FACULTY OF COMPUTING

SEMESTER 2 2024/2025

SECP2753 Data Mining

SECTION 02

Project 2

LECTURER: DR. ROZILAWATI BINTI DOLLAH @ MD. ZAIN

GROUP MEMBER

Student Name	Matric No.
AHMAD ZIYAAD	A23CS0206
GOE JIE YING	A23CS0224
TEH RU QIAN	A23CS0191

1.0 Lifecycle and Spend Segments

1.1 Objective

The aim of this analysis is to classify customers according to their spending patterns and lifecycle stage. Features that have been chosen include:

- Tenure is the period of time a customer has been with the business.
- MonthlyCharges are the total amount of money paid each month.
- TotalCharges is the total amount paid since registering.

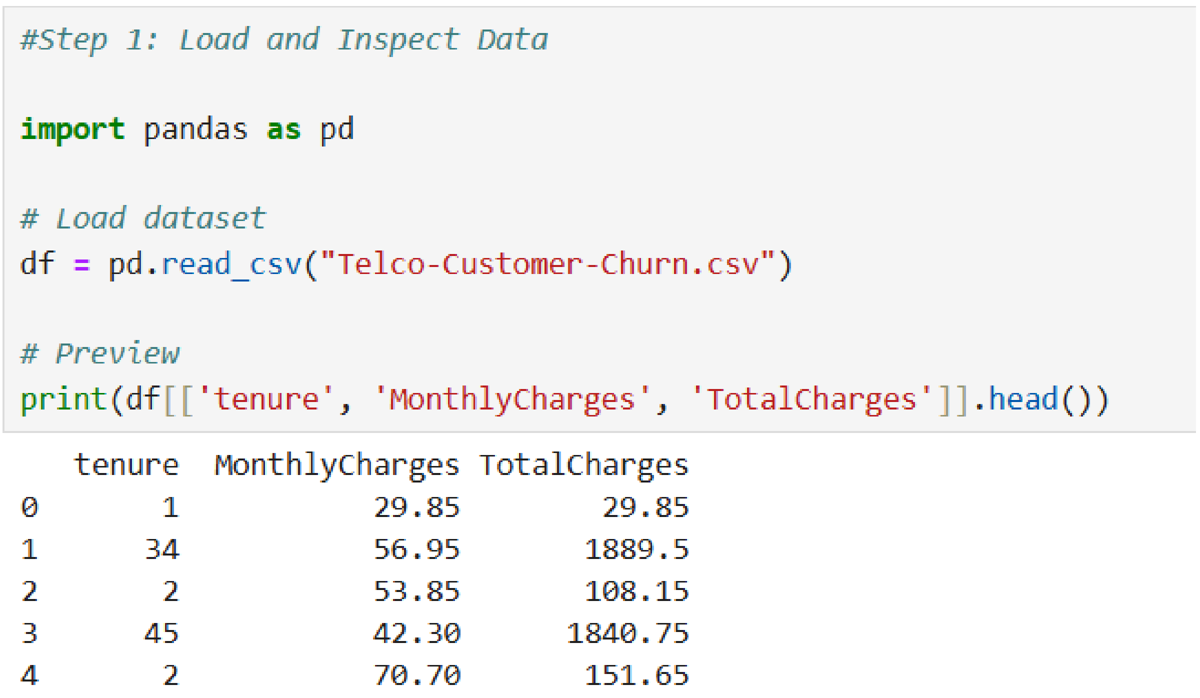
These features are important for differentiating between different customer groups, including *low-spending long-term customers, new high-paying consumers, and high-value loyal customers*.

1.2 Data Preprocessing

Data preparation and cleaning were required before doing clustering analysis. This procedure included choosing the appropriate features, addressing missing or invalid values, and scaling the data to fit clustering techniques. The following steps were taken:

Step 1: Load and Preview the Dataset

The dataset was loaded using the ‘pandas’ library shown in Figure 1.2.1. Only the relevant features for this clustering insight which are tenure, MonthlyCharges and TotalCharges were selected.



```
#Step 1: Load and Inspect Data

import pandas as pd

# Load dataset
df = pd.read_csv("Telco-Customer-Churn.csv")

# Preview
print(df[['tenure', 'MonthlyCharges', 'TotalCharges']].head())
```

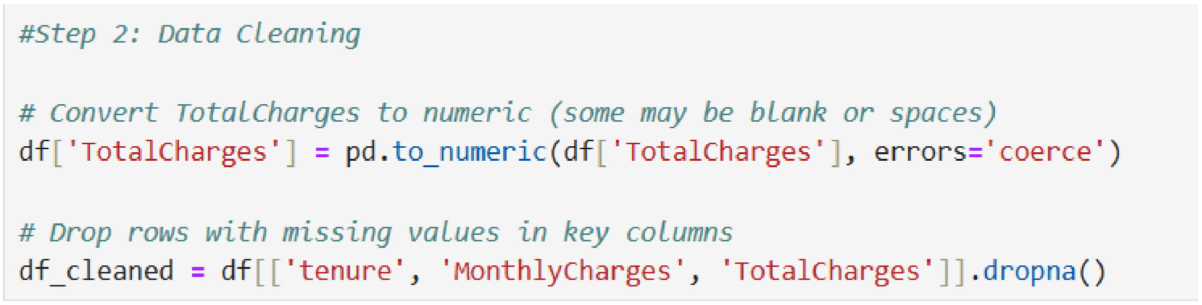
```
   tenure  MonthlyCharges TotalCharges
0       1           29.85        29.85
1      34           56.95       1889.5
2       2           53.85       108.15
3      45           42.30      1840.75
4       2           70.70       151.65
```

Figure 1.2.1 Load and Preview the Dataset

Step 2: Handle Missing or Invalid Values

Some values in the TotalCharges column in the original dataset were not numerically stored. These were possibly spaces or empty strings. Using pd.to_numeric(), the column was transformed to numeric and any invalid entries were replaced with NaN (missing values).

Then,.dropna() was used to remove rows that had missing values in the important columns. Figure 1.2.2 shows the process of data cleaning. This stage ensures that only valid and complete data is used for clustering.

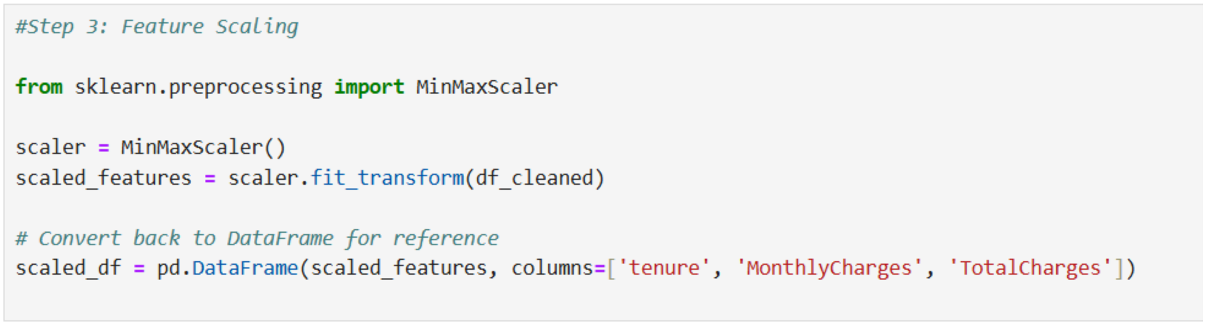


```
#Step 2: Data Cleaning

# Convert TotalCharges to numeric (some may be blank or spaces)
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

# Drop rows with missing values in key columns
df_cleaned = df[['tenure', 'MonthlyCharges', 'TotalCharges']].dropna()
```

Figure 1.2.2 Data Cleaning

Step 3: Feature Scaling (Normalization)

Distances are used by clustering algorithms such as Agglomerative Clustering and K-Means to group data. One feature (like TotalCharges) may dominate the clustering process if its range is much larger than that of another feature (like tenure). This was fixed by applying Min-Max Scaling to each of the three features. The technique provides equal weight to each feature in the clustering process by scaling each value to a range between 0 and 1. The normalized values of the three chosen features are stored in scaled_df, where they are prepared for clustering shown in Figure 1.2.3.



```
#Step 3: Feature Scaling

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaled_features = scaler.fit_transform(df_cleaned)

# Convert back to DataFrame for reference
scaled_df = pd.DataFrame(scaled_features, columns=['tenure', 'MonthlyCharges', 'TotalCharges'])
```

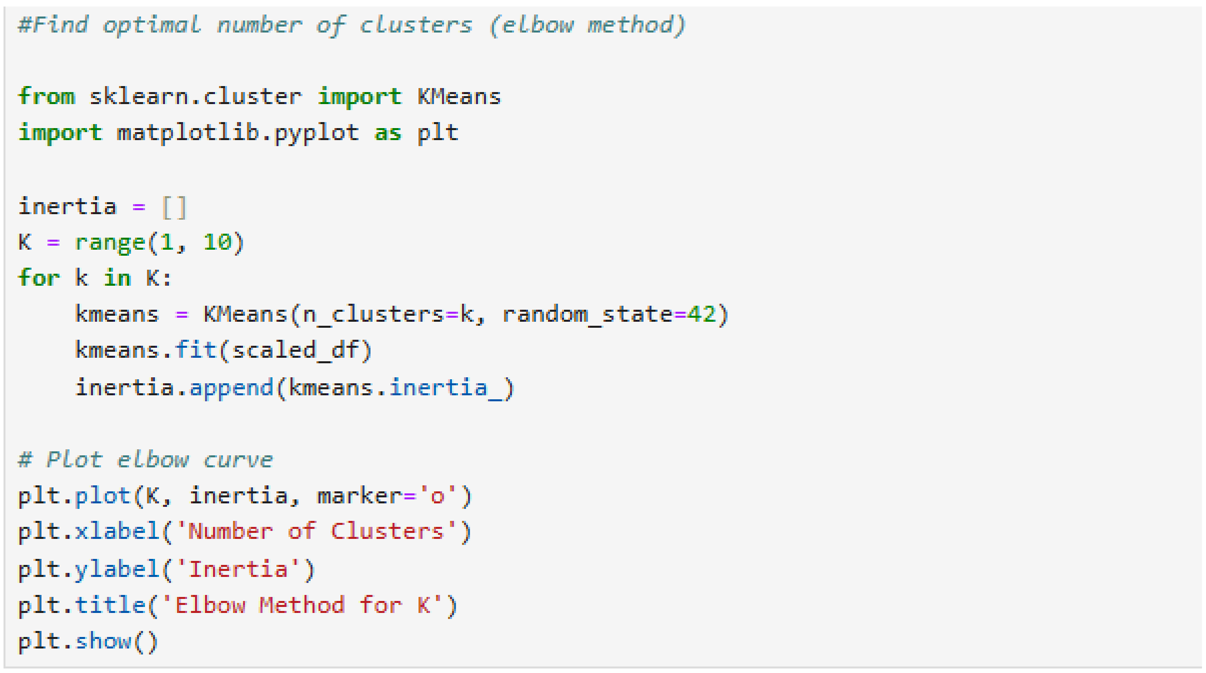
Figure 1.2.3 Feature Scaling

1.3 Clustering Algorithms Used

K-Means Clustering and Agglomerative Hierarchical Clustering were the clustering methods used after the data had been preprocessed. In order to compare clustering results and obtain more reliable insights, both used to categorize customers according to their tenure, monthly charges and total charges.

1.3.1 K-Means Clustering

Using K-Means clustering, customers were divided into different groups. Based on the chosen features, it determines cluster centers, also known as centroids and allocates each customer to the closest center. The elbow method was used to find the ideal number of clusters before running K-Means. Figure 1.3.1.1 displays the inertia values for different numbers of clusters, which are the sum of the distances between customers and their cluster centers. At k = 4, a strong curve occurs, indicating that using four clusters is the best choice. Following that, the inertia reduction decreases, indicating that the more clusters will only slightly improve the situation.



```python
#Find optimal number of clusters (elbow method)

from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

inertia = []
K = range(1, 10)
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_df)
    inertia.append(kmeans.inertia_)

# Plot elbow curve
plt.plot(K, inertia, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method for K')
plt.show()
```

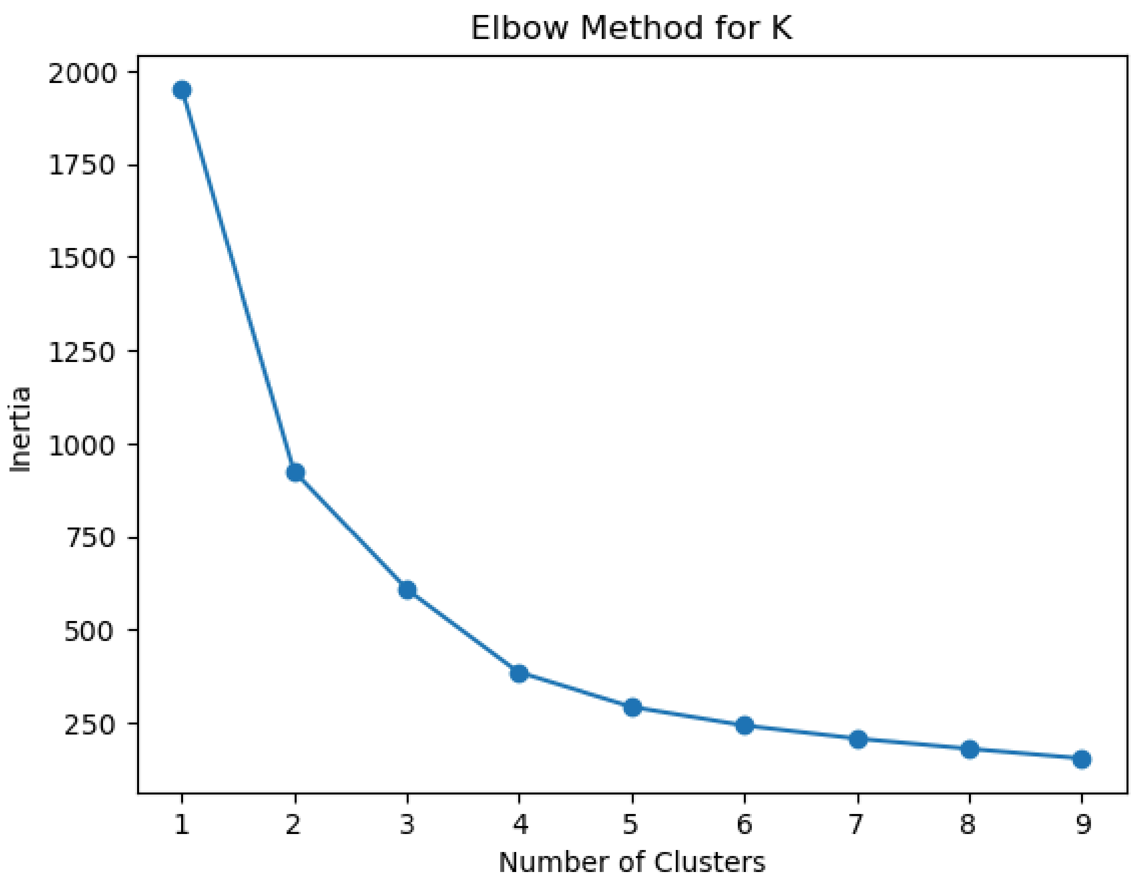
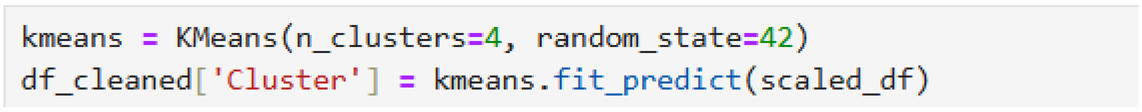


Figure 1.3.1.1 Elbow Method

After determining that 4 was the ideal number of clusters, the K-Means algorithm was used. Customers were assigned to one of four groups based on the similarity of their scaled features. The result is stored in the original unscaled DataFrame with new columns called Cluster displays in Figure 1.3.1.2. It helps in analysis because it keeps the real-world values.



```
kmeans = KMeans(n_clusters=4, random_state=42)
df_cleaned['Cluster'] = kmeans.fit_predict(scaled_df)
```

Figure 1.3.1.2 Apply K-Means with 4 clusters

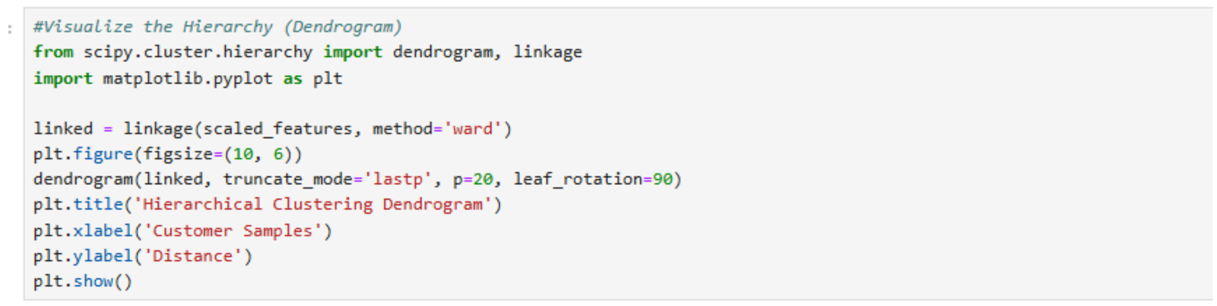
1.3.2 Agglomerative Hierarchical Clustering

Another method used to validate the segmentation was agglomerative clustering. This method repeatedly combines the two most similar groups to create clusters. It does not require the random initial point choosing and provides a hierarchical framework that demonstrates how data points are grouped.

A tree-like diagram called a dendrogram illustrates how Agglomerative Clustering was used to gradually group customers together. Each customer is initially treated as separate groups, and the most similar ones are gradually combined based on their tenure, monthly charges and total charges.

The labels on the x-axis indicate customer samples, with each representing a single customer or a small cluster of customers. The height also known as distance on the y-axis indicates the distance between the groups when they were joined. The greater the difference between two branches, the higher the connection point between those groups.

A clear split is visible near the top of the dendrogram, where the data is divided into larger, clearer groupings. This validates the decision to use 4 clusters, as there are significant differences in customer behavior. Overall , the dendrogram validates the groupings discovered in K-Means clustering and helps in discovering how customers are similar or different. Figure 1.3.2.1 shows the hierarchical clustering dendrogram.



```
#Visualize the Hierarchy (Dendrogram)
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt

linked = linkage(scaled_features, method='ward')
plt.figure(figsize=(10, 6))
dendrogram(linked, truncate_mode='lastp', p=20, leaf_rotation=90)
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Customer Samples')
plt.ylabel('Distance')
plt.show()
```

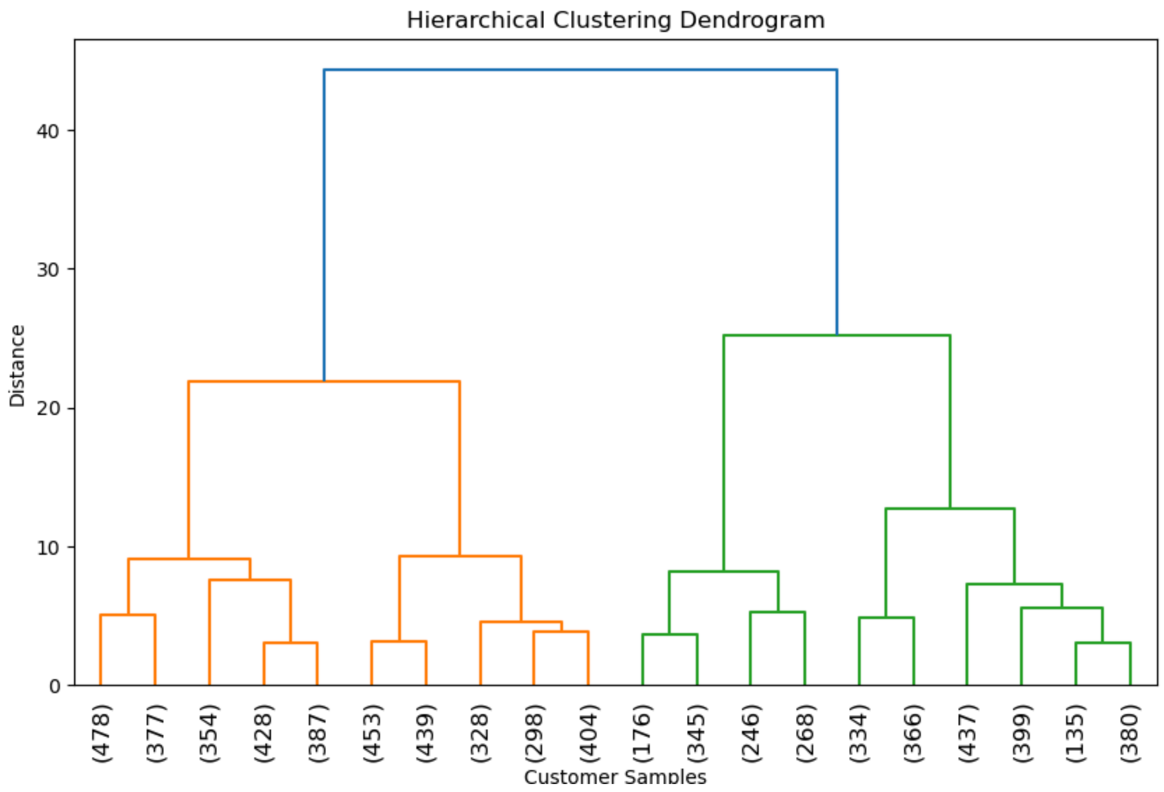
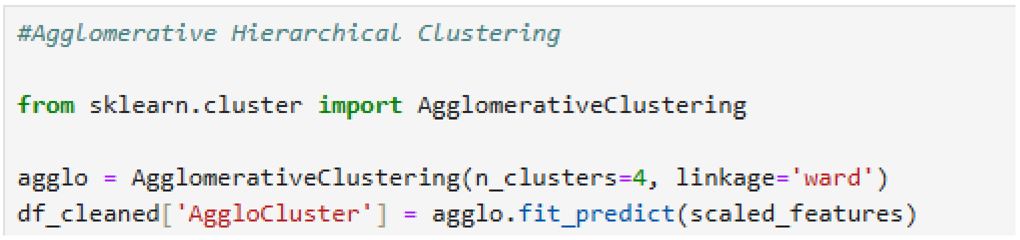


Figure 1.3.2.1 Hierarchical Clustering Dendrogram

After that, Agglomerative Clustering was applied with 4 clusters. The result is stored as a new column called AggloCluster shown in Figure 1.3.2.2.



```
#Agglomerative Hierarchical Clustering

from sklearn.cluster import AgglomerativeClustering

agglo = AgglomerativeClustering(n_clusters=4, linkage='ward')
df_cleaned['AggloCluster'] = agglo.fit_predict(scaled_features)
```

Figure 1.3.2.2 Agglomerative Hierarchical Clustering

1.4 Result and Interpretation

The code calculates the average of the three main features which are tenure, monthlyCharges and totalCharges. The result is displayed as a table form in Figure 1.3.4.1.

```
cluster_summary = df_cleaned.groupby('Cluster').mean()
print(cluster_summary)
```

```
            tenure  MonthlyCharges  TotalCharges
Cluster
0        59.434964       93.056217   5527.547607
1        10.158175       31.889485    303.595379
2        53.468048       34.530354   1815.149136
3        15.323763       80.826568   1244.615945
```

Figure 1.3.4.1 Average of Cluster

This cluster analysis revealed 4 different customer groups:

- Cluster 0 represents loyal high spenders. With a total average spending of RM5527.55, this group has the greatest average tenure (59.43 months) and monthly costs (RM93.06). These are loyal customers who make larger purchases and provide a major financial contribution. To maintain satisfaction and lower the chance of churn, they should be given priority for retention through loyalty programs, premium support and extra benefits.
- Cluster 1 consists of new budget users. This group probably consists of new customers trying with basic or minimal services, because of their short average tenure of 10.16 months, low-cost monthly charges (RM31.89), and total average spending of RM303.60. Their commitment and long-term value may be increased by providing attractive upgrade options, early engagement or welcome offers.
- Cluster 2 includes loyal low spenders. This group contains customers with a 53.47 month tenure, low monthly charges (RM34.53) and a total average spend of RM1815.15. Even though they use the service frequently, they do not spend much money. It is possible to increase revenue from this loyal base by using upselling options like bundling or discounts on premium features.
- Cluster 3 identifies new premium users. This group of customers spends an average of RM1244.62 in total, although they have a shorter tenure (15.32 months) but pay higher monthly charges (RM80.83). To increase trust and lower early-stage churn, these high-value new customers should get proactive service, early access benefits and strong onboarding support.

Besides, two plots show how customers are grouped visually using K-Means and Agglomerative Clustering, with different colors representing each cluster. The X-axis is tenure and the Y-axis is MonthlyCharges shown in Figure 1.3.4.2.

```
#K-Means Clustering
import seaborn as sns

sns.scatterplot(data=df_cleaned, x='tenure', y='MonthlyCharges', hue='Cluster', palette='Set2')
plt.title('Customer Clusters: Tenure vs Monthly Charges (K-Means Clustering)')
plt.show()
```

```
#Agglomerative Hierarchical Clustering
sns.scatterplot(data=df_cleaned, x='tenure', y='MonthlyCharges', hue='AggloCluster')
plt.title('Customer Clusters: Tenure vs Monthly Charges (Agglomerative Hierarchical Clustering)')
plt.show()
```

Figure 1.3.4.2 Code for Plotting

Based on their tenure and monthly charges, 4 different customer groups are displayed in the scatter plot generated by the K-Means clustering algorithm shown in Figure 1.3.4.3. Since these clusters are clearly defined, it is easy to differentiate between different customer groups based on their spending habits and duration of service. In contrast, the scatter plot as Figure 1.3.4.4 for Agglomerative Clustering shows similar clusters with slightly less defined boundaries. Some customer points appear to overlap, especially in the middle range of numbers, which can make interpretation more difficult.

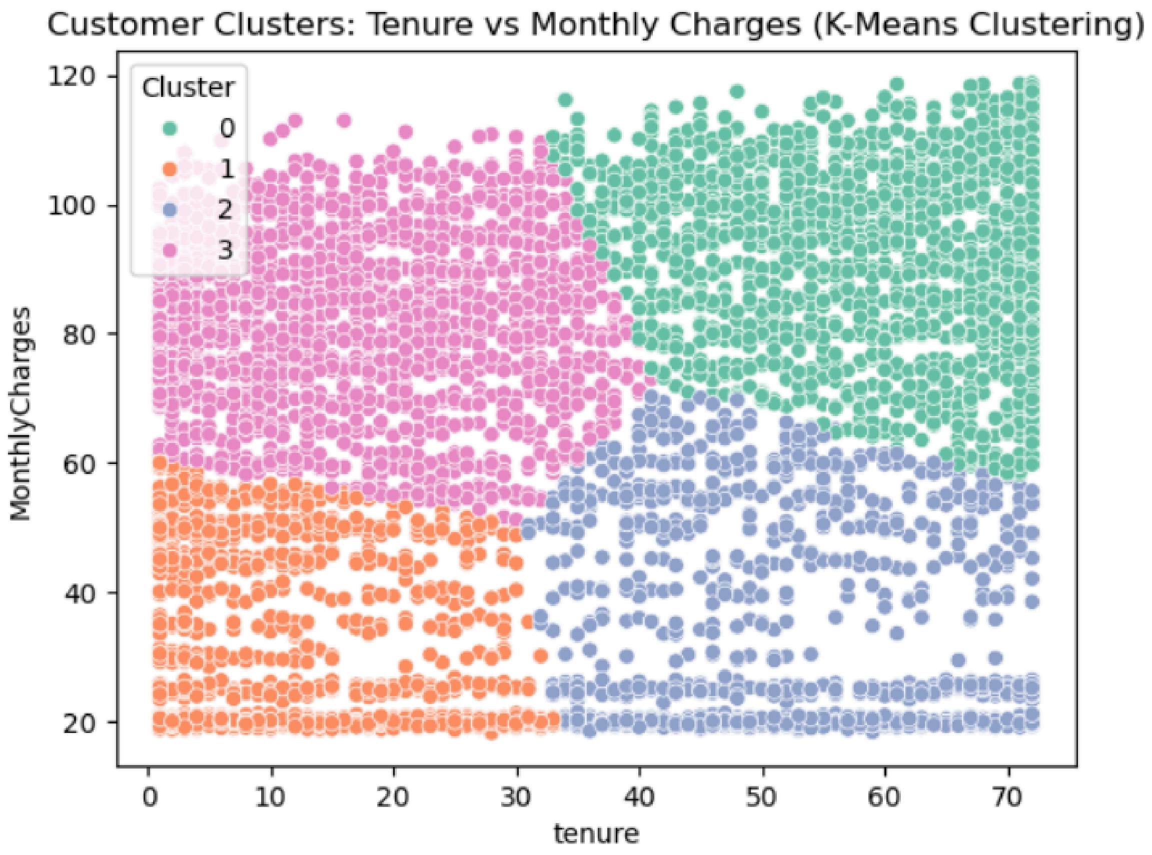


Figure 1.3.4.3 Plot for K-Means Clustering

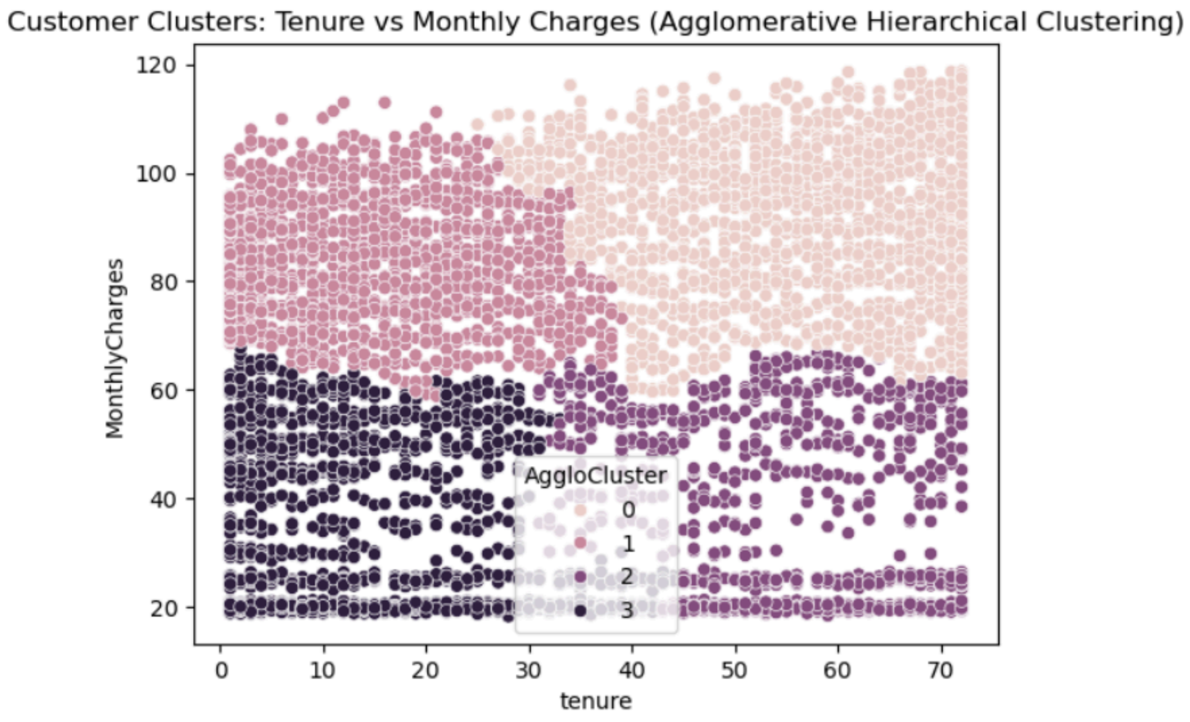


Figure 1.3.4.4 Plot for Agglomerative Hierarchical Clustering

1.5 Conclusion

As a conclusion, the clustering analysis divided the customers into 4 suitable groups based on their spending and usage of services. Loyal high customers (Cluster 0) should be rewarded with loyalty programs, while new premium customers (Cluster 3) need early support to stay longer. Budget-conscious customers (Clusters 1 and 2) can be encouraged to spend more through special discounts or service bundles. These insights can help the company reduce customer loss, grow revenue and improve customer satisfaction by offering the right approach to each group.

2.0 Digital Service Engagement Profiles

2.1 Objective

The aim of this analysis is to segment customers based on their engagement with digital services. By clustering customers according to their usage of features such as streaming, online security, and technical support, we can identify distinct usage profiles. These profiles help the company tailor product offerings, enhance user experience, and improve customer satisfaction by understanding digital behavior patterns.

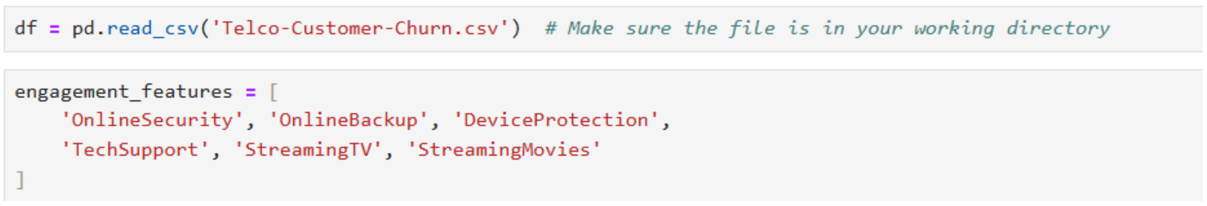
The selected features for this analysis are:

- OnlineSecurity
- OnlineBackup
- DeviceProtection
- TechSupport
- StreamingTV
- StreamingMovies

2.2 Data Preprocessing

Step 1: Load and Select Digital Service Features

The dataset Telco-Customer-Churn.csv was loaded using the pandas library. Only the six features related to digital service engagement were selected for clustering. These services represent whether customers subscribe to optional digital add-ons.



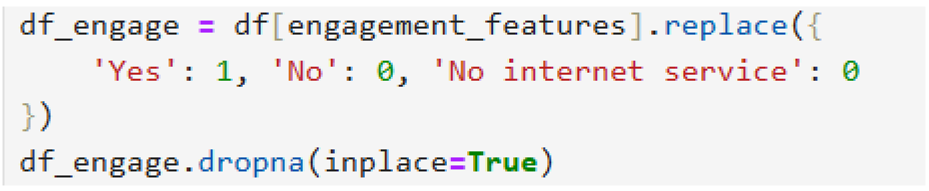
```
df = pd.read_csv('Telco-Customer-Churn.csv')  # Make sure the file is in your working directory
```

```
engagement_features = [
    'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
    'TechSupport', 'StreamingTV', 'StreamingMovies'
]
```

Figure 2.2.1 Loading dataset and selecting relevant features

Step 2: Clean Categorical Values

The original features in the dataset contained categorical string values such as "Yes", "No", and "No internet service", which are not suitable for direct use in clustering algorithms. To make the data numerically analyzable, these values were converted into binary numerical format. Specifically, "Yes" was encoded as 1 to indicate the presence of a service, while both "No" and "No internet service" were encoded as 0 to represent the absence of service. This transformation simplifies the data and ensures consistency, allowing the clustering algorithms to interpret and process the input correctly during analysis.

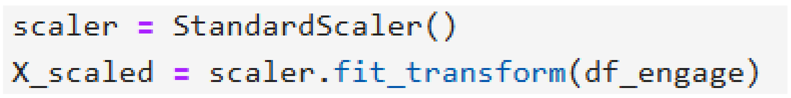


```
df_engage = df[engagement_features].replace({
    'Yes': 1, 'No': 0, 'No internet service': 0
})
df_engage.dropna(inplace=True)
```

Figure 2.2.2 Converting into binary numerical data

Step 3: Feature Scaling

To give equal importance to each feature, standardization was applied using the StandardScaler from sklearn. This scales the features so that they each have a mean of 0 and a standard deviation of 1. Clustering algorithms are sensitive to differences in scale, so this step prevents any feature from dominating the clustering results.



```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_engage)
```

Figure 2.2.3 Applying standardization

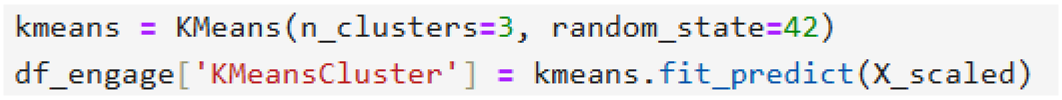
2.3 Clustering Algorithms Used

Two clustering methods were applied to the standardized data to uncover natural groupings among customers:

2.3.1 K-Means Clustering

K-Means is a partitioning-based clustering algorithm. We used it to divide the customers into three clusters (k = 3). The algorithm randomly initialized centroids and iteratively adjusted them to minimize the distance between customers and their assigned cluster centers.

The resulting cluster labels were added to the dataset in a new column called KMeansCluster.



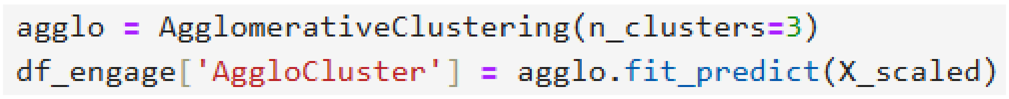
```
kmeans = KMeans(n_clusters=3, random_state=42)
df_engage['KMeansCluster'] = kmeans.fit_predict(X_scaled)
```

Figure 2.3.1.1 Applying K-Means Clustering

2.3.2 Agglomerative Hierarchical Clustering

Agglomerative clustering starts by treating each data point as an individual cluster and then merges the closest pairs iteratively based on distance. This continues until the desired number of clusters is reached. We selected 3 clusters to match the K-Means segmentation.

Additionally, a dendrogram was generated to visualize how clusters were formed step-by-step and to help validate the cluster separation visually.

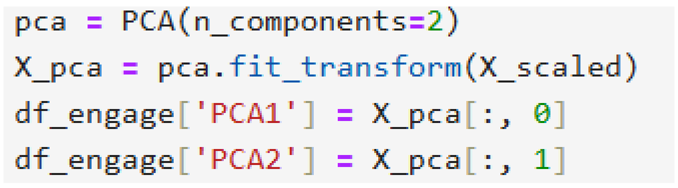


```
agglo = AgglomerativeClustering(n_clusters=3)
df_engage['AggloCluster'] = agglo.fit_predict(X_scaled)
```

Figure 2.3.2.1 Applying Agglomerative Clustering

2.4 Result and Interpretation

Since the data is high-dimensional (6 features), Principal Component Analysis (PCA) was used to project it into two dimensions for visualization. The two main components were plotted in scatter plots, with colors representing different clusters.



```python
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
df_engage['PCA1'] = X_pca[:, 0]
df_engage['PCA2'] = X_pca[:, 1]
```

Figure 2.4.1 Setting up PCA

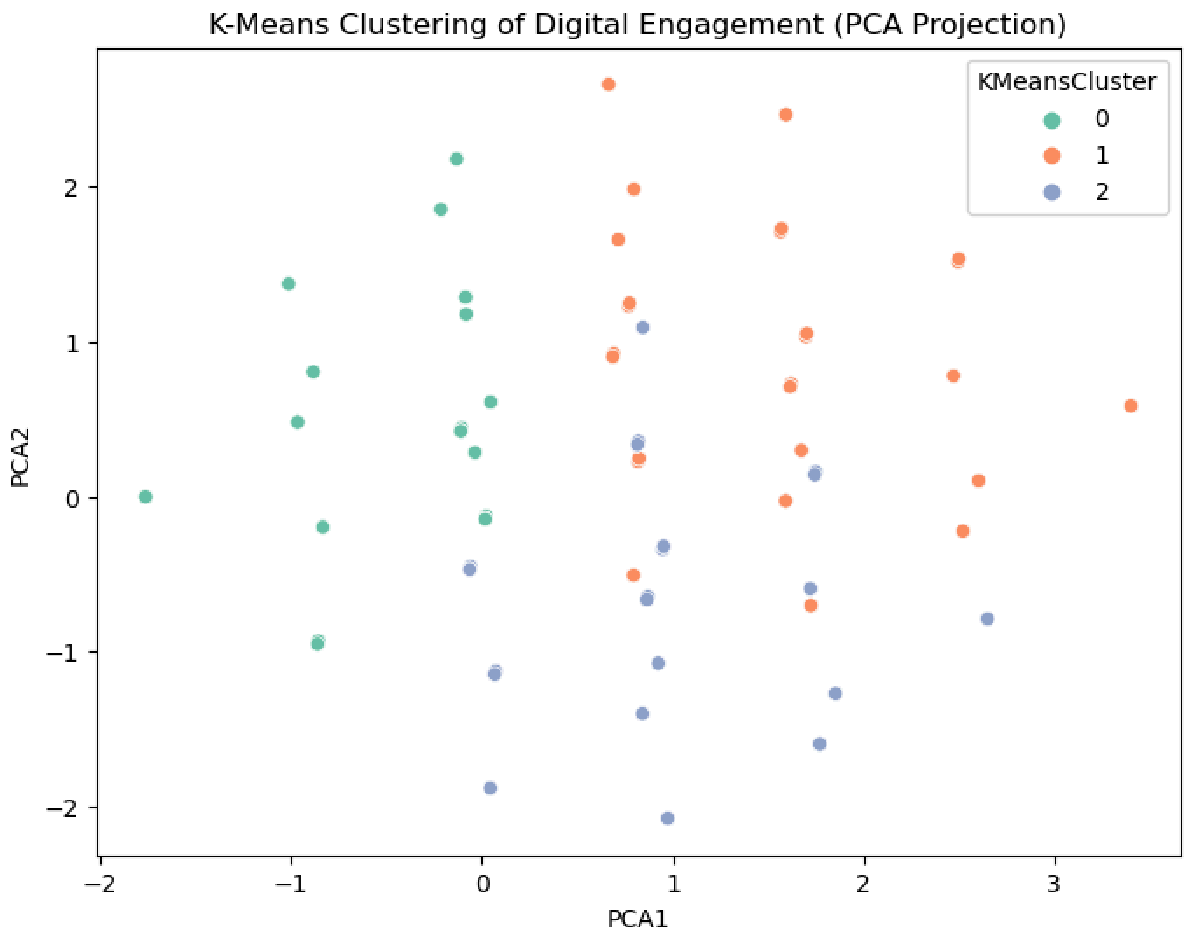


Figure 2.4.2 K-Means Clustering of Digital Engagement Scatter Plot

Figure 2.4.2 presents the output of K-Means clustering using a PCA (Principal Component Analysis) scatter plot to visualize the clusters in two dimensions. The plot shows three distinct clusters, each represented by a different color. These clusters correspond to different engagement levels: highly engaged users who actively use features such as security, backup, and tech support; low-engagement users who rarely subscribe to any digital services; and a third group that primarily uses streaming services like TV and movies, with little interest in security or support features. The clear separation between these clusters demonstrates the effectiveness of K-Means in grouping customers based on digital usage behavior.

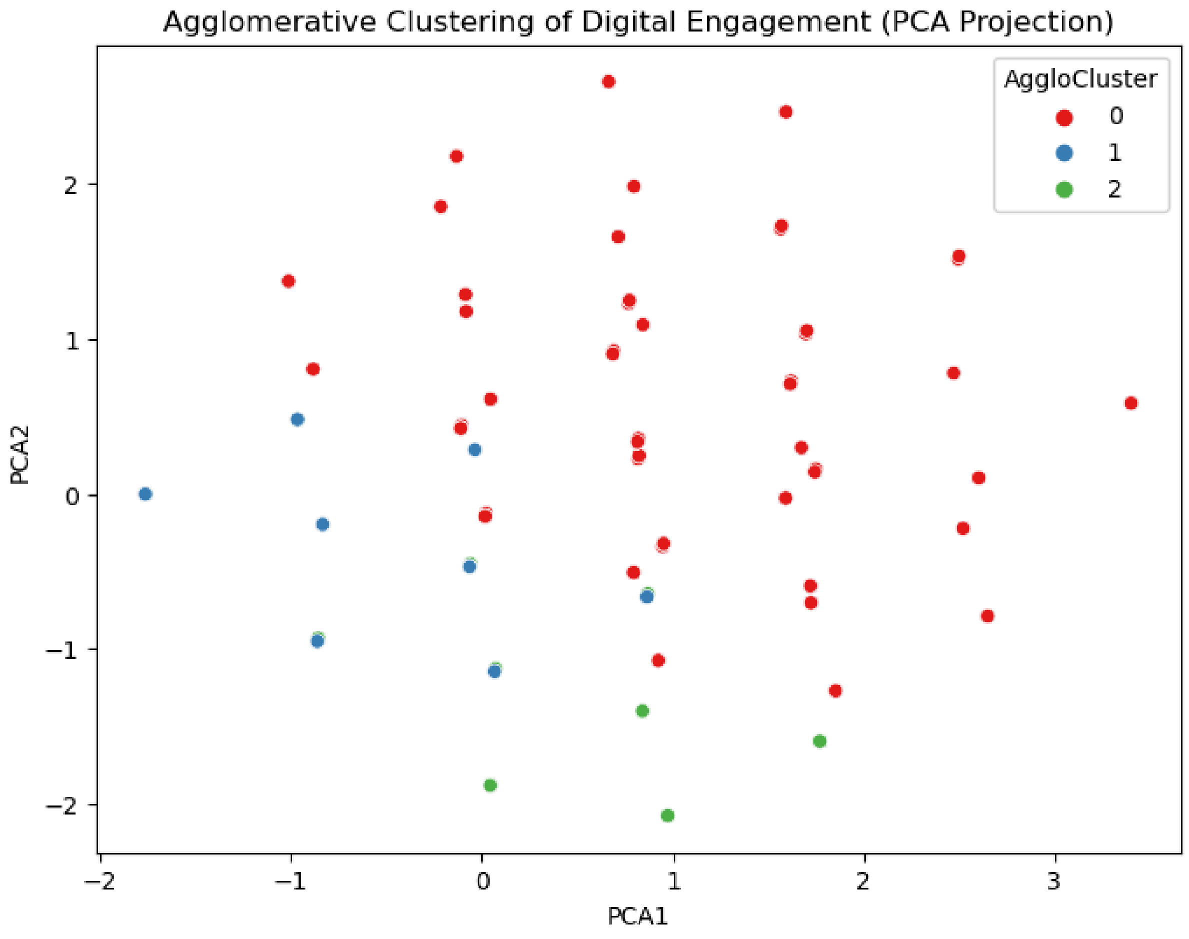


Figure 2.4.3 Agglomerative Clustering of Digital Engagement Scatter Plot

Figure 2.4.3 shows a similar PCA scatter plot, but the clusters were formed using Agglomerative Hierarchical Clustering instead of K-Means. Although the shape and distribution of clusters differ slightly, the overall segmentation is consistent. The same three engagement profiles are visible—high, low, and entertainment-focused users—suggesting that hierarchical clustering also captured the underlying patterns in customer behavior. While some data points appear closer together compared to K-Means, the cluster groupings are still distinguishable and support the same conclusions.

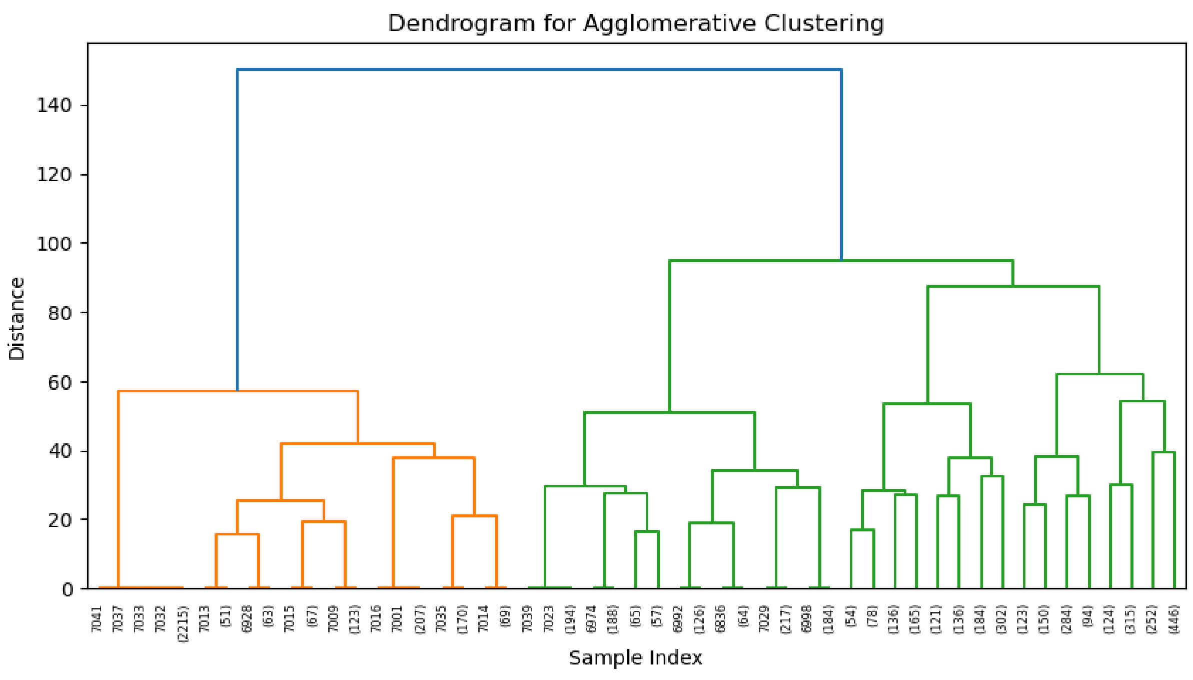


Figure 2.4.4 Dendogram for Agglomerative Clustering

Figure 2.4.4 displays the dendrogram resulting from hierarchical clustering. This tree-like diagram shows how individual customers were merged step by step into clusters. A significant vertical gap near the top of the dendrogram indicates a natural division into three main clusters, validating the decision to segment the data into three groups. The height of the connecting lines in the dendrogram reflects how different the merged clusters are—the taller the line, the greater the dissimilarity. This visualization further confirms the validity of the customer segmentation and provides deeper insight into how customers differ in their usage of digital services.

Cluster Interpretation

Based on the average values of digital service features within each cluster, three distinct customer profiles were identified through the K-Means clustering method. Cluster 0 consists of *highly engaged users* who actively subscribe to services such as online security, backup, device protection, and technical support. These customers are likely digitally savvy, value comprehensive service offerings, and may be more responsive to advanced features or premium add-ons.

Cluster 1, on the other hand, represents *low engagement users*. Customers in this group show minimal or no usage of any optional digital services. They may be new customers still exploring the platform or long-time users who prefer only the basic internet service without added features.

Cluster 2 includes *entertainment-only users*, who primarily use streaming services like StreamingTV and StreamingMovies but show little to no interest in other digital add-ons such as tech support or security features. This cluster highlights customers who engage with content but are less reliant on support or protection services. Notably, the Agglomerative Hierarchical Clustering approach produced similar segmentation, reinforcing the consistency and reliability of the three identified engagement profiles.

2.5 Conclusion

The clustering analysis successfully grouped customers into three meaningful engagement profiles: highly engaged users, low engagement users, and entertainment-only users. These findings provide valuable insights that can inform targeted business strategies. For example, highly engaged users could be prioritized for loyalty programs, premium plans, or early access to new features to maintain their satisfaction and reduce churn.

Low engagement users might benefit from onboarding support, tutorial prompts, or personalized promotions designed to introduce them to value-added services. Meanwhile, entertainment-focused users represent an opportunity to upsell services like online security or tech support, potentially through bundled packages or limited-time offers.

By tailoring engagement and marketing strategies according to these clusters, the company can enhance customer satisfaction, increase retention, and drive greater revenue through personalized, data-driven decision-making.

3.0 Payment Behaviour and Support Load

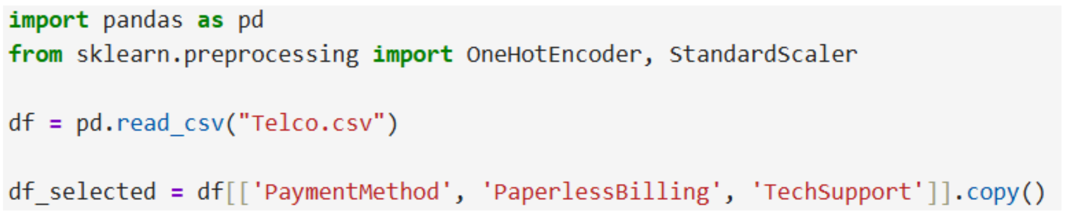
3.1 Objective

The objective of this analysis is to explore customer segments based on their payment behaviour and support load using unsupervised learning techniques. By clustering customers based on their choice of payment method, paperless billing preference, and use of technical support, we aim to identify meaningful groups for strategic business decisions. Specifically, we want to detect patterns such as customers who frequently contact support and prefer manual payments, which may indicate higher churn and service costs, versus customers who adopt auto-pay and require minimal assistance, which may present upselling opportunities.

3.2 Data Preprocessing

Before performing clustering analysis, data preparation and cleaning were carried out with following steps:

Step 1: Load the datasets and Select relevant feature for clustering

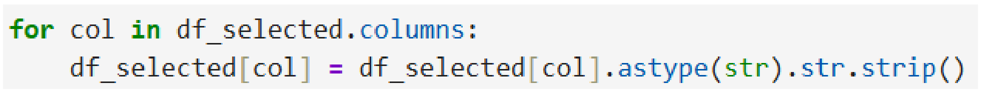


```python
import pandas as pd
from sklearn.preprocessing import OneHotEncoder, StandardScaler

df = pd.read_csv("Telco.csv")

df_selected = df[['PaymentMethod', 'PaperlessBilling', 'TechSupport']].copy()
```

Figure 3.2.1 Load and Select the Features in Datasets

Based on figure 3.2.1, the necessary libraries were imported, and the dataset was loaded for analysis. Only the relevant features, “PaymentMethod”, “PaperlessBilling” and “TechSupport” were selected. These features allow us to cluster customers based on how they choose to pay and how often they interact with technical support, which are key factors in identifying high-cost or low-engagement customer segments.

Step 2: Clean Categorical Values

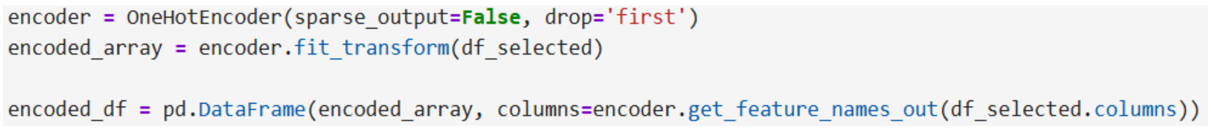


```python
for col in df_selected.columns:
    df_selected[col] = df_selected[col].astype(str).str.strip()
```

Figure 3.2.2 Clean String Column

Extra spaces in text values were removed using .strip() (shown in Figure 3.2.2), to ensure consistent categories. This step ensures that all categorical values are clean and consistent, which can prevent encoding errors and avoid duplicate columns during one-hot encoding.

Step 3: Encode Categorical Features Using One-Hot Encoding

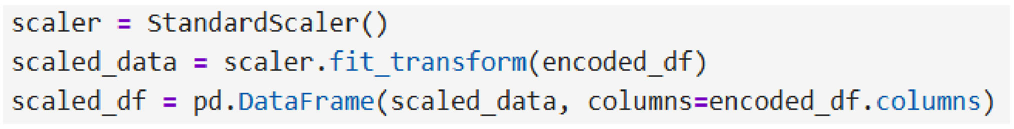


```python
encoder = OneHotEncoder(sparse_output=False, drop='first')
encoded_array = encoder.fit_transform(df_selected)

encoded_df = pd.DataFrame(encoded_array, columns=encoder.get_feature_names_out(df_selected.columns))
```

Figure 3.2.3 One-Hot Encoding

Clustering algorithms require numerical input. Therefore, all categorical features were transformed into numeric format using One-Hot Encoding, as shown in Figure 3.2.3. This technique creates a new binary column for each unique category (excluding the first to avoid redundancy). It allows the algorithm to accurately recognize different payment methods and support statuses without assuming any order or importance.

Step 4: Feature Scaling



```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(encoded_df)
scaled_df = pd.DataFrame(scaled_data, columns=encoded_df.columns)
```

Figure 3.2.4 Feature Scaling

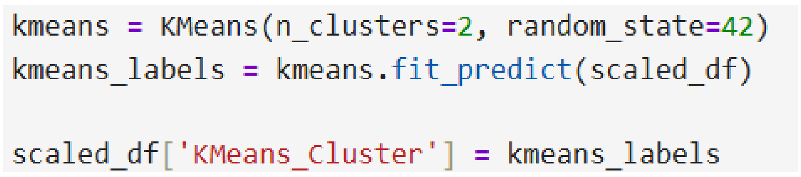
Clustering algorithms use distance to group data. To ensure all features contribute equally, the values were standardized using StandardScaler in Figure 3.2.4.

3.3 Clustering Algorithm

Two clustering techniques, K-Means and DBSCAN, were chosen in order to group consumers according to their support load and payment patterns. These methods represent different clustering approaches, which are partitioning and density-based. Both were applied to the preprocessed dataset containing encoded and scaled representations of PaymentMethod, PaperlessBilling, and TechSupport.

3.3.1 K-Means Clustering

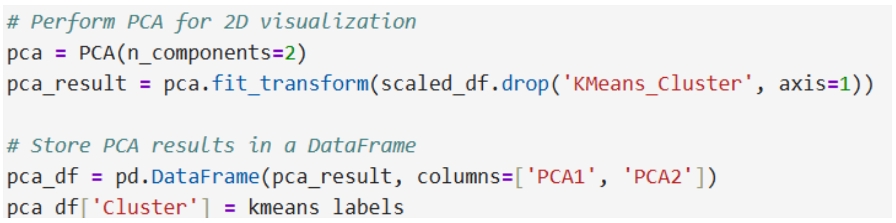
By reducing the distance between data points and their cluster centers, the K-Means partitioning algorithm divides data into 2 clusters, as Figure 3.3.1.1 shown. It was used to support usage patterns and group clients with comparable payment methods. The technique works effectively with data that is well-separated.



```
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans_labels = kmeans.fit_predict(scaled_df)

scaled_df['KMeans_Cluster'] = kmeans_labels
```

Figure 3.3.1.1 Application of K-Means

In Figure 3.3.1.2, PCA is used to reduce the high-dimensional data to two dimensions, making it possible to visualize the clusters on a 2D scatter plot.



```
# Perform PCA for 2D visualization
pca = PCA(n_components=2)
pca_result = pca.fit_transform(scaled_df.drop('KMeans_Cluster', axis=1))

# Store PCA results in a DataFrame
pca_df = pd.DataFrame(pca_result, columns=['PCA1', 'PCA2'])
pca_df['Cluster'] = kmeans_labels
```

Figure 3.3.1.2 Application of PCA

3.3.2 DBSCAN Clustering

DBSCAN is a density-based algorithm that forms clusters based on the proximity of data points and automatically identifies outliers. Unlike K-Means, it does not require specifying the number of clusters, making it suitable for detecting less obvious customer patterns. In this project, DBSCAN was selected to explore hidden structures in payment behaviour and support usage that may not be captured by partitioning methods. The clustering implementation is shown in Figure 3.3.2.1.

```
dbscan = DBSCAN(eps=0.9, min_samples=5)
dbscan_labels = dbscan.fit_predict(scaled_df.drop(['KMeans_Cluster'], axis=1))

scaled_df['DBSCAN_Cluster'] = dbscan_labels
```

Figure 3.3.2.1 Implementation of DBSCAN

DBSCAN successfully uncovered unique customer groups and isolated outliers, offering deeper insights into unusual or rare customer behaviours.

3.4 Result and Interpretation

3.4.1 K-Means Clustering

The K-Means algorithm was applied to partition the customer data into two distinct clusters (K = 2), focusing on payment method, paperless billing, and technical support usage. This clustering process is illustrated in Figure 3.3.1.1, which shows the implementation of the algorithm. To visualize the result, the clustering output was transformed using PCA, using the code shown in Figure 3.4.1.1. The final clustering outcome is presented in Figure 3.4.1.2, where the two customer segments are visually separated.

```
plt.figure(figsize=(8, 5))
sns.scatterplot(data=pca_df, x='PCA1', y='PCA2', hue='Cluster', palette='Set2')
plt.title('K-Means Clustering Result (PCA View)')
plt.show()
```

Figure 3.4.1.1 Code for Visualizing K-Means Clustering

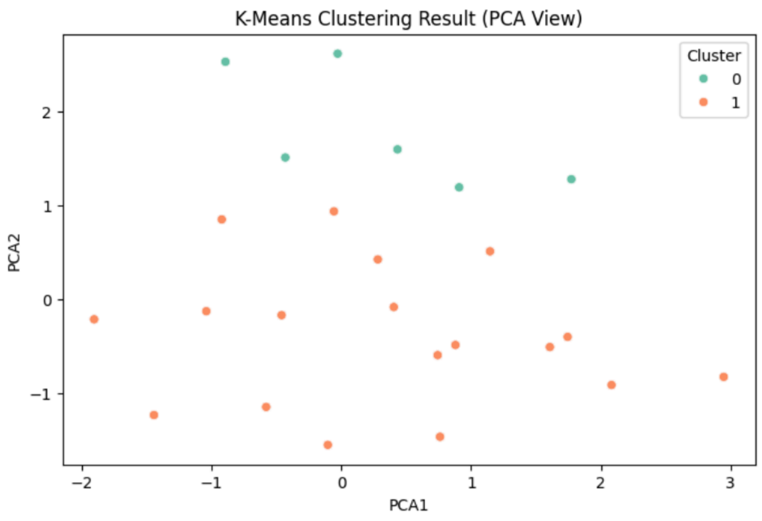


Figure 3.4.1.2 K-Means Clustering Output Based on Payment and Support Behaviour

Figure 3.4.1.2 K-Means Clustering Output Based on Payment and Support Behaviour

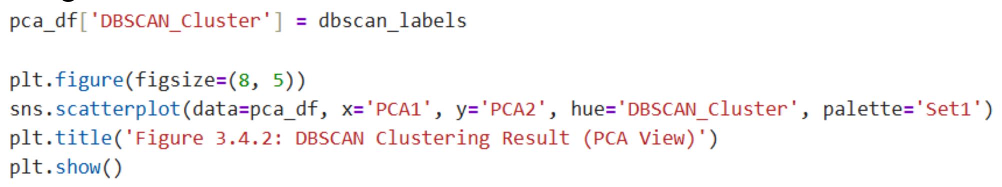
Cluster 0 (green) contains customers who appear more self-reliant, potentially those who use auto-pay methods and rarely require technical support. Cluster 1 (orange) includes customers who might prefer manual payments and have higher support interactions. The clusters are moderately well separated, though some points are close along the PCA1 axis, indicating some overlap in behaviour.

Overall, the silhouette score of 0.281 suggests a fair level of distinction between the two groups. Despite the score not being high, the clustering still offers valuable segmentation for tailoring customer engagement. For example, targeting Cluster 1 with proactive support or payment automation incentives.

3.4.2 DBSCAN Clustering

The DBSCAN algorithm was applied to identify customer clusters based on the density of data points, using the same features: payment method, paperless billing, and technical support usage. As shown in **Figure 3.3.2.1**, DBSCAN does not require specifying the number of clusters and is capable of detecting noise or outliers automatically.

The clustering output was visualized using PCA, and the corresponding code is provided in **Figure 3.4.2.1**. The final clustering result is presented in **Figure 3.4.2.2**, where distinct customer segments are shown in different colours.



```
pca_df['DBSCAN_Cluster'] = dbscan_labels

plt.figure(figsize=(8, 5))
sns.scatterplot(data=pca_df, x='PCA1', y='PCA2', hue='DBSCAN_Cluster', palette='Set1')
plt.title('Figure 3.4.2: DBSCAN Clustering Result (PCA View)')
plt.show()
```

Figure 3.4.2.1 Code for Visualizing DBSCAN Clustering

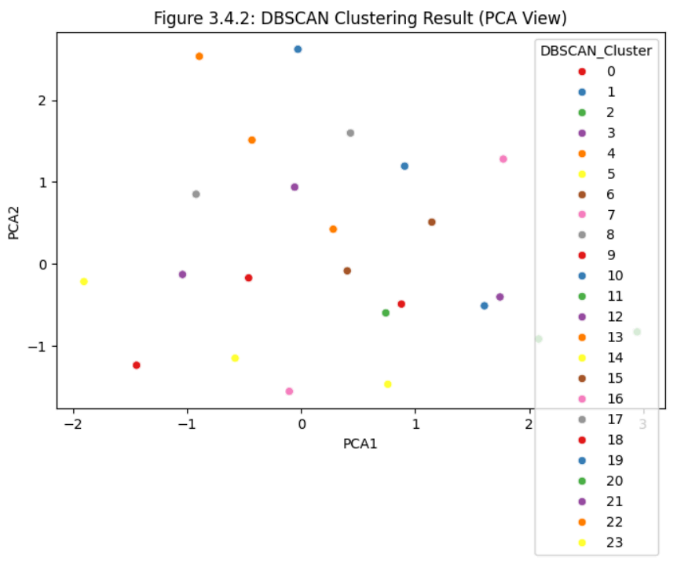


Figure 3.4.2.1 DBSCAN Clustering Output Based on Payment and Support Behaviour

A total of 24 different clusters were found. While the lack of outliers indicates that all records naturally fall into the identified patterns, each cluster consists of customers with comparable support and payment behaviors. DBSCAN provided more detailed categorization than K-Means, showing its ability to detect small behavioral variations.

Its silhouette score of 1.000 provides strong potential for deeper customer profiling and personalized marketing strategies based on support dependency and payment preferences.

3.5 Conclusion

K-Means offered a clear but general segmentation into two customer groups, while DBSCAN provided more detailed and well-separated clusters. DBSCAN's higher silhouette score indicates stronger performance, making it more suitable for uncovering subtle behavioural patterns.

4.0 Reflection

4.1 Project Reflection

Ahmad Ziyaad

For Project 2, I was mainly involved in clustering customer profiles based on their usage of digital services. This required selecting and preparing features such as OnlineSecurity, TechSupport, and StreamingTV, among others. I converted categorical entries into binary format and standardized the data to ensure all features contributed equally. I applied both K-Means and Agglomerative Hierarchical Clustering to uncover patterns in customer engagement. To support interpretation, I used Principal Component Analysis (PCA) to reduce the data into two dimensions for clear visualization.

From this task, I learned how clustering could uncover distinct customer behavior profiles without the need for labeled data. The consistency between K-Means and Agglomerative results gave me confidence in the segmentation. It also taught me how preprocessing can influence clustering performance. This experience helped me strengthen my Python skills and understand how to draw meaningful business insights from unsupervised learning techniques.

Goe Jie Ying

Through both projects, I gained a deeper understanding of how different machine learning techniques can be applied to real-world business problems. In Project 1, I explored how contract types and payment methods influence customer churn. By applying and comparing models like Decision Tree, Random Forest, and SVM, I learned how to evaluate classification results using key metrics such as accuracy, precision, recall, and F1-score. This taught me how model performance can vary based on the nature of the data and the business objective.

In Project 2, I focused on clustering techniques to segment customers based on payment behaviour and support load. Working with both K-Means and DBSCAN helped me understand the difference between partitioning and density-based clustering, and how preprocessing and PCA can influence the final output. I learned that clustering is not just about forming groups, but also about interpreting those segments in a business context to gain meaningful insights. Overall, these projects enhanced my practical skills in data handling, model evaluation, and visualization, while also improving my ability to translate data patterns into actionable conclusions.

Teh Ru Qian

In Project 1, which focused on classification, I developed a machine learning model to predict whether or not a customer would churn. The process began with preparing and cleaning the data, then selecting features and pattern analysis. Many classification models, such as K-Nearest Neighbours, Random Forest and Decision Tree, were used. These models were evaluated by accuracy, precision, recall and F1 score. One important insight was that customers were more likely to quit if they did not use optional services like TechSupport or

OnlineSecurity. This made it easier to understand how customers behave and how loyalty is impacted by service use.

In Project 2, I focused on clustering, where customers were grouped without labels according to similar characteristics. Prior to applying K-Means and Agglomerative Clustering, the data was normalised and features like tenure, monthly charges and total charges were used. The clusters displayed many consumer types, including new budget users, new premium users, and devoted high spenders. A challenge was selecting the appropriate number of clusters and correctly interpreting them. However, the project enhanced knowledge of data visualisation, unsupervised learning and how clustering can assist companies in developing focused marketing strategies.

4.2 Course Reflection

Ahmad Ziyaad

Taking this course gave me a deeper appreciation of the real-world impact of data mining. At first, the theoretical parts—especially around clustering and classification—were challenging. However, after engaging in practical projects and hands-on coding, those concepts became much clearer. I saw firsthand how crucial data preparation steps like encoding, cleaning, and scaling are to any analysis.

I especially enjoyed the clustering section, where we grouped customers based on behavior without having any labels. It gave me a glimpse into how companies personalize marketing or improve user experience using data-driven decisions. Overall, this course improved not just my technical ability with Python and machine learning tools, but also my critical thinking when approaching business problems. I now feel more confident using data to solve practical challenges in any industry.

Goe Jie Ying

This course has been a meaningful learning experience for me. At first, I found data mining quite boring, especially with all the theory in the slides. But through lectures, assignments, and projects, I started to understand the topics better and gained hands-on experience. I learned how to handle missing data, detect outliers, and prepare data properly, which showed me the importance of data quality.

I found classification and clustering topics the most interesting, as they showed how data can be used to make predictions or group customers. The group projects also helped me apply what I learned to real situations. Overall, this course improved my problem-solving and teamwork skills, and gave me a strong foundation in data analysis that will be useful in the future.

Teh Ru Qian

Throughout this course, the transition from basic data mining concepts to real-world machine learning approaches has been both difficult and rewarding. The topics discussed, which from data preprocessing, classification, clustering and association rule mining have helped to the building of a strong basis in data analysis and interpretation. The assignments and group projects made it possible to bring theory into reality, especially with tools like Python and machine learning libraries. Overall , the expected learning objectives were met, especially in terms of obtaining hands-on experience with data preparation, model evaluation and insight development. The development of new skills included handling missing values, creating models for classification and clustering and identifying patterns in the data.These skills will be useful for career positions requiring data-driven decision-making, especially for domains such as marketing, IT solutions or business analytics.