Final project

**SQL**

USE final\_project;

select \* from customer\_churn;

select

gender,

married,

Dependents,

Referrals,

Phone\_Service,

Multiple\_Lines,

Internet\_Service,

Internet\_Type,

Online\_Security,

Online\_Backup

Device\_Protection,

Tech\_Support,

Streaming\_TV,

Streaming\_Movies,

Streaming\_Music,

Unlimited\_Data,

Contract,

Paperless\_Billing,

Customer\_Status,

count(\*) as churn\_count

from telecom\_customer\_churn

WHERE customer\_status='churned'

Group by

gender,

married,

Dependents,

Referrals,

Phone\_Service,

Multiple\_Lines,

Internet\_Service,

Internet\_Type,

Online\_Security,

Online\_Backup,

Device\_Protection,

Tech\_Support,

Streaming\_TV,

Streaming\_Movies,

Streaming\_Music,

Unlimited\_Data,

Contract,

Paperless\_Billing;

-- 2. Demographic and behavioral characteristics of churned customers:

-- Comparison churn and avtive customers

SELECT gender, married, dependents,

avg(age) as avg\_age,

avg(monthly\_charge) as avg\_monthly\_charges,

avg(total\_charges) as avg\_total\_charge

FROM telecom\_customer\_churn

GROUP by gender,married, dependents;

-- 3. services or contract terms

select

Phone\_Service,

Multiple\_Lines,

Internet\_Service,

Internet\_Type,

Online\_Security,

Online\_Backup,

Device\_Protection,

Tech\_Support,

Streaming\_TV,

Streaming\_Movies,

Streaming\_Music,

Unlimited\_Data,

Contract,

Paperless\_Billing,

count(\*) as churn\_count

from telecom\_customer\_churn

WHERE customer\_status='churned'

Group by

Phone\_Service,

Multiple\_Lines,

Internet\_Service,

Internet\_Type,

Online\_Security,

Online\_Backup,

Device\_Protection,

Tech\_Support,

Streaming\_TV,

Streaming\_Movies,

Streaming\_Music,

Unlimited\_Data,

Contract,

Paperless\_Billing;

**Python**

[telecom\_customer\_churn@12 | Kaggle](https://www.kaggle.com/code/somasun/telecom-customer-churn-12)

*# Import the library*

import pandas as pd

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

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

data=pd.read\_csv('/kaggle/input/telecom-customer-churn12/telecom\_customer\_churn.csv')

data.shape

data.describe()

Out[16]:

data.info()

*# Split the data into features(X) and target variable(y)*

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

y=data['Customer\_Status']

X.head()

In [20]:

y.head()

Out[21]:

*# Encode categorical data variables*

*# Text/String convertion into numeric format*

label\_encoder=LabelEncoder()

X\_encoded=X.apply(label\_encoder.fit\_transform)

X\_encoded.head()

*#split the data into training and testing sets*

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

In [25]:

*# initialize the model*

model=DecisionTreeClassifier()

X\_train.shape

X\_test.shape

Out[26]:

*# Train the model*

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

*# make Predictions on the test set*

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

In [30]:

*# Evaluate the model*

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

print("Accuracy:",accuracy)