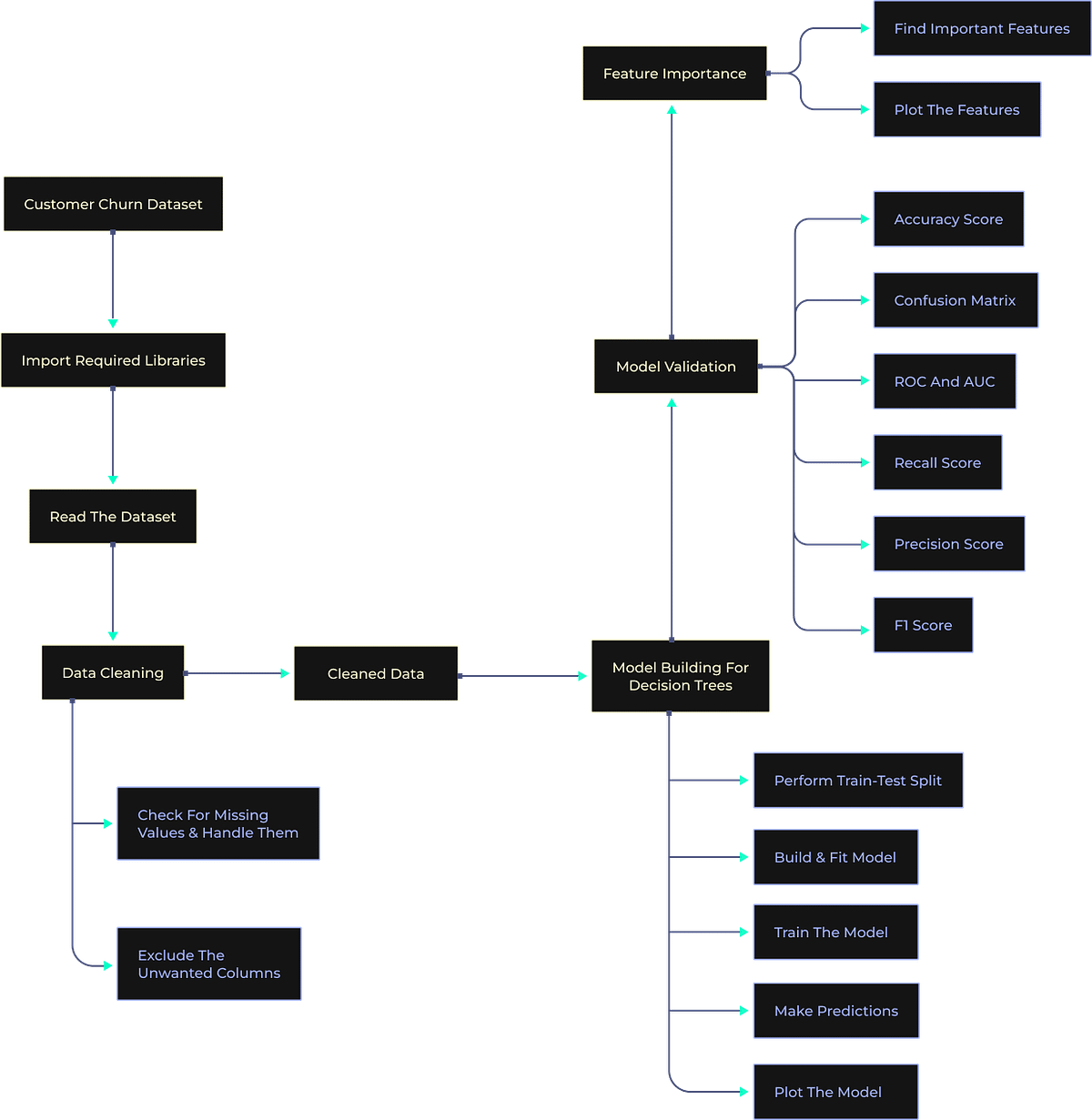
**Build a Customer Churn Prediction Model using Decision Trees**

Develop a customer churn prediction model using decision tree machine learning algorithms and data science on streaming service data.

**ARCHITECTURE DIAGRAM**



# DATASET AT KAGGLE Bank Customer Churn Prediction

<https://www.kaggle.com/datasets/shantanudhakadd/bank-customer-churn-prediction?resource=download>

SOURCE CODE :

import pandas as pd

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

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.preprocessing import LabelEncoder

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

import matplotlib.pyplot as plt

# Load your dataset

df = pd.read\_csv('Churn\_Modelling.csv')

# Identify categorical columns (assuming they are of type 'object')

categorical\_cols = df.select\_dtypes(include=['object']).columns

# Apply Label Encoding to each categorical column

for col in categorical\_cols:

le = LabelEncoder()

df[col] = le.fit\_transform(df[col])

# Assuming 'churn' is your target variable

X = df.drop('churn', axis=1)

y = df['churn']

# Split the dataset 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 and train the Decision Tree model

model = DecisionTreeClassifier(random\_state=42)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

# Evaluate the model

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

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

classification\_rep = classification\_report(y\_test, y\_pred)

# Print the results

print(f"Accuracy: {accuracy:.2f}")

print("Confusion Matrix:\n", conf\_matrix)

print("Classification Report:\n", classification\_rep)

# Visualize the Decision Tree

plt.figure(figsize=(12,8))

plot\_tree(model, filled=True, feature\_names=X.columns, class\_names=['Not Churned', 'Churned'], rounded=True)

plt.show()

#pre pruned decision tree

model1 = DecisionTreeClassifier(max\_depth=4, min\_samples\_split=50, random\_state=42)

# Train the model

model1.fit(X\_train, y\_train)

model1.fit(X\_train, y\_train)

y\_pred1 = model1.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred1)

conf\_matrix = confusion\_matrix(y\_test, y\_pred1)

classification\_rep = classification\_report(y\_test, y\_pred1)

# Print the results

Print(“pre pruned decision tree”)

print(f"Accuracy of pre pruned decision tree: {accuracy:.2f}")

print("Confusion Matrix of pre pruned decision tree:\n", conf\_matrix)

print("Classification Report of pre pruned decision tree:\n", classification\_rep)

# Visualize the pre-pruned Decision Tree

plt.figure(figsize=(12,8))

plot\_tree(model1, filled=True, feature\_names=X.columns, class\_names=['Not Churned', 'Churned'], rounded=True)

plt.show()

#post pruned tree

model2 = DecisionTreeClassifier(max\_depth=4, min\_samples\_split=50, random\_state=42)

# Train the model

model2.fit(X\_train, y\_train)

model2.fit(X\_train, y\_train)

y\_pred2 = model2.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred2)

conf\_matrix = confusion\_matrix(y\_test, y\_pred2)

classification\_rep = classification\_report(y\_test, y\_pred2)

# Print the results

Print(“post pruned trees”)

print(f"Accuracy of post pruned tree: {accuracy:.2f}")

print("Confusion Matrix of post pruned tree:\n", conf\_matrix)

print("Classification Report of post pruned tree:\n", classification\_rep)

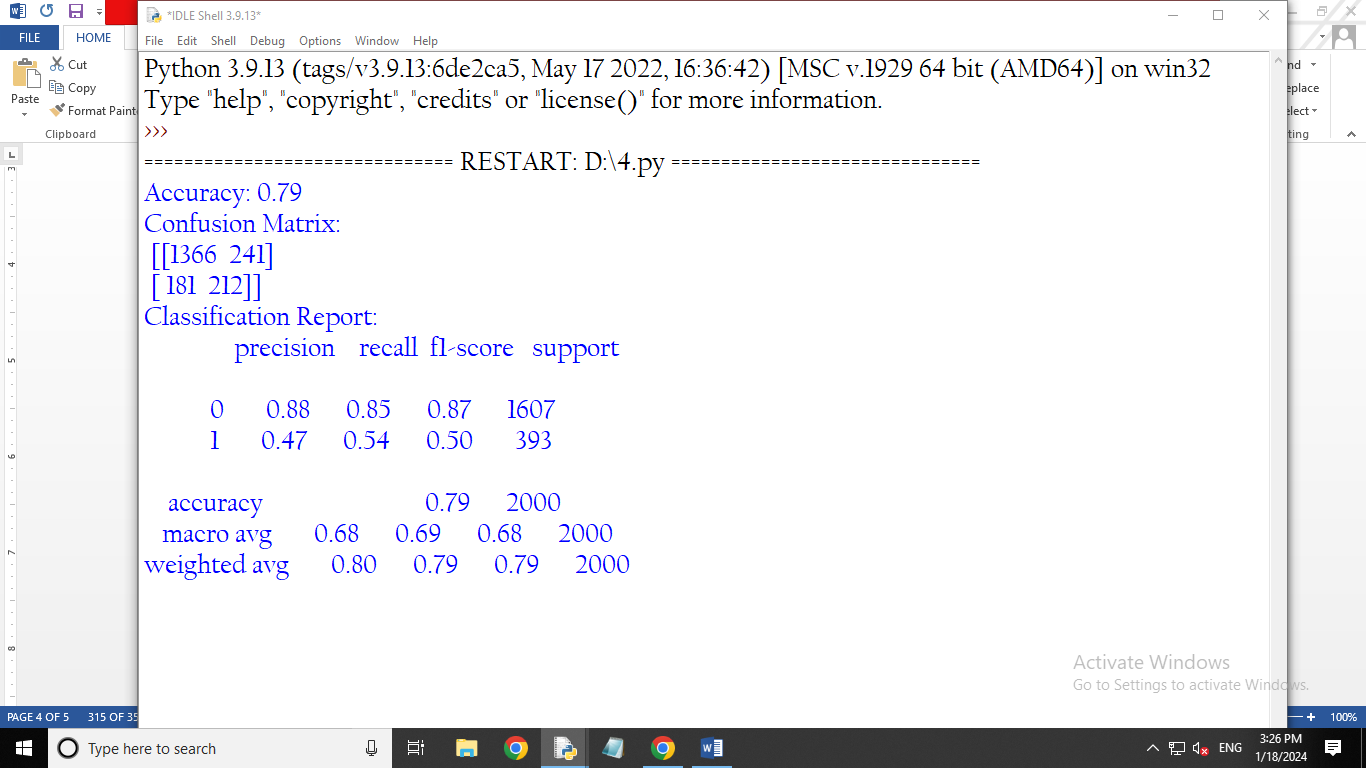
# Visualize the pre-pruned Decision Tree

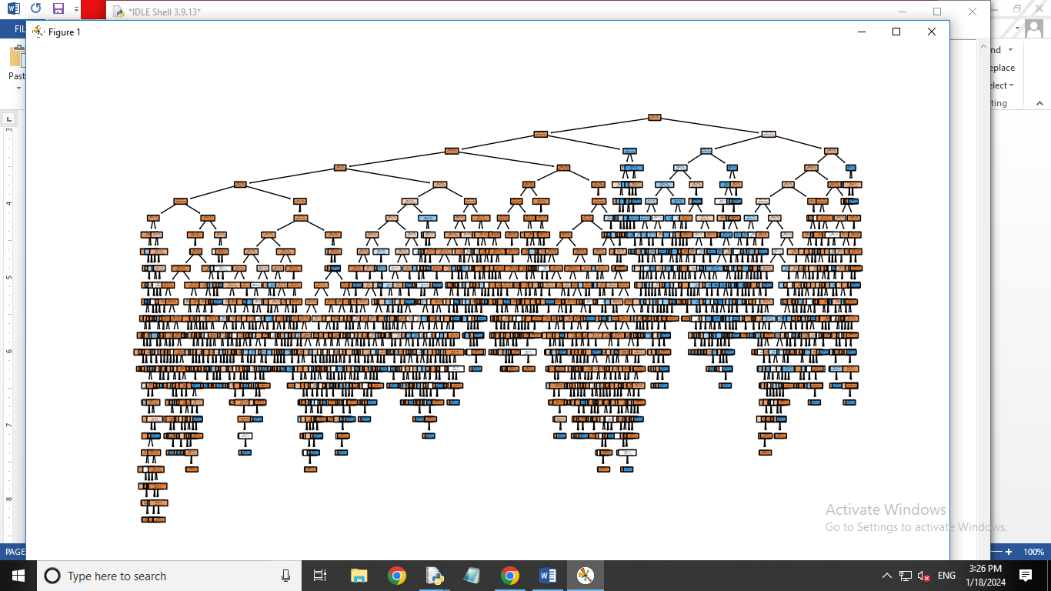
plt.figure(figsize=(12, 8))

plot\_tree(model2, filled=True, feature\_names=X.columns, class\_names=['Not Churned', 'Churned'], rounded=True)

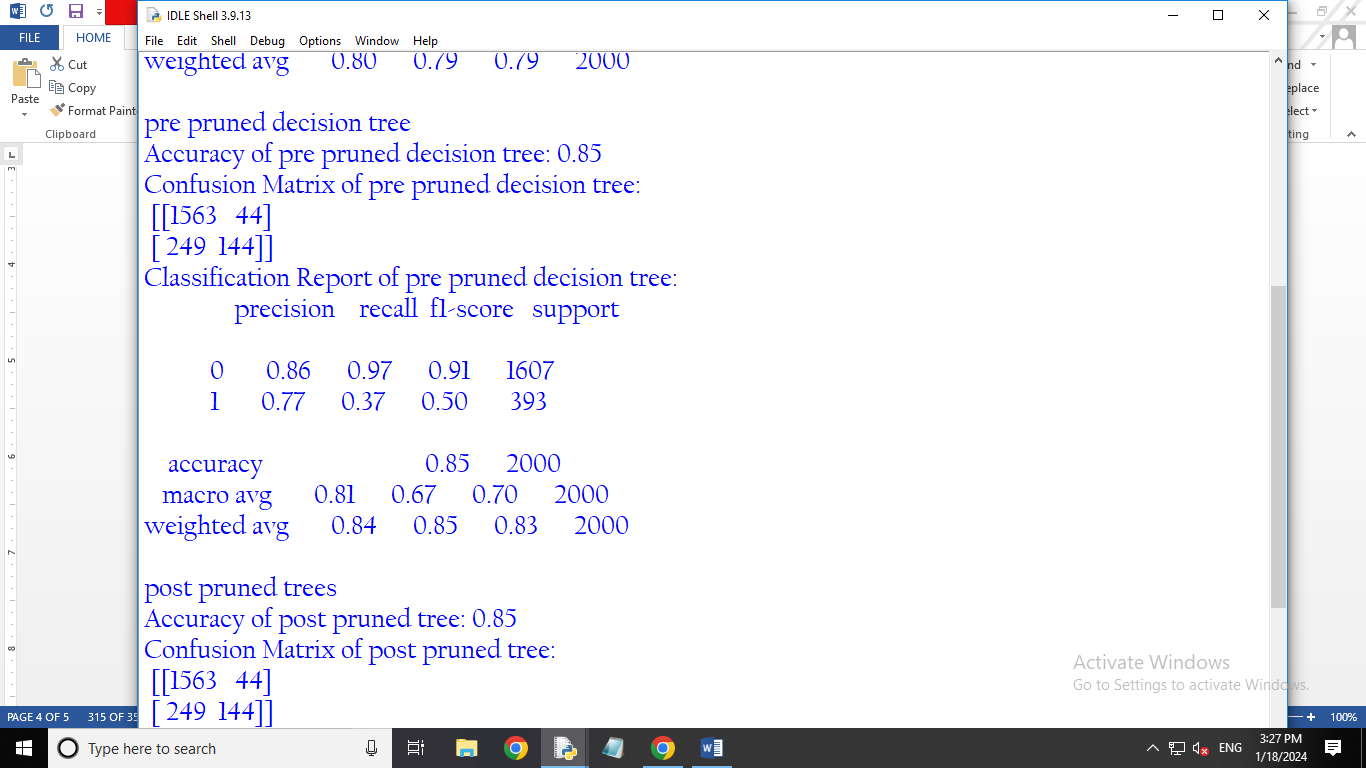
plt.show()

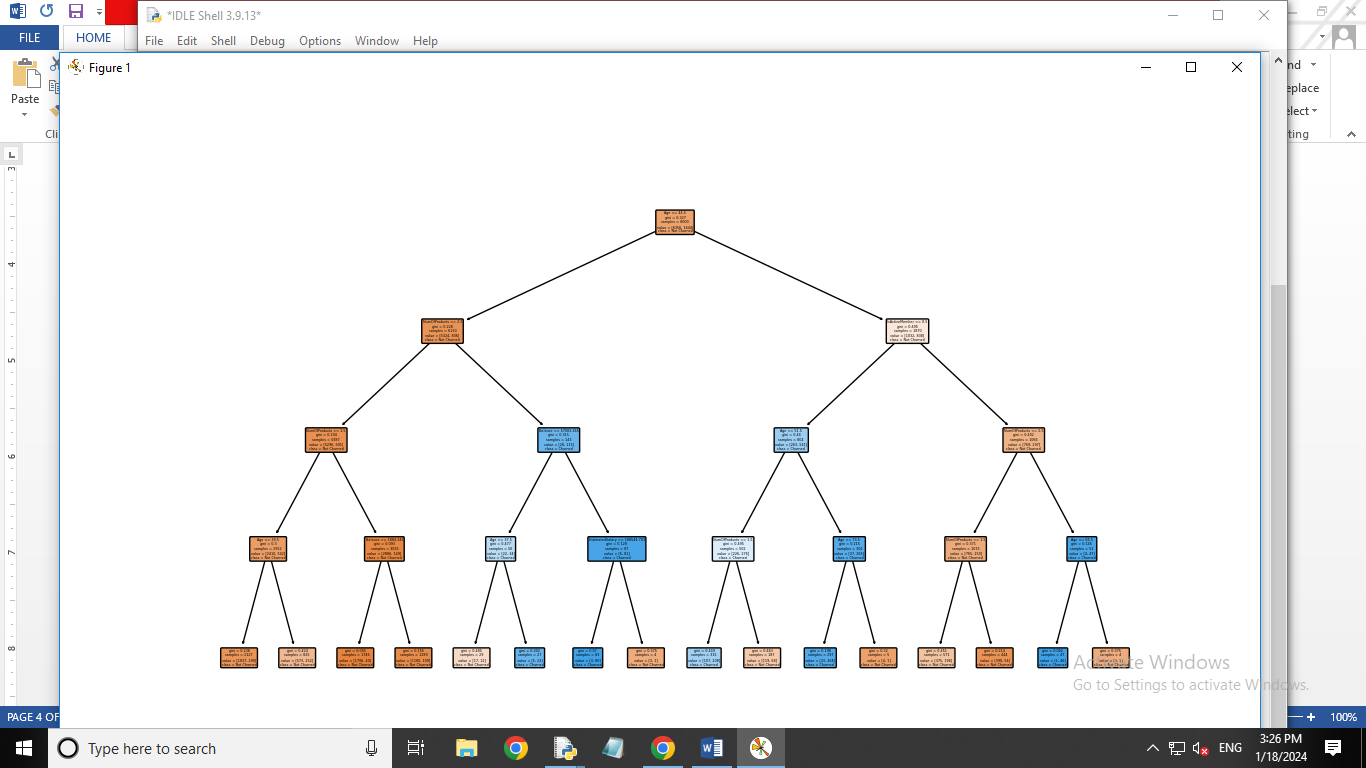
OUTPUT:





PRE PRUNED TREES:





POST PRUNED TREES:

