1. **Linear Regression a. Compute y = ex for x = 1 to 50. b. Save the values generated in the CSV File [through Pandas] c. Predict the values of y for x = 51 to 60 using linear regression. d. Find out the Model Parameters (Slope and Intercept). e. Evaluate the error in terms of training set.**

**Source Code:**

import math

l1 = range(1,51,1)

l2 = [math.exp(x) for x in l1]

#print(l2)

matrix = []

for i in range(len(l1)):

matrix.append([l1[i],l2[i]])

import csv

with open ('exponential.csv', 'w', newline = '') as csvfile:

writer = csv.writer(csvfile, delimiter = ',')

writer.writerow(['x', 'exp\_x'])

writer.writerows(matrix)

with open ('exponential.csv', 'r') as fin:

print(list(csv.reader(fin)))

import matplotlib.pyplot as plt

from scipy import stats

slope, intercept, r, p, std\_err = stats.linregress(l1,l2)

def myfunc(z):

return slope\*z + intercept

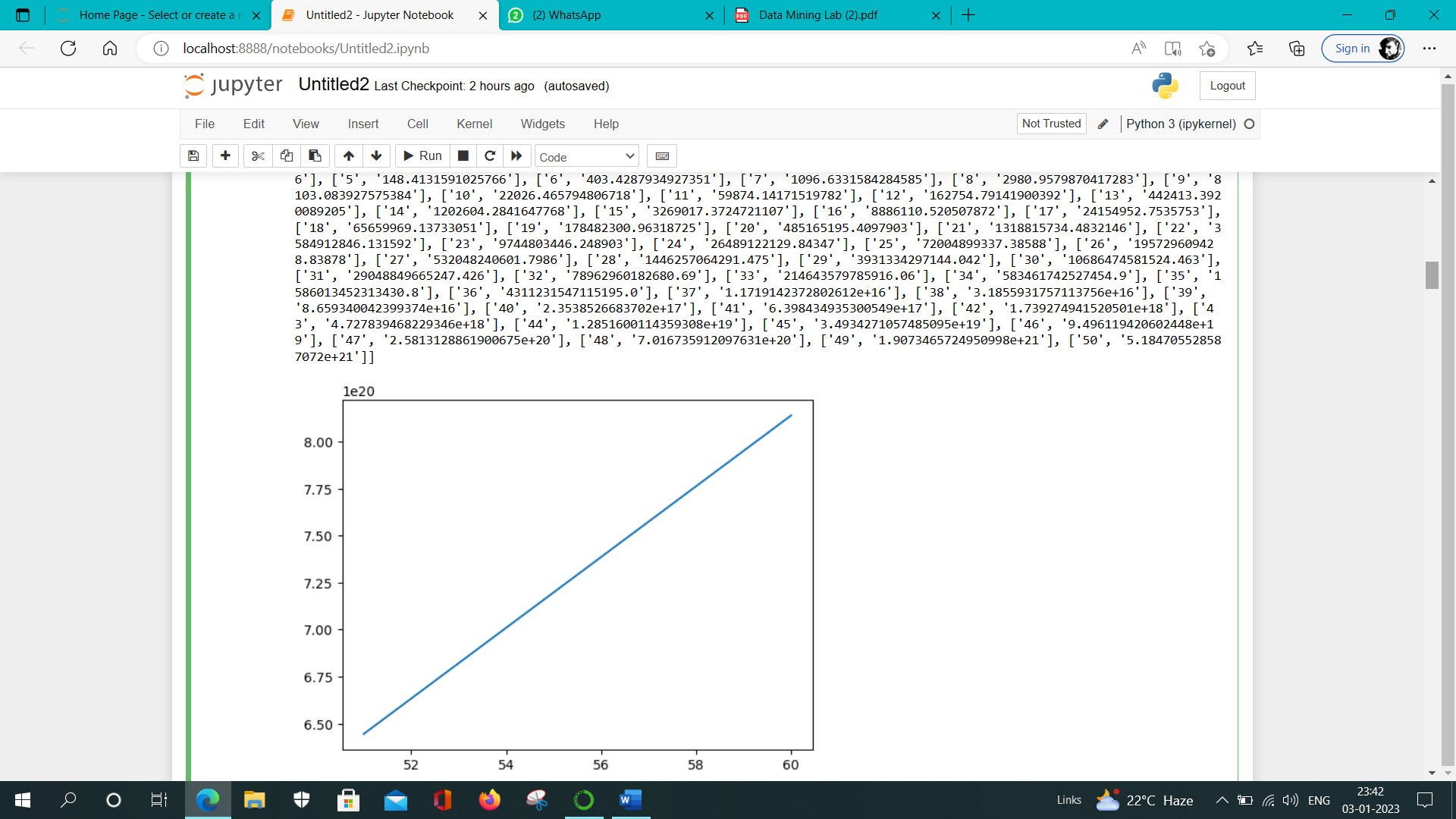
l3 = range(51,61,1)

l4 = [myfunc(x) for x in l3]

plt.plot(l3, l4)

plt.show()

**OUTPUT:**



1. **Dataset Import a. Import Titanic: Machine Learning from Disaster dataset. b. Find the total number of records present in the dataset. c. Find the attributes and types of attributes present in the dataset. d. Identify and print categorical and numerical attributes present in the dataset. e. Find unique values for string variables in the dataset. f. Find the range of values for numerical variables present in the dataset.**

**SOURCE CODE:**

import pandas as pd

df = pd.read\_csv("titanic.csv")

print(df.count(axis = 0))

print("\n")

print(df.index)

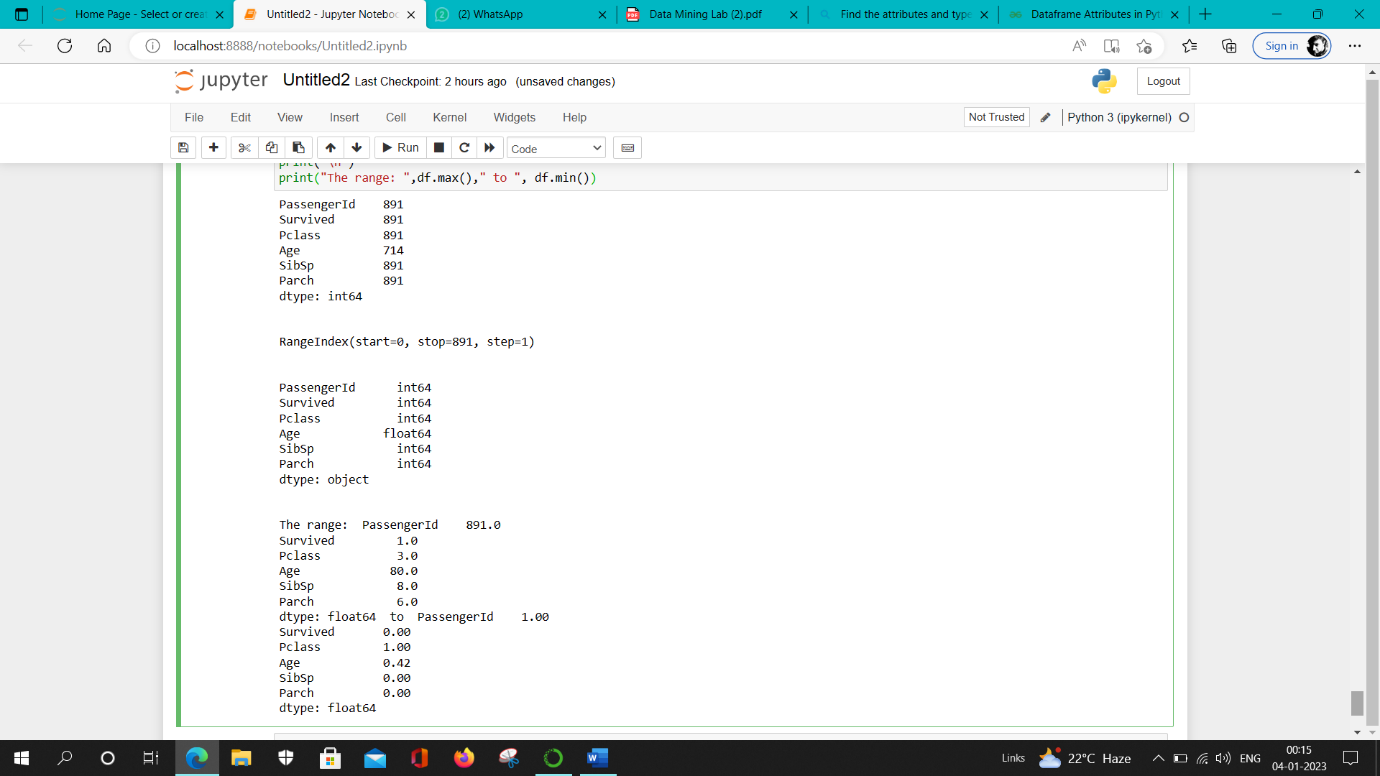
print("\n")

print(df.dtypes)

print("\n")

print("The range: ",df.max()," to ", df.min())

**OUTPUT:**

****

1. **Dataset Pre-processing a. Find the number of missing values. b. Replace these missing values with relevant values (either using mean or 0) c. Scale numeric variables into specified range. d. Replace string variables using numeric values. (Use Label encoder or one hot encoding) e. Remove irrelevant attributes (if any).**

**SOURCE CODE:**

#importing the required libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

#importing the datasets

data = pd.read\_csv('missing\_data.csv')

print(data.isnull().sum().sum())

data = data.fillna(0)

data

scaler = MinMaxScaler()

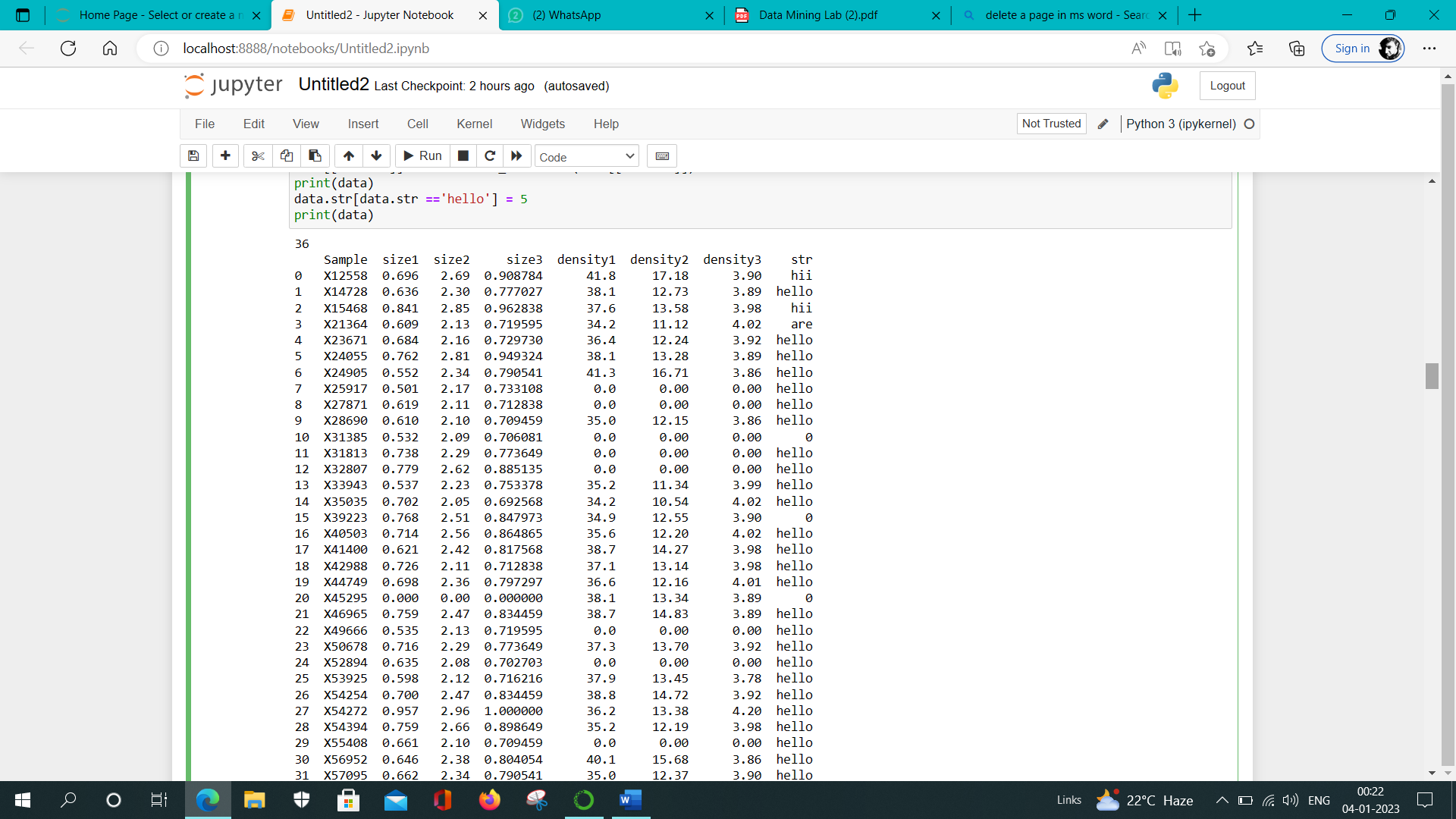
data[["size3"]] = scaler.fit\_transform(data[["size2"]])

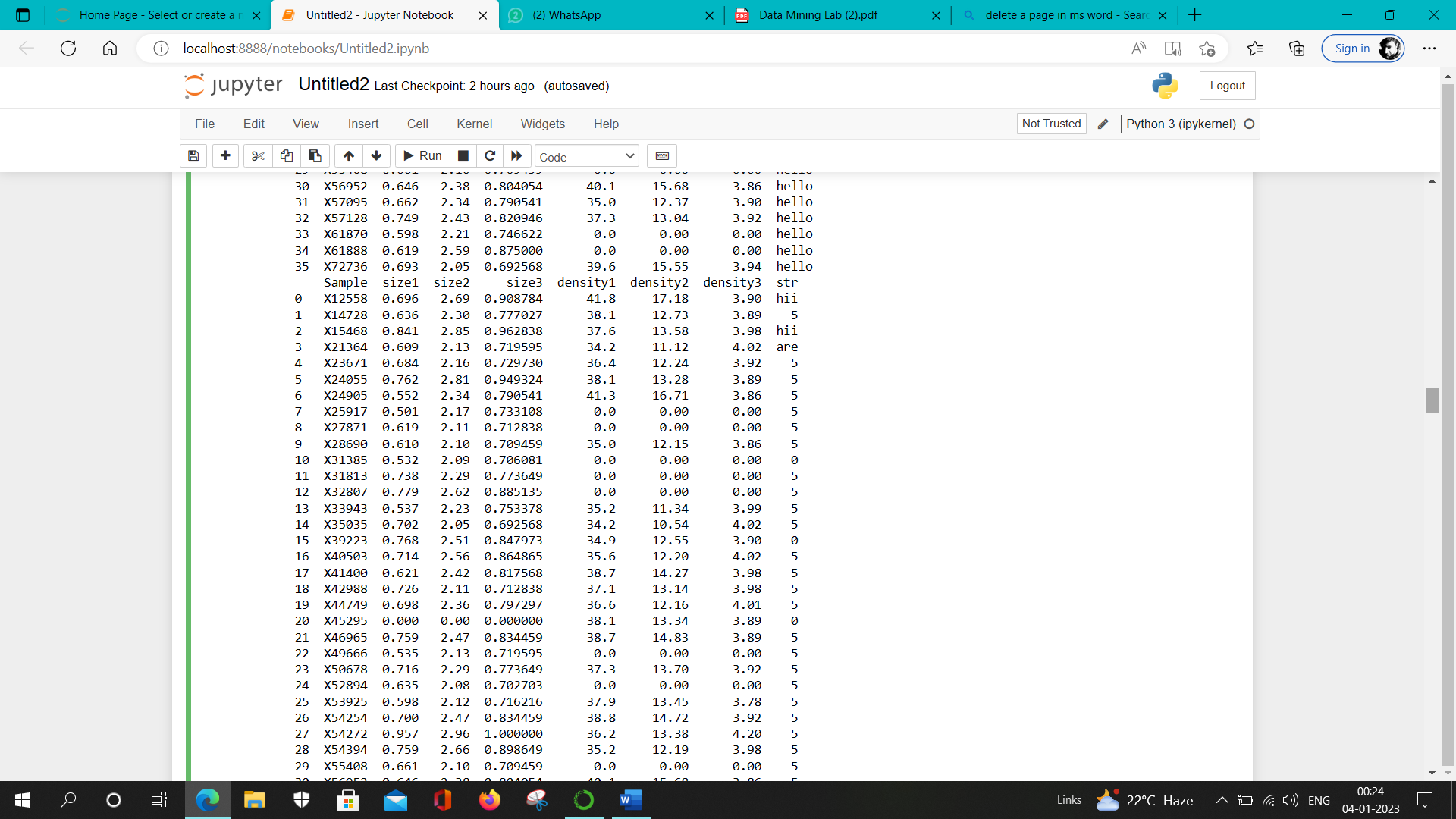
print(data)

data.str[data.str =='hello'] = 5

print(data)

**OUTPUT:**





1. **Understanding of the Dataset a. Use bar chart for plotting number of missing values for each attribute. b. Use pie chart for plotting percentage of survivals. c. Plot the relationship among attribute ‘age’ and ‘survival’.**

**SOURCE CODE:**

import pandas as pd

import missingno as msno

from matplotlib import pyplot as plt

df = pd.read\_csv("train.csv")

#plotting bar graph

msno.bar(df, color = "red")

#plotting pie char

df.head()

cal = list(df['Survived'])

sur = dead = 0

for i in cal:

if(i==0):

dead+=1

else:

sur+=1

fig = plt.figure(figsize=(10,7))

p = []

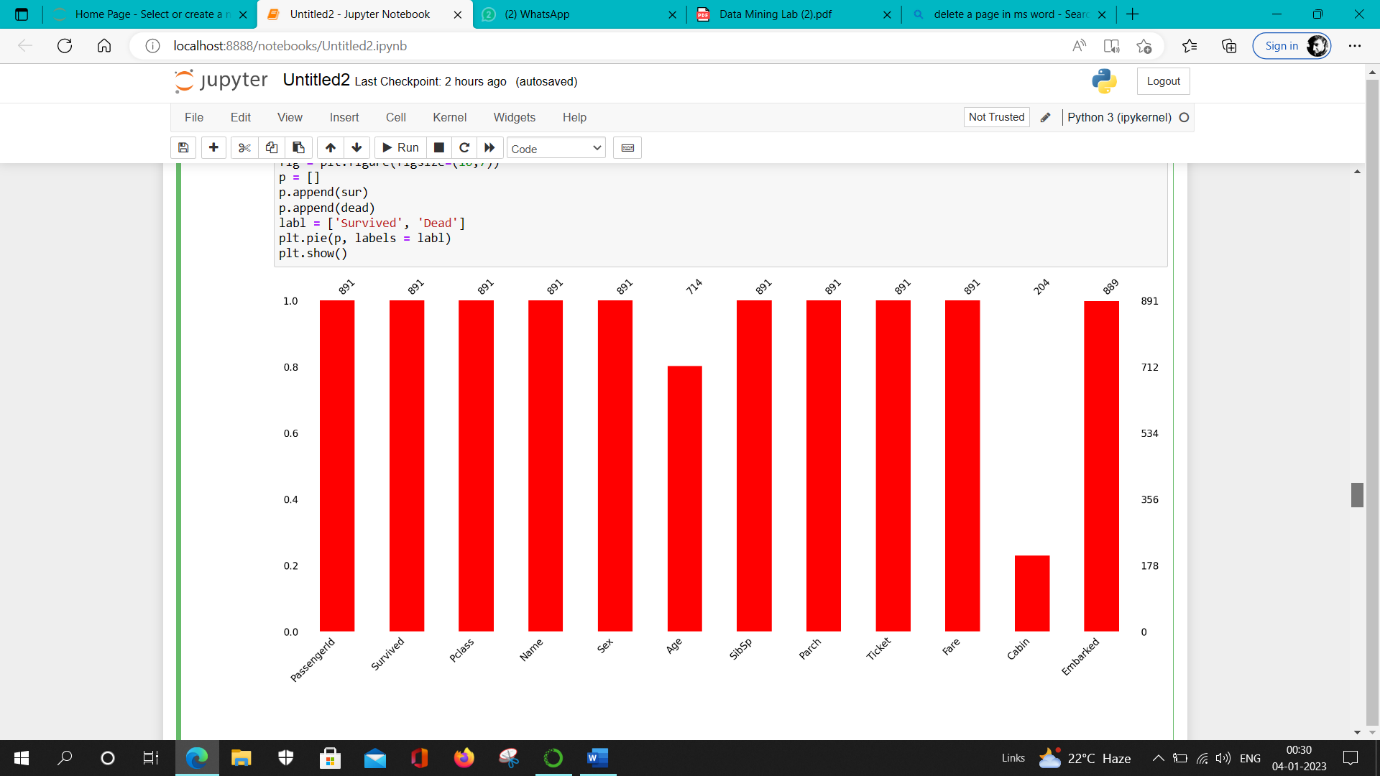
p.append(sur)

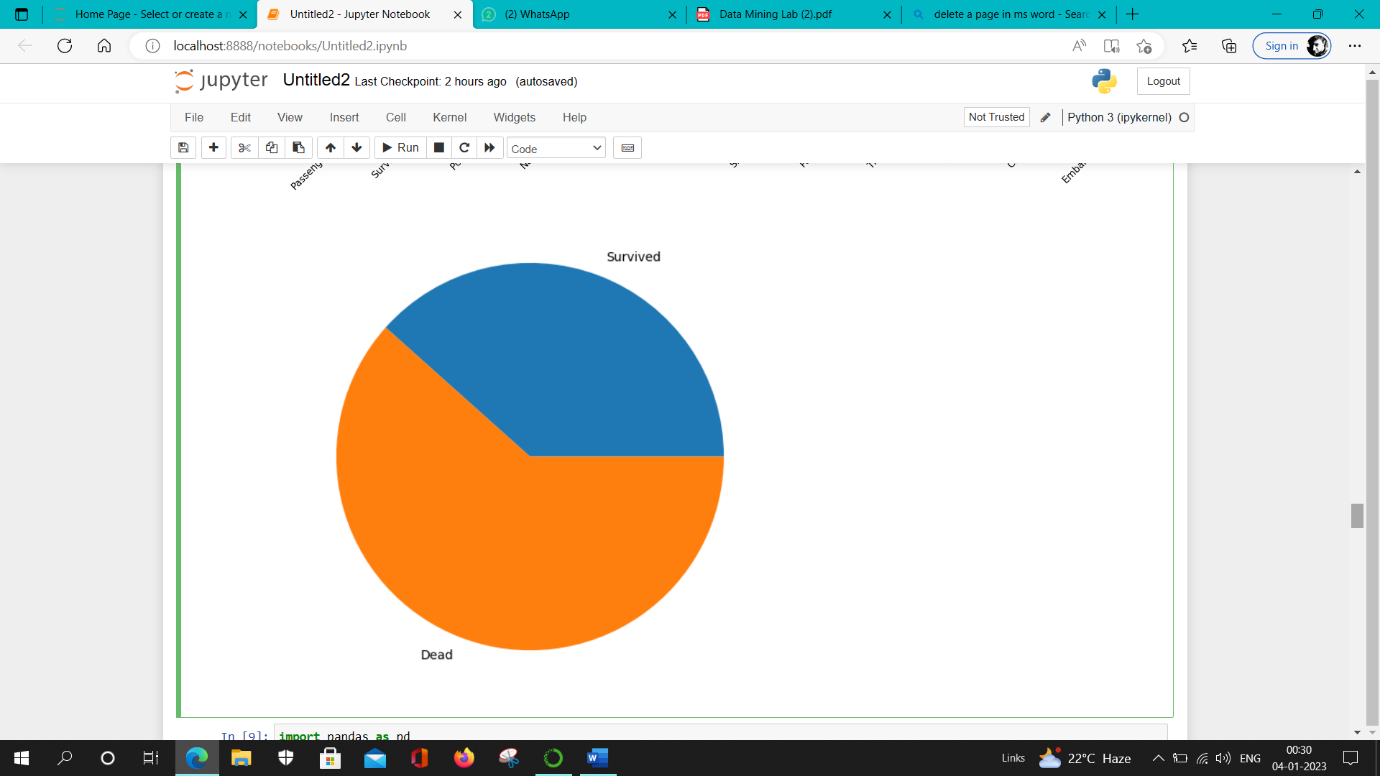
p.append(dead)

labl = ['Survived', 'Dead']

plt.pie(p, labels = labl)

plt.show()

**Output:**



**5. Retrieve Train and Test Dataset. a. Partition the original dataset into train and test dataset with the ratio of 8:2. b. Partition the original dataset into train and test dataset with the ratio of 7:3.**

**Source Code:**

#importing required libraries

import pandas as pd

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

data = pd.read\_csv('missing\_data.csv')

#data.head()

y = data.size1

x = data.drop('size1', axis = 1)

#x.head()

#y.head()

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y, test\_size=0.2) #splitting the dataset

print("shape of original dataset :", data.shape)

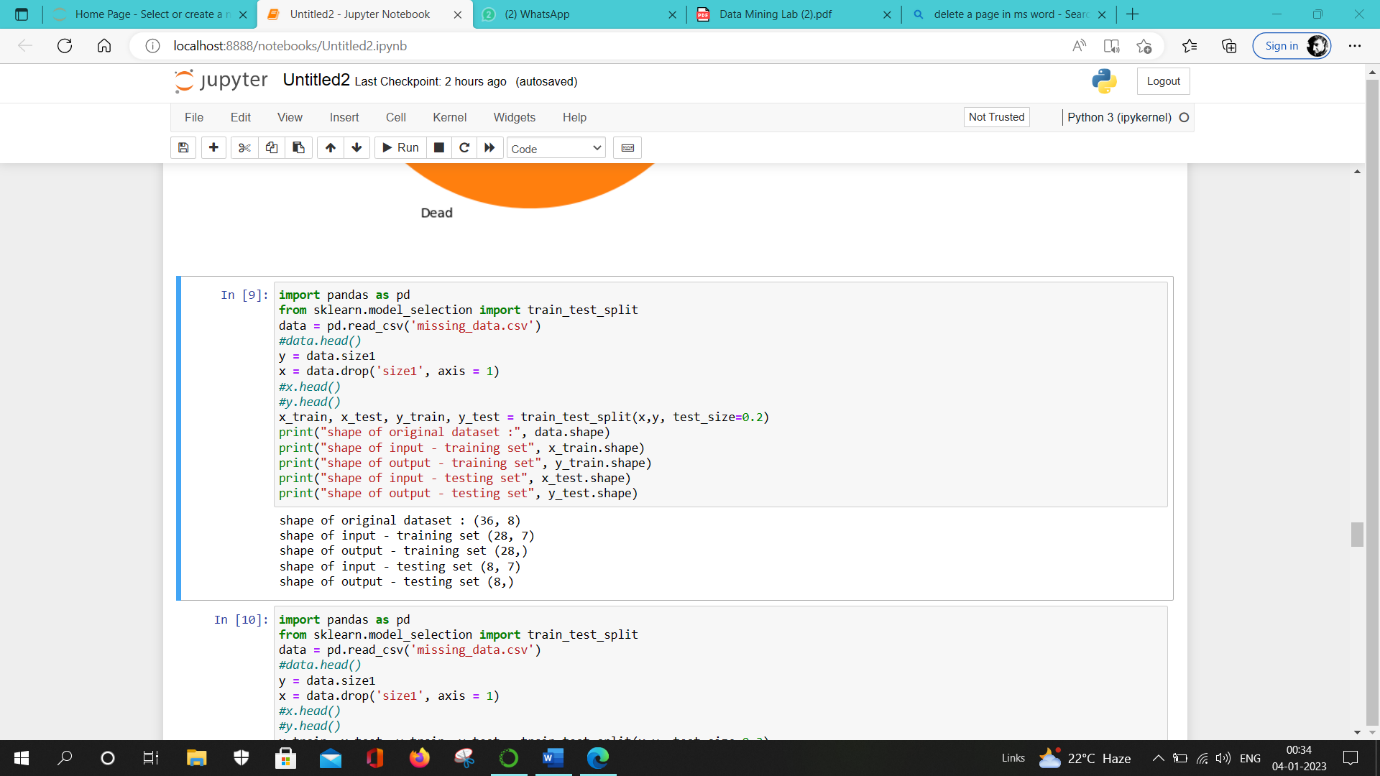
print("shape of input - training set", x\_train.shape)

print("shape of output - training set", y\_train.shape)

print("shape of input - testing set", x\_test.shape)

print("shape of output - testing set", y\_test.shape)

**Output:**



**6. Predict the chance of survival. a. Apply Bayesian classification methods [Use Multinomial and Gaussian Naïve Bayes for classification] b. Apply Minimum Distance classification Use K-NN. Vary the values of K and obtain prediction. c. Apply Decision Tree classifier Try this classifier model using different criterions and splitter.**

**Source Code:**

#Gaussian classifier

import numpy as np

import pandas as pd

from sklearn.naive\_bayes import GaussianNB

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

from sklearn import metrics

#from sklearn.cross\_validation import train\_test\_split

data = pd.read\_csv('titanic.csv')

data = data.fillna(0)

y = data.Survived

x = data.drop('Survived', axis = 1)

gaussian = GaussianNB()

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y, test\_size=0.3)

model = GaussianNB()

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

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

k=s=d=0

for i in y\_pred:

k+=1

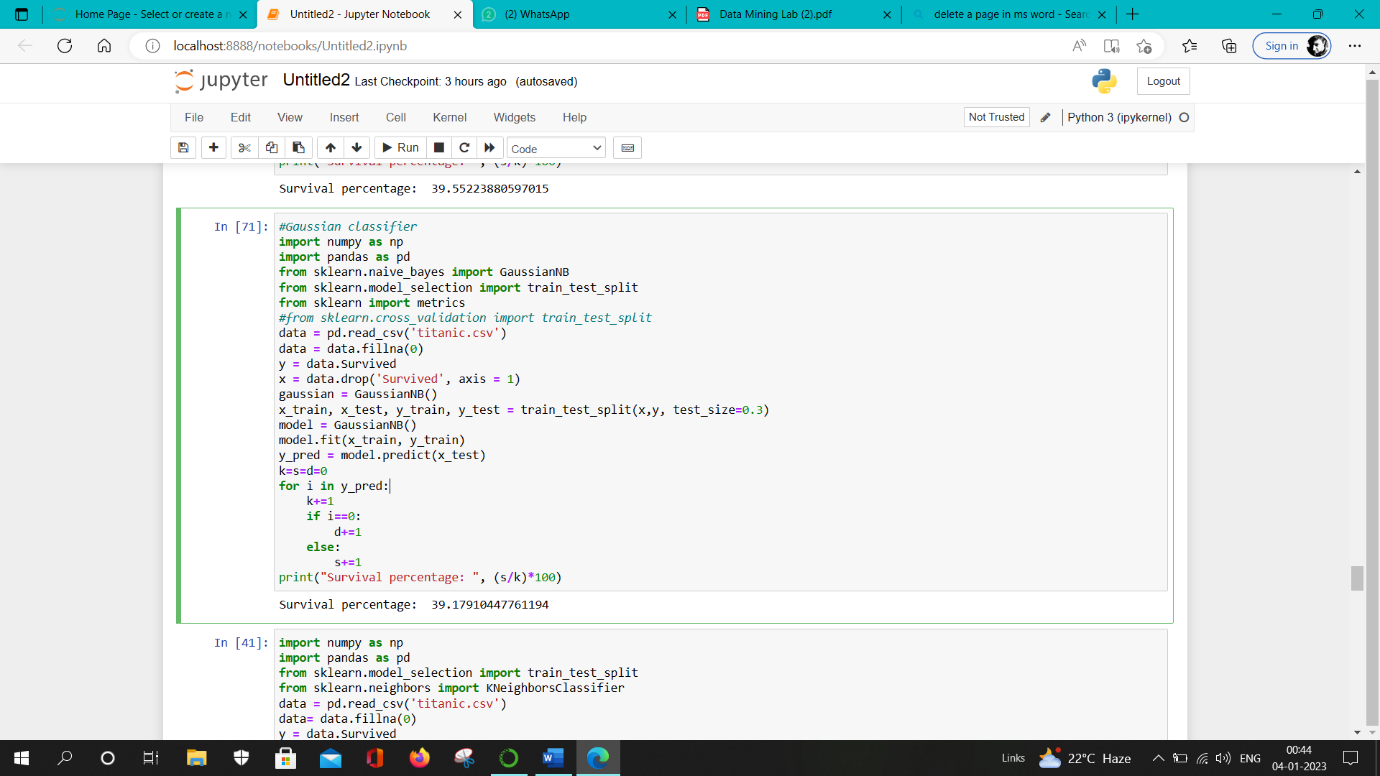
if i==0:

d+=1

else:

s+=1

print("Survival percentage: ", (s/k)\*100)

**Output:**

**Source Code:**

#kNN means of classification

import numpy as np

import pandas as pd

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

from sklearn.neighbors import KNeighborsClassifier

data = pd.read\_csv('titanic.csv')

data= data.fillna(0)

y = data.Survived

x = data.drop('Survived', axis = 1)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y, test\_size=0.3)

neigh = range(1,11)

clf = []

prediction = []

for k in neigh:

clf = KNeighborsClassifier(n\_neighbors = k)

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

prediction = clf.predict(x\_test)

#print(prediction)

j=s=d=0

for i in prediction:

j+=1

if i==0:

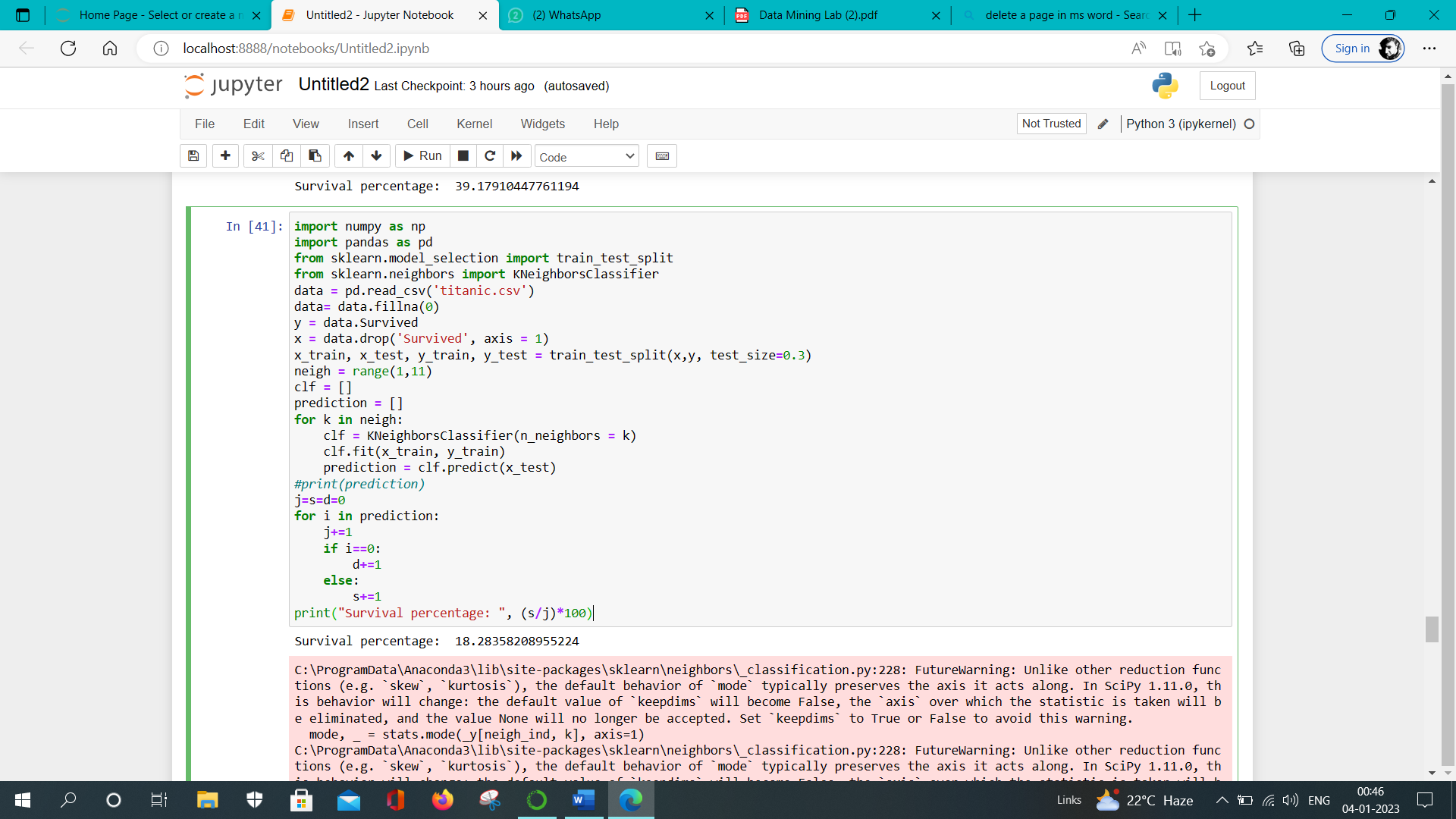
d+=1

else:

s+=1

print("Survival percentage: ", (s/j)\*100)

**Output**:



**Source Code:**

#decission tree

import pandas as pd

from sklearn import tree

from sklearn.tree import DecisionTreeClassifier

df = pd.read\_csv("titanic.csv")

data= data.fillna(0)

y = data.Survived

x = data.drop('Survived', axis = 1)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y, test\_size=0.3)

dtree = DecisionTreeClassifier()

dtree = dtree.fit(x\_train, y\_train)

prediction = dtree.predict(x\_test)

s=d=k=0

for i in y\_pred:

k+=1

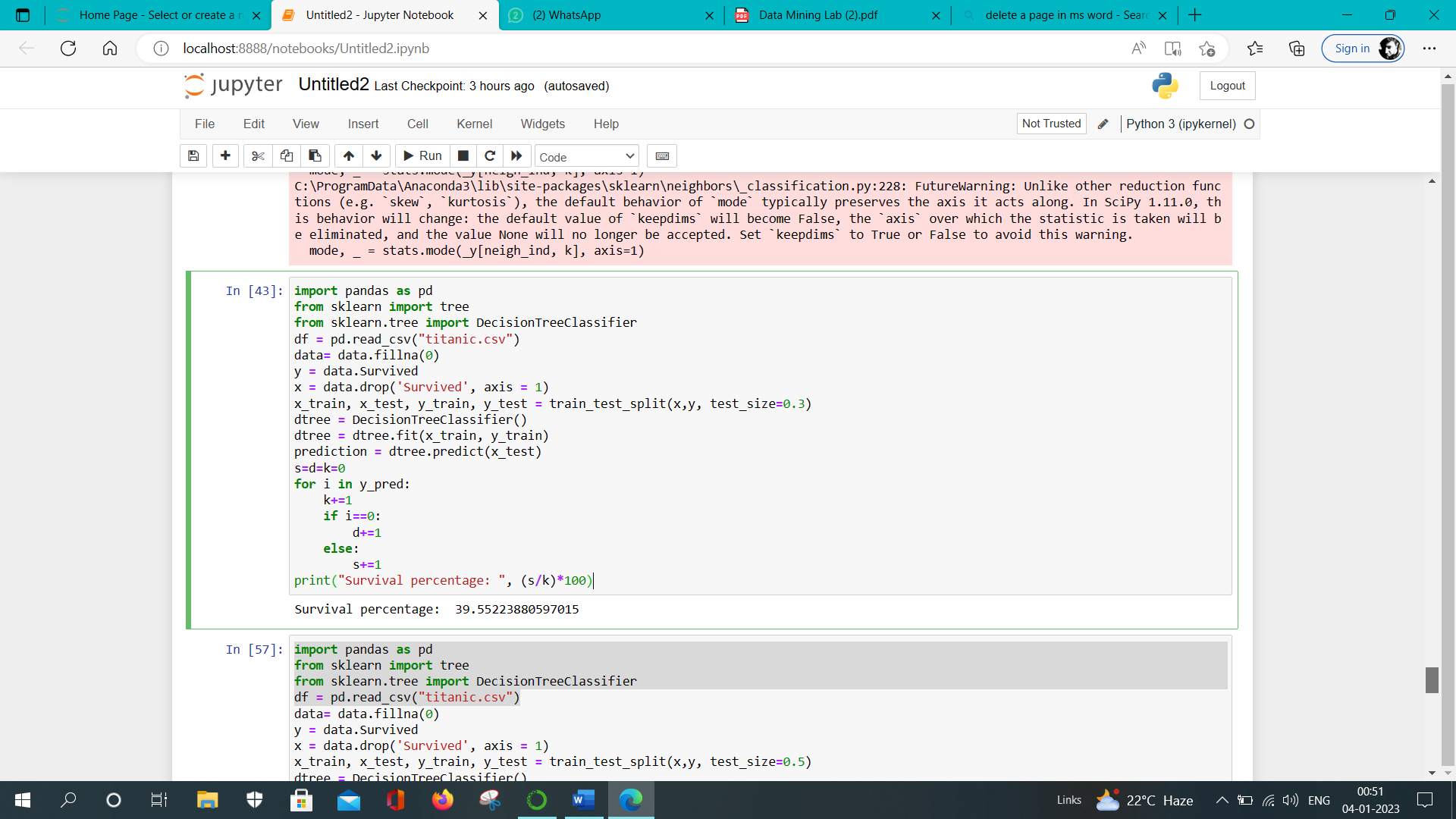
if i==0:

d+=1

else:

s+=1

print("Survival percentage: ", (s/k)\*100)

**Output:**

**7. Justify the performance of classifier models designed in Q7 using: a. Obtain Confusion Matrix b. Find out True positive, True Negative, False Positive, False Negative. c. Calculate Accuracy d. Calculate Precision. e. Calculate Recall. f. Calculate F1-score/ F-measure.**

**Source Code:**

import pandas as pd

from sklearn import tree

from sklearn.tree import DecisionTreeClassifier

df = pd.read\_csv("titanic.csv")

data= data.fillna(0)

y = data.Survived

x = data.drop('Survived', axis = 1)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y, test\_size=0.5)

dtree = DecisionTreeClassifier()

dtree = dtree.fit(x\_train, y\_train)

prediction = dtree.predict(x\_test)

import matplotlib.pyplot as plt

import numpy

from sklearn import metrics

confusion\_matrix = metrics.confusion\_matrix(y\_test, prediction)

cm\_display = metrics.ConfusionMatrixDisplay(confusion\_matrix = confusion\_matrix, display\_labels = [False, True])

cm\_display.plot()

plt.show()

Accuracy = metrics.accuracy\_score(y\_test, prediction)

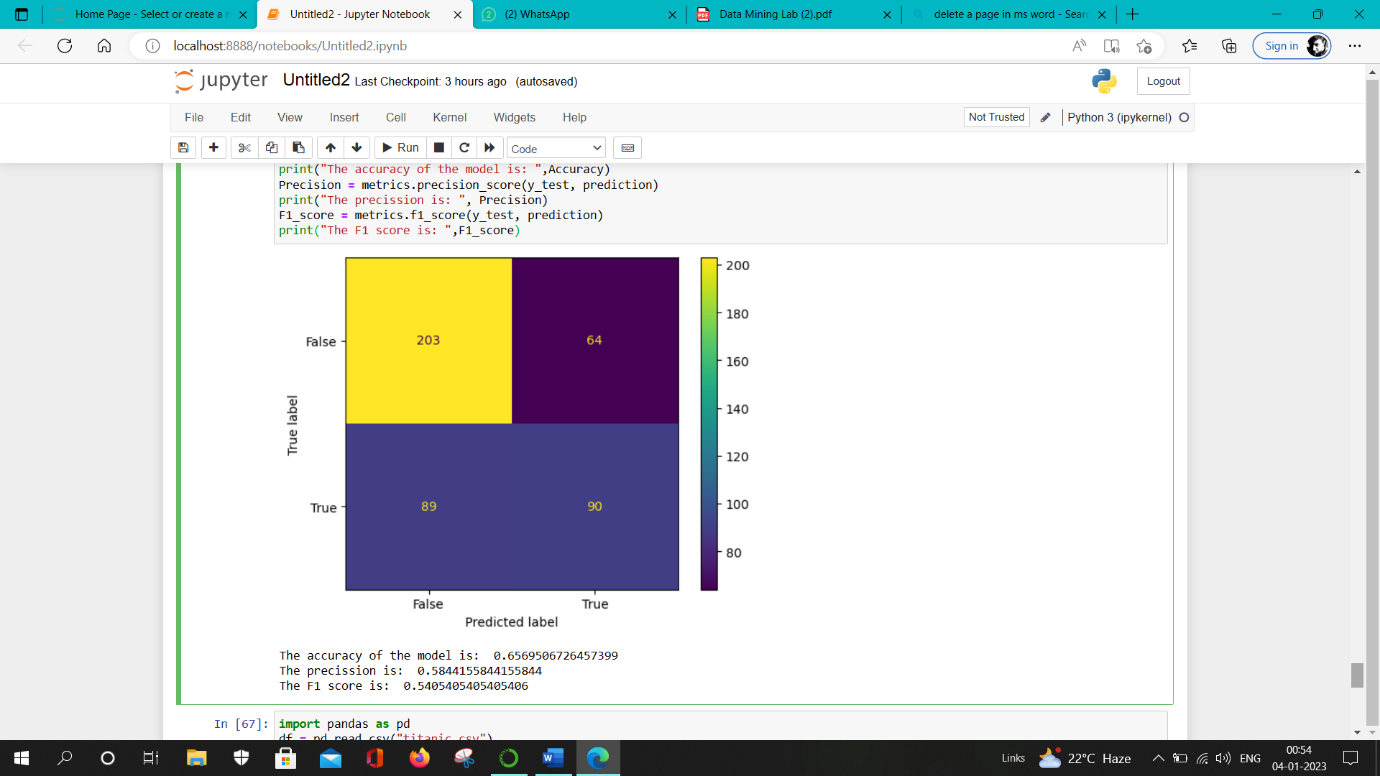
print("The accuracy of the model is: ",Accuracy)

Precision = metrics.precision\_score(y\_test, prediction)

print("The precission is: ", Precision)

F1\_score = metrics.f1\_score(y\_test, prediction)

print("The F1 score is: ",F1\_score)

**Output:**