## Create Churn Prediction Model – Random Forest

Now we will work with an application called Jupyter Notebook and we will coding our ML model in Python. Easiest way to install both them is to install the ANACONDA Software Package. You can follow the below link to do so:

<https://docs.anaconda.com/anaconda/install/>

## Installing Libraries

Open the Anaconda Command Prompt and run below code:pip install pandas numpy matplotlib seaborn scikit-learn joblib openpyxl

Open Jupyter Notebook, create a new notebook and write below code:

1. **Importing Libraries & Data Load**

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.metrics import classification\_report, confusion\_matrix  
from sklearn.preprocessing import LabelEncoder  
import joblib

# Define the path to the Excel file

file\_path = r"C:\yourpath\Prediction\_Data.xlsx"

# Define the sheet name to read data from  
sheet\_name = 'vw\_ChurnData'

# Read the data from the specified sheet into a pandas DataFrame  
data = pd.read\_excel(file\_path, sheet\_name=sheet\_name)

# Display the first few rows of the fetched data  
print(data.head())

1. **Data Preprocessing**

# Drop columns that won't be used for prediction

data = data.drop(['Customer\_ID', 'Churn\_Category', 'Churn\_Reason'], axis=1)

# List of columns to be label encoded

columns\_to\_encode = [

'Gender', 'Married', 'State', 'Value\_Deal', 'Phone\_Service', 'Multiple\_Lines',

'Internet\_Service', 'Internet\_Type', 'Online\_Security', 'Online\_Backup',

'Device\_Protection\_Plan', 'Premium\_Support', 'Streaming\_TV', 'Streaming\_Movies',

'Streaming\_Music', 'Unlimited\_Data', 'Contract', 'Paperless\_Billing',

'Payment\_Method'

]

# Encode categorical variables except the target variable

label\_encoders = {}

for column in columns\_to\_encode:

label\_encoders[column] = LabelEncoder()

data[column] = label\_encoders[column].fit\_transform(data[column])

# Manually encode the target variable 'Customer\_Status'

data['Customer\_Status'] = data['Customer\_Status'].map({'Stayed': 0, 'Churned': 1})

# Split data into features and target

X = data.drop('Customer\_Status', axis=1)

y = data['Customer\_Status']

# Split 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)

1. **Train Random Forest Model**

# Initialize the Random Forest Classifier

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

# Train the model

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

1. **Evaluate Model**

# Make predictions

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

# Evaluate the model

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

# Feature Selection using Feature Importance

importances = rf\_model.feature\_importances\_

indices = np.argsort(importances)[::-1]

# Plot the feature importances

plt.figure(figsize=(15, 6))

sns.barplot(x=importances[indices], y=X.columns[indices])

plt.title('Feature Importances')

plt.xlabel('Relative Importance')

plt.ylabel('Feature Names')

plt.show()

1. **Use Model for Prediction on New Data**

# Define the path to the Joiner Data Excel file

file\_path = r"C:\yourpath\Prediction\_Data.xlsx"

# Define the sheet name to read data from

sheet\_name = 'vw\_JoinData'

# Read the data from the specified sheet into a pandas DataFrame

new\_data = pd.read\_excel(file\_path, sheet\_name=sheet\_name)

# Display the first few rows of the fetched data

print(new\_data.head())

# Retain the original DataFrame to preserve unencoded columns

original\_data = new\_data.copy()

# Retain the Customer\_ID column

customer\_ids = new\_data['Customer\_ID']

# Drop columns that won't be used for prediction in the encoded DataFrame

new\_data = new\_data.drop(['Customer\_ID', 'Customer\_Status', 'Churn\_Category', 'Churn\_Reason'], axis=1)

# Encode categorical variables using the saved label encoders

for column in new\_data.select\_dtypes(include=['object']).columns:

new\_data[column] = label\_encoders[column].transform(new\_data[column])

# Make predictions

new\_predictions = rf\_model.predict(new\_data)

# Add predictions to the original DataFrame

original\_data['Customer\_Status\_Predicted'] = new\_predictions

# Filter the DataFrame to include only records predicted as "Churned"

original\_data = original\_data[original\_data['Customer\_Status\_Predicted'] == 1]

# Save the results

original\_data.to\_csv(r"C:\yourpath\Predictions.csv", index=False)