"Churn.csv" dataset, here's a comprehensive multistep summary:

1. \*\*Dataset Overview:\*\*

- The "Churn.csv" dataset contains information about customer attributes and behaviors in the telecom sector.

- It includes features such as 'Account length', 'Total day minutes', 'Total evening minutes', 'Customer service calls', among others.

- The target variable 'Churn' indicates whether a customer has churned (TRUE) or not (FALSE).

2. \*\*Data Exploration:\*\*

- The dataset comprises both numerical and categorical features, offering insights into various aspects of customer behavior and service usage.

- Initial exploration revealed the presence of features such as call durations, service-related attributes, and customer service interactions.

3. \*\*Objective and Context:\*\*

- The primary objective is to understand factors associated with customer churn in the telecom sector.

- The dataset provides an opportunity to assess patterns and correlations between different customer attributes and the likelihood of churn.

4. \*\*Data Preprocessing:\*\*

- Upon loading the dataset, data preprocessing tasks such as handling missing values, encoding categorical variables, and scaling numerical features were performed to ensure data readiness for subsequent analysis.

5. \*\*Visualizations and Insights:\*\*

- Visualizations were created to analyze relationships and distributions within the dataset, providing insights into key features and their potential impact on customer churn.

- Visualization techniques included box plots, pair plots, correlation heatmaps, and count plots, revealing patterns and potential correlations between features and churn.

6. \*\*Next Steps:\*\*

- The dataset exhibits a rich array of customer attributes and behaviors, allowing for more detailed exploratory analysis and modeling techniques.

Data Preparation, including Label Encoding and Feature Scaling.

import pandas as pd  
from sklearn.preprocessing import LabelEncoder  
# Load the datasetfile\_path = "path\_to\_your\_dataset.csv"  
df = pd.read\_csv("Churn.csv")  
# Print column names  
print(df.columns)  
 #Verify the existence of 'Categorical\_Column'  
if 'Categorical\_Column' in df.columns:  
 label\_encoder = LabelEncoder()  
 df['Categorical\_Column'] = label\_encoder.fit\_transform(df ['Categorical\_Column'])  
else:  
 print("Column 'Categorical\_Column' does not exist in the DataFrame.")

OUTPUT

Index(['Account length', 'Area code', 'Total day minutes', 'Total day calls',

'Total day charge', 'Total eve minutes', 'Total eve calls',

'Total eve charge', 'Total night minutes', 'Total night calls',

'Total night charge', 'Total intl minutes', 'Total intl calls',

'Total intl charge', 'Customer service calls', 'Churn'],

dtype='object')

Column 'Categorical\_Column' does not exist in the DataFrame.

VISUALIZATIONS

import seaborn as sns

import matplotlib.pyplot as plt

# Visualization 1: Boxplot of Total day minutes vs. Churn

plt.figure(figsize=(10, 6))

sns.boxplot(x='Churn', y='Total day minutes', data=df)

plt.title('Distribution of Total Day Minutes by Churn')

plt.show()

# Visualization 2: Pairplot of Numerical Features with Churn

numerical\_features = ['Total day minutes', 'Total eve minutes', 'Total night minutes', 'Total intl minutes', 'Customer service calls']

sns.pairplot(df, vars=numerical\_features, hue='Churn', kind='scatter', diag\_kind='kde', plot\_kws={'alpha':0.6})

plt.suptitle('Pairplot of Numerical Features with Churn', y=1.02)

plt.show()

# Visualization 3: Correlation Heatmap

plt.figure(figsize=(10, 8))

correlation\_matrix = df.corr()

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm')

plt.title('Correlation Heatmap')

plt.show()

# Visualization 4: Countplot of Customer Service Calls and Churn

plt.figure(figsize=(8, 6))

sns.countplot(data=df, x='Customer service calls', hue='Churn')

plt.title('Count of Customer Service Calls by Churn')

plt.show()

1. \*\*Boxplot of Total day minutes vs. Churn:\*\* This plot allows you to compare the distribution of total day minutes for customers who churned and those who did not.

import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
# Load the CSV file into a DataFrame  
file\_path = "Churn.csv"  
df = pd.read\_csv(file\_path)  
# Visualization 1: Boxplot of Total day minutes vs. Churn  
plt.figure(figsize=(10, 6))  
sns.boxplot(x='Churn', y='Total day minutes', data=df)  
plt.title('Distribution of Total Day Minutes by Churn')  
plt.show()

A diagram of a distribution of a number of blue rectangular objects

Description automatically generated

1. \*\*Pairplot of Numerical Features with Churn:\*\* A pairplot visualizes relationships between multiple numerical features and the target 'Churn', providing insights into any patterns or differences between churned and non-churned customers.

import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
file\_path = "Churn.csv"  
df = pd.read\_csv(file\_path)  
# Pairplot of Numerical Features with Churn  
numerical\_features = ['Total day minutes', 'Total eve minutes', 'Total night minutes', 'Total intl minutes', 'Customer service calls']  
sns.pairplot(df, vars=numerical\_features, hue='Churn', kind='scatter', diag\_kind='kde', plot\_kws={'alpha':0.6})  
plt.suptitle('Pairplot of Numerical Features with Churn', y=1.02)  
plt.show()

A collage of graphs and diagrams

Description automatically generated

1. \*\*Correlation Heatmap:\*\* The heatmap visualizes the correlation matrix of numerical features, revealing potential relationships and multicollinearity, providing a deeper understanding of feature interactions.

import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
DataFramefile\_path ="Churn.csv"  
df = pd.read\_csv(DataFramefile\_path)  
# Correlation Heatmap  
correlation\_matrix = df.corr()  
plt.figure(figsize=(10, 8))  
sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm')  
plt.title('Correlation Heatmap')  
plt.show()

A screenshot of a graph

Description automatically generated

4. \*\*Countplot of Customer Service Calls and Churn:\*\* This plot showcases the count of customer service calls for churned and non-churned customers, offering insights into the impact of customer service inter

actions on churn.

import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
file\_path = "Churn.csv"  
df = pd.read\_csv(file\_path)  
# Countplot of Customer Service Calls and Churn  
plt.figure(figsize=(8, 6))  
sns.countplot(data=df, x='Customer service calls', hue='Churn')  
plt.title('Count of Customer Service Calls by Churn')  
plt.show()

A graph of customer service calls

Description automatically generated

THE SPLIT

The shape of X is (667, 6), which means there is 667 samples and 6 features, and the shape of y is (667,), signifying that there is 667 samples in the target variable. This aligns with the expected outcome, indicating that the data has been structured appropriately for the next steps in your analysis.

import pandas as pd  
# Load the dataset  
file\_path = "Churn2.csv"  
df = pd.read\_csv(file\_path)  
# Define features and the target variable  
feature\_columns = ["Total day minutes", "Total eve minutes", "Total night minutes", "Total intl minutes", "Customer service calls", "Account length"]  
target\_variable = "Churn"  
# Create the feature matrix (X) and the target vector (y)  
X = df[feature\_columns]  
y = df[target\_variable]  
# Optional: Display the shape of X and y to verify the split  
print("Shape of X:", X.shape)  
print("Shape of y:", y.shape)

OUTPUT

Shape of X: (667, 6)

Shape of y: (667,)

K NEAREST NEIGHBORS

K Nearest Neighbors (KNN) model achieved an accuracy of approximately 89.55% on the testing data. This means that the model's predictions aligned with the actual churn outcomes in the test set with high accuracy.

The accuracy metric indicates the proportion of correctly classified instances out of the total instances in the test set. An accuracy of 89.55% suggests that the KNN model, based on the provided features, is reasonably effective at predicting customer churn.

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.metrics import accuracy\_score  
  
# Load the dataset  
file\_path = "Churn2.csv"  
df = pd.read\_csv(file\_path)  
  
# Define features and the target variable  
feature\_columns = ["Total day minutes", "Total eve minutes", "Total night minutes", "Total intl minutes", "Customer service calls", "Account length"]  
target\_variable = "Churn"  
  
# Create the feature matrix (X) and the target vector (y)  
X = df[feature\_columns]  
y = df[target\_variable]  
  
# Split the data into training and testing sets  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Initialize KNN classifier (with, for example, k=3)  
knn = KNeighborsClassifier(n\_neighbors=3)  
  
# Fit the model on the training data  
knn.fit(X\_train, y\_train)  
  
# Make predictions on the testing data  
y\_pred = knn.predict(X\_test)  
  
# Evaluate the accuracy of the model  
accuracy = accuracy\_score(y\_test, y\_pred)  
print("Accuracy:", accuracy)

OUTPUT

Accuracy: 0.8955223880597015

CROSS VALIDATION

Implementing cross-validation is a valuable step in assessing the robustness of my model. Here's an example of how you can incorporate k-fold cross-validation using scikit-learn's cross\_val\_score to obtain a more comprehensive evaluation of my K Nearest Neighbors (KNN) model:

import pandas as pd  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.model\_selection import cross\_val\_score  
import numpy as np  
  
# Load the dataset  
file\_path = "Churn.csv"  
df = pd.read\_csv(file\_path)  
  
# Define features and the target variable  
feature\_columns = ["Total day minutes", "Total eve minutes", "Total night minutes", "Total intl minutes", "Customer service calls", "Account length"]  
target\_variable = "Churn"  
  
# Create the feature matrix (X) and the target vector (y)  
X = df[feature\_columns]  
y = df[target\_variable]  
  
# Initialize KNN classifier (with, for example, k=3)  
knn = KNeighborsClassifier(n\_neighbors=3)  
  
# Perform 5-fold cross-validation  
cv\_scores = cross\_val\_score(knn, X, y, cv=5)  
  
# Print the cross-validation scores  
print("Cross-validation Scores:", cv\_scores)  
print("Mean CV Score:", np.mean(cv\_scores))

OUTPUT

Cross-validation Scores: [0.85074627 0.88059701 0.87218045 0.81954887 0.89473684]

Mean CV Score: 0.8635618897991246

The output indicates that the K Nearest Neighbors (KNN) model achieved an average cross-validation score of approximately 86.36% across the 5 folds. This score provides a more comprehensive assessment of the model's performance and how well it generalizes to new data.

The cross-validation scores of individual folds, along with the mean score, offer valuable insights into the model's consistency and robustness.

GRAPHIC of OPTIMAL NUMBER of NEIGHBORS

GridSearchCV is employed to perform a grid search over the specified range of neighbors (k) and extract the mean cross-validation scores, which are then plotted to visualize the optimal number of neighbors for the KNN algorithm.

A graph with a line

Description automatically generated