

# TELECOM CUSTOMER CHURN ANALYSIS (POWER BI + SQL + MACHINE LEARNING)

## PYTHON CODE - CREATE CHURN PREDICTION MODEL – RANDOM FOREST

### 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\ Churn Prediction.csv "

# Define the sheet name to read data from
sheet_name = 'vw_ChurnData'

# Read the data from the specified sheet into a panda DataFrame
data = pd.read_csv(file_path)

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

### 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)

```

## 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)

```

## 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()

```

## Use Model for Prediction on New Data

```

# Define the path to the Joiner Data Excel file
file_path = r"C:\YOURPATH\ Joined Prediction.csv "

# 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_csv(file_path)

# 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\ Joined Prediction.csv ", index=False)
```