#%%

print(1/3)

print(1//3)

#%%

a = 5

b = 3.2

print(type(b))

#%%

#user Input

name = input("Enter you name:")

print("Hello", name)

#%%

x=3; y = 5

if x<y:

print("x is less than y")

elif x>y:

print("x is greater than y")

else:

print("x and y are equal")

print("Done")

#%%

if x == y:

print("x and y are equal")

else:

if x <y :

print("x is less than y")

else:

print("x is greater than y")

print("Done")

#%%

choice = int(input("Enter 1/2/3: "))

if choice == 1 or 2 or 3:

if choice == 1:

print("bad guess")

elif choice == 2:

print("good guess")

elif choice == 3:

print("close, but not correct")

else:

print("Enter a correct number")

print("Done")

#%%

x = 5

y = 8

if x < 7 and y >7:

print("x is a positive single-digit number")

#%%

def square(a):

print("value is = ", a)

return a\*a

square(4)

#%%

count= 4

while(count >0):

print(count)

count += -1

else:

print("count value reached", count)

#%%

friends = ["A", "B", 'C', 3.6]

for f in friends:

print('happy diwali', f)

#%%

a= 10+2J

print(type(a))

#%%

print(int(-2.8))

print(float(5))

#%%

import math

print(math.sqrt(4))

print(math.floor(-10.7))

print(math.pow(2,3))

print(math.pi)

print(math.factorial(6))

#%%

import random

print(random.random())

print(random.randrange(50,100))

#%%

my\_string = 'hello world'

print(my\_string)

my\_string = """py is interesting language.

yes that is correct."""

print(my\_string)

my\_string = "py is good.\nWow."

print(my\_string)

#%%%

my\_string = "my name is navdeep"

print(my\_string[0])

print(my\_string[-3])

print(my\_string[5:10])

print(my\_string[:10])

print(my\_string[10:])

print(my\_string[:])

print(my\_string[100])

#%%

str1 = 'hello'

str2 = 'world'

str3 = str1+" "+str2

print(str3)

#%%

print("PYthon".lower())

string = "python, is, language good"

print(string.split(sep = ","))

print('Happy Diwali'.replace("p", "c"))

#%%

?str.count()

#%%

a = (5, 'python', 3.5)

print(type(a))

print(a[0])

#%%

print(tuple("python"))

#%%

tuple1 = (2,4,5)

c = tuple(tuple1)

print(c)

#%%

n\_tuple = ("mouse", [1,2,3], (5,6,7))

print(n\_tuple[0])

print(n\_tuple[1][2])

#%%

import pandas as pd

titanic\_df = pd.read\_excel(r'C:\Users\Administrator\Desktop\Data Science Program\Python\Titanic\Titanic\_Survival\_Train.xls', header = 0, index\_col = 0)

titanic\_df.head()

titanic\_df.tail()

#%%

import numpy as np

titanic\_df.dtypes

titanic\_df.info()

#%%

titanic\_df.describe(include = [np.object])

#%%

titanic\_df.describe(include = 'all')

#%%

my\_df = titanic\_df[["Sex", "Pclass", "Age"]]

#%%

df\_aremorethan60 = titanic\_df[titanic\_df['Age']>60]

df\_aremorethan60.shape

#%%

my\_df1 = titanic\_df[(titanic\_df["Age"]>60) & (titanic\_df['Sex'] == 'male') & (titanic\_df['Survived']==1)]

my\_df1.shape[0]

#%%

titanic\_df['Pclass'].value\_counts()

#%%

pd.crosstab(titanic\_df['Sex'], titanic\_df['Survived'])

#%%

print(titanic\_df['Sex'][titanic\_df['Survived']==1].value\_counts())

#%%

print(titanic\_df['Sex'][titanic\_df['Survived']==0].value\_counts())

#%%

PassengerAge = titanic\_df['Age']

PassengerAge = PassengerAge.dropna()

Bins = [0,15,21,60, PassengerAge.max()]

Binlabels = ['Child', "Adolescent", "Adult" , "Senior" ]

Categories = pd.cut(PassengerAge, Bins, labels = Binlabels, include\_lowest = True)

print(Categories.value\_counts())

#%%

print(titanic\_df.isnull().sum())

#%%

print(titanic\_df.isnull().any(axis = 1).sum())

#%%

print(titanic\_df['Age'].isnull().sum())

#%%

half\_count = len(titanic\_df) / 2

titanic\_df = titanic\_df.dropna(thresh = half\_count, axis = 1)

print(titanic\_df)

#%%

titanic\_df['Embarked'].fillna(titanic\_df['Embarked'].mode()[0], inplace = True)

#%%

titanic\_df['Age'].mean()

#%%

titanic\_df['Age'].fillna(titanic\_df['Age'].mean(), inplace = True)

print(titanic\_df.isnull().sum())

#%%

adult\_df = pd.read\_csv(r'C:\Users\Administrator\Desktop\Data Science Program\Python\Adult\adult\_data.csv', header = None, delimiter = ' \*, \*', engine = 'python')

adult\_df.head()

#%%

pd.set\_option('display.max\_columns', None)

#%%

#preprocessing the data

adult\_df.columns = ['age', 'workclass', 'fnlwgt', 'education', 'education\_num',

'marital\_status', 'occupation', 'relationship',

'race', 'sex', 'capital\_gain', 'capital\_loss',

'hours\_per\_week', 'native\_country', 'income']

adult\_df.head()

#%%

adult\_df.isnull().sum()

#%%

adult\_df = adult\_df.replace(['?'], np.nan)

#%%

adult\_df.isnull().sum()

#%%

adult\_df\_rev = pd.DataFrame.copy(adult\_df)

#%%

adult\_df\_rev.describe(include = "all")

#%%

for value in ['workclass', 'occupation', 'native\_country']:

adult\_df\_rev[value].fillna(adult\_df\_rev[value].mode()[0], inplace = True)

#%%

adult\_df\_rev.isnull().sum()

#%%

print(adult\_df\_rev['education'].dtype)

#%%

print(adult\_df\_rev.dtypes)

#%%

colname = ['workclass','education','marital\_status','occupation',

'relationship', 'race', 'sex',

'native\_country', 'income']

print(colname)

colname

#%%

from sklearn import preprocessing

#%%

le = {}

for x in colname:

le[x] = preprocessing.LabelEncoder()

for x in colname:

adult\_df\_rev[x] = le[x].fit\_transform(adult\_df\_rev[x])

#%%

adult\_df\_rev.describe(include= 'all')

#%%

adult\_df\_rev.head()

#%%

#0 --> <=50K

#1 --> >50K

#%%

x = adult\_df\_rev.values['age', 'workclass', 'fnlwgt', 'education', 'education\_num',

'marital\_status', 'occupation', 'relationship',

'race', 'sex', 'capital\_gain', 'capital\_loss',

'hours\_per\_week', 'native\_country', 'income']

y = adult\_df\_rev.values[:,-1]

print(x.dtype)

print(y.dtype)

#%%

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(x)

x = scaler.transform(x)

print(x)

#%%%

y = y.astype(int)

#%%

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size = 0.3, random\_state = 10)

#%%

from sklearn.linear\_model import LogisticRegression

#creating model

classifier = (LogisticRegression())

#fitting training data to the model

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

#predicting y values

#%%

classifier.score(x\_test, y\_test)

#%%

import statsmodels.api as sm

logit\_model=sm.Logit(y,x)

result=logit\_model.fit()

print(result.summary())

#%%

print(adult\_df\_rev.info())

#%%

print(c.summary())

#%%

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

print(list(zip(y\_test, y\_pred)))

#%%

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

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

print(cfm)

print()

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

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

print()

print(acc)

#%%

print(7009/(7009+1318))

print(1028/(1028+414))

#%%

print(7009/(7009+414))

print(1028/(1028+1318))

#%%

titanic\_1 = pd.read\_excel(r'C:\Users\Administrator\Desktop\Data Science Program\Python\Titanic\Titanic\_Survival\_Train.xls', header = 0, index\_col = None)

print(titanic\_1.head())

titanic\_1.info()

#%%

titanic\_1['Cabin', 'Embarked'].value\_counts()

titanic\_1.apply(pd.Series.value\_counts)

#%%

titanic\_2 = titanic\_1[['Age', 'Cabin', 'Embarked']]

titanic\_2.describe(include = 'all')

#%%

print(titanic\_1.isnull().any(axis = 1).sum())

#%%

print(titanic\_df['Age'].isnull().sum())

#%%

half\_count = len(titanic\_1) / 2

titanic\_1 = titanic\_1.dropna(thresh = half\_count, axis = 1)

titanic\_1.head()

#%%

titanic\_1['Embarked'].fillna(titanic\_1['Embarked'].mode()[0], inplace = True)

#%

titanic\_1['Age'].fillna(titanic\_1['Age'].mean(), inplace = True)

#%%

titanic\_1.isnull().sum()

#%%

titanic\_1 = titanic\_1.drop(['PassengerId', 'Ticket'], axis = 1)

#%%

print(titanic\_1.head())

#%%

le1 = {}

for x in titanic\_1[['Sex', 'Embarked']]:

le1[x] = preprocessing.LabelEncoder()

for x in titanic\_1[['Sex', 'Embarked']]:

titanic\_1[x] = le1[x].fit\_transform(titanic\_1[x])

print(titanic\_1.head())

#%%

x = titanic\_1.values[:,1:]

#%%

y = titanic\_1.values[:,0]

#%%

scaler1 = StandardScaler()

scaler1.fit(x)

x = scaler1.transform(x)

#%%

y.dtype

x.dtype

#%%

y = y.astype(int)

#%%

x\_train1, x\_test1, y\_train1, y\_test1 = train\_test\_split(x, y, test\_size = 0.3, random\_state = 10)

#%%

LR1 = LogisticRegression()

LR1.fit(x\_train1, y\_train1)

y\_pred1 = LR1.predict(x\_test1)

#%%

print(list(zip(y\_test1, y\_pred1)))

#%%

cm1 = confusion\_matrix(y\_test1, y\_pred1)

print(cm1)

#%%

(152+64)/(152+22+30+64)

#%%

print(classification\_report(y\_test1, y\_pred1))

#%%

accuracy\_score(y\_test1, y\_pred1)

#%%

hepd = pd.read\_csv(r'C:\Users\Administrator\Desktop\Data Science Program\Python\my data\hepatitis.csv', header = None, index\_col = None)

hepd.head()

#%%

hepd.columns = ['Class', 'AGE', 'SEX', 'STEROID', 'ANTIVIRALS', 'FATIGUE', 'MALAISE', 'ANOREXIA', 'LIVER\_BIG', 'LIVER\_FIRM', 'SPLEEN\_PALPABLE', 'SPIDERS', 'ASCITES', 'VARICES', 'BILIRUBIN', 'ALK\_PHOSPHATE', 'SGOT', 'ALBUMIN', 'PROTIME','HISTOLOGY']

#%%

hepd.head()

#%%

hepd.info()

#%%

hepd.replace(['?'], np.nan, inplace = True)

#%%

hepd.isnull().sum()

#%%

hepd.describe(include = 'all')

#%%

m\_o = ['STEROID', 'FATIGUE', 'MALAISE', 'ANOREXIA', 'LIVER\_BIG', 'LIVER\_FIRM', 'SPLEEN\_PALPABLE' ,'SPIDERS', 'ASCITES', 'VARICES', 'BILIRUBIN', 'ALK\_PHOSPHATE', 'SGOT', 'ALBUMIN' ,'PROTIME']

#%%

for x in m\_o:

hepd.fillna(hepd[x].mode()[0], inplace = True)

#%%

hepd.isnull().sum()

#%%

hepd.STEROID.astype(int)

hepd.FATIGUE.astype(int)

#%%

hepd.MALAISE.astype(int)

#%%

hepd.ANOREXIA.astype(int)

#%%

hepd.LIVER\_BIG.astype(int)

#%%

hepd.LIVER\_FIRM.astype(int)

#%%

hepd.SPLEEN\_PALPABLE.astype(int)

#%%

hepd.SPIDERS.astype(int)

#%%

hepd.ASCITES.astype(int)

#%%

hepd.VARICES.astype(int)

#%%

hepd.ALK\_PHOSPHATE.astype(int)

#%%

hepd.SGOT.astype(int)

#%%

hepd.PROTIME.astype(int)

#%%

#%%

# 2 = live

# 1 = die

#%%

x = hepd.values[:,1:]

y = hepd.values[:,0]

#%%

train\_x, test\_x, train\_y, test\_y = train\_test\_split(x, y, test\_size = 0.3, random\_state = 10)

#%%

lrmodel = LogisticRegression()

lrmodel.fit(train\_x, train\_y)

#%%

y\_pred= lrmodel.predict(test\_x)

#%%

from sklearn.datasets import load\_breast\_cancer

#%%

from sklearn.neighbors