**Practical File**

**Machine Learning**



B.Sc. (H) Computer Science

Semester 6th

Name: Kartikey Sharma

Roll No: CSC/21/18

Examination Roll No: 21059570020

PRACTICAL QUESTION 1 :-

#Peform elementary mathematical Learning like addition ,multiplication,

division and expnential

import numpy as np

import array as arr

x = 1

y = 2

print(x+y)

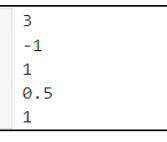
print(x-y)

print(x\*\*y)

print(x/y)

print(x%y)

#output



Pratical Ques.2 :-

#Perfom elementary logiacal operations like OR,AND,checking for

equality,NOT,XOR).

x = 2

y = 5

print(x&amp;y)

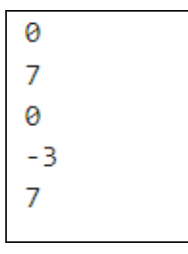
print(x|y)

print(x&amp;y)

print(~x)

print(x^y)

#output



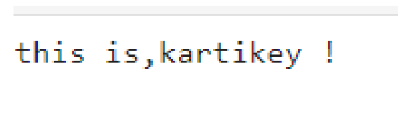
Practical ques.3 :-

#Create initialize and display simple variables and use simple formatting for

variable

N = "kartikey"

print("this is,%s !"%N)



Practical Ques. P4

#create single dimension / multi-dimension arrays and array with specific

values like array of all zeroes ,

#array with random values within a range o diagonal matix.

a = [2,3,4,5]

x = np.zeros([3,3])

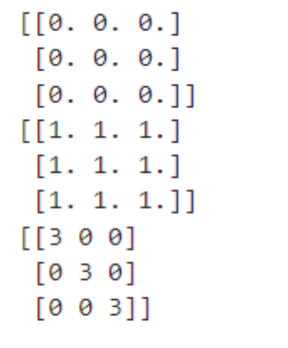
print(x)

y = np.ones([3,3])

print(y)

z = np.diag([3,3,3])

print(z)



Practical 5

#Use command to compute the size /length of particular row/column,load data from a text

file , store matrix data to a text file finding

# out variables and their features in the current scope .

import numpy as np

matrix = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

# Compute the length of a particular row

row\_index = 1

row\_length = len(matrix[row\_index])

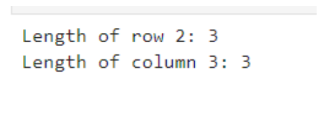
print(f"Length of row {row\_index + 1}: {row\_length}")

# Compute the length of a particular column (e.g., column 2)

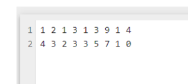
col\_index = 2

col\_length = len(matrix[:, col\_index])

print(f"Length of column {col\_index + 1}: {col\_length}")

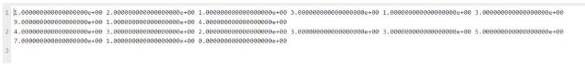


data = np.loadtxt('data.txt')



# Store the matrix data back to a text file

np.savetxt('output.txt', data)



var1 = 10

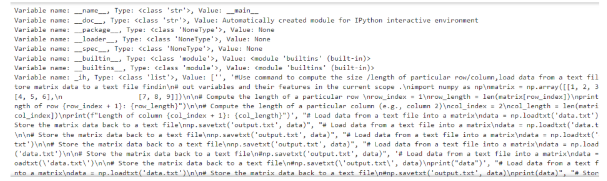
var2 = 'hello'

var3 = [1, 2, 3]

current\_scope = locals().copy()

for var\_name, var\_value in current\_scope.items():

print(f"Variable name: {var\_name}, Type: {type(var\_value)}, Value: {var\_value}")



Practical Ques. P6

#perform basic operations on matrices and display specific rows and

columns of the matix

import numpy as np

matrix1 = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

matrix2 = np.array([[9, 8, 7],

[6, 5, 4],

[3, 2, 1]])

# Addition

addition\_result = matrix1 + matrix2

print("Addition Result:")

print(addition\_result)

# Subtraction

subtraction\_result = matrix1 - matrix2

print("\nSubtraction Result:")

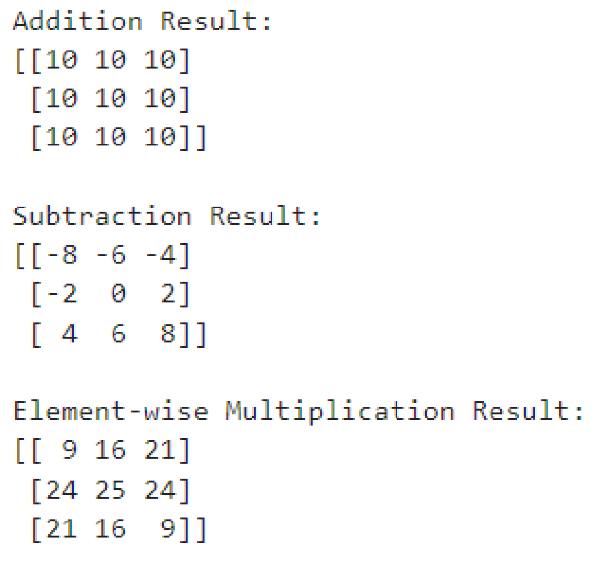
print(subtraction\_result)

# Multiplication

multiplication\_result = matrix1 \* matrix2

print("\nElement-wise Multiplication Result:")

print(multiplication\_result)



Practical Ques. P7

#7. Perform other matrix operations like converting matrix data to absolute

values, taking the negative of matrix values, additing/removing rows/columns

from a matrix, finding the maximum

#or minimum values in a matrix or in a row/column, and finding the sum of

some/all

#elements in a matrix.

import numpy as np

# Define a matrix

matrix = np.array([[1, -2, 3],

[-4, 5, -6],

[7, 8, -9]])

# Convert matrix data to absolute values

absolute\_matrix = np.abs(matrix)

print("Absolute Values of Matrix:")

print(absolute\_matrix)

# Take the negative of matrix values

negative\_matrix = -matrix

print("\nNegative of Matrix Values:")

print(negative\_matrix)

# Add a row to the matrix

new\_row = np.array([10, 11, 12])

matrix\_with\_new\_row = np.vstack([matrix, new\_row])

print("\nMatrix with Added Row:")

print(matrix\_with\_new\_row)

# Remove a row from the matrix

matrix\_with\_removed\_row = np.delete(matrix, 1, axis=0) # Remove row at

index 1

print("\nMatrix with Removed Row:")

print(matrix\_with\_removed\_row)

# Add a column to the matrix

new\_column = np.array([10, 11, 12]).reshape(-1, 1)

matrix\_with\_new\_column = np.hstack([matrix, new\_column])

print("\nMatrix with Added Column:")

print(matrix\_with\_new\_column)

# Remove a column from the matrix

matrix\_with\_removed\_column = np.delete(matrix, 1, axis=1) # Remove

column at index 1

print("\nMatrix with Removed Column:")

print(matrix\_with\_removed\_column)

# Find the maximum value in the matrix

max\_value = np.max(matrix)

print("\nMaximum Value in Matrix:", max\_value)

# Find the minimum value in the matrix

min\_value = np.min(matrix)

print("Minimum Value in Matrix:", min\_value)

# Find the maximum value in each row

max\_values\_in\_rows = np.max(matrix, axis=1)

print("\nMaximum Values in Each Row:")

print(max\_values\_in\_rows)

# Find the minimum value in each column

min\_values\_in\_columns = np.min(matrix, axis=0)

print("Minimum Values in Each Column:")

print(min\_values\_in\_columns)

# Find the sum of all elements in the matrix

total\_sum = np.sum(matrix)

print("\nSum of All Elements in Matrix:", total\_sum)

# Find the sum of elements in each row

sum\_in\_rows = np.sum(matrix, axis=1)

print("Sum of Elements in Each Row:")

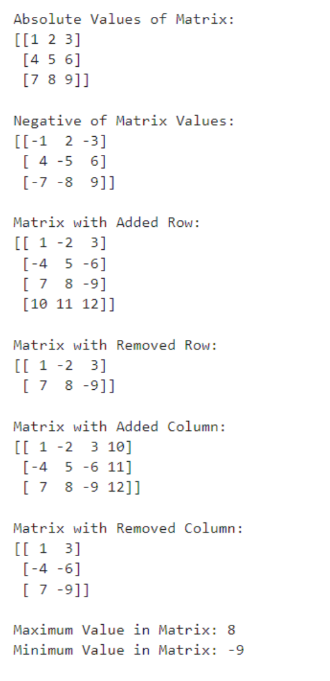
print(sum\_in\_rows)

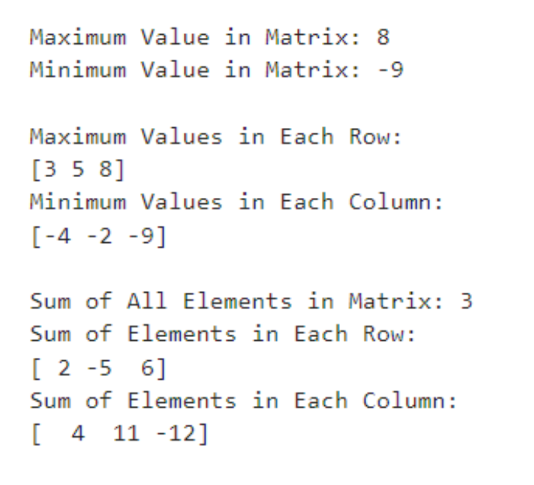
# Find the sum of elements in each column

sum\_in\_columns = np.sum(matrix, axis=0)

print("Sum of Elements in Each Column:")

print(sum\_in\_columns)





Practical Ques. P8

#Create various type of plots/charts like histograms, plot based on sine/cosine function

based on data from a matrix.

#Further label different axes in a plot and data in a plot.

import numpy as np

import matplotlib.pyplot as plt

# Create sample data

data = np.random.randn(1000) # Random data for histogram

x = np.linspace(0, 10, 100) # Data for sine and cosine functions

y\_sin = np.sin(x)

y\_cos = np.cos(x)

# Create histogram

plt.figure(figsize=(8, 6))

plt.hist(data, bins=30, color='skyblue', edgecolor='black')

plt.title('Histogram of Random Data')

plt.xlabel('Value')

plt.ylabel('Frequency')

plt.grid(True)

plt.show()

# Create plot for sine and cosine functions

plt.figure(figsize=(8, 6))

plt.plot(x, y\_sin, label='Sine', color='blue')

plt.plot(x, y\_cos, label='Cosine', color='red')

plt.title('Plot of Sine and Cosine Functions')

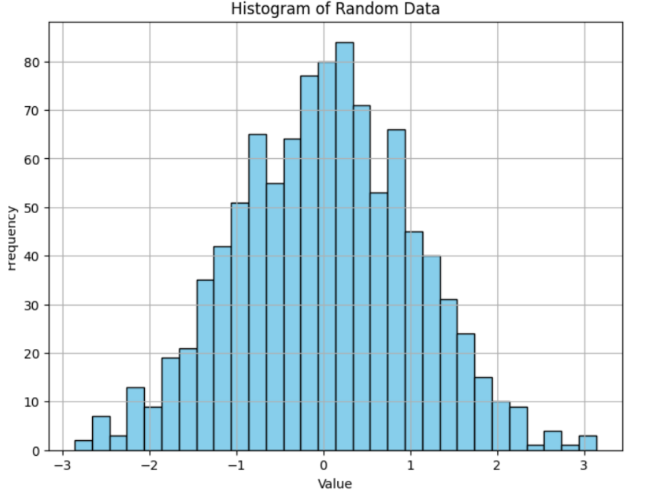
plt.xlabel('X')

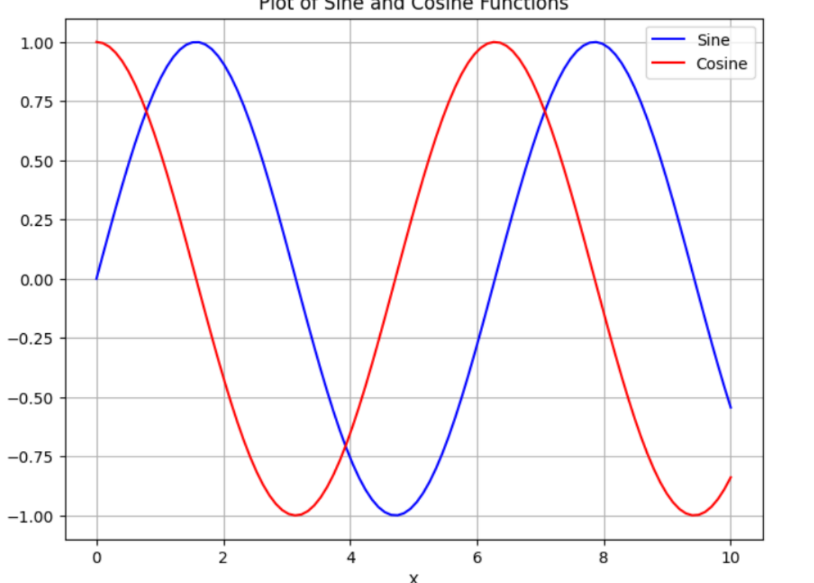
plt.ylabel('Y')

plt.legend()

plt.grid(True)

plt.show()





Practical Ques. P9

#Generate different subplots from a given plot and color plot data.

import numpy as np

import matplotlib.pyplot as plt

# Create sample data

x = np.linspace(0, 10, 100)

y1 = np.sin(x)

y2 = np.cos(x)

y3 = np.tan(x)

# Create subplots

fig, axs = plt.subplots(3, 1, figsize=(8, 12))

# Plot data in each subplot with different colors

axs[0].plot(x, y1, color='blue')

axs[0].set\_title('Sine Function')

axs[0].set\_xlabel('X')

axs[0].set\_ylabel('Y')

axs[1].plot(x, y2, color='red')

axs[1].set\_title('Cosine Function')

axs[1].set\_xlabel('X')

axs[1].set\_ylabel('Y')

axs[2].plot(x, y3, color='green')

axs[2].set\_title('Tangent Function')

axs[2].set\_xlabel('X')

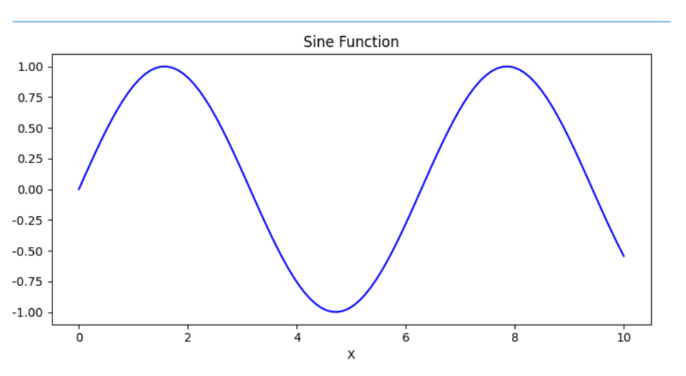
axs[2].set\_ylabel('Y')

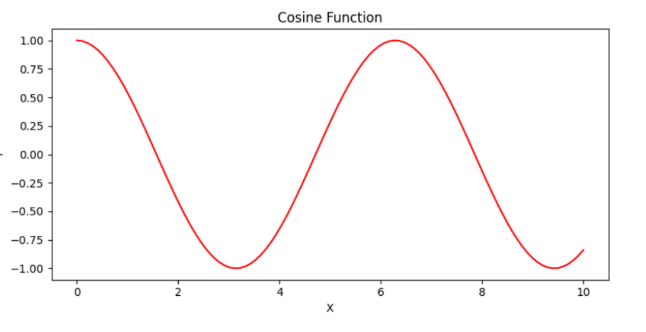
# Adjust layout

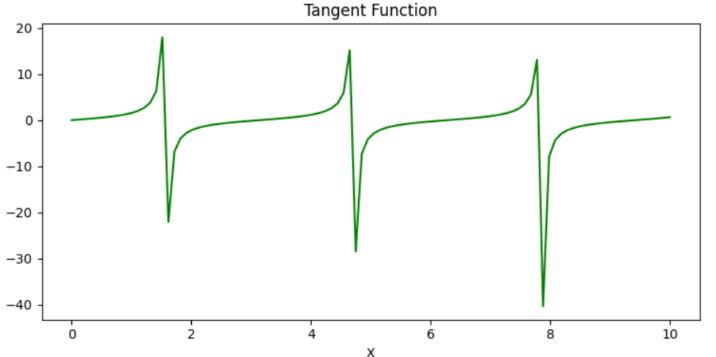
plt.tight\_layout()

# Show plot

plt.show()







Practical Ques. P10

#Use conditional statements and different type of loops based on simple examples

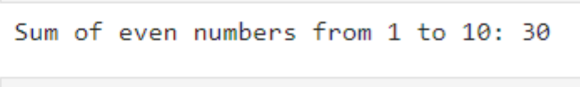
sum\_even = 0

for i in range(1, 11):

if i % 2 == 0:

sum\_even += i

print("Sum of even numbers from 1 to 10:", sum\_even)



Practical Ques. P11

#11. Perform vectorized implementation of simple matrix operation like finding the transpose of a matrix, adding, subtracting or multiplying two matrices.

import numpy as np

matrix1 = np.array([[1, 2, 3],

[4, 5, 6]])

matrix2 = np.array([[7, 8, 9],

[10, 11, 12]])

transpose\_matrix1 = np.transpose(matrix1)

# Adding two matrices

sum\_matrix = matrix1 + matrix2

# Subtracting two matrices

difference\_matrix = matrix1 - matrix2

# Multiplying two matrices (element-wise multiplication)

product\_matrix = matrix1 \* matrix2

# Multiplying two matrices (matrix multiplication)

dot\_product\_matrix = np.dot(matrix1, np.transpose(matrix2))

# Printing the results

print("Original Matrix 1:")

print(matrix1)

print("\nTranspose of Matrix 1:")

print(transpose\_matrix1)

print("\nMatrix 2:")

print(matrix2)

print("\nSum of Matrix 1 and Matrix 2:")

print(sum\_matrix)

print("\nDifference of Matrix 1 and Matrix 2:")

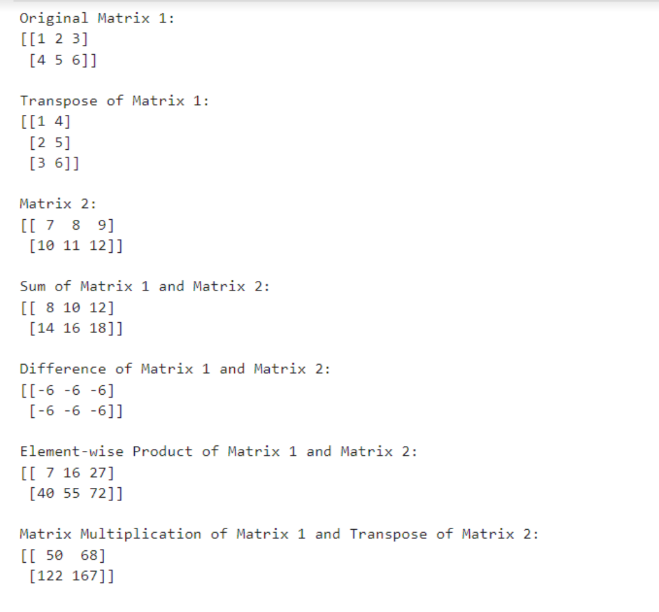
print(difference\_matrix)

print("\nElement-wise Product of Matrix 1 and Matrix 2:")

print(product\_matrix)

print("\nMatrix Multiplication of Matrix 1 and Transpose of Matrix 2:")

print(dot\_product\_matrix)



Q12) Implement Linear regression problem . For example , based on a dataset comprising of existing set of prices and area/size of the houses, predict the estimated price of a given house.

Dataset = pd.read\_csv(housing.csv)

Dataset.head()

df = Dataset[[‘area’,’price’]]

x= df[[‘area’]]

y= df[[‘price’]]

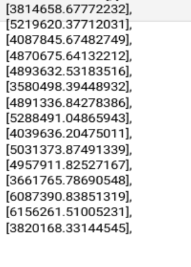
from sklearn.model\_selection import train\_test\_split

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

Simple\_regression = LinearRegression()Simple\_regression.fit(x\_train,y\_train)

Y\_pred = Simple\_regression.predict(x\_test)

Y\_pred



Q13) Based on multiple features/variables perform Linear Regression. For example, based on a number of additional features like number of bedrooms, servant room, number of balconies, number of houses of years a house has been built - predict the price of a house.

Dataset = pd.read\_csv(housing.csv)

Dataset.head()

df = Dataset[[‘area’,’price’]]

x= df[[‘area’,’bedroom’,bathroom’,’parking’]]

y= df[[‘price’]]

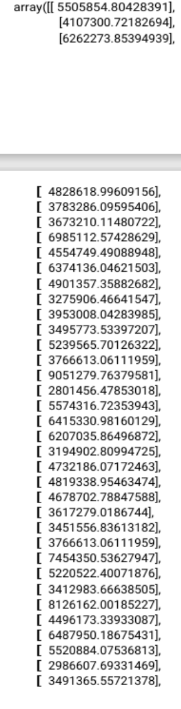
from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size = 0.30,shuffle = True)

MLR = LinearRegression()MLR.fit(x\_train,y\_train)

Y\_pred = MLR.predict(x\_test)

Y\_pred



Q14 implement a logistic regression problem. for example based on different features students data classify whethe a student is sutible for particular activity .

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sb

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

from sklearn.linear\_model import LogisticRegression

df = pd.read\_csv('Downloads/StudentsPerformance.csv')

print(df.head())

df = df.replace({'yes':1, 'no':0, '?':'Others', 'others':'Others'})

df['test preparation course'] = df['test preparation course'].replace({'completed': 1, 'none': 0})

x = df.iloc[:, [5, 6, 7]]

y = df.iloc[:, 4]

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size = 0.25,random\_state = 42)

from sklearn.metrics import accuracy\_score

x\_train, x\_test,\

y\_train, y\_test = train\_test\_split(x, y,

test\_size=0.20,

random\_state=23)

# LogisticRegression

clf = LogisticRegression(random\_state=0)

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

# Prediction

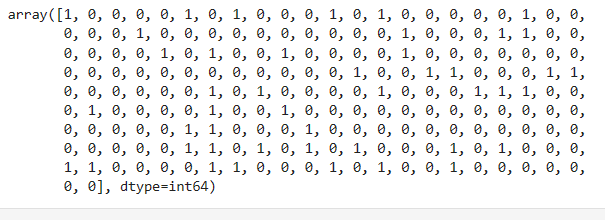
y\_pred = clf.predict(x\_test)

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

print("Logistic Regression model accuracy (in %):", acc\*100)



y\_pred



Q15) Use some function for regularization of dataset based on problem 14.

from sklearn import linear\_model

lasso\_reg= linear\_model.Lasso(alpha = 50,max\_iter = 100,tol = 0.1)

lasso\_reg.fit(x\_train, y\_train)

from sklearn.linear\_model import Ridge

ridge\_reg= Ridge(alpha = 50,max\_iter = 100,tol = 0.1)

ridge\_reg.fit(x\_train, y\_train)

Q16) Use some function for neural networks, like Stochastic Gradient Descent or backpropagation algorithm to predict the value of a variable based on the dataset of problem 14.

Def sigmoid(x):

Return 1/(1+np.exp(-x))

Training\_input = np.array([[2,1,3],[1,0,0]])

Training\_outputs = np.array([[1,1,0,1]]])

Np.random.seed(1)

Synaptic\_weights = 2\*np.random.random((3,1))

for iteration in range(20000):

Input\_layer = training\_inputs

outputs = sigmoid(np.dot(input\_layer,synaptic\_weights))

Error = training\_outputs – outputs

Adjustments = error\*sigmoid\_derivative(outputs)

Synaptic\_weights += np.dot(inpput\_layer.T,adjustments)

Naive bayes :-

From sklearn.naive\_bayes import GaussianNB

Classifier = GaussianNB()

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

Y\_train = classifier predict(x\_test)

From sklearn import matrix

Print(matrices.accuracy.score(y\_test,y\_predict)\*100)

Decision Tree :-

Sklearn tree import DecisionTreeClassifier

From sklearn.parallel selection input

From sklearn import confusion matrix, accuracy matrix

Like this only :-

Dataset.isna().sum()

Cols\_fill\_0 = [‘’,’’,’’,’’,’’]

Datasetname[Cols\_fill\_0] = Datasetname[Cols\_fill\_0].fillna(0)

Dataset.isna().sum()

Dataset[‘’] = dataset[‘’].fillna(dataset.landsize.mean())

Dataset[‘Bu’] = dataset[‘Bu’].fillna(dataset.Bu.mean())

Dataset.dropna(inplace=True)

Dataset = pd.get\_dummies(dataset, drop\_first = True)

Dataset.head()

Dataset.drop(‘price’,axis=1) # na values will get dropped