

Customer Market Segmentation using Cluster Analysis

Machine Learning Project Report
April 17th, 2022

TABLE OF CONTENT

S. No	TOPICS	Page No.
1	INTRODUCTION	03
2	DATASET	03
3	DATA PREPROCESSING	04
4	PRINCIPAL COMPONENT ANALYSIS	07
4	ELBOW METHOD AND SCALING	08
5	CLUSTER ANALYSIS	08
6	FEATURE SELECTION	11
7	MODEL IMPLEMENTATION	14
8	CONCLUSION AND LIMITATIONS	15

INTRODUCTION

Market segmentation is a key component of any effective marketing plan since it enables businesses to categorize their clients according to their specific wants and needs. Clustering is an effective form of market segmentation that involves putting clients into groups according to how similar they are in terms of attributes, such as demographic data, purchasing tendencies, or psychographic characteristics. Businesses can better adapt their marketing initiatives and product offers to each group's unique needs by recognizing and understanding these diverse client segments. In this project, we will split clients into specific clusters using clustering methodologies to create more efficient marketing plans and eventually promote corporate growth.

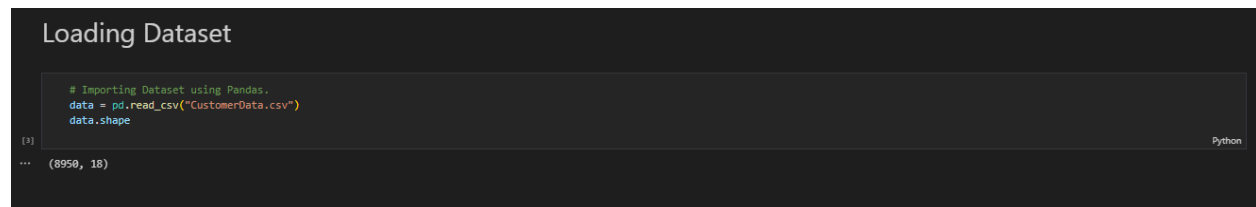
To segment customer, we are using different techniques such as data preprocessing, feature scaling, principal component analysis, clustering, and supervised learning model to accurately predict from which market the customer belongs. In the end, we are creating a webapp using streamlet API.

Objective

Market segmentation is the practice of partitioning a large consumer or commercial market based on shared traits, typically including current and potential clients. In this project our aim is to segment our customer based on similarity and liked characteristics. For this purpose, we are using unsupervised learning techniques such as clustering and principal component analysis (PCA) to divide customers into segments.

About The Dataset

The data set which we are using in this project is taken from an online Kaggle repository. The dataset contains around 9000 records of customers and 18 independent features. Dataset does not contain any class label or target variable, so this is basically an unsupervised learning project in which we create our target variable by performing clustering analysis.



```
Loading Dataset

# Importing Dataset using Pandas.
data = pd.read_csv("CustomerData.csv")
data.shape

[3]
... (8950, 18)
```

Python

Data Preprocessing

Before using the data in machine learning algorithms, preparing the data is a crucial step. Data preprocessing is a data mining technique used to turn raw data into a format that is both practical and effective.

For our dataset, we are also performing data preprocessing. The steps we performed for preprocessing of data are below:

1- Checking Data Type of the Features

The first thing we need to check after loading our data set are the data types of the features. All independent features must be in numerical format before performing any machine learning on the dataset.

In our dataset, as we see all the features are in numerical format.

```
data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   CUST_ID                               8950 non-null   object
1   BALANCE                               8950 non-null   float64
2   BALANCE_FREQUENCY                     8950 non-null   float64
3   PURCHASES                             8950 non-null   float64
4   ONEOFF_PURCHASES                     8950 non-null   float64
5   INSTALLMENTS_PURCHASES               8950 non-null   float64
6   CASH_ADVANCE                         8950 non-null   float64
7   PURCHASES_FREQUENCY                 8950 non-null   float64
8   ONEOFF_PURCHASES_FREQUENCY          8950 non-null   float64
9   PURCHASES_INSTALLMENTS_FREQUENCY    8950 non-null   float64
10  CASH_ADVANCE_FREQUENCY               8950 non-null   float64
11  CASH_ADVANCE_TRX                     8950 non-null   int64
12  PURCHASES_TRX                       8950 non-null   int64
13  CREDIT_LIMIT                         8949 non-null   float64
14  PAYMENTS                             8950 non-null   float64
15  MINIMUM_PAYMENTS                    8637 non-null   float64
16  PRC_FULL_PAYMENT                     8950 non-null   float64
17  TENURE                               8950 non-null   int64
dtypes: float64(14), int64(3), object(1)
memory usage: 1.2+ MB
```

2- Checking Null Values

Next and most important step in data processing is checking null values. Null values are the values which are empty records in the data set. We have checked the null values in our dataset, and we can see that in one of the columns there are some null values present, so we have to compute mean of that column and replace the null values with the average of the feature values so that the distribution of the attribute doesn't get disturbed.

```

# Checking Null values in all features.
data.isnull().sum()

```

```

[7]
... CUST_ID                0
    BALANCE              0
    BALANCE_FREQUENCY    0
    PURCHASES            0
    ONEOFF_PURCHASES     0
    INSTALLMENTS_PURCHASES 0
    CASH_ADVANCE         0
    PURCHASES_FREQUENCY  0
    ONEOFF_PURCHASES_FREQUENCY 0
    PURCHASES_INSTALLMENTS_FREQUENCY 0
    CASH_ADVANCE_FREQUENCY 0
    CASH_ADVANCE_TRX     0
    PURCHASES_TRX        0
    CREDIT_LIMIT         1
    PAYMENTS             0
    MINIMUM_PAYMENTS     313
    PRC_FULL_PAYMENT     0
    TENURE               0
    dtype: int64

```

```

# Computing Mean of every column.
columns_means = data.mean()
print(columns_means)

```

```

[8]
... BALANCE                1564.474828
    BALANCE_FREQUENCY      0.877271
    PURCHASES              1003.204834
    ONEOFF_PURCHASES       592.437371
    INSTALLMENTS_PURCHASES 411.067645
    CASH_ADVANCE           978.071112
    PURCHASES_FREQUENCY    0.490351
    ONEOFF_PURCHASES_FREQUENCY 0.202458
    PURCHASES_INSTALLMENTS_FREQUENCY 0.364437
    CASH_ADVANCE_FREQUENCY 0.135144
    CASH_ADVANCE_TRX       3.248827
    PURCHASES_TRX         14.709832
    CREDIT_LIMIT           4494.449450
    PAYMENTS              1733.143852
    MINIMUM_PAYMENTS       864.206542
    PRC_FULL_PAYMENT       0.153715
    TENURE                11.517318
    dtype: float64

```

```

# Adding mean value in null spaces of column Genetic Preigne.
data['MINIMUM_PAYMENTS'] = data['MINIMUM_PAYMENTS'].fillna(864.206)
data['CREDIT_LIMIT'] = data['CREDIT_LIMIT'].fillna(4494.44)

```

```

[9]

```

3- Dropping Unnecessary Column

We are dropping ID column from our dataset, because such variable and not much impactful in predicting the label or performing clustering, although these features may cause overfitting in thr model.

```

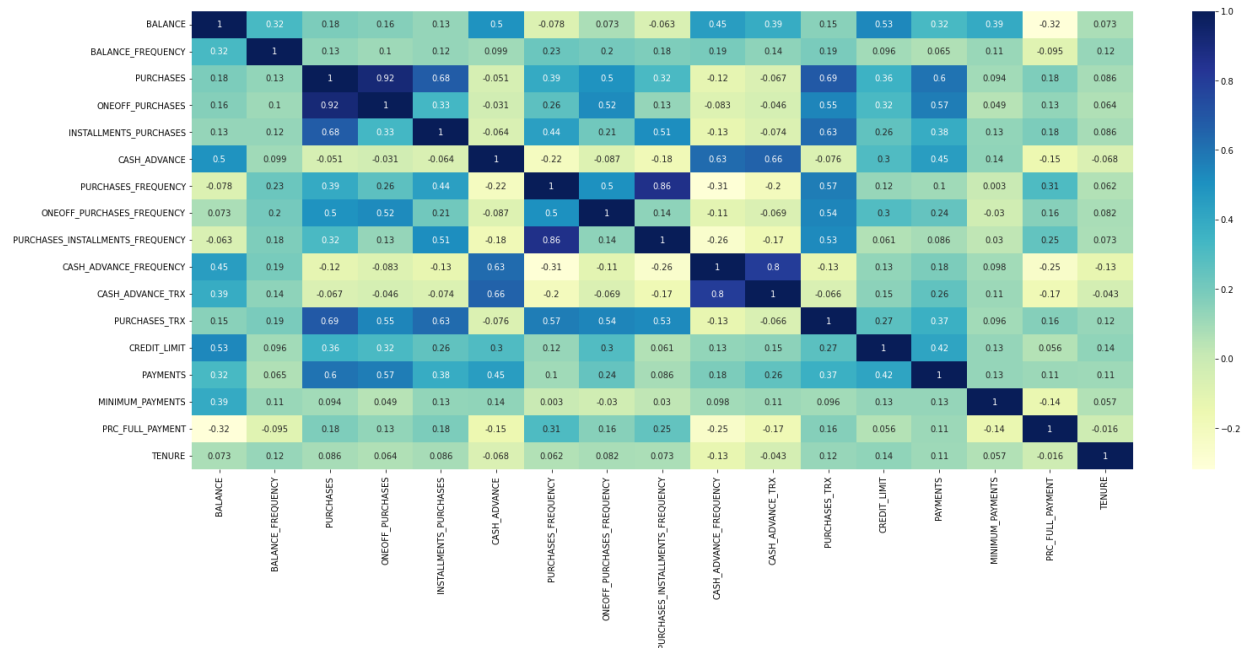
# Dropping Unnecessary Column.
data.drop('CUST_ID',axis=1,inplace=True)

```

```

[10]

```



From the correlation matrix, we can observe that the purchases column is strongly correlated with the other purchases columns such as on off purchases etc. Similarly balance variables are highly correlated with the other balance features such as balance frequency, credit card limit etc. There are also multiple multicollinearities present in the data which we can see in the plot as blue blocks.

SCALING DATASET

In machine learning, scaling is a common preprocessing step that involves converting the values of the dataset's feature values to a standard scale. This is done to make sure that every feature, regardless of scale or measurement units used in the past, contributes equally to the model.

In our project, we are also performing Min Max scaling technique.

```
Scaling Dataset

from sklearn.preprocessing import MinMaxScaler
scaling = MinMaxScaler()

scaling.fit(data)
data_scaled = scaling.transform(data)
```

[14] Python

Principal Component Analysis (PCA)

Dimensionality reduction in machine learning and data analysis is accomplished by using the Principal Component Analysis (PCA) method. It entails turning a dataset of potentially correlated variables into a new set of uncorrelated variables, known as principal components, that account for the most variance in the original dataset.

Identifying the underlying structure in the data and capturing as much variance as feasible with fewer features are the two objectives of PCA. The performance of a machine learning model can be improved, the computational complexity of the model can be decreased, and high-dimensional data can be visualized in a lower-dimensional environment, among other uses.

The clustering algorithm's performance and interpretability can both be enhanced with PCA. The clustering algorithm may be able to discover significant clusters more precisely and quickly by lowering the dimensionality of the dataset and determining the most crucial factors. Principal components can also be used to create new clustering features that may be more understandable and appealing to the eye than the original variables.

We are creating two principal components from the dataset of 18 features. So now our features information will be represented by only two components.

```
Principal Component Analysis

#use pca to reduce dimensionality
from sklearn.decomposition import PCA
from matplotlib import pyplot as plt

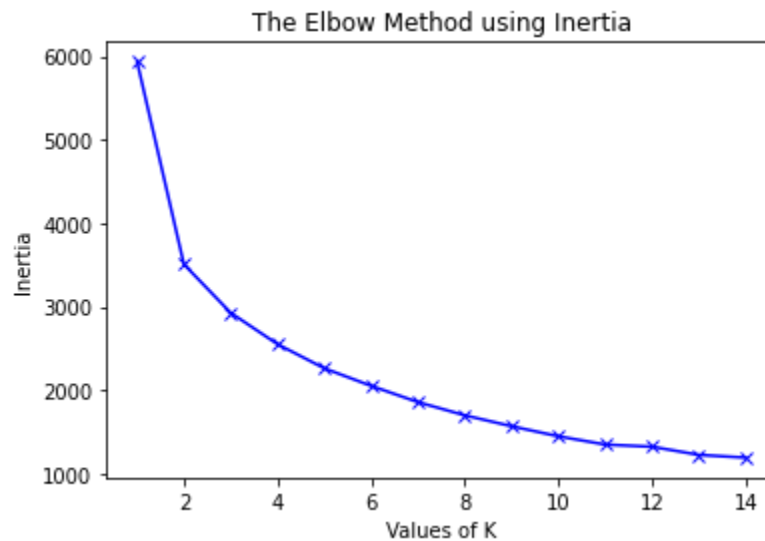
pca = PCA(n_components=2)
reduced_matrix = pca.fit_transform(data_scaled)
pca_df = pd.DataFrame(data=reduced_matrix, columns=['PCA 1', 'PCA 2'])
pca_df
```

	PCA 1	PCA 2
0	-0.482164	-0.097656
1	-0.608577	-0.019379
2	0.304507	0.920946
3	-0.588622	-0.005605
4	-0.554354	0.052965
...
8945	0.646921	-0.463044
8946	0.549893	-0.406711
8947	0.360389	-0.422647
8948	-0.646095	-0.204332
8949	-0.094598	0.399578

8950 rows x 2 columns

Elbow Method

The Elbow approach is a well-liked strategy for choosing the ideal number of clusters in a dataset in machine learning and data analysis. In this method, the number of clusters is chosen at the "elbow" of the plot, where the rate of reduction in WSS starts to slow down, by graphing the within-cluster sum of squares (WSS) versus the number of clusters.



The Elbow method is suggesting us with 2 number of clusters, but we will try for 3 and 4 as well because elbow method is not accurate all the time.

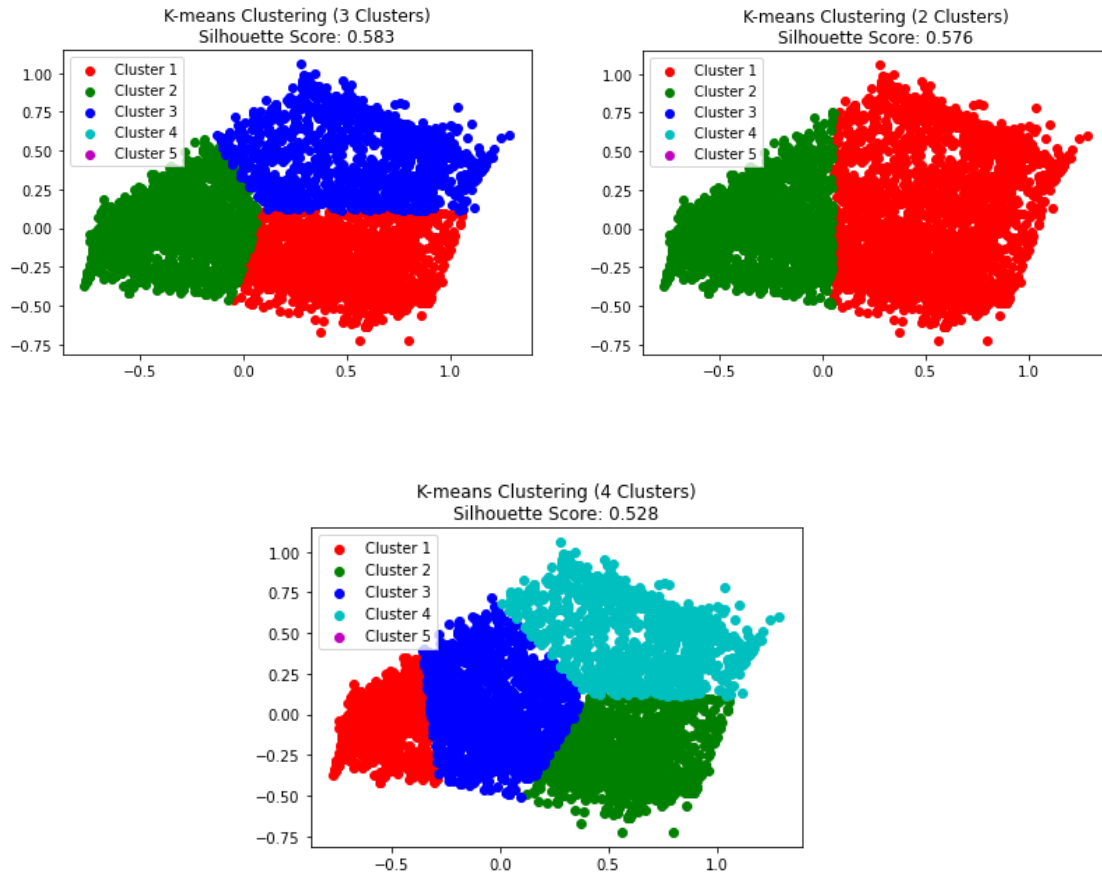
Clustering Analysis

for customer segmentation, we are using different clustering algorithms which are discussed below:

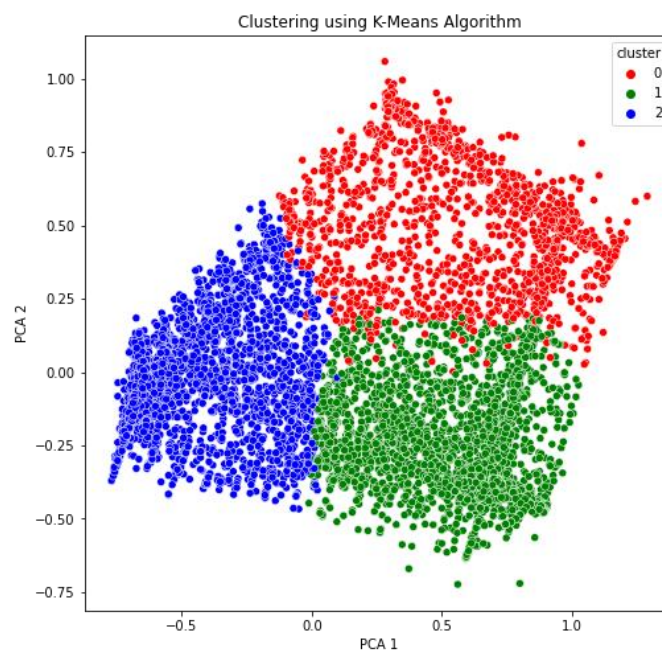
1- K Mean Clustering

Unsupervised machine learning technique K-means clustering is frequently used to divide a dataset into k clusters. It is a quick and effective algorithm that, depending on how closely its features resemble the cluster centroids, iteratively assigns each observation to one of the k clusters.

In our project, we got elbow score of 2 clusters, so we are implementing K Mean with 2, 3 and 4 clusters and checking out the silhouette score of each.



As we can see, we are getting the highest silhouette score with 3 numbers of clusters.

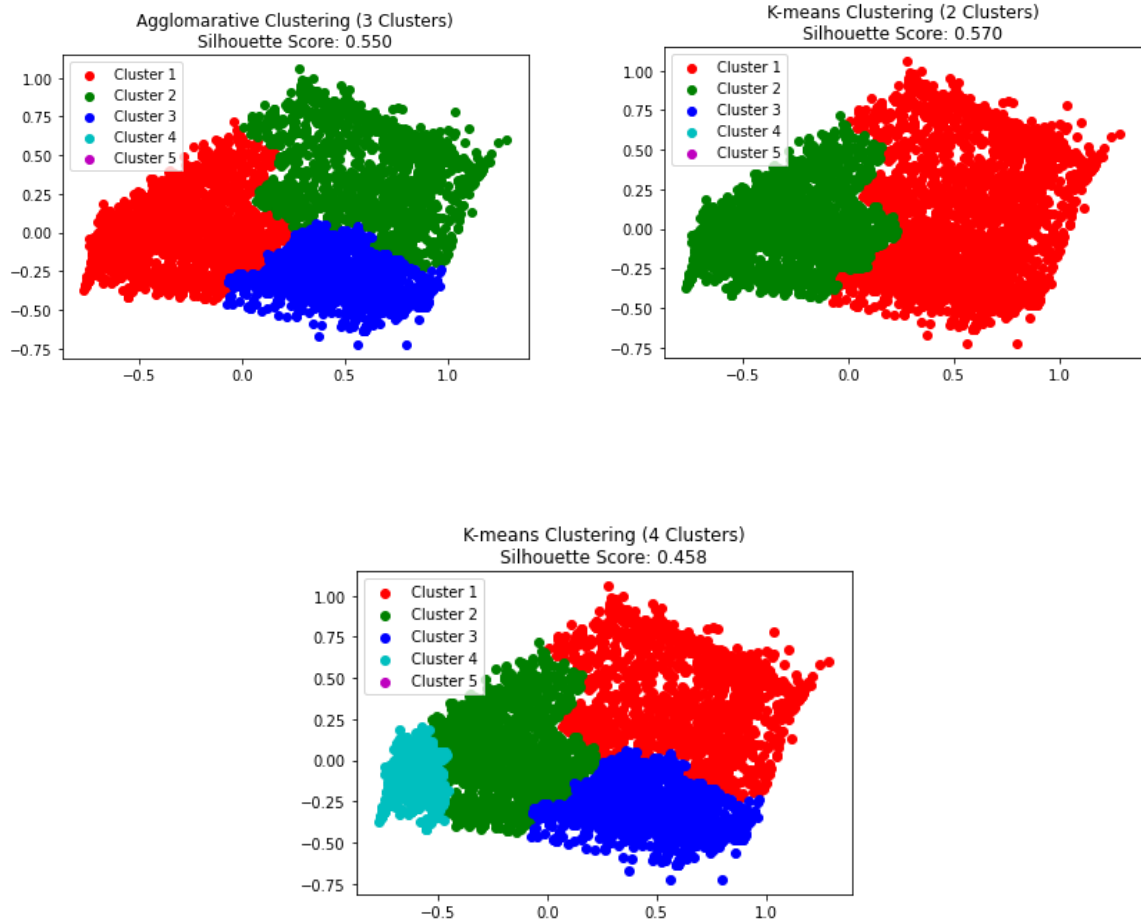


2- Agglomerative Clustering

Agglomerative clustering is a hierarchical clustering technique in which each observation is first placed in a distinct cluster, and then the algorithm repeatedly merges the two clusters that are closest to one another until only one cluster is left.

The technique gathers the nearest two observations into a cluster by measuring the distance between each pair of observations in the dataset. The linking criterion, which establishes how to calculate the distance between two clusters based on the distances between their individual observations, is then used to determine the distance between clusters.

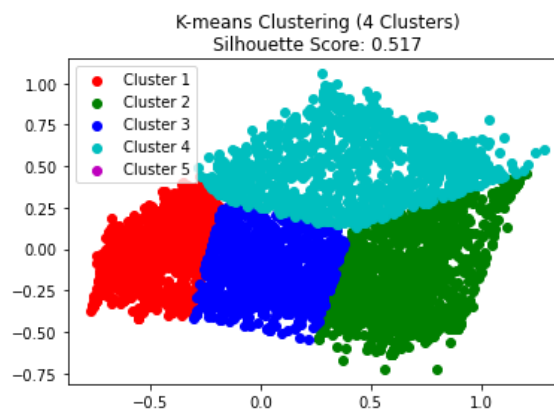
For agglomerative Clustering we are getting the best score with 2 number of clusters, but its score is less than K means.



3- Mean Shift Clustering

A non-parametric clustering approach called mean shift clustering allocates each data point to the closest cluster center, which is the mean of the data points within a specified radius. Up until convergence, the procedure, which is iterative, moves the cluster centers to the median of the data points within a predetermined radius.

The procedure begins by choosing a set of initial cluster centers, which may be done at random or using knowledge from the past. The technique calculates the mean of the data points within a specific radius (referred to as the bandwidth) surrounding each cluster center. After shifting the cluster center to the new mean, the procedure is repeated until convergence.



After performing clustering analysis, we have concluded that we are getting best scores on creating 3 clusters using k mean clustering algorithm.

Inverse Scaling

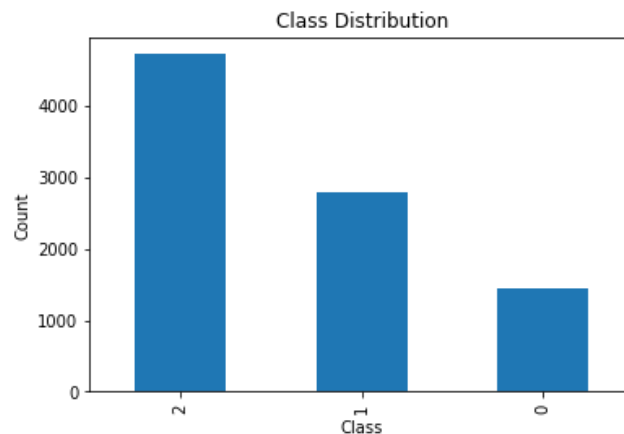
A scaled or normalized value is converted back to its original scale or units through the process of "inverse scaling." To guarantee that all features are on a comparable size and to enhance the performance of machine learning models, it is common practice in data preprocessing to apply scaling or normalization techniques like min-max scaling or standardization.

```
~ Inverse Scaling
Now we are scaling back to original form of the data.

# find all cluster centers
cluster_centers = pd.DataFrame(data=kmeans_model.cluster_centers_, columns=[data.columns])
# inverse transform the data
cluster_centers = scaling.inverse_transform(cluster_centers)
cluster_centers = pd.DataFrame(data=cluster_centers, columns=[data.columns])
cluster_centers
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURCHASES_INSTALLMENTS_FREQ
0	1924.914811	0.973706	3197.352536	2393.314193	804.038344	658.917943	0.893007	0.798486	0
1	1139.550204	0.914001	1066.589684	268.991991	798.242293	503.092784	0.870993	0.098667	0
2	1704.586025	0.826193	295.799164	232.894470	63.093763	1356.714019	0.143265	0.081543	0

Now after doing inverse scaling, we are creating a new feature in a dataset and assigning records with their respective clusters which are allotted to them using K mean clustering. After this step, we have our target label which is a cluster. So now we can implement supervised learning to predict the target variable 'cluster'. We have 3 numbers of clusters so it's a multi class classification problem now.



TRAIN TEST SPLIT

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. Here we are splitting our dataset into train dataset and test dataset with the ratio of 70-30% where 70% is our train dataset and 30% is our test dataset.

```
X = cluster_df.loc[:, cluster_df.columns != 'Cluster'] # All columns except target variable.
y = cluster_df[['Cluster']] # Target Variable.

Train Test Split

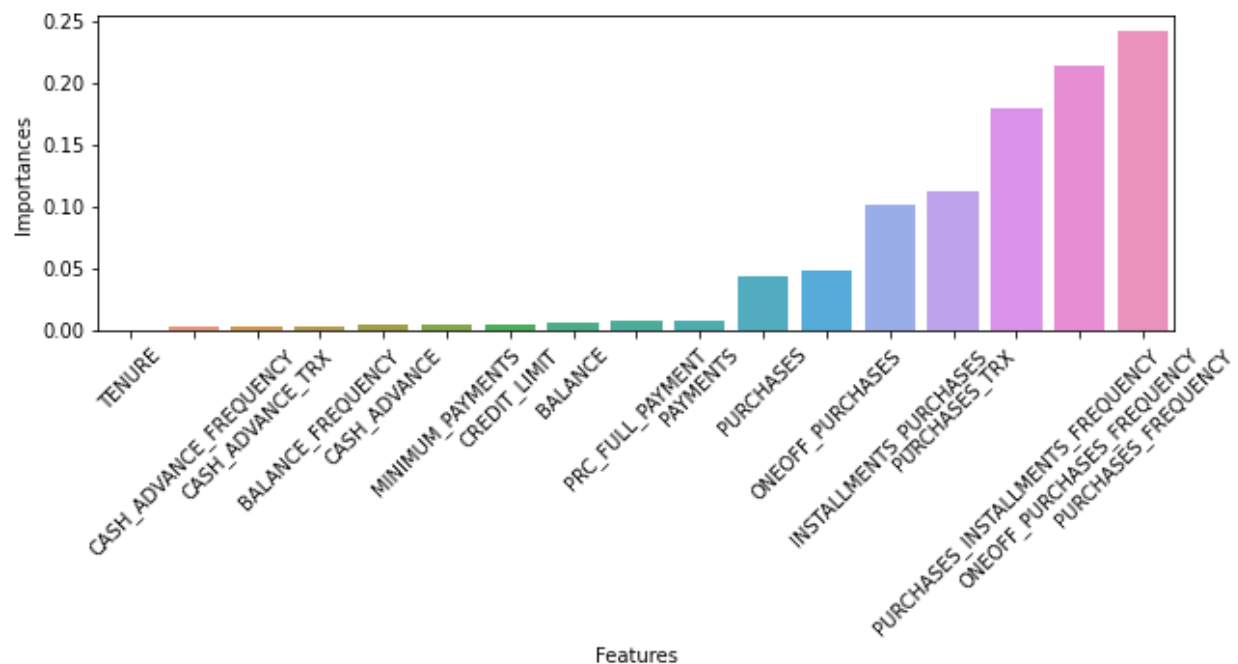
# we are splitting dataset with 70-30 ratio.
from sklearn.model_selection import train_test_split
trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.3, random_state=2)
print(trainX.shape)
print(testX.shape)
print(trainy.shape)
print(testy.shape)
```

```
... (6265, 17)
(6265, 1)
(2685, 17)
(2685, 1)
```

Feature Selection Using Random Forest

Feature selection using random forest involves using a random forest model to identify the most impactful features in a data. Random forest is an ensemble learning algorithm that works by forming multiple decision trees and combining their outputs predictions. In feature selection using random forest, the model determines the importance of each feature by calculating the decrease in impurity (e.g., Gini index or entropy) that is achieved when that feature is used to split the data at a node in the decision tree.

The random forest algorithm builds the decision trees and then determines the average decrease in impurity for each feature across all the trees. Characteristics are more significant if their average impurity decrease is higher. A subset of the most crucial traits might be chosen for additional study based on this rating. Because of the decreased overfitting and enhanced interpretability, this can improve model performance.



Model Implementation: -

At first, we are implementing different machine learning classifier on our dataset first without any feature selection and then after doing selection with different methods of feature selection. We have 25 attributes at first, and our data is 70-30% divided into train and test sets.

1- Decision Tree Classifier

The first model we implemented is the decision tree classifier. A decision tree is a supervised machine learning model that is used for classification and regression tasks. It is a tree-like model that consists of nodes and branches, where each node represents a feature or attribute, and each branch represents a decision rule or condition based on that feature or attribute.

In a classification task, the goal of a decision tree is to divide the dataset into smaller and more homogeneous groups, where each group corresponds to a particular class or category.

The decision tree gives us accuracy of 94% on our full feature dataset, but its accuracy increases after reducing dimension and use only top performing features. The highest accuracy obtained by decision tree is after selecting feature 97%.

2- Random Forest Classifier

The second model we are implementing after decision tree is random forest classifier. A Random Forest Classifier is a supervised machine learning algorithm that uses an ensemble of decision trees to perform classification tasks. The random forest algorithm is an extension of the decision tree algorithm, which overcomes some of the limitations of decision trees such as overfitting and instability.

On our dataset, random forest is performing well, giving accuracy of 98% accuracy on full feature dataset and 98.6% accuracy after feature selection.

3- Gradient Boosting Classifier

One of the famous boosting algorithms, which combines different weak learners into strong by minimizing the loss function. It is an ensemble technique. The goal of gradient boosting classifier is to iteratively add new models to the ensemble that correct the errors of the previous models.

In our project Gradient boosting is performing well on both full feature and selected feature dataset. It is getting the highest accuracy of 98.4% after feature selection.

4- XGBoost Classifier

A well-known machine learning technique that is a member of the gradient boosting family is called XGBoost (Extreme Gradient Boosting) Classifier. It is a development of the conventional gradient boosting approach, which minimizes a loss function to combine several weak models into a strong one. High performance and accuracy are hallmarks of XGBoost, particularly when applied to structured data issues like feature engineering and tabular data. In comparison to other gradient boosting methods, XGBoost key benefit is its scalability, which enables it to handle huge datasets with millions of rows and columns.

In our project Xtreme Gradient boosting is performing better than all the other models, maybe because of the reason that XGBoost can capture non-linear relationships between the features and the target variable. It is getting the highest accuracy of 98.7% after feature selection.

5- K Nearest Neighbor

K Nearest Neighbor (KNN) is a simple machine learning algorithm used for classification tasks. The algorithm works by finding the K nearest neighbors of a new data point based on some distance metric, where K is a hyperparameter specified by the user. The class of the new data point is then determined by majority vote among its K nearest neighbors. In our project, KNN is working fine, and its accuracy was 76% with full feature dataset and 81% after doing feature selection.

Conclusion

So, in this project, we have a data set without any label, so we implemented how do you spell wise lighting techniques such as PCA and perform cluster analysis to allocate a label in our data set. We have got are highest silhouette score on three number of clusters using K mean clustering algorithm whereas an elbow method suggests two numbers of clusters. The other clustering algorithms which we are using are agglomerative clustering and mean shift algorithm.

After getting the label attached in a dataset, we are performing machine learning to predict our class label which is a cluster using different supervised learning algorithms both on full features and after selecting impactful features using random forest model.

Then we are saving our best model which is a random forest model because it is performing well as compared to the other models both before the feature selection and after the feature selection and it is more generalizable as compared to boosting algorithms.

We observed that KNN is performing low out of all the other models maybe because KNN model prone to overfitting and not much robust as compared to the other models like random forest and boosting algorithms. KNN requires calculating distances between each pair of data points, which can be computationally expensive when the dataset is large. This can result in longer training and prediction times, making KNN less practical for some applications and Time complexity of KNN increase with the number of features.

Limitations

- 1- For future work, we can also implement other clustering algorithms like optics, DB scan and spectral clustering etc.
- 2- We can also implement these experiments on large-scale data where the records of the customers are more so that our model trained well.
- 3- For this project only 5 algorithms were used, there might be other supervised algorithms such as Naïve Bayes, SVM etc. that can also be used to check if they are performing better or low. Other than that, hyper parameter tuning of models can also be used to increase efficiency.