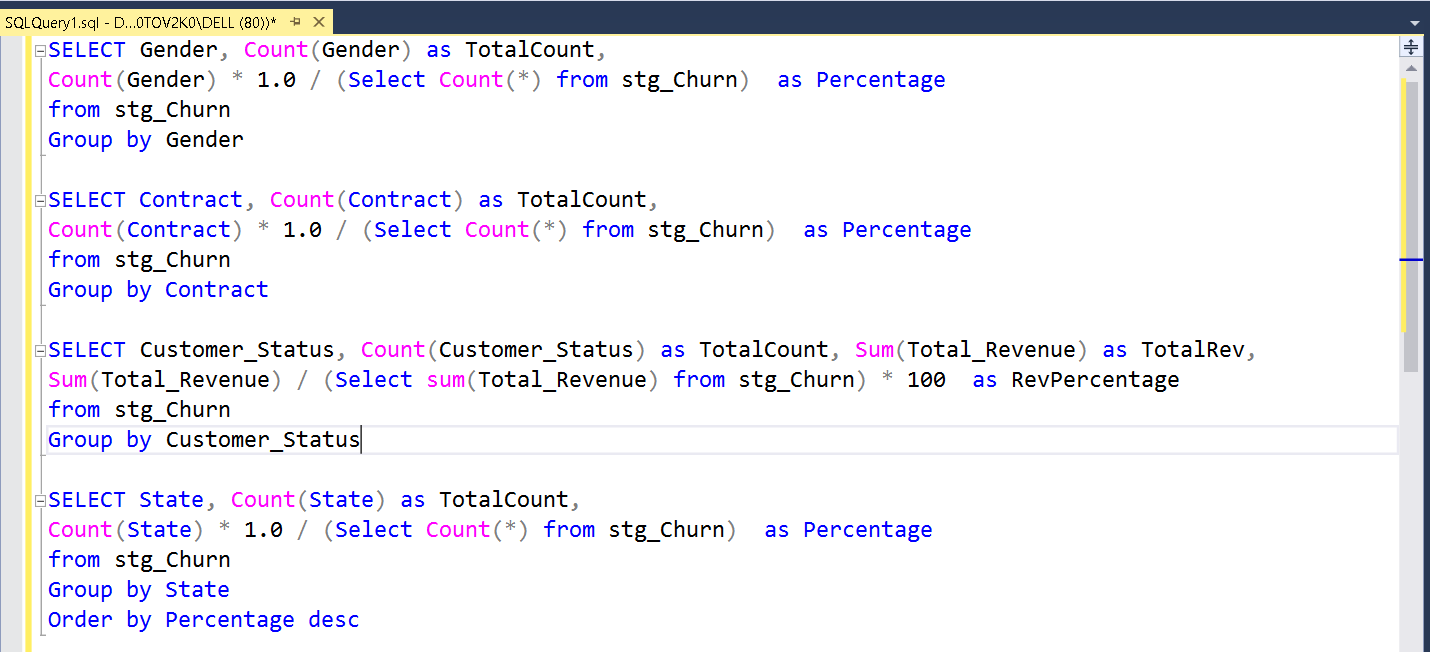
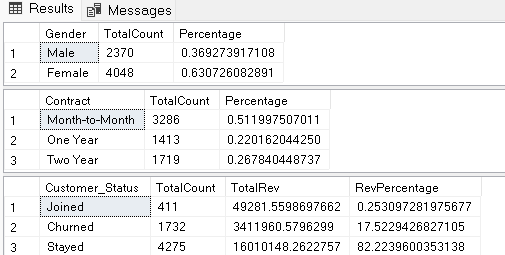
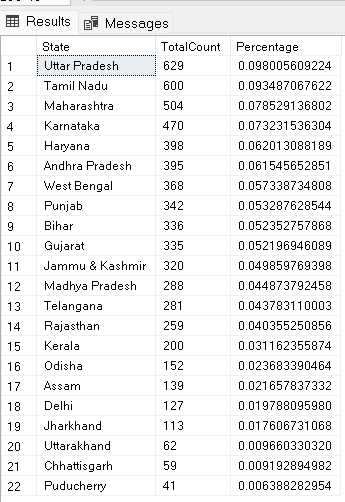
**Project Title**

Customer Churn Analysis and Prediction Using Machine Learning Models**.**

**Introduction:** In this project, I predicted customer churn for a telecom company using data sourced from Kaggle. SQL was employed for ETL processes, cleaning and preprocessing the data to ensure it was analysis-ready. Power BI was utilized to create an interactive dashboard, visualizing key metrics and providing valuable insights into customer behaviour. The Random Forest Classifier was then applied to predict churners, offering the company actionable insights to improve customer retention strategies.

**Data Exploration – Using SQL**



****

**Data Exploration** -,**Remove null and insert the new data into Prod table**

SELECT

Customer\_ID,

Gender,

Age,

Married,

State,

Number\_of\_Referrals,

Tenure\_in\_Months,

ISNULL(Value\_Deal, 'None') AS Value\_Deal,

Phone\_Service,

ISNULL(Multiple\_Lines, 'No') As Multiple\_Lines,

Internet\_Service,

ISNULL(Internet\_Type, 'None') AS Internet\_Type,

ISNULL(Online\_Security, 'No') AS Online\_Security,

ISNULL(Online\_Backup, 'No') AS Online\_Backup,

ISNULL(Device\_Protection\_Plan, 'No') AS Device\_Protection\_Plan,

ISNULL(Premium\_Support, 'No') AS Premium\_Support,

ISNULL(Streaming\_TV, 'No') AS Streaming\_TV,

ISNULL(Streaming\_Movies, 'No') AS Streaming\_Movies,

ISNULL(Streaming\_Music, 'No') AS Streaming\_Music,

ISNULL(Unlimited\_Data, 'No') AS Unlimited\_Data,

Contract,

Paperless\_Billing,

Payment\_Method,

Monthly\_Charge,

Total\_Charges,

Total\_Refunds,

Total\_Extra\_Data\_Charges,

Total\_Long\_Distance\_Charges,

Total\_Revenue,

Customer\_Status,

ISNULL(Churn\_Category, 'Others') AS Churn\_Category,

ISNULL(Churn\_Reason , 'Others') AS Churn\_Reason

INTO [db\_Churn].[dbo].[prod\_Churn]

FROM [db\_Churn].[dbo].[stg\_Churn];

**Create View for Power BI**

Create View vw\_ChurnData as

select \* from prod\_Churn where Customer\_Status In ('Churned', 'Stayed')

Create View vw\_JoinData as

select \* from prod\_Churn where Customer\_Status = 'Joined'

**Power Query Transformations**

**Add a new column in prod\_Churn**

1. Churn Status = if [Customer\_Status] = "Churned" then 1 else 0
2. Change Churn Status data type to numbers
3. Monthly Charge Range = if [Monthly\_Charge] < 20 then "< 20" else if [Monthly\_Charge] < 50 then "20-50" else if [Monthly\_Charge] < 100 then "50-100" else "> 100"

**Create a New Table Reference for mapping\_AgeGrp**

1. Keep only Age column and remove duplicates
2. Age Group = if [Age] < 20 then "< 20" else if [Age] < 36 then "20 - 35" else if [Age] < 51 then "36 - 50" else "> 50"
3. AgeGrpSorting = if [Age Group] = "< 20" then 1 else if [Age Group] = "20 - 35" then 2 else if [Age Group] = "36 - 50" then 3 else 4
4. Change data type of AgeGrpSorting

**Create a new table reference for mapping\_TenureGrp**

1. Keep only Tenure\_in\_Months and remove duplicates
2. Tenure Group = if [Tenure\_in\_Months] < 6 then "< 6 Months" else if [Tenure\_in\_Months] < 12 then "6-12 Months" else if [Tenure\_in\_Months] < 18 then "12-18 Months" else if [Tenure\_in\_Months] < 24 then "18-24 Months" else ">= 24 Months"
3. TenureGrpSorting = if [Tenure\_in\_Months] = "< 6 Months" then 1 else if [Tenure\_in\_Months] = "6-12 Months" then 2 else if [Tenure\_in\_Months] = "12-18 Months" then 3 else if [Tenure\_in\_Months] = "18-24 Months " then 4 else 5
4. Change data type of TenureGrpSorting

**Create a new table reference for prod\_Services**

1. Unpivot services columns
2. Rename Column –
   1. Attribute >> Services
   2. Value >> Status

**Summary Page - Measures**

Total Customers = Count (prod\_Churn[Customer\_ID])

New Joiners = CALCULATE(COUNT(prod\_Churn[Customer\_ID]), prod\_Churn[Customer\_Status] = "Joined")

Total Churn = SUM(prod\_Churn[Churn Status])

Churn Rate = [Total Churn] / [Total Customers]

**Churn Prediction Page - Measures**

Count Predicted Churner = COUNT(Predictions[Customer\_ID]) + 0

Title Predicted Churners = "COUNT OF PREDICTED CHURNERS : " & COUNT(Predictions[Customer\_ID])

**# Importing the Library**

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.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

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

from sklearn.preprocessing import LabelEncoder , StandardScaler

import joblib

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import cross\_val\_score

--------------------------------------------------------------------------------------------------------------------------------------

**# Reading the Data**

data = pd.read\_excel(r"C:\Users\DELL\Downloads\CHURN\_data.xlsx")

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

**#Standardize the Data**

data\_to\_scale = ['Tenure\_in\_Months','Age','Monthly\_Charge','Total\_Charges',

'Number\_of\_Referrals','Total\_Refunds','Total\_Extra\_Data\_Charges',

'Total\_Long\_Distance\_Charges','Total\_Revenue']

scaler\_ = StandardScaler()

data[data\_to\_scale] = scaler\_.fit\_transform(data[data\_to\_scale])

-------------------------------------------------------------------------------------------------------------------------------------

**#Encoding the Categorical Data**

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'

]

label\_encoders = {}

for column in columns\_to\_encode:

label\_encoders[column] = LabelEncoder()

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

**#Seperately Encoding the Target Variable**

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

-------------------------------------------------------------------------------------------------------------------------------------

**#Train\_Test\_Split**

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)

**#Random Forest Classifier**

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

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

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

print("Confusion Matrix:")

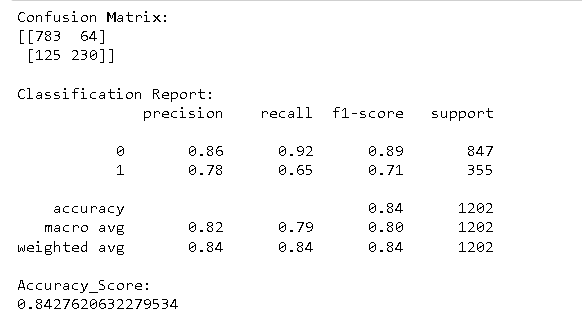
print(confusion\_matrix(y\_test, y\_pred))

print("\nClassification Report:")

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

print("Accuracy\_Score:")

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



**#Feature Importance**

importances = rf\_model.feature\_importances\_

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

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

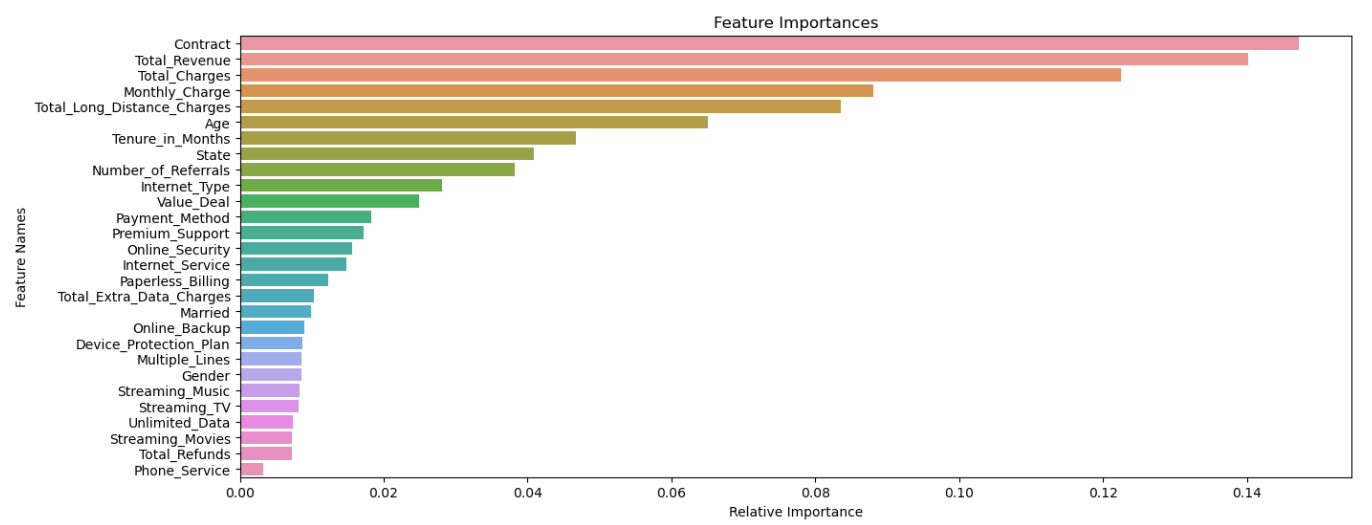
sns.barplot(x=importances[indices], y=X.columns[indices])

plt.title('Feature Importances')

plt.xlabel('Relative Importance')

plt.ylabel('Feature Names')

plt.show()



# Perform 5-fold cross-validation

cv\_scores = cross\_val\_score(rf\_model, X\_train, y\_train, cv=5, scoring='accuracy')

# Print the cross-validation accuracy scores for each fold

print(f'Cross-Validation Scores: {cv\_scores}')

**Cross-Validation Scores: [0.85327784 0.85119667 0.86056191 0.85952133 0.83662851]**

**--------------------------------------------------------------------------------------------------------------------------------------**

**# Print the average cross-validation accuracy**

print(f'Average Cross-Validation Accuracy: {cv\_scores.mean():.4f}')

**Average Cross-Validation Accuracy: 0.8522**

**--------------------------------------------------------------------------------------------------------------------------------------**

**# Define the parameter grid**

param\_grid = {

'n\_estimators': [100, 200, 300], # Number of trees in the forest

'max\_depth': [None, 10, 20, 30], # Maximum depth of the trees

'min\_samples\_split': [2, 5, 10], # Minimum number of samples required to split a node

'min\_samples\_leaf': [1, 2, 4], # Minimum number of samples required at a leaf node

'max\_features': ['auto', 'sqrt', 'log2'] # Number of features to consider when looking for the best split

}

# Initialize the GridSearchCV with your RandomForest model and param\_grid

grid\_search = GridSearchCV(estimator=rf\_model, param\_grid=param\_grid, cv=5, scoring='accuracy', n\_jobs=-1, verbose=2)

# In[87]:

# Fit the grid search model

grid\_search.fit(X\_train, y\_train)

# Get the best parameters

best\_params = grid\_search.best\_params\_

print(f'Best Hyperparameters: {best\_params}')

**Best Hyperparameters: {'max\_depth': 20, 'max\_features': 'auto', 'min\_samples\_leaf': 1, 'min\_samples\_split': 5, 'n\_estimators': 300}**

--------------------------------------------------------------------------------------------------------------------------------

# Evaluate the tuned model on the test data

tuned\_accuracy = grid\_search.score(X\_test, y\_test)

print(f'Tuned Model Accuracy: {tuned\_accuracy:.4f}')

**Tuned Model Accuracy: 0.8411**

==================================================================================

new\_data = pd.read\_csv(r"C:\Users\DELL\Downloads\JOIN\_data.csv")

original\_data = new\_data.copy()

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

data\_to\_scale = ['Tenure\_in\_Months','Age','Monthly\_Charge','Total\_Charges',

'Number\_of\_Referrals','Total\_Refunds','Total\_Extra\_Data\_Charges',

'Total\_Long\_Distance\_Charges','Total\_Revenue']

scaler\_ = StandardScaler()

new\_data[data\_to\_scale] = scaler\_.fit\_transform(new\_data[data\_to\_scale])

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

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

new\_predictions = grid\_search.predict(new\_data)

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

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

original\_data.to\_csv(r"C:\Users\DELL\Downloads\RCF\_Predictions.csv", index=False)