**How effectively can machine learning models predict customer churn based on demographic and service plan data, and what are the key features influencing churn in telecom customers?**

**PROJECT NAME: Customer Churn Prediction Using Machine Learning Techniques: A Data-Driven Approach**

CODE:

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from seaborn import pairplot

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

from sklearn.preprocessing import LabelEncoder

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import plot\_tree

from sklearn.metrics import classification\_report, accuracy\_score, confusion\_matrix

np.random.seed(0)

n=200

customer\_id=np.arange(1,n+1)

gender=np.random.choice(['Male','Female','Others'],size=n)

age=np.random.randint(18,80,size=n)

tenure=np.random.randint(1,20,size=n)

service\_plan=np.random.choice([1,2,3,4],size=n)

churn=np.random.choice(['yes','no'],size=n,p=[0.2,0.8])

df=pd.DataFrame({

    'customer\_id':customer\_id,

    'gender':gender,

    'age':age,

    'tenure':tenure,

    'service\_plan':service\_plan,

    'churn':churn

})

print("First data overview:")

print(df.head())

print("\n Missing Values:")

print(df.isnull().sum())

# check for outliers in age and tenure

sns.boxplot(x=df['age'],color='olive')

plt.title('Boxplot of Age')

plt.show()

sns.boxplot(x=df['tenure'],color='pink')

plt.title('Boxplot of tenure')

plt.show()

#Gender distribution

print("\nGender distribution ")

print(df['gender'].value\_counts())

sns.countplot(x='gender',data=df,palette='pastel')

plt.title('Gender distribution')

plt.show()

#Service Plan Distribution

print("\nService Plan Distribution")

print(df['service\_plan'].value\_counts)

sns.countplot(x='service\_plan',data=df,palette='pastel')

plt.title('Service Plan Distribution')

plt.show()

# Summary Statistics for Age and Tenure

print("\n Age Summary")

print(df['age'].describe())

print("\nTenure Summary")

print(df['tenure'].describe())

#Histogram for Age and Tenure

sns.histplot(df['age'],kde=True,color='orange')

plt.title('Age Distribution')

plt.show()

sns.histplot(df['tenure'],kde=True,color='red')

plt.title('Tenure Distribution')

plt.show()

#Bivarite Analysis

#Gender vs Churn

sns.countplot(x='gender',hue='churn',data=df)

plt.title('Gender vs churn')

plt.show()

#Age vs Tenure

sns.scatterplot(x='age',y='tenure',data=df)

plt.title('Age vs Tenure')

plt.show()

#Chure vs Age

sns.boxplot(y='age',x='churn',data=df,palette='pastel')

plt.title('Age vs Tenure')

plt.show()

#Multivariate Analysis

#Scatter Plot Matrix (Age,Tenure,Service Plan)

sns.pairplot(df[['age', 'tenure', 'service\_plan', 'gender']], hue='gender', palette='pastel')

plt.suptitle('Pairplot of Age, Tenure, Service Plan by Gender', y=1.02)

plt.show()

#Correlation Heatmap(Age and Tenure)

correlation=df[['age','tenure']].corr()

sns.heatmap(correlation,annot=True)

plt.title('Correlation Heatmap of Age and Tenure')

plt.show()

# Encode categorical variables

le\_gender = LabelEncoder()

df['gender\_encoded'] = le\_gender.fit\_transform(df['gender'])

le\_churn = LabelEncoder()

df['churn\_encoded'] = le\_churn.fit\_transform(df['churn'])

# Features and target

X = df[['age', 'tenure', 'service\_plan', 'gender\_encoded']]

y = df['churn\_encoded']

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# ---------------------------

# Logistic Regression Model

# ---------------------------

lr\_model = LogisticRegression()

lr\_model.fit(X\_train, y\_train)

lr\_pred = lr\_model.predict(X\_test)

print("\n🔷 Logistic Regression Results:")

print("Accuracy:", accuracy\_score(y\_test, lr\_pred))

print("Classification Report:\n", classification\_report(y\_test, lr\_pred))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, lr\_pred))

# ---------------------------

# Decision Tree Model

# ---------------------------

dt\_model = DecisionTreeClassifier(random\_state=42)

dt\_model.fit(X\_train, y\_train)

dt\_pred = dt\_model.predict(X\_test)

print("\n🔷 Decision Tree Results:")

print("Accuracy:", accuracy\_score(y\_test, dt\_pred))

print("Classification Report:\n", classification\_report(y\_test, dt\_pred))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, dt\_pred))

# ---------------------------

# Random Forest Model

# ---------------------------

rf\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_model.fit(X\_train, y\_train)

rf\_pred = rf\_model.predict(X\_test)

print("\n🔷 Random Forest Results:")

print("Accuracy:", accuracy\_score(y\_test, rf\_pred))

print("Classification Report:\n", classification\_report(y\_test, rf\_pred))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, rf\_pred))

models = ['Logistic Regression', 'Decision Tree', 'Random Forest']

accuracies = [accuracy\_score(y\_test, lr\_pred),

              accuracy\_score(y\_test, dt\_pred),

              accuracy\_score(y\_test, rf\_pred)]

comparison\_df = pd.DataFrame({'Model': models, 'Accuracy': accuracies})

print(comparison\_df)

importances = rf\_model.feature\_importances\_

feature\_names = X.columns

sns.barplot(x=importances, y=feature\_names, palette='pastel')

plt.title('Feature Importances (Random Forest)')

plt.show()