
    if clf_name in ["RandomForestClassifier", "DecisionTreeClassifier"]:
        clf_feature_relevance_clus.append(clf_clus.feature_importances_.tolist())
    elif clf_name == "LinearSVC":
        clf_feature_relevance_clus.append(clf_clus.coef_[0].tolist())
# Store information in list for better visibility
clf_report_clus = pd.DataFrame(clf_report_clus, columns=['classifier', 'Cluster_train_accuracy', 'Cluster_train_time'])
```

In [ ]:

```
pd.DataFrame(clf_feature_relevance_clus, columns=X_train_clus.columns, index=['LinearSVC',
                                                                              'DecisionTreeClassifier',
                                                                              'RandomForestClassifier'])
```

]

Fig 23. Passing the training data generated after GMM clustering using the models

Accuracy score of GMM Clustered Training data passing through models:

Accuracy and time taken by different models

In [ ]:

```
clf_report_clus = clf_report_clus.iloc[:,0:3]
clf_report_clus.sort_values('Cluster_train_accuracy', ascending=False)
```

Fig 24. Accuracy score calculated after passing GMM clustered Training data through models

Passing the testing data generated after GMM Clustering using the models:\

Using Testing Data

In [ ]:

```
# Let's start the prediction
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC, SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score
from sklearn import model_selection
from time import time

# Let's keep 20 % of the data for testing purposes
test_size = .35
random_state = 42

#create X and y, X will be feature set and y is the label - LTV
X_all_clus = df_sample_data.drop(['LTVCluster','m6_Revenue'],axis=1)
y_all_clus = df_sample_data['LTVCluster']

# Use for testing later and don't touch: X_test_clus / y_test_clus
X_train_clus, X_test_clus, y_train_clus, y_test_clus = cross_validation.train_test_split(
    X_all_clus, y_all_clus, test_size = test_size, random_state = random_state, stratify=y_all_clus)

clf_dict_clus_test = {}
clf_report_clus_test = []
clf_feature_relevance_clus_test = []

for clf_clus in [LinearSVC(random_state = random_state),
                 LogisticRegression(random_state = random_state),
                 DecisionTreeClassifier(random_state = random_state),
                 SVC(random_state = random_state),
                 RandomForestClassifier(random_state = random_state)]:
    # Extract name of estimator
    clf_name = clf_clus.__class__.__name__
    print ("Testing", clf_name, "...")
    # Fit model on training data
    clf_dict_clus[clf_name] = clf_clus.fit(X_train_clus, y_train_clus)
    # Predict based on it
    # y_pred = clf.predict(X_train)

    # Perform cross validation
    start = time()
    scores = cross_validation.cross_val_score(clf_clus, X_test_clus, y_test_clus, cv=5, scoring='accuracy')
    end = time()
    duration = end - start
    print ("Average CV performance for {}: {:.6} (in {:.6} seconds)".format(clf_name, scores.mean(), duration))
    clf_report_clus_test.append([clf_name, scores.mean(), duration])

    # Store feature relevance information
    if clf_name in ["RandomForestClassifier", "DecisionTreeClassifier"]:
        clf_feature_relevance_clus_test.append(clf_clus.feature_importances_.tolist())
    elif clf_name == "LinearSVC":
        clf_feature_relevance_clus_test.append(clf_clus.coef_[0].tolist())
# Store information in list for better visibility

clf_report_clus_test = pd.DataFrame(clf_report_clus_test, columns=['classifier', 'Cluster_test_accuracy', 'Cluster_test_
```

Fig 25. Passing testing data generated after GMM Clustering through models

Accuracy score of GMM Clustered Testing data passing through models:

Accuracy and time taken by different models

In [ ]:
```
clf_report_clus_test = clf_report_clus_test.iloc[:,0:3]
clf_report_clus_test.sort_values('Cluster_test_accuracy', ascending=False)
```

Fig 26. Accuracy score calculated after passing GMM clustered testing data through models

Visualizing the feature importance without cluster information:

VISUALIZATION

Feature importance without cluster information

In [ ]:
```
predictor_list = []
for relevance in clf_dict_base['RandomForestClassifier'].feature_importances_:
    predictor_list.append(relevance)


columns_list = X_all_base.columns.values.tolist()

new = pd.DataFrame(predictor_list, columns=['importance'], index=columns_list)
new.sort_values(by='importance', ascending=False, inplace=True)
new['features'] = new.index
new = new[:6]


new

sns.set(rc = {'figure.figsize':(10,5)})
#sns.set_style('whitegrid')
sns.set(font_scale=1)
plt.title('Feature importance without cluster information', size=16,color='BLACK')
plt.xlabel('features', size=13)
plt.ylabel('importance', size=13)
g = sns.barplot(data=new,x = 'features', y = 'importance', color='crimson')
```

Fig 27. Visualizing the Feature Importance without cluster information

Visualizing the feature importance with cluster information:

Feature importance with cluster information

In [ ]:

```
predictor_list_clus = []
for relevance in clf_dict_clus['RandomForestClassifier'].feature_importances_:
    predictor_list_clus.append(relevance)
new_clus = pd.DataFrame(predictor_list_clus, columns=['importance'], index=X_all_clus.columns.values.tolist())
new_clus.sort_values(by='importance', ascending=False, inplace=True)
new_clus['features'] = new_clus.index

new_clus = new_clus[:6]

sns.set(rc = {'figure.figsize':(10,5)})
#sns.set_style('whitegrid')
sns.set(font_scale=1)
plt.title('Feature importance with cluster information', size=16,color='BLACK')
plt.xlabel('features', size=13)
plt.ylabel('importance', size=13)
g = sns.barplot(data=new_clus,x = 'features', y = 'importance', color='crimson')


```

In [ ]:

```
clf_report_base
```

Fig 28. Visualizing the Feature Importance with cluster information

Copying accuracy scores to a result dataframe:

FINAL RESULTS

In [ ]:

```
final_result=pd.DataFrame()
final_result['classifier']=clf_report_base['classifier']
final_result['Train_accuracy']=clf_report_base['Train_accuracy']
final_result['Train_time']=clf_report_base['Train_time']
final_result['Test_accuracy']=clf_report_base_test['Test_accuracy']
final_result['Test_time']=clf_report_base_test['Test_time']
final_result['Cluster_train_accuracy']=clf_report_clus['Cluster_train_accuracy']
final_result['Cluster_train_time']=clf_report_clus['Cluster_train_time']
final_result['Cluster_test_accuracy']=clf_report_clus_test['Cluster_test_accuracy']
final_result['Cluster_test_time']=clf_report_clus_test['Cluster_test_time']
```

In [ ]:

```
final_result
```

In [ ]:

```
final_accuracy=pd.DataFrame()
final_accuracy['classifier']=final_result['classifier']
final_accuracy['Train_accuracy']=final_result['Train_accuracy']
final_accuracy['Test_accuracy']=final_result['Test_accuracy']
final_accuracy['Cluster_train_accuracy']=final_result['Cluster_train_accuracy']
final_accuracy['Cluster_test_accuracy']=final_result['Cluster_test_accuracy']

```

In [ ]:

```
final_accuracy
```

Fig 29. Converting final prediction accuracies into dataframe

ROC Curve Analysis:

ROC CURVE ANALYSIS

In [ ]:
```
# Import the classifiers
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC, SVC
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.calibration import CalibratedClassifierCV
```

In [ ]:
```
# Instantiate the classfiers and make a list
classifiers = [LogisticRegression(random_state=42),
               DecisionTreeClassifier(random_state=42),
               RandomForestClassifier(random_state=42),
               SVC(random_state=42),
               LinearSVC(random_state=42)]

# Define a result table as a DataFrame
result_table = pd.DataFrame(columns=['classifiers', 'fpr','tpr','auc'])

# Train the models and record the results
for cls in classifiers:
    clf = CalibratedClassifierCV(cls)
    model = clf.fit(X_train_clus, y_train_clus)
    yproba = model.predict_proba(X_test_clus)[::,1]

    fpr, tpr, _ = roc_curve(y_test_clus,  yproba)
    auc = roc_auc_score(y_test_clus, yproba)

    result_table = result_table.append({'classifiers':cls.__class__.__name__,
                                        'fpr':fpr,
                                        'tpr':tpr,
                                        'auc':auc}, ignore_index=True)

result_table.set_index('classifiers', inplace=True)
#print(result_table)
```

In [ ]:
```
fig = plt.figure(figsize=(8,6))

for i in result_table.index:
    plt.plot(result_table.loc[i]['fpr'],
             result_table.loc[i]['tpr'],
             label="{}, AUC={:.3f}".format(i, result_table.loc[i]['auc']))
plt.plot([0,1], [0,1], color='orange', linestyle='--')
plt.xticks(np.arange(0.0, 1.1, step=0.1))
plt.xlabel("False Positive Rate", fontsize=15)
plt.yticks(np.arange(0.0, 1.1, step=0.1))
plt.ylabel("True Positive Rate", fontsize=15)
plt.title('ROC Curve Analysis', fontweight='bold', fontsize=15)
plt.legend(prop={'size':13}, loc='lower right')
plt.show()
```

Fig 30. ROC Curve Analysis

6.2 Design document and flow chart

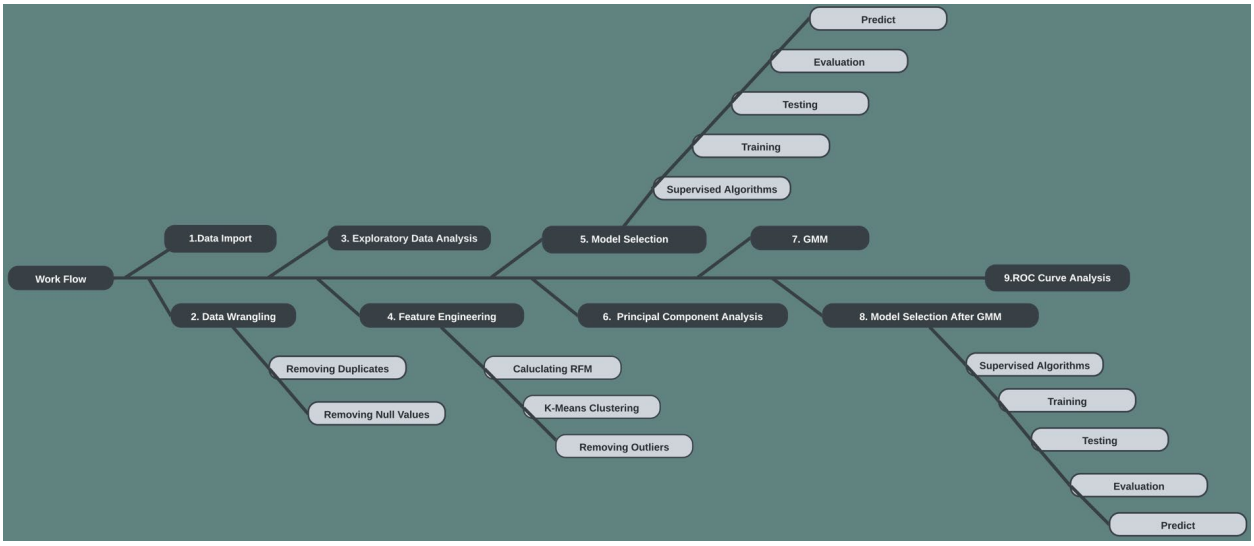

Fig 31. Workflow chart

The first step is to download the necessary dataset. Once downloaded , the data is put through the wrangling part. In this phase, the duplicates and null values are removed and we get a condensed dataset with higher quality. In order to understand the dataset better, Exploratory data analysis is performed. In this step, scenarios like, top 8 countries with most transactions, 8 countries with least transactions, Purchases based on month, year, date and day , etc. are measured. Insights from this step will be helpful in determining how the data needs to be handled.

The dataset contains records for 2 years, thus, to carry out with predictions, the year 1 value is taken as the base sample to train and the year 2 data is categorized as the one where the prediction is going to happen and get verified. Year 1 data is now used to calculate the recency, frequency and monetary benefits factors. Kmeans clustering is used to classify the data into clusters after removing the labels. Elbow plot for each of the frequency, monetary benefits and recency factors

is used to decide on the number of clusters to be used. Using the calculated RFM Scores, the overall scoring is calculated and revenue is derived using the year 2 dataset.

This simplified dataset is now passed into different machine learning models such as Logistic Regression, SVC, Linear SVC, DecisionTreeClassifier and Random Classifier. The accuracy score of each of these models is recorded.

Following this, the dataset is further experimented by passing it into Gaussian Mixture Model again and calculate the silhouette score. Dimensions decide the no of clusters to be used here.

The refined dataset coming out of the above step is then passed on to different machine learning models such as Logistic Regression, SVC, Linear SVC, DecisionTreeClassifier and Random Classifier. The accuracy score of each of these models is recorded.

Comparison of all the calculated accuracy score will help in determining the most optimal solution for the problem. An ROC Curve is plotted to display the accuracy score of all the models

7. Data Analysis and Discussion

Data analysis is a crucial part when dealing with a large dataset. It is helpful in attaining critical insights which will be handy when dealing with the problems that occur down the road. The interpretation of data gathered through logical reasoning will help determining patterns , relationship, and trends. Along with this, it will also help the personnel in gaining significant knowledge regarding the dataset itself. The outputs generated by the code and the insights gathered are discussed in this section.

7.1 Output Generation

The dataset that was chosen for this project consists of information regarding the purchase behavior of customers spread across a couple of years. The data has been split into two parts, the first one being the one used to generate the RFM clustered values and overall score. The second one is used to calculate the Revenue and its respective cluster value. Which will then be used as the target part in the prediction. The predictions will be equated to the cluster values generated in the second part which in turn will be reflected in the accuracy Score.

7.1.1 Exploratory Data Analysis

In order to get a better understanding of the data, some exploratory analysis operations were performed. The output generated from those operations can be seen in the following section

First analysis done is to check the number of Transactions per country.

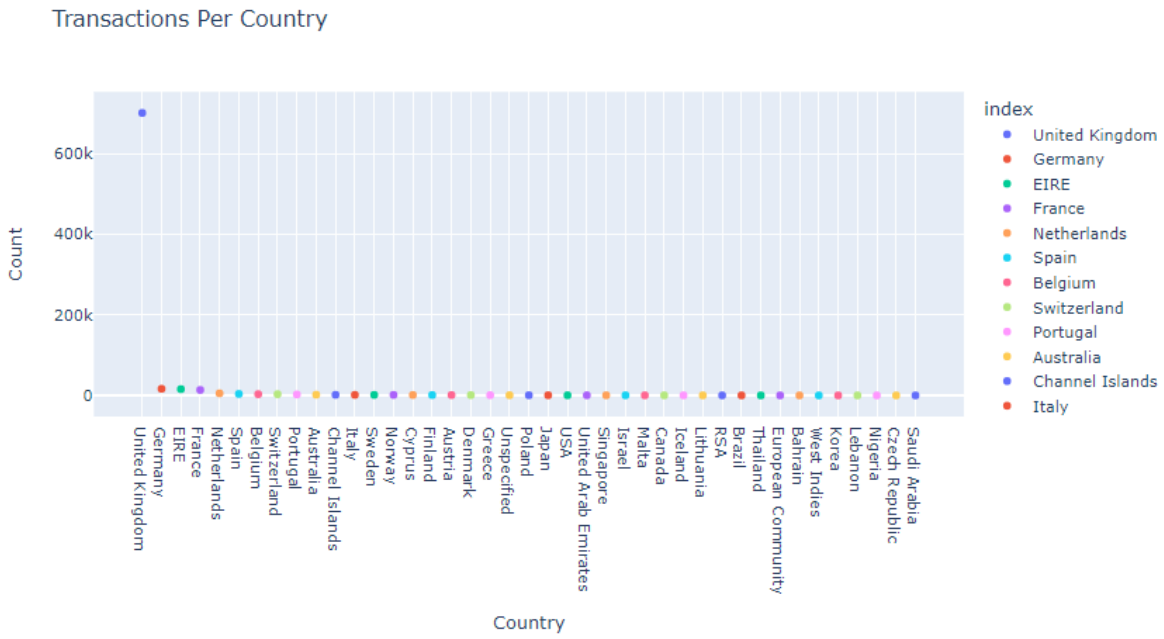

Fig 32. Count of transactions per country

Figure 32 depicts the number of transactions recorded per Country. From this we shall see that United Kingdom recorded the most transactions followed by Germany , EIRE, France and Netherlands

Following the above analysis, the next one to come up is to find the average price per country.

Figure 33 depicts the list of countries and the average of prices per country. In transactions UK had the highest number, whereas here it doesn't seem to be so. Spain has the highest average price per country. The nations to follow Spain and arrive at the 2nd and 3rf spot are Norway and Malta.

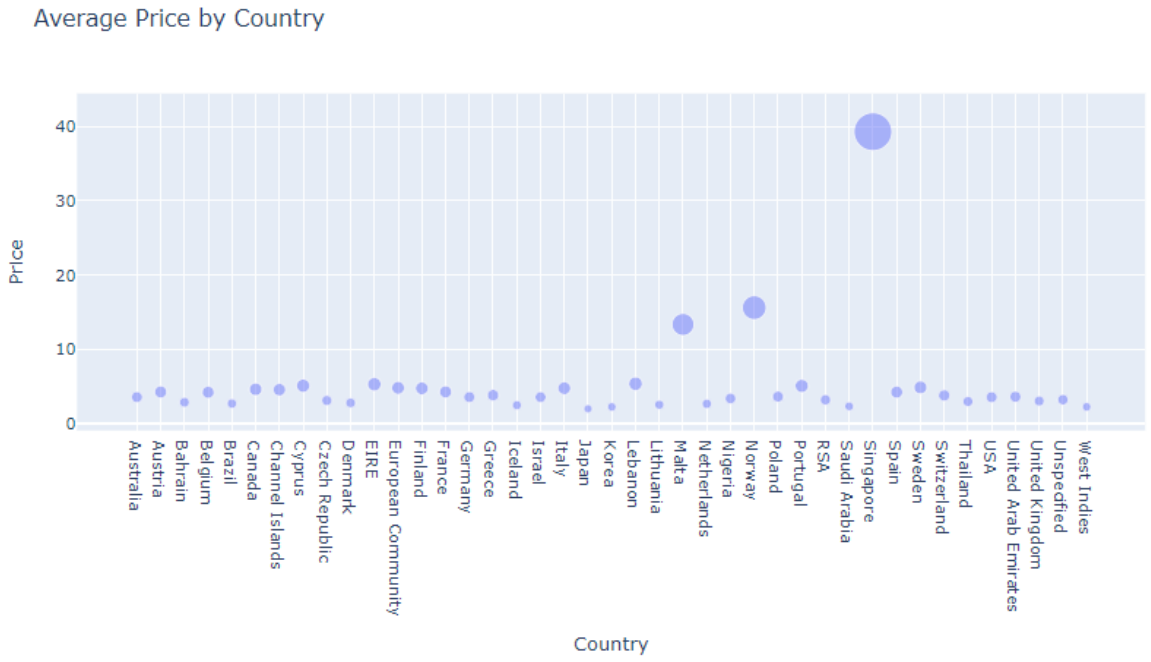

Fig 33. Average price per country

In order to gain insights on the dataset and come up with the most optimal approach to find the customer lifetime value predictions, there are a few factors which needs to be considered. The most crucial one is, finding out the purchase patterns based on the time of year. Hence the listed scenarios were taken into consideration:

(a) Transactions based on month

(b) Transactions based on year

(c) Transactions based on Quarter

(d) Transactions based on Day and

(e) Transactions based on Week

The results of these scenarios are depicted in the Figure 34.

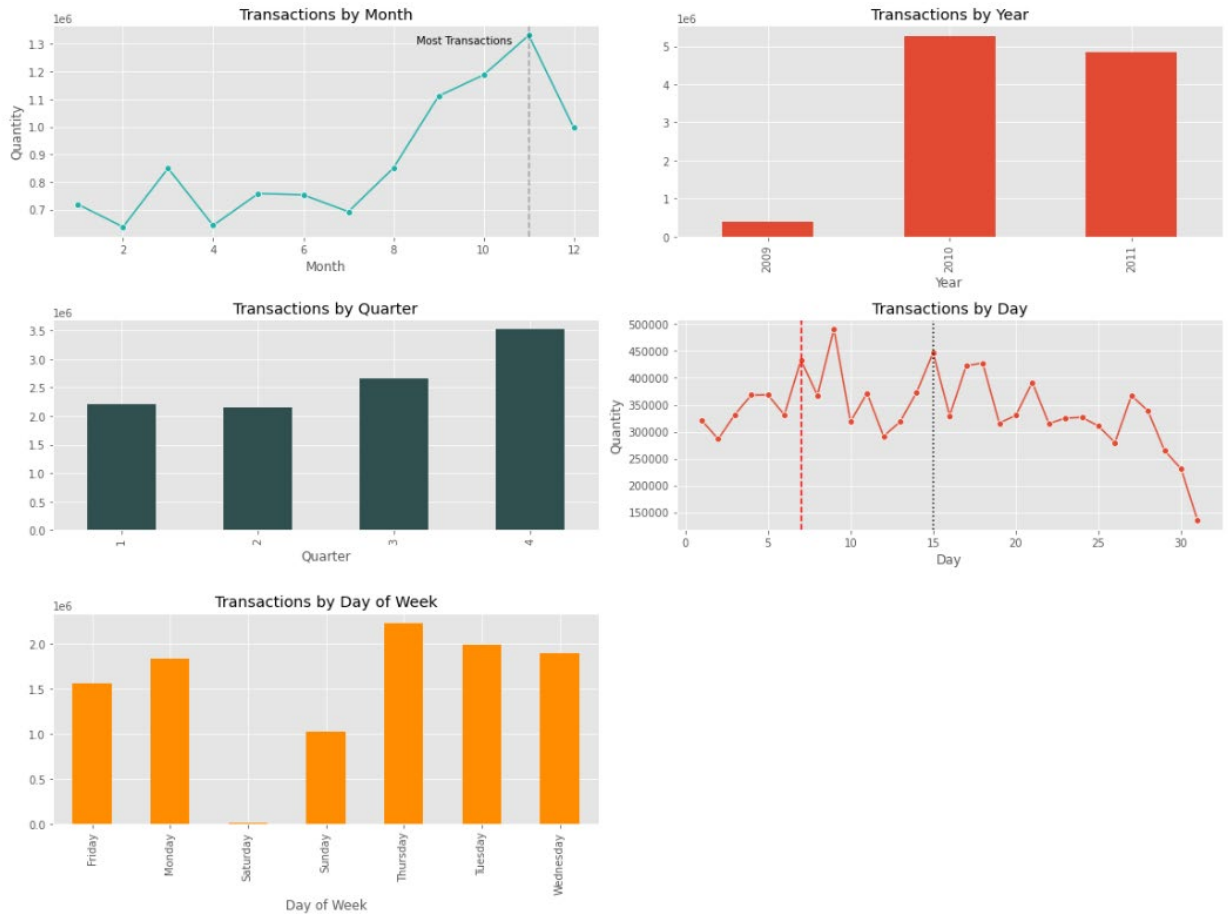

Fig 34. Exploratory analysis based on time period

From this, the following insights are gathered

(a) Most purchases tend to happen on the month of November, this is most likely due to the thanksgiving and year end holidays coming forward

(b) The above reason also justifies why most purchases are tracked in last quarter

(c) 2010 seems to have more purchases happening when compared with the other two years. The most probable reason for this is, the lack of data in the year prior and due to the closure of a shop in the year later

(d) People also tend to purchase more between the 7^{th} and 10^{th} of the month. The reason for this could be, the dispatch of salary on those dates

Figure 35 and 36 depicts the products which are on demand in the top 8 countries where the most transactions are happening

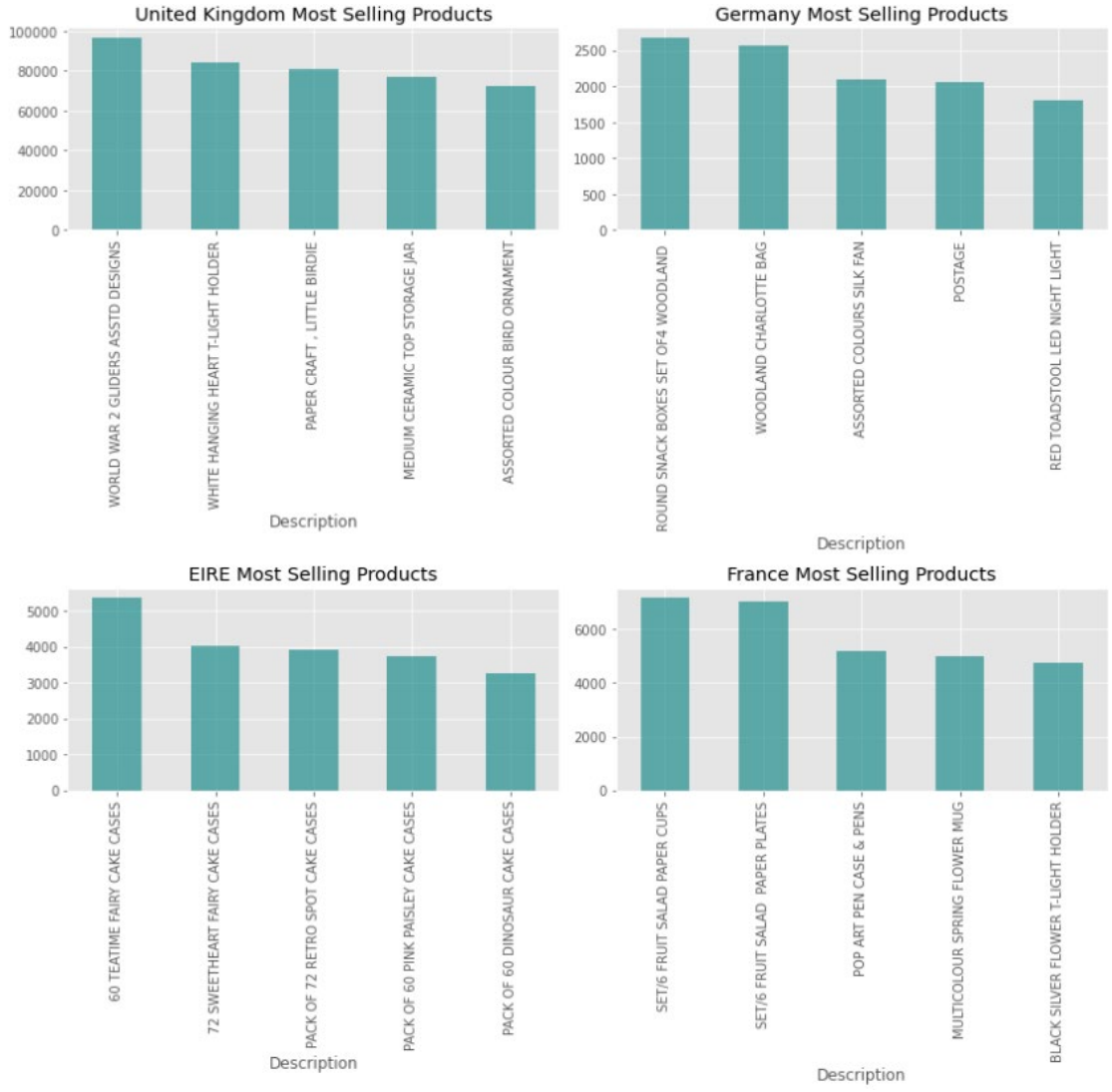

Fig 35. Top 8 Countries with most transactions – part 1

The countries represented in figure 35 are UK , Germany, EIRE, France . And in Figure 35, the countries namely Netherlands, Spain, Belgium and Switzerland are represented.

Fig 36. Top 8 Countries with most transactions – part 2

Figure 37 and 38 depicts the products which are on demand in the top 8 countries where the least transactions are happening.

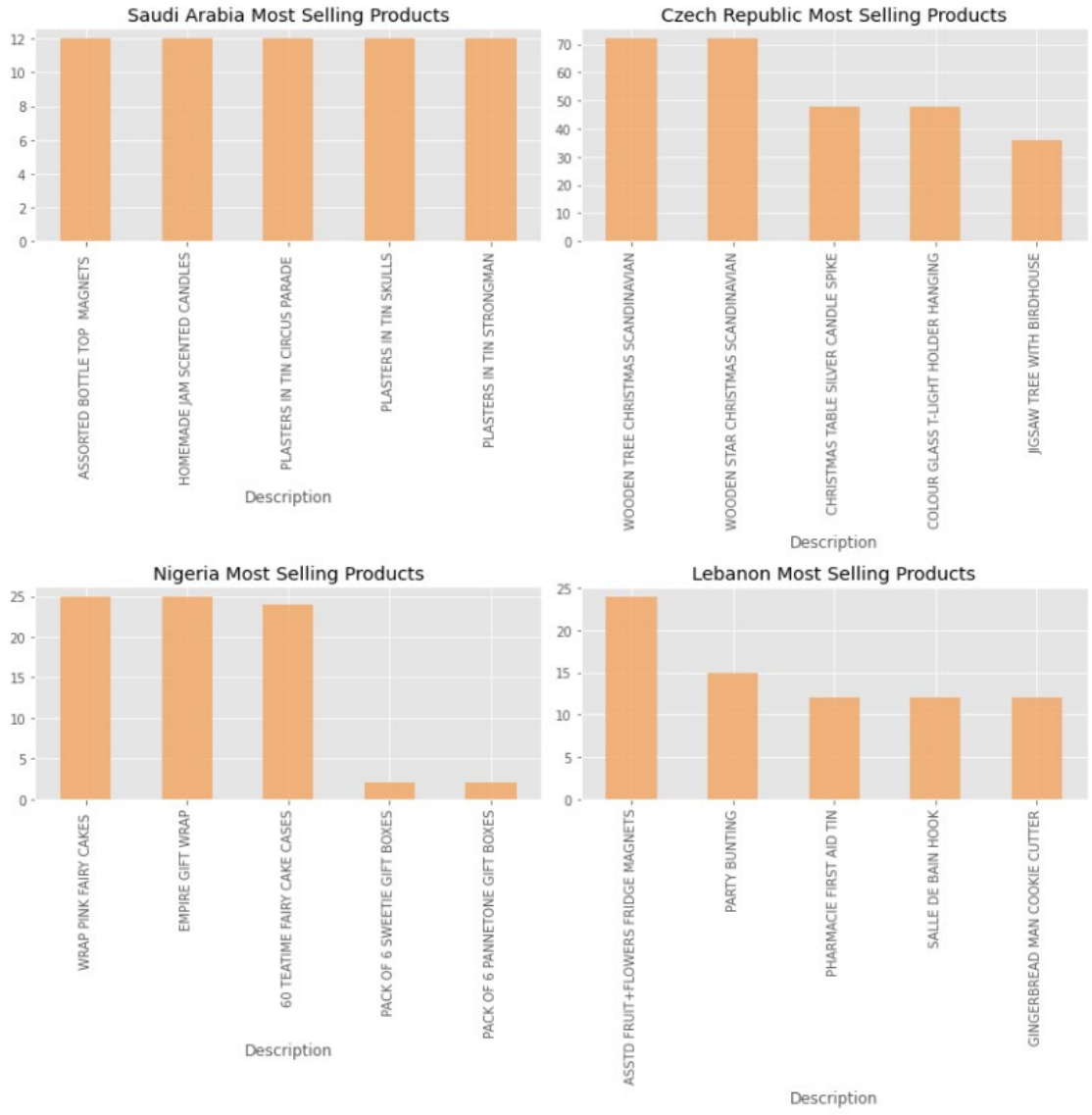

Fig 37. Top 8 Countries with least transactions – part 1

The countries represented in figure 37 are Saudi Arabia, Czech Republic , Nigeria and Lebanon . And in Figure 35, the countries namely Korea, West Indies, Bahrain and European Community are represented

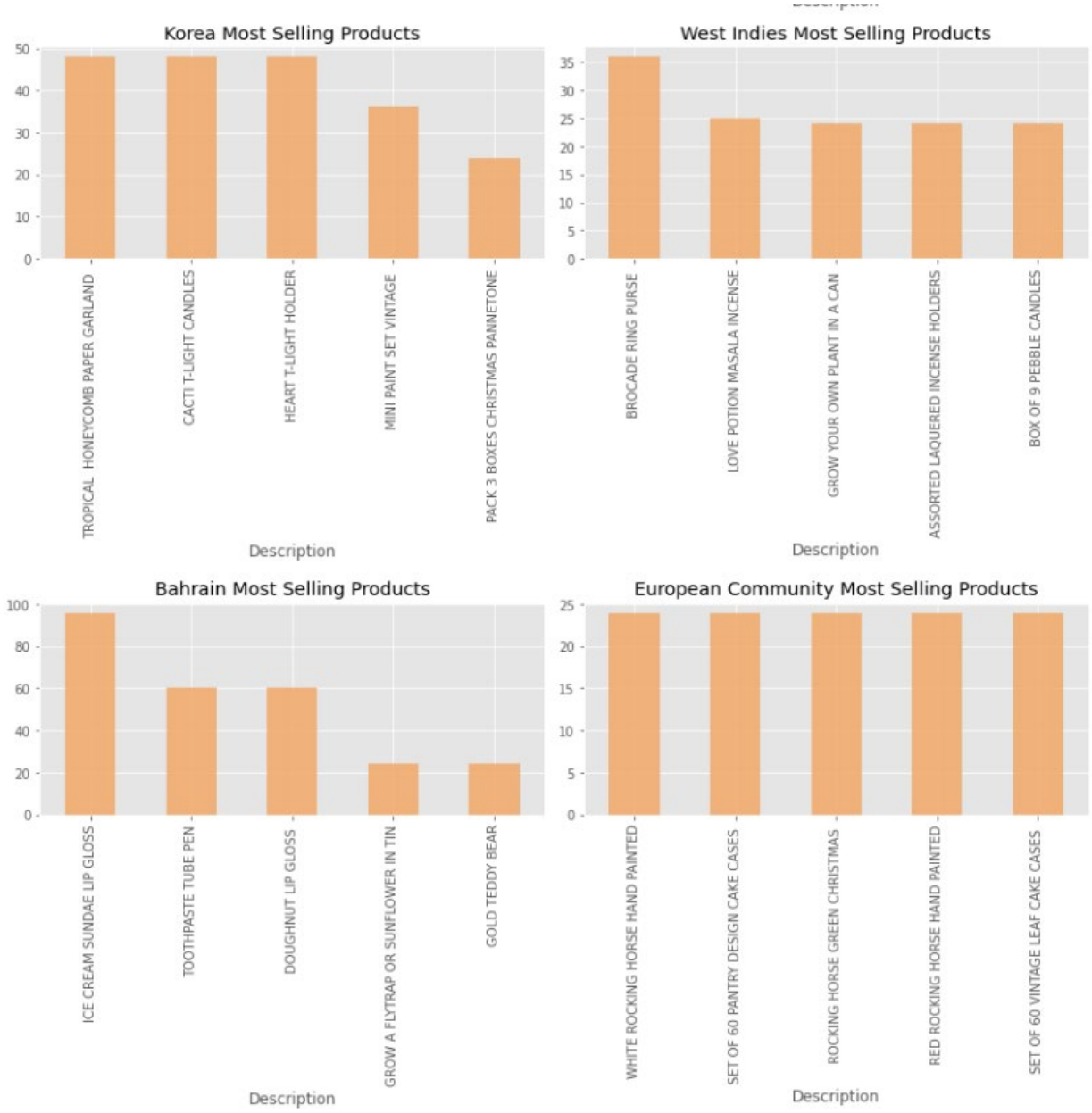

Fig 38. Top 8 Countries with least transactions – part 2

A correlation matrix is simply a table which displays the correlation. It is best used in variables that demonstrate a linear relationship between each other Figure 39 depicts the correlation matrix.

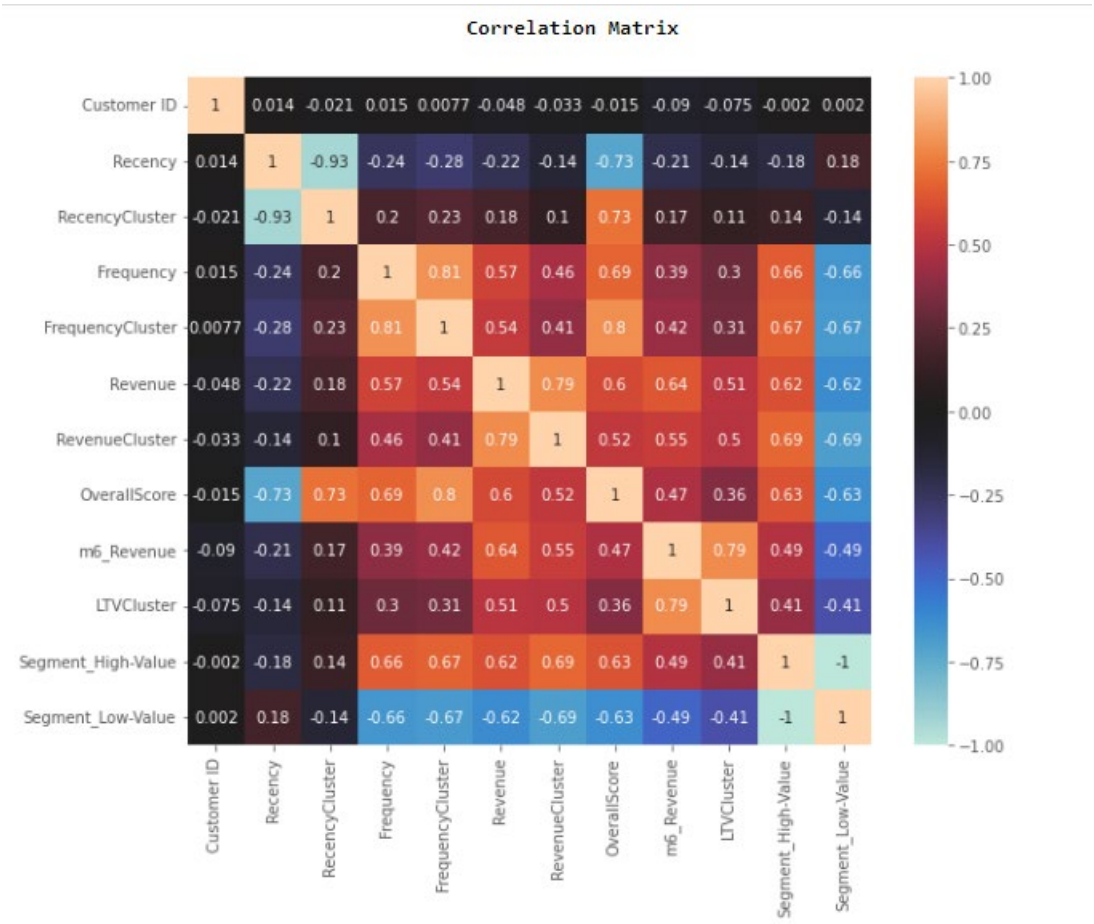

Fig 39. Correlation Matrix

7.1.2 Training and Testing

The Recency , Frequency and Monetary benefit factors that were clustered using Kmeans will now be split into train and test sets. Two different executions will take place here by passing the value through different models that were chosen for this project. The models are Logistic Regression, Linear SVC, SVC, Decision Tree Classifiers, Random Forest Classifiers. First, the training set is passed and then the testing set. The accuracy Scores are measured for these.

```
Training LinearSVC ...
Average CV performance for LinearSVC: 0.942274 (in 0.189645 seconds)
Training LogisticRegression ...
Average CV performance for LogisticRegression: 0.942271 (in 0.0724354 seconds)
Training DecisionTreeClassifier ...
Average CV performance for DecisionTreeClassifier: 0.918997 (in 0.031569 seconds)
Training SVC ...
Average CV performance for SVC: 0.939938 (in 0.139932 seconds)
Training RandomForestClassifier ...
Average CV performance for RandomForestClassifier: 0.936689 (in 0.754307 seconds)
```

Fig 40. Accuracy scores returned by models when training data is passed through

```
Testing LinearSVC ...
Average CV performance for LinearSVC: 0.936691 (in 0.0480022 seconds)
Testing LogisticRegression ...
Average CV performance for LogisticRegression: 0.942264 (in 0.050642 seconds)
Testing DecisionTreeClassifier ...
Average CV performance for DecisionTreeClassifier: 0.927362 (in 0.0187457 seconds)
Testing SVC ...
Average CV performance for SVC: 0.931118 (in 0.0309997 seconds)
Testing RandomForestClassifier ...
Average CV performance for RandomForestClassifier: 0.932883 (in 0.433957 seconds)
```

Fig 41. Accuracy scores returned by models when testing data is passed through

Figure 40 and 41 depicts the accuracy scores returned by the two distinguishes runs of all the models with one having the training data as input while other having the testing data as input.

Following this we have also experiment with double layer clustering, this time using GMM. The KMeans clustered values are now again passed into Gaussian Mixture Model. This experiment is done to check if it has any effect on increasing the accuracy. Figure 42 depicts the principal component analysis . Figure 43 and 44 depicts the 2D and 3D representation of the data clusters passed through GMM.

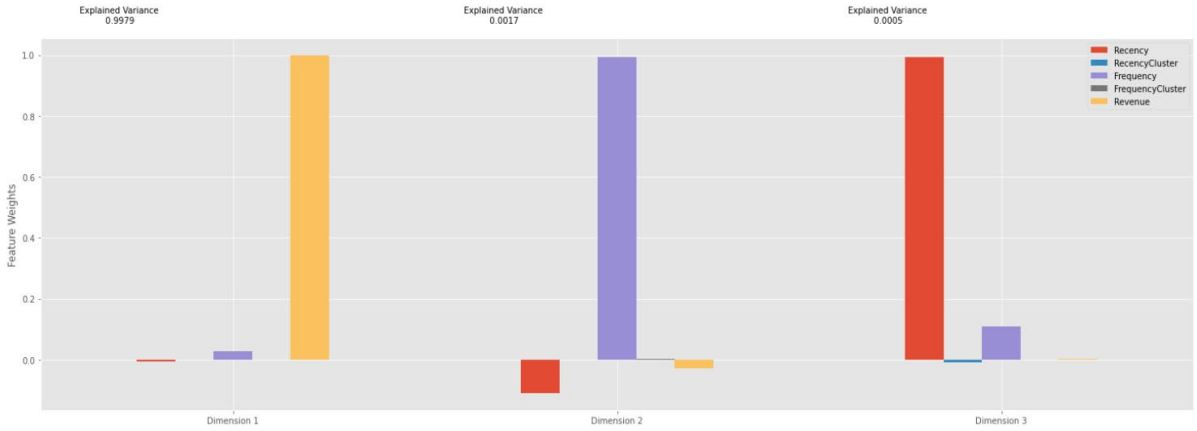

Fig 42. Principal Component Analysis

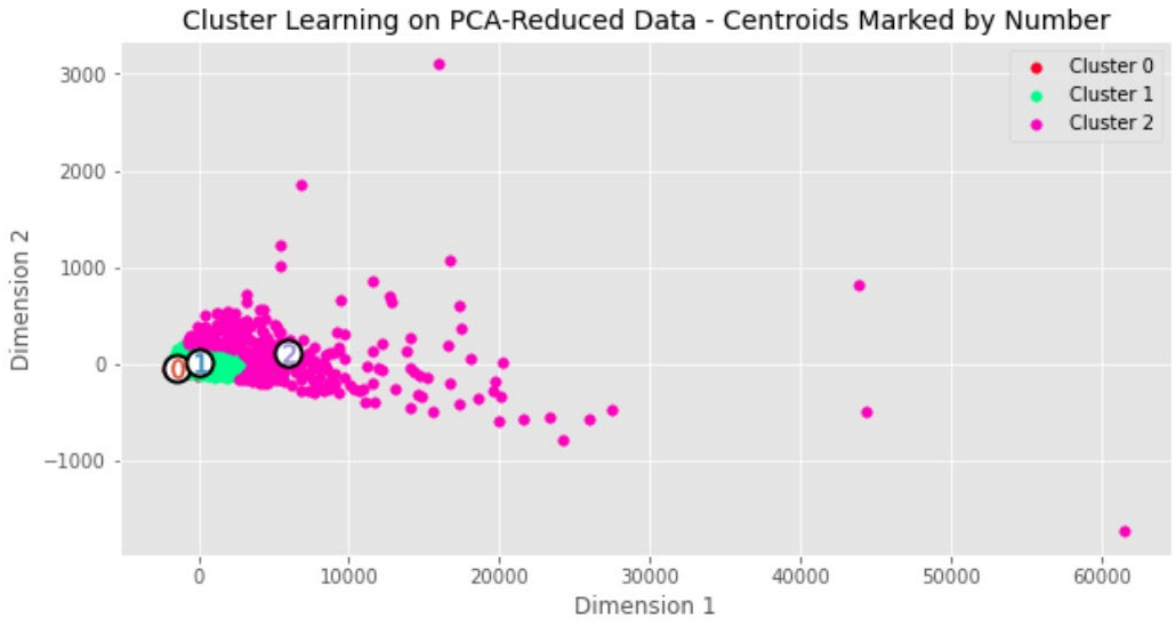

Fig 43. Principal Component Analysis

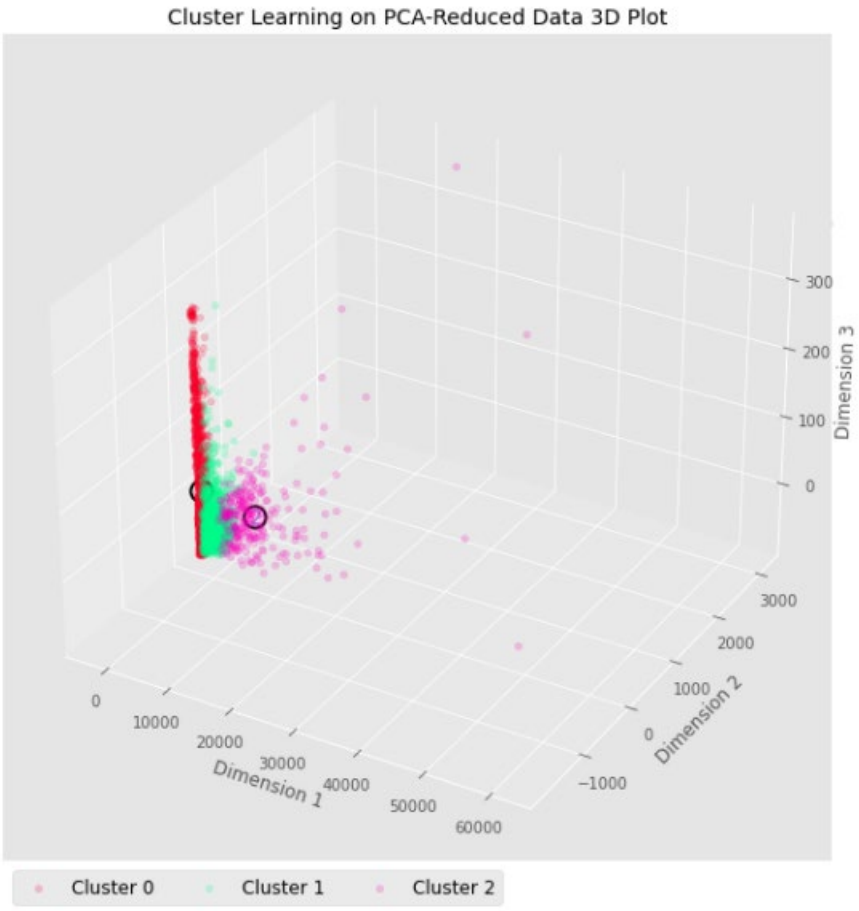

Fig 44. 3D Representation of the clustered Values

These clustered values are then split into train and test sets and passed into different models. Accuracy score is measured for both the train and test runs. Figure 45 and 46 depicts the accuracy scores returned by first run with the training data and second one with the test data.

```
Training LinearSVC ...
Average CV performance for LinearSVC: 0.941801 (in 0.194093 seconds)
Training LogisticRegression ...
Average CV performance for LogisticRegression: 0.943662 (in 0.0781603 seconds)
Training DecisionTreeClassifier ...
Average CV performance for DecisionTreeClassifier: 0.909674 (in 0.0311661 seconds)
Training SVC ...
Average CV performance for SVC: 0.939939 (in 0.170426 seconds)
Training RandomForestClassifier ...
Average CV performance for RandomForestClassifier: 0.940872 (in 0.744311 seconds)
```

Fig 45. Accuracy scores returned by models when training data is passed through

```
Testing LinearSVC ...
Average CV performance for LinearSVC: 0.937234 (in 0.0703406 seconds)
Testing LogisticRegression ...
Average CV performance for LogisticRegression: 0.944681 (in 0.0518496 seconds)
Testing DecisionTreeClassifier ...
Average CV performance for DecisionTreeClassifier: 0.912766 (in 0.0223699 seconds)
Testing SVC ...
Average CV performance for SVC: 0.939362 (in 0.0409722 seconds)
Testing RandomForestClassifier ...
Average CV performance for RandomForestClassifier: 0.945745 (in 0.515517 seconds)
```

Fig 46. Accuracy scores returned by models when training data is passed through

The results of this section can be analyzed in the following part.

7.2 Output analysis

The results gathered by the executions explained in the above sections are gathered together into a single table. This table can be analyzed further and the optimal solution for this specific scenario with these specific parameters can also be discussed here .

	classifier	Train_accuracy	Test_accuracy	Cluster_train_accuracy	Cluster_test_accuracy
0	LinearSVC	0.942274	0.936691	0.941801	0.937234
1	LogisticRegression	0.942271	0.942264	0.943662	0.944681
2	DecisionTreeClassifier	0.918997	0.927362	0.909674	0.912766
3	SVC	0.939938	0.931118	0.939939	0.939362
4	RandomForestClassifier	0.936689	0.932883	0.940872	0.945745

Fig 47. Consolidated results of all the runs

In the above Figure 47, the data shows the different machine learning algorithms that were used in this project, we used five different supervised machine learning algorithm and the accuracy score of each of those when passing the same exact data through it is recorded. As expected, we shall see that the Decision tree classifier has the lowest accuracy. The more enhanced version of the decision tree which is the random forest classifier seem to have an accuracy , as high that is on par if not a bit higher than the other three models. We can also see that, the GMM Clustering has

slight improvements in few models, whereas it remains the same in the others. The highest recorded accuracy is from RandomForestClassifier, whereas Logistic Regression seem to have the most consistent one.

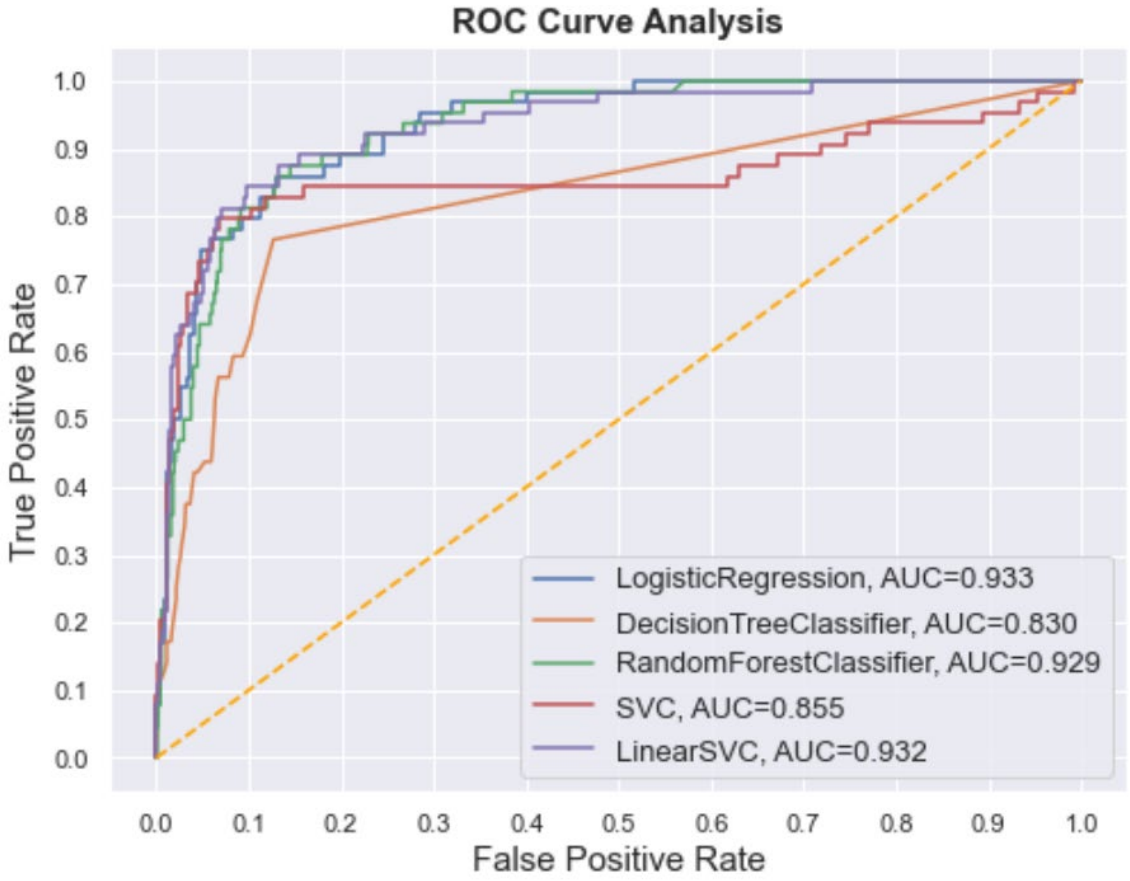

Fig 48. ROC Curve Analysis

Figure 48 depicts the ROC Curve, which measures how well the predictions are ranked rather than their absolute value. And it's a probability curve. The difference between the scores presents in the manually calculated table and the ROC Curve is mostly because, the ROC Curve depicts how the predictions are ranked.

Thus, we shall concluded with this analysis that the product is off higher accuracy.

7.3 Discussion

The success of the algorithm can be attributed to a lot of deciding factor which played its part and aided in arriving at a cohesive product with higher accuracy. Some of the factors can be, the exploratory analysis done, which helped in understanding the data better. The next one would be the formulation of RFM clusters and passing it through different models. This way, we have a variety of options and the sample run results to back the choice of the algorithm. The most optimal one for the scenario can be chosen from the varied tests. These factors, work cohesively and the result of that is a product with high accuracy.

8. Conclusions and recommendations

8.1 Summary

CLV is an important business metric, especially in retail, as it allows companies to make data-driven decisions in areas such as customer segmentation, resource allocation and evaluation of marketing campaigns. However, the modeling of CLV for retail is a challenge due to the lack of access to historical data of all the factors that affect customer purchase decisions, like the launching of a new product, tax rates, competitors' pricing, etc.

In this research, we implemented and evaluated supervised and unsupervised machine learning techniques to predict customer loyalty using a dataset of customer purchase history belonging to an online retailer based in the UK. After establishing theoretical groundwork, several machine learning solutions were explored. We began by calculating RFM scores and segmenting customers accordingly using K-means clustering. This clustered data was then fed to several supervised machine learning models, whereby the data was trained, tested, and the models evaluated. Next, an additional layer of clustering was applied to the clustered RFM data in order to give additional insights into the data and improve prediction. For this, GMM clustering was used. This double-clustered data was then fed to supervised machine learning models, whereby the data was again trained, tested, and the models evaluated.

8.2 Conclusion

After performing all the different scenario runs, we have observed Logistic Regression appears to be the most consistent solution of all the tested techniques. It is also observed that the RFM statistics comes prominent when dealing with customer lifetime value prediction. Alan

Turing once quoted that “What we want is a machine that can learn from experience”. Abiding to this quote, we shall observe that the mode than data is fed into the system, the learning increases which in turn also raises the accuracy percentage. interpretation of data using the feature engineering plays a major part in how accurate the results are. There are a few limitations of our research that should be highlighted. The proposed frameworks for predicting customer loyalty were developed for existing customers only. Our frameworks were developed using 2-year data and tested only on a faction of that. Our predictions can be significantly improved using a dataset with a broader date range. Another limitation, as mentioned previously, is that our dataset lacked all the factors that could affect purchase patterns, including the cultural aspect of each country represented in our dataset. Taking such factors into consideration would vastly improve our prediction accuracy.

8.3 Recommendations for future studies

As with any research, there is always room for improvement and further study. Below we provide recommendations for future studies:

1. CLV prediction can be applied not just to the retail industry but also to a variety of other industries, including tech and gaming. Therefore, we recommend extending our modeling into other industries.
2. Explore Markov Chain probabilistic model for making churn predictions.
3. Segment our customers not just based on standard RFM scores but also based on their preference features.

9 Bibliography

(a) A. Kasprova, “Customer lifetime value for retail based on transactional and Loyalty Card Data,” Домівка, 01-Jan-1970. [Online].

(b) Josef Bauer and Dietmar Jannach," Improved Customer Lifetime Value Prediction With Sequence-To-Sequence Learning and Feature-Based Models", ACM Transactions on Knowledge Discovery from Data, Vol. 15, No. 5, Article 80. April 2021.

(c) Michael C. Mozer, Richard Wolniewicz, David B. Grimes, Eric Johnson, and Howard Kaushansky,"Predicting Subscriber Dissatisfaction and Improving Retention in the Wireless Telecommunications Industry", IEEE Transactions

(d) Gaurav Sharma, Prashant Singh Rana and Seema Bawa, “Hybrid machine learning models for predicting types of Human T-cell Lymphotropic Virus”, IEEE Transactions on Computational Biology and Informatics, pp.1-12, July 2019.

(e) https://archive.ics.uci.edu/ml/datasets/Online+Retail+II

(f) https://cloud.google.com/architecture/clv-prediction-with-offline-training-train

(g) https://github.com/andirs/hr_predict

(h) https://towardsdatascience.com/understanding-and-forecasting-customer-lifetime-value-cltv-634fe34f522b

(i) https://towardsdatascience.com/data-driven-growth-with-python-part-3-customer-lifetime-value-prediction-6017802f2e0f