**Using Pandas & Matplotlib**

**(for Data Manipulation, Data Visualization, and Machine Learning)**

import pandas a pd

import numpy as np

from matplotlib import pylot as plt

customer\_churn = pd.read\_csv(‘customer\_churn.csv’)

customer\_churn.head()

Table

Description automatically generated with low confidence

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

a. c\_5 = customer\_churn.iloc[:,4]

Result: 0 No, 1 No, 2 No, 3 No, ……

b. c\_15 = customer\_churn.iloc[:,14]

Result: 0 No, 1 No, 2 No, 3 No, …… Name: StreamingMovies, type: object

c. senior\_male\_electronic = customer\_churn[(customer\_churn[‘gender’]== “Male”) & (customer\_churn[‘SeniorCitizen’]==1) & (customer\_churn[‘PaymentMethod’] == “Electronic check”)]

senior\_male\_electronic.head() -> result

d. customer\_total\_tenure = customer\_churn[(customer\_churn[’tenure’]>70) | (customer\_churn[‘MonthlyCharges’]>100)]

e. two\_mail\_yes = customer\_churn[(customer\_churn[’Contract’]==“Two year”) & (customer\_churn[‘PaymentMethod’]==“Mailed check”) & (customer\_churn[‘Churn’]==“Yes”]

two\_mail\_yes -> 3 rows satisfied

f. **Random sampling**

customer\_333 = customer\_churn.sample(n=333)

g. customer\_churn[‘Churn’].value\_counts()

A screenshot of a computer

Description automatically generated with low confidence

Text

Description automatically generated

plt.bar(customer\_churn[‘InternetService’].value\_counts().keys().tolist(),customer\_churn[‘InternetService’].value\_counts().tolist(),color=“orange")

plt.xlabel(“Categories of Internet Service”)

ptl.ylabel(“Count”)

plt.title(“Distribution of Internet Service”)

Chart, bar chart

Description automatically generated

\*1st parameter in the (): names of the bars, 2nd parameter: values of those bars

Text

Description automatically generated

Plt.hist(customer\_churn[‘tenure’=, bins=30,color=“green”)

Plt.title(“Distribution of tenure”)

Text, letter

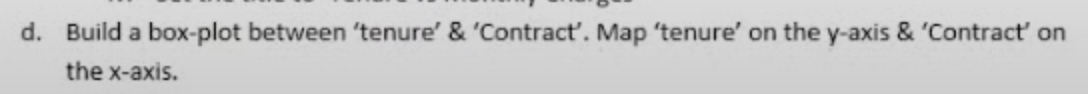
Description automatically generated

plt.scatter(x=customer churn[‘tenuree’], y=customer\_churn[‘MonthlyCharges’])

plt.xlabel(“Tenure”)

plt.ylabel(“Monthly Charges”)

plt.title(“Tenue vs Monthly Charges”)



**Machine Learning**

Text

Description automatically generated

from sklearn import linear\_model

from sklearn.linear\_model import LinearRegression

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

y=customer\_churn[[‘MonthlyCharges’]]

x=customer\_churn[[‘tenure’]]

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size=0.30, random\_state = 0)

x\_train.shape, y\_train.shape, x\_test.shape, y\_test.shape

**regressor = LinearRegression()**

**regressor.fit(x\_train, y\_train)**

y\_pred=regressor.predict(x\_test)

from sklearn.metrics import mean\_squared\_error #lower mqe, better accuracy

np.sqrt(mean\_squared\_error(y\_test,y\_pred))

Graphical user interface, text, application

Description automatically generated

x=customer.churn[[‘MonthlyCharges’]]

y=customer\_churn[[‘Churn’]]

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size=0.35, random\_state = 0)

From sklearn.linear\_model import LogisticRegression

log\_model = LogisticRegression()

log.model.fit(x\_train, y\_train)

y\_pred=log\_model.predict(x\_test)

**#Classification Model - Confusion matrix**

From sklearn.metrics import confusion\_matrix, accuracy\_score

confusion\_matrix(y\_test,y\_pred),accuracy\_score(y\_test,y\_pred)

Graphical user interface, text, application

Description automatically generated~73.6% accuracy

Text

Description automatically generated

x=customer.churn[[‘MonthlyCharges’,’tenure’]]

y=customer\_churn[[‘Churn’]]

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size=0.20, random\_state = 0)

From sklearn.linear\_model import LogisticRegression

log\_model = LogisticRegression()

log.model.fit(x\_train, y\_train)

y\_pred=log\_model.predict(x\_test)

**#Classification Model - Confusion matrix**

From sklearn.metrics import confusion\_matrix, accuracy\_score

confusion\_matrix(y\_test,y\_pred),accuracy\_score(y\_test,y\_pred)

Graphical user interface, text, application

Description automatically generated

Text, letter

Description automatically generated

x=customer\_churn[[‘tenure’]]

y=customer\_churn[[‘Churn’]]

from sklearn.tree import DecisionTreeClassifier

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size=0.20, random\_state = 0)

my\_tree = DecisionTreeClassifier()

my\_tree.fit(x\_train,y\_train)

y\_pred=my\_tree.predict(x\_test)

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

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

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

Graphical user interface, text, application

Description automatically generated

from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()

rf.fit(x\_train,y\_train)

rf.predict(x\_test)

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

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