Market Segmentation In Insurance

WISDOM IZUCHUKWU ADIKE

Contents

Dataset Description

Introduction

In marketing, market segmentation is the process of dividing a broad consumer or business market, normally consisting of existing and potential customers, into subgroups of consumers based on some type of shared characteristics. This case requires developing a customer segmentation to give recommendations like saving plans, loans, wealth management, etc. on target customer groups. The sample Dataset summarizes the usage behavior of about 9000 active credit cardholders during the last 6 months. The file is at a customer level with 18 behavioral variables.

Dataset Description 01

	0	1	2	3	4
CUST_ID	C10001	C10002	C10003	C10004	C10005
BALANCE	40.900749	3202.467416	2495.148862	1666.670542	817.714335
BALANCE_FREQUENCY	0.818182	0.909091	1.0	0.636364	1.0
PURCHASES	95.4	0.0	773.17	1499.0	16.0
ONEOFF_PURCHASES	0.0	0.0	773.17	1499.0	16.0
INSTALLMENTS_PURCHASES	95.4	0.0	0.0	0.0	0.0
CASH_ADVANCE	0.0	6442.945483	0.0	205.788017	0.0
PURCHASES_FREQUENCY	0.166667	0.0	1.0	0.083333	0.083333
ONEOFF_PURCHASES_FREQUENCY	0.0	0.0	1.0	0.083333	0.083333
PURCHASES_INSTALLMENTS_FREQUENCY	0.083333	0.0	0.0	0.0	0.0
CASH_ADVANCE_FREQUENCY	0.0	0.25	0.0	0.083333	0.0
CASH_ADVANCE_TRX	0	4	0	1	0
PURCHASES_TRX	2	0	12	1	1
CREDIT_LIMIT	1000.0	7000.0	7500.0	7500.0	1200.0
PAYMENTS	201.802084	4103.032597	622.066742	0.0	678.334763
MINIMUM_PAYMENTS	139.509787	1072.340217	627.284787	NaN	244.791237
PRC_FULL_PAYMENT	0.0	0.222222	0.0	0.0	0.0
TENURE	12	12	12	12	12

About the features

- We have 18 variables in Market Segmentation dataset with more focus on the 'TENURE' columns.
- (rows, columns) | (8950, 18)

Dataset Description 02

- **CUST_ID:** Customer ID or identifier.
- **BALANCE:** Current account balance.
- **BALANCE_FREQUENCY:** Frequency of maintaining a balance.
- **PURCHASES:** Total purchases made.
- **ONEOFF_PURCHASES:** Purchases made as one-time transactions.
- **INSTALLMENTS_PURCHASES:** Purchases made in installments.
- **CASH_ADVANCE:** Total cash advances taken.
- **PURCHASES_FREQUENCY:** Frequency of purchase transactions.
- **ONEOFF_PURCHASES_FREQUENCY:** Frequency of one-off purchase transactions.
- **PURCHASES_INSTALLMENTS_FREQUENCY:** Frequency of installment purchase transactions.
- **CASH_ADVANCE_FREQUENCY:** Frequency of cash advances.
- **CASH_ADVANCE_TRX:** Number of cash advance transactions.
- **PURCHASES_TRX:** Number of purchase transactions.
- **CREDIT_LIMIT:** Credit limit on the account.
- **PAYMENTS:** Total payments made.
- **MINIMUM_PAYMENTS:** Minimum required payments.
- **PRC_FULL_PAYMENT:** Percentage of full payments.
- **TENURE:** Account tenure in months.

Main Objective of the analysis:

In this section, we will conduct a thorough examination of the dataset using various Exploratory Data Analysis (EDA) methods. These methods will involve investigating null values, assessing data skewness, and employing data visualization. Additionally, we will illustrate the relationships between the dataset's features to facilitate feature engineering and data cleaning processes.

Exploratory Data Analysis (EDA) and Feature Engineering

Exploratory Data Analysis 01

```
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                        0
PAYMENTS                            0
MINIMUM_PAYMENTS                    0
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

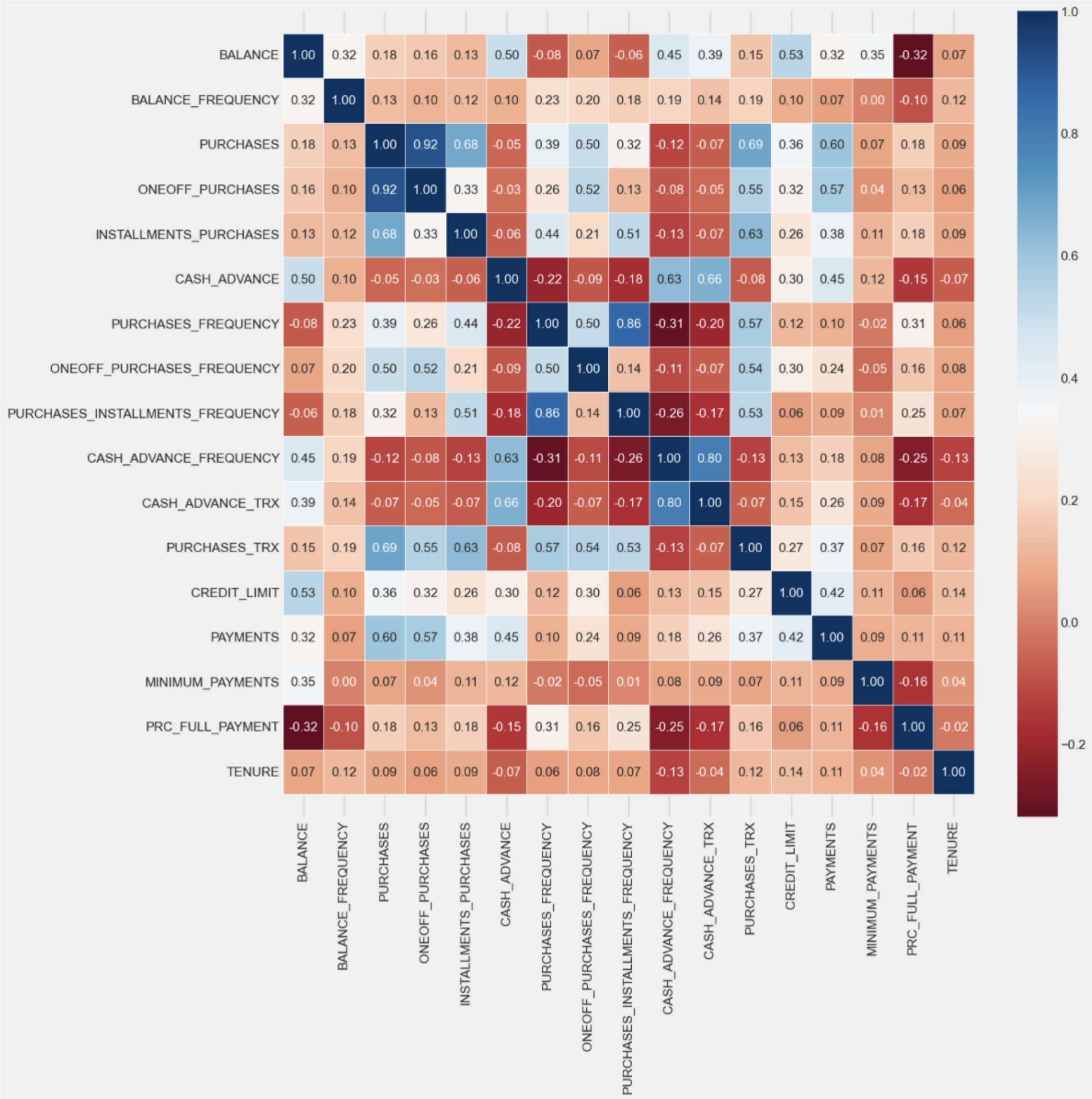
- First of all, we dropped the CUST_ID column since it will be irrelevant for our machine learning algorithm.
- Then, we treated the missing values by replacing the NaN with the mean value of the numerical columns and the modal value of the categorical columns with where MINIMUM_PAYMENTS and CREDIT_LIMIT columns contains 313 and 1 missing values respectively.

Exploratory Data Analysis 02

```
BALANCE                             float64
BALANCE_FREQUENCY                   float64
PURCHASES                           float64
ONEOFF_PURCHASES                    float64
INSTALLMENTS_PURCHASES              float64
CASH_ADVANCE                        float64
PURCHASES_FREQUENCY                 float64
ONEOFF_PURCHASES_FREQUENCY          float64
PURCHASES_INSTALLMENTS_FREQUENCY    float64
CASH_ADVANCE_FREQUENCY              float64
CASH_ADVANCE_TRX                      int64
PURCHASES_TRX                         int64
CREDIT_LIMIT                        float64
PAYMENTS                            float64
MINIMUM_PAYMENTS                    float64
PRC_FULL_PAYMENT                    float64
TENURE                                int64
dtype: object
```

- Identifying numerical variables that are categorical by their data types which is int64.
- While float64 are numerical variables that will be suitable for scaling.

Exploratory Data Analysis 03



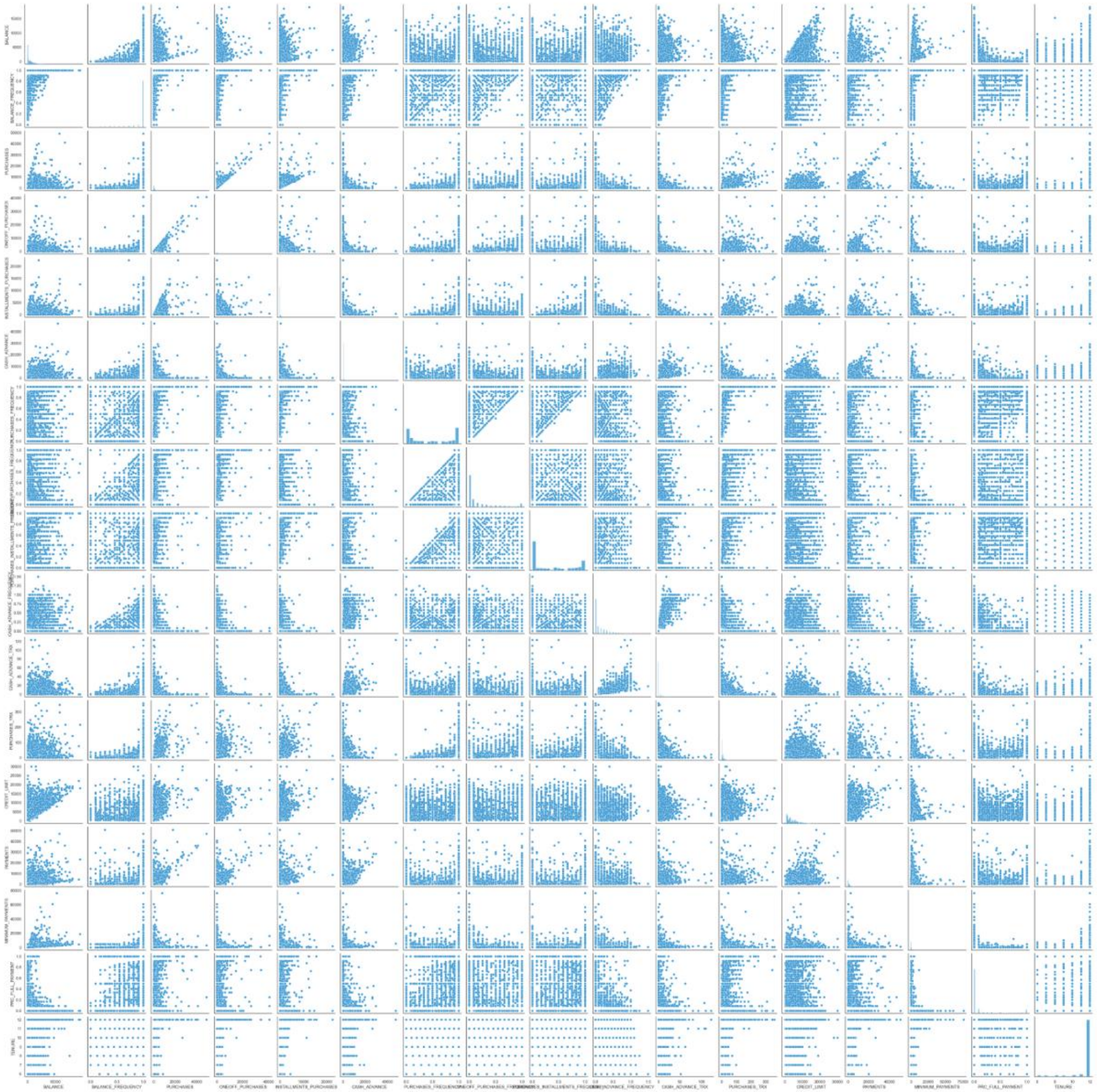
The objective of this matrix is to display the associations among features, which proves beneficial for upcoming feature engineering techniques. In the following slides, we will provide a more detailed presentation of these correlations using data frames for better clarity.

Exploratory Data Analysis 03

	1st Feature	2nd Feature	Correlation
0	PURCHASES	ONEOFF_PURCHASES	0.916845
1	ONEOFF_PURCHASES	PURCHASES	0.916845
2	PURCHASES_INSTALLMENTS_FREQUENCY	PURCHASES_FREQUENCY	0.862934
3	PURCHASES_FREQUENCY	PURCHASES_INSTALLMENTS_FREQUENCY	0.862934
4	CASH_ADVANCE_FREQUENCY	CASH_ADVANCE_TRX	0.799561
5	CASH_ADVANCE_TRX	CASH_ADVANCE_FREQUENCY	0.799561
6	PURCHASES_TRX	PURCHASES	0.689561
7	INSTALLMENTS_PURCHASES	PURCHASES	0.679896
8	CASH_ADVANCE	CASH_ADVANCE_TRX	0.656498
9	PAYMENTS	PURCHASES	0.603264
10	ONEOFF_PURCHASES_FREQUENCY	PURCHASES_TRX	0.544869
11	CREDIT_LIMIT	BALANCE	0.531267
12	BALANCE	CREDIT_LIMIT	0.531267
13	MINIMUM_PAYMENTS	BALANCE	0.353675
14	BALANCE_FREQUENCY	BALANCE	0.322412
15	PRC_FULL_PAYMENT	BALANCE	0.318959
16	TENURE	CREDIT_LIMIT	0.139034

It's evident that there exists a strong correlation among the features, primarily because a significant portion of them pertains to geometric characteristics that describe the shape of the dataset.

Exploratory Data Analysis 04



By utilizing a pair plot, we can examine the linearity between the features and also gain insight into how the dataset are distributed, forming distinct clusters.

Exploratory Data Analysis 05

	Skewness Value
MINIMUM_PAYMENTS	12.283207
ONEOFF_PURCHASES	10.045083
PURCHASES	8.144269
INSTALLMENTS_PURCHASES	7.299120
PAYMENTS	5.907620
CASH_ADVANCE_TRX	5.721298
CASH_ADVANCE	5.166609
PURCHASES_TRX	4.630655
BALANCE	2.393386
PRC_FULL_PAYMENT	1.942820
CASH_ADVANCE_FREQUENCY	1.828686
ONEOFF_PURCHASES_FREQUENCY	1.535613
CREDIT_LIMIT	1.522549
PURCHASES_INSTALLMENTS_FREQUENCY	0.509201
PURCHASES_FREQUENCY	0.060164
BALANCE_FREQUENCY	-2.023266
TENURE	-2.943017

	skewness_value
MINIMUM_PAYMENTS	12.283207
ONEOFF_PURCHASES	10.045083
PURCHASES	8.144269
INSTALLMENTS_PURCHASES	7.299120
PAYMENTS	5.907620
CASH_ADVANCE_TRX	5.721298
CASH_ADVANCE	5.166609
PURCHASES_TRX	4.630655
BALANCE	2.393386
PRC_FULL_PAYMENT	1.942820
CASH_ADVANCE_FREQUENCY	1.828686
ONEOFF_PURCHASES_FREQUENCY	1.535613
CREDIT_LIMIT	1.522549

Assessing skewness in the features as a step for transformations.

We consider the following aspects:

- (nearly 0) or ($-0.75 < \text{value} < 0.75$): No skewness
- Positive (value > +0.75): Right skewness
- negative (value < -0.75): left skewness

Featuring Engineering 01

Applying Log transformation to right skewed data.

	Skewness Value
PRC_FULL_PAYMENT	1.746046
CASH_ADVANCE_FREQUENCY	1.455462
ONEOFF_PURCHASES_FREQUENCY	1.290617
CASH_ADVANCE_TRX	0.940131
PURCHASES_INSTALLMENTS_FREQUENCY	0.509201
MINIMUM_PAYMENTS	0.269565
CASH_ADVANCE	0.262594
ONEOFF_PURCHASES	0.185854
PURCHASES_FREQUENCY	0.060164
PURCHASES_TRX	0.032697
INSTALLMENTS_PURCHASES	-0.024981
CREDIT_LIMIT	-0.101564
PURCHASES	-0.764492
BALANCE	-0.861021
PAYMENTS	-1.778312
BALANCE_FREQUENCY	-2.023266
TENURE	-2.943017

Note that when we apply log transformation to the features, it effectively addresses only positive skewness. This transformation aims to convert right skewness into left skewness, ultimately achieving a symmetric or normal distribution.

Featuring Engineering 02

Applying feature scaling for Machine Learning

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVA
8945	-1.379634	0.518084	0.266574	-0.987090	0.754107	-0.93
8946	-1.568051	0.518084	0.276841	-0.987090	0.763821	-0.93
8947	-1.473834	-0.185477	0.027374	-0.987090	0.527794	-0.93
8948	-1.733775	-0.185477	-1.679855	-0.987090	-1.087454	0.08
8949	-0.118301	-0.889033	0.719365	1.168619	-1.087454	0.42

Feature scaling is particularly important in machine learning models that rely on distance metrics. When employing clustering methods such as K-means, which heavily relies on distance metrics, it's crucial to scale our features appropriately to ensure accurate clustering results.

Machine Learning Analysis and Findings

Machine Learning Analysis and Findings:

In the upcoming slides, we will conduct a comparison among four distinct clustering methods: K-means, Agglomerative Hierarchical clustering, DBSCAN, and MeanShift. Our evaluation will focus on determining the optimal number of clusters and comparing the clustered observations with the expected target variable across the entire dataset.

Machine Learning Analysis 01

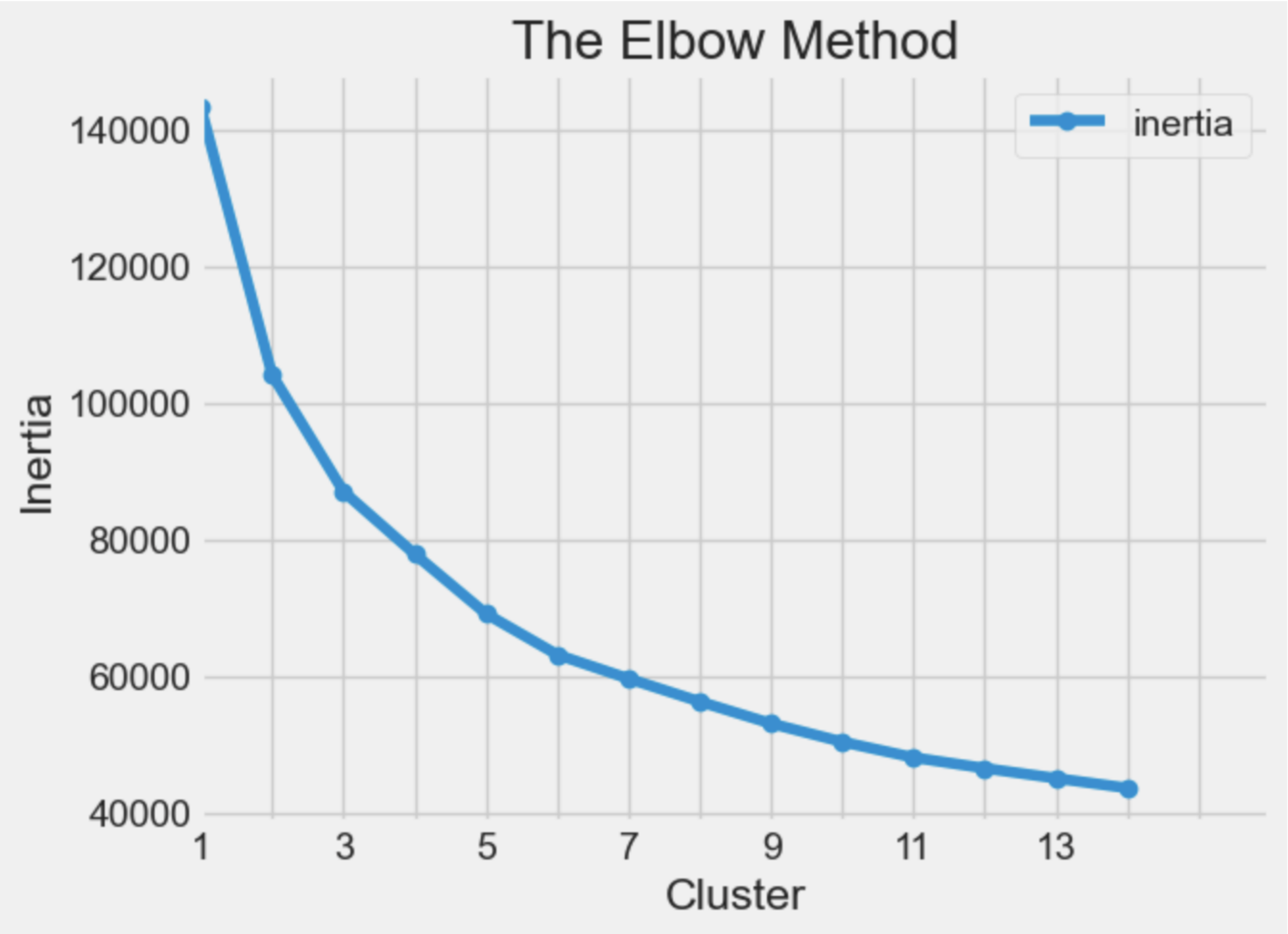
Performing K-Means

```
from sklearn.cluster import KMeans

# Create and fit a range of models
km_list = list()

for clust in range(1,15):
    km = KMeans(n_clusters=clust, random_state=42)
    km = km.fit(data[float_columns])

    km_list.append(pd.Series({'clusters': clust,
                              'inertia': km.inertia_,
                              'model': km}))
```

In this section, we have applied the K-means algorithm to the dry bean dataset, testing a range of different K values from 0 to 20. The aim is to determine the most suitable number of clusters. To assess the model's quality, we've employed the inertia metric and the elbow method. Inertia is defined as the sum of squared distances from each data point (Xi) to its assigned cluster (Ck).

Machine Learning Analysis 02

Elbow Method



As depicted in the graph, when following the elbow method approach, it becomes evident that the suitable number of clusters falls within the range of 3 to 5. This observation aligns with our dataset, which inherently comprises four distinct clusters.

Machine Learning Analysis 03

Applying K = 4

```
[66]: km = KMeans(n_clusters=4, random_state=42)
      km = km.fit(data[float_columns])
```

```
[67]: data['k-means'] = km.predict(data[float_columns])
      data.sample(7)
```

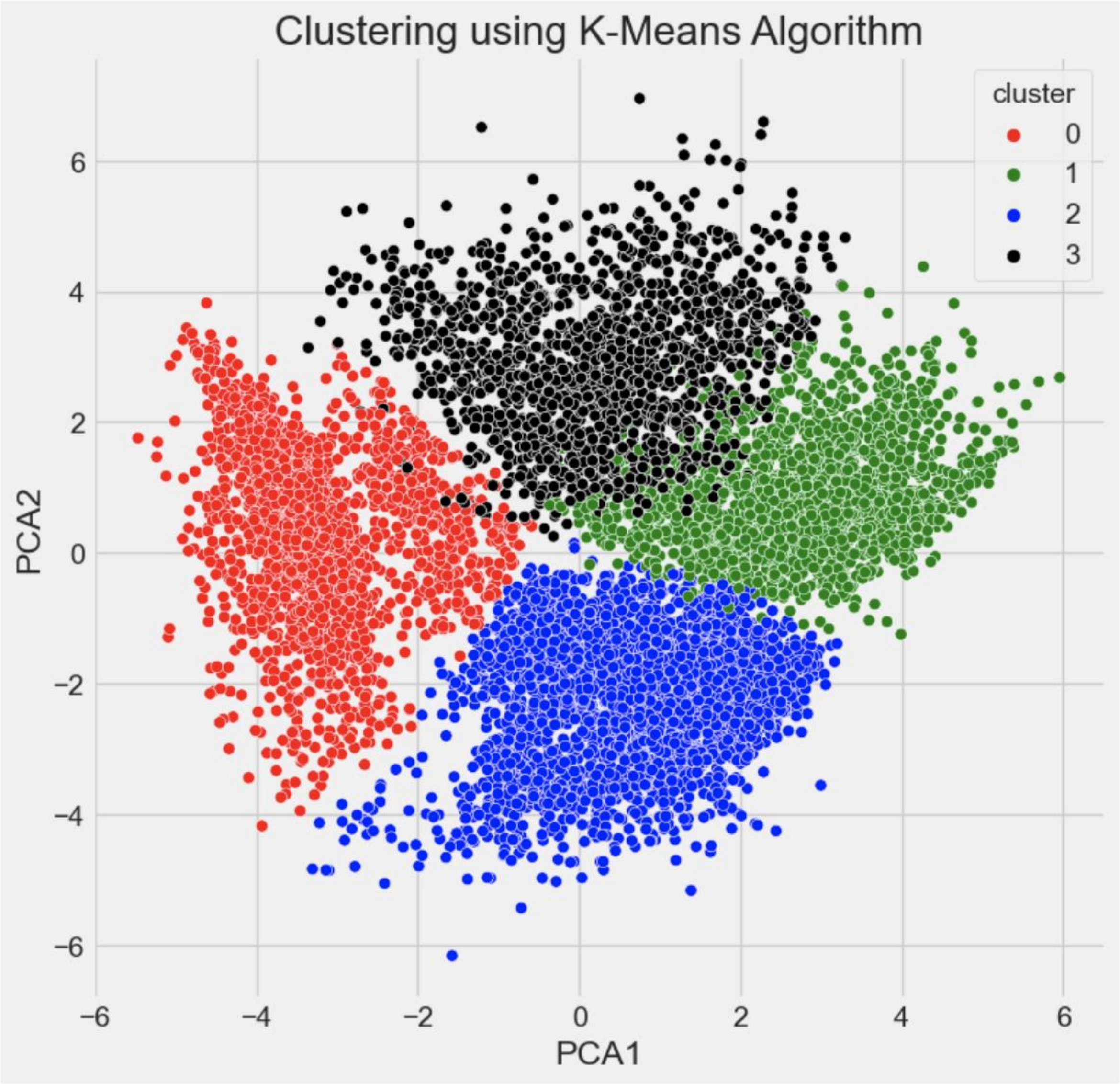
[67]:

/ANCE_TRX	PURCHASES_TRX	CREDIT_LIMIT	PAYMENTS	MINIMUM_PAYMENTS	PRC_FULL_PAYMENT	TENURE	k-means
-0.810069	0.487865	-1.447207	-0.525098	0.320577	-0.556368	12	0
2.296657	-0.874655	0.835591	1.386995	1.083956	-0.556368	12	1
-0.810069	1.143564	1.361133	0.164368	-0.565026	0.803155	12	2
0.784603	-1.379210	1.088901	0.163000	0.630187	-0.556368	12	1
0.563506	-0.074955	-0.602091	-0.085019	0.280016	-0.556368	12	3
0.278464	-1.379210	-0.107577	1.784719	1.180512	1.359774	12	1
1.565826	-1.379210	-0.107577	0.146156	1.513802	-0.556368	12	1

Upon applying the K-means algorithm with the number of clusters set to 4 on the dataset, the algorithm will categorize each observation into one of the four clusters as defined in the model. Subsequently, we can assess how many of these observations have been correctly classified into their respective clusters.

Machine Learning Analysis 04

Grouping clusters

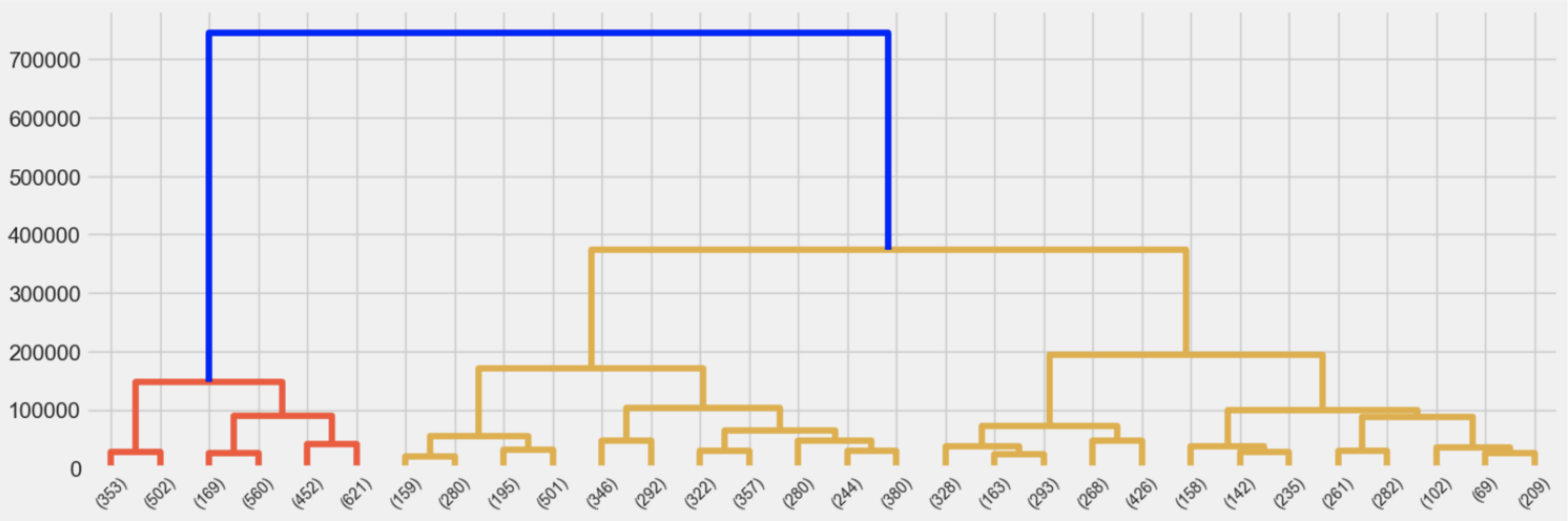


After performing PCA to 2 dimensions, we applied k = 4 and grouped the clusters in order to validate our clustering model with the actual classes labeling.

Machine Learning Analysis 05

Agglomerative Clustering

```python
from sklearn.cluster import AgglomerativeClustering
agglo = AgglomerativeClustering(n_clusters=4, linkage='ward', compute_full_tree=True)
agglo = agglo.fit(data[float_columns])
```



Applying agglomerative hierarchical clustering with number of clusters = 4

Machine Learning Analysis 06

Grouping the agglomerative clusters

TENURE	agglo	Number
6	0	58
	1	9
	2	78
	3	59
7	0	68
	1	14
	2	53
	3	55
8	0	66
	1	15
	2	59
	3	56
9	0	51
	1	13
	2	60
	3	51
10	0	71
	1	25
	2	71
	3	69
11	0	112
	1	47
	2	82
	3	124
12	0	2243
	1	1629
	2	2138
	3	1574

We grouped the clusters in order to validate our clustering model with the actual classes labeling from TENURE.

Machine Learning Analysis 07

DBSCAN and MeanShift Clustering

```
from sklearn.cluster import MeanShift
ms  = MeanShift(bandwidth=3.6, n_jobs=-1)
ms = ms.fit(data[float_columns])
```

```
np.unique(ms.labels_)
```

```
array([0, 1, 2, 3])
```

```
data['MeanShift'] = ms.fit_predict(data[float_columns])
data.tail()
```

TENURE	MeanShift	Number
6	0	101
	1	87
	2	16
7	0	89
	1	87
	2	14
8	0	93
	1	89
	2	14
9	0	83
	1	80
	2	12
10	0	118
	1	105
	2	12
	3	1
11	0	176
	1	180
	2	8

TENURE	dbscan	Number
6	-1	204
7	-1	190
8	-1	195
	0	1
9	-1	174
	0	1
10	-1	230
	0	6
11	-1	351
	0	13
	1	1
12	-1	7337
	0	217
	1	12
	2	18

```
from sklearn.cluster import DBSCAN
dbs = DBSCAN(eps=0.5, min_samples=18, metric='euclidean')
dbs = dbs.fit(data[float_columns])
```

```
np.unique(dbs.labels_)
```

```
array([-1,  0,  1,  2])
```

After conducting multiple iterations of these two algorithms with various parameter settings, we encountered unsatisfactory results. Despite the algorithms correctly predicting the optimal number of clusters, our validation process revealed a high error rate when comparing the clustered observations to the classes.

Machine Learning Findings

Model comparisons

```
from sklearn.model_selection import train_test_split
X = data.drop('k-means', axis=1)
y = data['k-means']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3)
```

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion="entropy")
dt.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

```
y_pred = dt.predict(X_test)
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(metrics.confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[832  11  34   9]
 [  5 713   0  15]
 [ 28   0 603   7]
 [  6  11  10 401]]
              precision    recall  f1-score   support

           0       0.96      0.94      0.95       886
           1       0.97      0.97      0.97       733
           2       0.93      0.95      0.94       638
           3       0.93      0.94      0.93       428

    accuracy                           0.95      2685
   macro avg       0.95      0.95      0.95      2685
weighted avg       0.95      0.95      0.95      2685
```

If you'd notice from the picture, we used Decision Tree to fit our K-Means in the entire dataset and check for accuracy which is 95%. As mentioned in the previous slide, DBSCAN and MeanShift algorithms yielded unsatisfactory results when it came to clustering observations. Therefore, we will exclude them from the comparison, focusing solely on assessing the performance of the K-means and Agglomerative Clustering algorithms. K-MEANS is the best clustering algorithm for this dataset.

Model Flaws and Strength and Findings

Machine flaws and Strength:

Both the K-means and Agglomerative Hierarchical Clustering methods demonstrated efficiency and accuracy in determining the suitable number of clusters. Additionally, they successfully clustered the majority of observations with an accuracy rate exceeding 90%.

In contrast, DBSCAN and MeanShift algorithms demanded significant time and effort to identify the appropriate number of clusters. This process involved repeatedly adjusting parameters to achieve the desired cluster count. Furthermore, these algorithms produced subpar results in terms of accurately clustering the observations.

Advanced Steps:

To improve the clustering process with DBSCAN and MeanShift, we have a couple of options. One approach is to use image data from the scanned images, as both algorithms are well-suited for computer vision applications. Alternatively, we can employ grid search and hyperparameter tuning to find the best parameters for these algorithms. However, it's important to note that tuning parameters, especially for the MeanShift model, can be time-consuming.

Thank you

WISDOM IZUCHUKWU ADIKE