**Report on Customer Churn Analysis and Clustering**

**1. Introduction**

This report outlines the methodology used for data preparation and clustering analysis of customer churn data. The aim is to preprocess the dataset, apply K-Means clustering, and visualize the results. Understanding customer segments can help in strategizing retention efforts and improving service offerings.

**2. Data Preparation**

**2.1 Dataset Overview**

The dataset, named Dataset.csv, contains various customer attributes related to service usage, demographics, and churn status. The key features include:

* gender: Gender of the customer.
* Dependents: Whether the customer has dependents.
* PhoneService: Whether the customer has a phone service.
* MultipleLines: Whether the customer has multiple phone lines.
* InternetService: Type of internet service.
* Contract: Type of contract the customer is under.
* Churn: Target variable indicating whether the customer has churned.

**2.2 Data Loading**

The dataset was loaded using pandas:

python

Copy code

df = pd.read\_csv('Dataset.csv')

**2.3 Handling Missing Values**

Missing values in the dataset were handled using the forward fill method (commented out in the provided code):

python

Copy code

# df.fillna(method='ffill', inplace=True)

**2.4 Encoding Categorical Variables**

Categorical columns were encoded using LabelEncoder to convert them into numerical format. The following columns were transformed:

* gender
* Dependents
* PhoneService
* MultipleLines
* InternetService
* Contract
* Churn

This was implemented with the following code:

python

Copy code

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for column in categorical\_columns:

df[column] = le.fit\_transform(df[column])

**2.5 Saving Preprocessed Data**

The preprocessed dataset was saved as preprocessed\_dataset.csv for future use:

python

Copy code

df.to\_csv('Data\_Preparation/preprocessed\_dataset.csv', index=False)

**3. Data Splitting**

**3.1 Train-Test Split**

The dataset was split into training and testing sets, with 70% of the data allocated for training and 30% for testing. The target variable is Churn.

python

Copy code

from sklearn.model\_selection import train\_test\_split

X = df.drop(['Churn'], axis=1)

y = df['Churn']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

**3.2 Saving Split Data**

The splits were saved into separate CSV files:

python

Copy code

X\_train.to\_csv('Data\_Preparation/X\_train.csv', index=False)

X\_test.to\_csv('Data\_Preparation/X\_test.csv', index=False)

y\_train.to\_csv('Data\_Preparation/y\_train.csv', index=False)

y\_test.to\_csv('Data\_Preparation/y\_test.csv', index=False)

**4. Scaling Data**

**4.1 Standardization**

The features were standardized using StandardScaler, which normalizes the data to have a mean of 0 and a standard deviation of 1. This ensures that all features contribute equally to the distance calculations in clustering.

python

Copy code

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

**4.2 Saving Scaled Data**

The scaled training and testing datasets were saved as CSV files:

python

Copy code

pd.DataFrame(X\_train\_scaled).to\_csv('Data\_Preparation/X\_train\_scaled.csv', index=False)

pd.DataFrame(X\_test\_scaled).to\_csv('Data\_Preparation/X\_test\_scaled.csv', index=False)

**5. Clustering Analysis**

**5.1 K-Means Clustering**

K-Means clustering was performed to segment the customer data into distinct clusters. The Elbow method was used to determine the optimal number of clusters by plotting the within-cluster sum of squares (WCSS).

python

Copy code

wcss = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters=i, init='k-means++', random\_state=42)

kmeans.fit(X\_train\_scaled)

wcss.append(kmeans.inertia\_)

**5.2 Elbow Method Visualization**

The Elbow method was visualized with a line plot to identify the optimal number of clusters:

python

Copy code

plt.plot(range(1, 11), wcss)

plt.title('Elbow Method')

plt.xlabel('Number of Clusters')

plt.ylabel('WCSS')

plt.savefig('Clustering\_Analysis/elbow\_method.png')

**5.3 Training K-Means Model**

Based on the Elbow method, the K-Means model was trained with 3 clusters:

python

Copy code

kmeans = KMeans(n\_clusters=3, init='k-means++', random\_state=42)

kmeans.fit(X\_train\_scaled)

**5.4 Saving the K-Means Model**

The trained K-Means model was saved using joblib for future predictions:

python

Copy code

import joblib

joblib.dump(kmeans, 'Clustering\_Analysis/kmeans\_model.pkl')

**6. Cluster Visualization**

**6.1 Assigning Clusters**

The clusters were assigned to the training data points, and a new column Cluster was added to the original DataFrame:

python

Copy code

clusters\_train = kmeans.predict(X\_train\_scaled)

df\_train = X\_train.copy()

df\_train['Cluster'] = clusters\_train

**6.2 Saving Training Data with Clusters**

The training data with the assigned clusters was saved:

python

Copy code

df\_train.to\_csv('Data\_Preparation/train\_with\_clusters.csv', index=False)

**6.3 Scatter Plot of Clusters**

Clusters were visualized based on tenure and MonthlyCharges using a scatter plot:

python

Copy code

sns.scatterplot(x=df\_train['tenure'], y=df\_train['MonthlyCharges'], hue=df\_train['Cluster'], palette='coolwarm')

plt.title('Clusters based on Tenure and MonthlyCharges')

plt.savefig('Clustering\_Analysis/clusters\_visualization.png')

**7. Conclusion**

The analysis successfully prepared the dataset, applied K-Means clustering, and visualized the results. The clustering helps to identify distinct customer segments based on their service usage and demographic features, which can guide retention strategies.