Python Programming & Data Science LAB (20A05101P)

SAMPLE RECORD

I - B. TECH & II- SEM

Prepared by:

P Khatija Khan, Assistant Professor

K Kushboo, Assistant Professor

Department of Computer Science & Engineering

VEMU INSTITUTE OF TECHNOLOGY

(Approved By AICTE, New Delhi and Affiliated to JNTUA, Anantapur) Accredited By NAAC, NBA (EEE, ECE & CSE) & ISO: 9001-2015 Certified Institution

Experiments List

1. Write a program to demonstrate

a) Different numeric data types and

b) To perform different Arithmetic Operations on numbers in Python

2. Write a program to create, append, and remove lists in Python

3. Write a program to demonstrate working with tuples in Python

4. Write a program to demonstrate working with dictionaries in Python

5. Write a program to demonstrate

a) arrays

b)array indexing such as slicing, integer array indexing and Boolean array indexing along with their basic operations in NumPy.

6.Write a program to demonstrate

a) arrays

b)array indexing such as slicing, integer array indexing and Boolean array indexing along with their basic operations in NumPy.

7.Write a program to compute summary statistics such as mean, median, mode, standard deviation and variance of the given different types of data.

8.Write a script named copyfile.py. This script should prompt the user for the names of two text files. The contents of the first file should be the input that to be written to the second file.

9.Write a program to demonstrate Regression analysis with residual plots on a given data set.

10.Write a program to demonstrate the working of the decision tree-based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

11.Write a program to implement the Naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

12.Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions using Java/Python ML library classes.

Additional Experiments

1. Write a program to find factorial of a given number.
2. Write a program to check whether the given string is palindrome or not.
3. Write a program to demonstrate function calling in python.

EXPERIMENT-1

Write a program to demonstrate
a) Different numeric data types and
b) To perform different Arithmetic Operations on numbers in Python

a) Different numeric data types

SOURCE CODE:

```
a=5
b=5.7
c=2+7j
d="python"
print("avalueis:",a,"\t\tdatatypeis:",type(a))
print("bvalue is:",b,"\tdatatypeis:",type(b))
print("c value is:",c,"\t data type is:",type(c))
print("dvalueis:",d,"\tdatatypeis:",type(d))
```

OUTPUT:

```
avalueis: 5                datatypeis: <class 'int'> bvalue is: 5.7
datatypeis: <class 'float'>
c value is: (2+7j)         data type is: <class 'complex'>
dvalueis: python           datatypeis: <class 'str'>
```

RESULT: Hence the program has been Succesfully Implemented

b) To perform different Arithmetic Operations on numbers in Python

SOURCE CODE:

```
n1=int(input("Enter first number:"))
n2=int(input("Entersecondnumber:"))
sum=n1+n2
sub=n1-n2
mul=n1*n2
div=n1/n2
mdiv=n1%n2
fdiv=n1//n2
exp=n1**n2
print("The addition of",n1,"and",n2,"is:",sum)
print("The subtraction of",n1,"and",n2,"is:",sub)
print("The multiplication of",n1,"and",n2,"is:",mul)
print("The division of",n1,"and",n2,"is:",div)
print("The modulo division of",n1,"and",n2,"is:",mdiv)
print("The floor division of",n1,"and",n2,"is:",fdiv)
print("The exponent of",n1,"and",n2,"is:",exp)
```

OUTPUT:

```
Enter first number:2
Entersecondnumber:4
The addition of 2 and 4 is: 6
The subtraction of 2 and 4 is: -2
The multiplication of 2 and 4 is: 8
The division of 2 and 4 is: 0.5
The modulo division of 2 and 4 is: 2
The floor division of 2 and 4 is: 0
The exponent of 2 and 4 is: 16
```

RESULT: Hence the program has been Succesfully Implemented

EXPERIMENT-2

Write a program to create, append, and remove lists in Python.

SOURCE CODE:

```
program to create,append, and remove lists in python
#creating an empty list
empty_List=[]
print("Empty List is:", empty_List)
#creating a list with elements
my_List=[10,507,"python"]
print("created list is:",my_List)
#Inserting new elements using append()
my_List.append(20)
my_List.append("program")
my_List.append([3,7])
print("After adding elements the new list is:",my_List)
#deleting elements using pop() method
d1=my_List.pop()
d2=my_List.pop(2)
#deleting the elements using remove
my_List.remove(10)
print("After deleting elements the new list is:",my_List)
#Removing all the elements using clear()
my_List.clear()
print("After removing all the elements the new list is:",my_List)
```

OUTPUT:

```
Empty List is: []
created list is: [10, 507, 'python']
After adding elements the new list is: [10, 507, 'python', 20, 'program', [3, 7]]
After deleting elements the new list is: [507, 20, 'program']
After removing all the elements the new list is: []
```

RESULT: Hence the program has been Succesfully Implemented

EXPERIMENT – 3

Write a program to demonstrate working with tuples in Python

SOURCE CODE:

write a program to demonstrate working with tuples in python

```
#creating an empty tuple
empty_tup=()
print("Empty tuple=",empty_tup)
#creating single element tuple
single_tup=(10,)
print("single element tuple=",single_tup)
#creating a tuple with multiple elements
my_Tup=(10,3.7,'program','a')
print("Tuple with multiple elements is:",my_Tup)
print("Length of the tuple is:",len(my_Tup))
T1=(10,20,30,40,70.5,33.3)
print("Maximum value of the tuple T1 is:",max(T1))
print("Minimum value of the tuple T1 is:",min(T1))
str1='tuple'
T=tuple(str1)#convering string into tuple
print("After converting a string into tuple,the new tuple is:",T)
L=[2,4,6,7,8]
```

```
T2=tuple(L)#convering string into tuple

print("After converting a List into tuple,the new tuple is:",T2)
```

OUTPUT:

```
Empty tuple= ()
single element tuple= (10,)
Tuple with multiple elements is: (10, 3.7, 'program', 'a')
Length of the tuple is: 4
Maximum value of the tuple T1 is: 70.5
Minimum value of the tuple T1 is: 10
After converting a string into tuple,the new tuple is: ('t', 'u', 'p', 'l', 'e')
After converting a List into tuple,the new tuple is: (2, 4, 6, 7, 8)
```

RESULT: Hence the program has been Succesfully Implemented

EXPERIMENT-4

Write a program to demonstrate working with dictionaries in Python.

SOURCE CODE:

```
#write a program to demonstrate working with dictionaries in python
#empty dictionary
my_dict={}
print("Empty dictionary is:",my_dict)
#dictionary with integer keys
my_dict={1:'apple',2:'ball'}
print("dictionary with integer keys",my_dict)
#dictionary with mixed keys
my_dict={'name':'rishi',1:[2,4,3]}
print("dictionary with mixed keys",my_dict)
#using dicy.fromkeys()
my_dict=dict.fromkeys("abcd",'alphabet')
print("dictionary created by using dict.fromkeys method=",my_dict)
#using get method
my_dict={'name':'jack','age':25}
print(my_dict['name']) #ouput jack
#changing and adding dictionary elements
```

```
my_dict['age']=18 #upadte vaue

my_dict['class']="B.Tech" #updating vlaue

print("After changing and adding the values,the new dictionary=",my_dict)

#using items()

print("items in the dictionary is:",my_dict.items())

#using keys()

print("Keys in the dictionary is:",my_dict.keys())

#using values()

print("values in the dictionary is:",my_dict.values())
```

OUTPUT:

```
Empty dictionary is: {}
dictionary with integer keys {1: 'apple', 2: 'ball'}
dictionary with mixed keys {'name': 'rishi', 1: [2, 4, 3]}
dictionary created by using dict.fromkeys method= {'a': 'alphabet', 'b': 'alphabet', 'c': 'alphabet', 'd':
'alphabet'}
jack
After changing and adding the values,the new dictionary= {'name': 'jack', 'age': 18, 'class':
'B.Tech'}
items in the dictionary is: dict_items([('name', 'jack'), ('age', 18), ('class', 'B.Tech')])
Keys in the dictionary is: dict_keys(['name', 'age', 'class'])
values in the dictionary is: dict_values(['jack', 18, 'B.Tech'])
```

RESULT: Hence the program has been Succesfully Implemented

EXPERIMENT – 5

Write a program to demonstrate a) arrays b) array indexing such as slicing, integer array indexing

and Boolean array indexing along with their basic operations in NumPy.

A)ARRAYS

SOURCE CODE:

```
import numpy as np

a = np.array(42)

b = np.array([1, 2, 3, 4, 5])

c = np.array([[1, 2, 3], [4, 5, 6]])

d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print("entered array is:",a,"and its dimension is:",a.ndim)

print("entered array is:",b,"and its dimension is:",b.ndim)

print("entered array is:",c,"and its dimension is:",c.ndim)

print("entered array is:",d,"and its dimension is:",d.ndim)
```

OUTPUT:

```
entered array is: 42 and its dimension is: 0
entered array is: [1 2 3 4 5] and its dimension is: 1
entered array is: [[1 2 3]
 [4 5 6]] and its dimension is: 2
entered array is: [[[1 2 3]
  [4 5 6]]

 [[1 2 3]
 [4 5 6]]] and its dimension is: 3
```

RESULT: Hence the program has been Succesfully Implemented

B)array indexing such as slicing, integer array indexing and Boolean array indexing along with their basic operations in NumPy.

SOURCE CODE:

```
import numpy as np

a=np.arange(10,1,-2)

print("a sequential array with nagative step value:",a)

newarr=[a[3],a[1],a[2]]

print("elements at these indices are:",newarr)

a=np.arange(20)

print("Array is:",a)

print("a[-8:17:1]=",a[-8:17:1])

print("a[10:]=",a[10:])
```

OUTPUT:

```
a sequential array with nagative step value: [10  8  6  4  2]
elements at these indices are: [4, 8, 6]
Array is: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
a[-8:17:1]= [12 13 14 15 16]
a[10:]= [10 11 12 13 14 15 16 17 18 19]
```

RESULT: Hence the program has been Succesfully Implemented

EXPERIMENT-6

Write a program to compute summary statistics such as mean, median, mode, standard deviationand variance of the given different types of data.

SOURCE CODE:

```
import numpy as np

a= np.array([[1,23,78],[98,60,75],[79,25,48]])

print("Entered array=",a)
#Minimum Function

print("minimum=",np.amin(a))

#Maximum Function

print("maximum=",np.amax(a))

#Mean Function

print("mean=",np.mean(a))

#Median Function

print("median=",np.median(a))

#std Function

print("standarad deviation=",np.std(a))

#var Function

print("variance=",np.var(a))
```

OUTPUT:

```
Entered array= [[ 1 23 78]
 [98 60 75]
 [79 25 48]]
    minimum= 1
    maximum= 98
    mean= 54.111111111111114
    median= 60.0
    standarad deviation= 30.296477405960523
    variance= 917.8765432098766
```

RESULT: Hence the program has been Succesfully Implemented

EXPERIMENT-7

Write a script named copyfile.py. This script should prompt the user for the names of two text files. The contents of the first file should be the input that to be written to the second file.

SOURCE CODE:

```
infile=input("enter first file name:")

outfile=input("enter second file name:")

f1=open("firstfile.txt",'r')

f2=open("secondfile.txt",'w+')

content=f1.read()

f2.write(content)

 f1.close()

 f2.close()
```

OUTPUT:

```
enter first file name:TOPPERS LIST enter second file
name:exp 2

FileNotFoundError                         Traceback (most recent call last)
<ipython-input-14-83c708416bfd> in <cell line: 3>()
     1 infile=input("enter first file name:")
     2 outfile=input("enter second file name:")
----> 3 f1=open("firstfile.txt",'r')
     4 f2=open("secondfile.txt",'w+')
     5 content=f1.read()
     6 FileNotFoundError: [Errno 2] No such file or directory: 'firstfile.txt'
---------------------------------------------------------------------------
```

RESULT: Hence the program has been Succesfully Implemented

EXPERIMENT-8

Write a program to demonstrate Regression analysis with residual plots on a given data set.

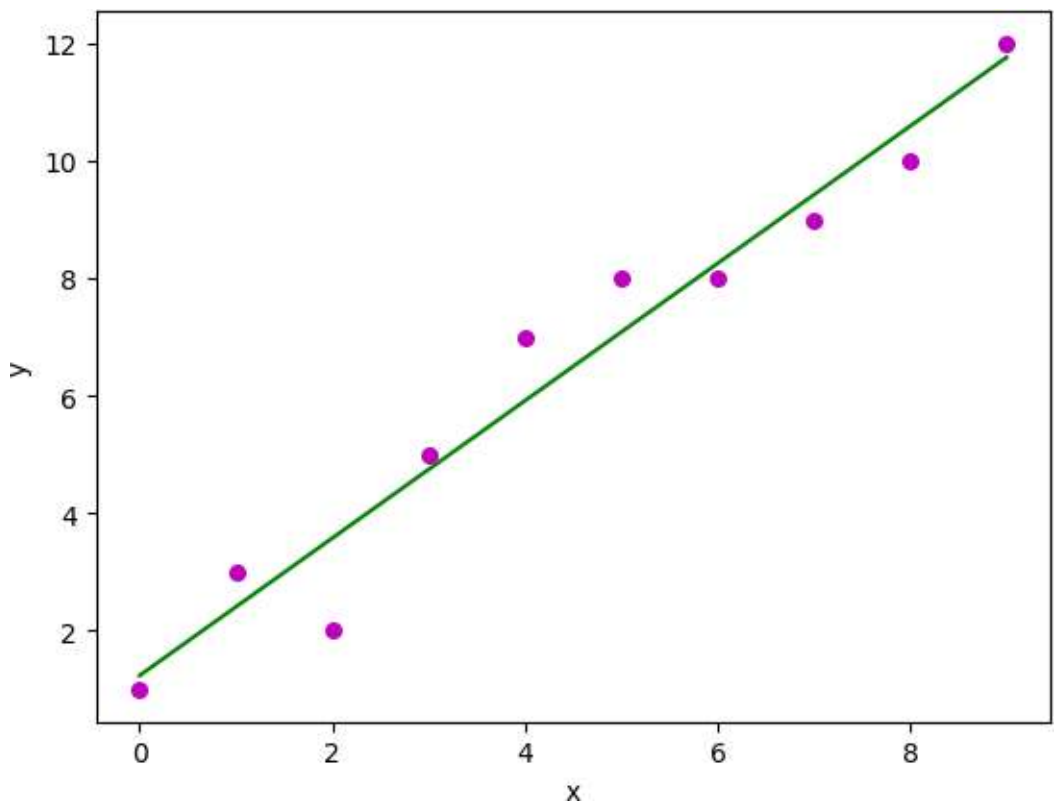
SOURCE CODE:

```
import numpy as np
import matplotlib.pyplot as plt
def estimate_coef(x, y):
    # number of observations/points
    n = np.size(x)
    # mean of x and y vector
    mx = np.mean(x)
    my = np.mean(y)
    # calculating cross-deviation and deviation about x
    sxy = np.sum(y*x) - n*my*mx
    sxx = np.sum(x*x) - n*mx*mx
    # calculating regression coefficients
    b1 = sxy / sxx
    b0 = my - b1*mx
    return (b0, b1)
def plot_regression_line(x, y, b):
    # plotting the actual points as scatter plot
    plt.scatter(x, y, color = "m",marker = "o", s = 30)
    # predicted response vector
    ypred = b[0] + b[1]*x
    # plotting the regression line
    plt.plot(x, ypred, color = "g")

    # putting labels
    plt.xlabel('x')
    plt.ylabel('y')
    # function to show plot
    plt.show()
def main():
    # observations or data
```

```
    x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
    y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

    # estimating coefficients
    b = estimate_coef(x, y)
    print("Estimated coefficients:\nb0 = {} \nb1 = {}".format(b[0], b[1]))
    # plotting regression line
    plot_regression_line(x, y, b)
main()
```

OUTPUT:

```
Estimated coefficients:
b0 = 1.2363636363636363
b1 = 1.1696969696969697
```



RESULT: Hence the program has been Succesfully Implemented

EXPERIMENT-9

Write a program to demonstrate the working of the decision tree-based ID3 algorithm.

SOURCE CODE:

```
Importing the required packages
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
#from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import  accuracy_score
from sklearn.metrics import classification_report
# Function importing Dataset
def importdata():
   balance_data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-'+
'databases/balance-scale/balance-scale.data',sep= ',', header = None)
   # Printing the dataswet shape
   print ("Dataset Length: ", len(balance_data))
   print ("Dataset Shape: ", balance_data.shape)
   # Printing the dataset obseravtions
   print ("Dataset: ",balance_data.head())
   return balance_data
   # Function to split the dataset
def splitdataset(balance_data):
   # Separating the target variable
   X = balance_data.values[:, 1:5]
   Y = balance_data.values[:, 0]

   # Splitting the dataset into train and test
   X_train, X_test, y_train, y_test = train_test_split(
         X, Y, test_size = 0.3, random_state = 100)
   return X, Y, X_train, X_test, y_train, y_test
         # Function to perform training with giniIndex.
def train_using_gini(X_train,  X_test, y_train):
```

```
        # Creating the classifier object
        clf_gini = DecisionTreeClassifier(criterion = "gini",
                        random_state = 100,max_depth=3, min_samples_leaf=5)
        # Performing training
        clf_gini.fit(X_train, y_train)
        return clf_gini
        # Function to perform training with entropy.
def tarin_using_entropy(X_train, X_test, y_train):
        # Decision tree with entropy
        clf_entropy = DecisionTreeClassifier(
                        criterion = "entropy", random_state = 100,
                        max_depth = 3, min_samples_leaf = 5)
        # Performing training
        clf_entropy.fit(X_train, y_train)
        return clf_entropy
# Function to make predictions
def prediction(X_test, clf_object):
        # Predicton on test with giniIndex
        y_pred = clf_object.predict(X_test)
        print("Predicted values:")
        print(y_pred)
        return y_pred
# Function to calculate accuracy
def cal_accuracy(y_test, y_pred):
                print("Confusion Matrix: ",
                confusion_matrix(y_test, y_pred))
                print ("Accuracy : ", accuracy_score(y_test,y_pred)*100)
                print("Report : ",classification_report(y_test, y_pred))
def main():
        # Building Phase
        data = importdata()
        X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
        clf_gini = train_using_gini(X_train, X_test, y_train)
        clf_entropy = tarin_using_entropy(X_train, X_test, y_train)
        # Operational Phase
        print("Results Using Gini Index:")
        # Prediction using gini
        y_pred_gini = prediction(X_test, clf_gini)
        cal_accuracy(y_test, y_pred_gini)
        print("Results Using Entropy:")
        # Prediction using entropy
        y_pred_entropy = prediction(X_test, clf_entropy)
        cal_accuracy(y_test, y_pred_entropy)
        # Calling main function
if__name_=="_main_":
```

```
main()
```

OUTPUT:

```
Dataset Length: 625
Dataset Shape:  (625, 5)
Dataset:     0  1  2  3  4
0  B  1  1  1  1
1  R  1  1  1  2
2  R  1  1  1  3
3  R  1  1  1  4
4  R  1  1  1  5
```

RESULT: Hence the program has been Succesfully Implemented

EXPERIMENT-10

Write a program to implement the Naïve Bayesian classifier for a sample training data set stored as

a .CSV file.

SOURCE CODE:

.CSVfile.

```
classNaiveBayesClassifier:
def__init__(self,X,y):
'''Xandydenotesthefeaturesandthetargetlabelsrespectively'''self.X,se
lf.y =X,y
self.N=len(self.X)#Lengthofthetraining set
self.dim=len(self.X[0])#Dimensionofthevectoroffeatures
self.attrs = [[] for _ in range(self.dim)] # Here we'll store the columns of the training
setself.output_dom={}#Outputclasseswiththenumberofocurrencesinthetrainingset.Inthiscase
wehaveonly 2classes
self.data=[]#Tostoreeveryrow[Xi,yi]for
iinrange(len(self.X)):
forjinrange(self.dim):
#ifwehaveneverseenthisvalueforthisattrbefore,
#thenweadditttotheattrsarrayinthecorrespondingpositionifnot
self.X[i][j]inself.attrs[j]:self.attrs[j].append(self.X[i][j])
#ifwehaveneverseenthisoutputclassbefore,
#thenweadditttotheoutput_domandcountoneoccurrence
fornowifnotself.y[i]inself.output_dom.keys():
```

```
self.output_dom[self.y[i]]=1
#otherwise,weincrementtheoccurrenceofthisoutputinthetrainingsetby1else:
self.output_dom[self.y[i]]+=1

# store the
rowself.data.append([self.X[i],self.y[i
]])defclassify(self, entry):
solve = None # Final
resultmax_arg=-
1#partialmaximumforyinself.o
utput_dom.keys():

prob=self.output_dom[y]/self.N#P(y)fori
inrange(self.dim):
cases=[xforxinself.dataifx[0][i]==entry[i]andx[1]==y]#allrowswithXi=
xin=len(cases)
prob*=n/self.N# P*=P(Xi=xi)
#ifwehaveagreaterprobforthisoutputthanthepartialmaximum...ifpro
b>max_arg:
max_arg =
probsolve=y
```

OUTPUT:

Array(['Iris_virgincia','Iris_versicolor','Iris_setosa',' Iris_virgincia',' Iris_setosa', Iris_virgincia', ' Iris_setosa', Iris_virgincia', 'Iris_versicolor', Iris_virgincia', Iris_virgincia', Iris_virgincia', (['Iris_virgincia','Iris_versicolor','Iris_setosa',' Iris_virgincia',' Iris_setosa', Iris_virgincia', ' Iris_setosa', Iris_virgincia', 'Iris_versicolor', Iris_virgincia', Iris_virgincia', Iris_virgincia'],dtype='<U15')

RESULT: Hence the program has been Succesfully Implemented

EXPERIMENT-11

Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set.

SOURCE CODE:

```
importnumpyasnpi
mportpandasaspd
import matplotlib.pyplot as
pltplt.rcParams['font.sans-
serif']=['SimHei']#Datageneration
train_num=200
test_num=100
config={
    'Corn':[[150,190],[40,70],[2,4]],
    'Potato':[[30,60],[7,10],[1,2]],
    'grass':[[10,40],[10,40],[0,1]]
}
plants=list(config.keys())
dataset=pd.DataFrame(columns=['height(cm)','Leaf
length(cm)','Stemdiameter(cm)','type'])index=0
#Natural
forpinconfig:
    foriinrange(int(train_num/3-
        3)):row=[]
        forj,[min_val,max_val]inenumerate(config[p]):
v=round(np.random.rand()*(max_val-
min_val)+min_val,2)whilevindataset[dataset.columns[j]]:
                v=round(np.random.rand()*(max_val-
            min_val)+min_val,2)row.append(v)
        row.append(p)dataset.l
        oc[index]=rowindex+=
```

```
        1
#Wrongdata
foriinrange(train_num-
    index):k=np.random.randint(
    3)
    p =
    plants[k]ro
    w=[]
    forj,[min_val,max_val]inenumerate(config[p]):
        v=round(np.random.rand()*(max_val-
        min_val)+min_val,2)whilevindataset[dataset.columns[j]]:
            v=round(np.random.rand()*(max_val-
        min_val)+min_val,2)row.append(v)
    row.append(plants[(k+1)%3])

    dataset.loc[index]=rowi
    ndex+=1
#dataset=dataset.infer_objects()
dataset=dataset.reindex(np.random.permutation(len(dataset))
)dataset.reset_index(drop=True,inplace=True)
dataset.iloc[:int(train_num),:-
1].to_csv('potato_train_data.csv',index=False)dataset.iloc[:int(train_num):,[-
1]].to_csv('potato_train_label.csv',index=False)

defvisualize(dataset,labels,features,classes,fig_size=(10,10),layout=None):plt.figure(figsize=
    fig_size)
    index=1
    iflayout==None:
        layout=[len(features),1]
    foriinrange(len(features)):
```

```
        forjinrange(i+1,len(features)):
p=plt.subplot(layout[0],layout[1],index)plt.subplots_
adjust(hspace=0.4)p.set_title(features[i]+'&'+featur
es[j])p.set_xlabel(features[i])p.set_ylabel(features[
j])
forkinrange(len(classes)):
   p.scatter(dataset[labels==k,i],dataset[labels==k,j],label=classes[k])p.legend()
            index+=1
     plt.show()

 dataset
 =pd.read_csv('potato_train_data.csv')labels
 =pd.read_csv('potato_train_label.csv')feature
 s=list(dataset.keys())
 classes     =     np.array(['Corn',     'Potato',
 'grass'])foriinrange(3):
    labels.loc[labels['type']==classes[i],'type']=i

 dataset =
 dataset.valueslabels=labels[
 'type'].values
 visualize(dataset,labels,features,classes)
```

EXPERIMENT-12

Write a program to implement k-Means clustering algorithm to cluster the set of data stored in .CSV file.

SOURCE CODE:

```
fromsklearn.clusterimportKMeansimport
pandasaspd
importnumpyasnpi
mport pickle

#readcsvinputfile
input_data=pd.read_csv("input_data.txt",sep="\t")

 #initializeKMeansobjectspecifyingthenumberofdesiredclusterskmeans
=KMeans(n_clusters=4)

#learningtheclusteringfromtheinputdatek
means.fit(input_data.values)
#outputthelabelsfortheinputdatapr
int(kmeans.labels_)

#predicttheclassificationforgivendatasamplepredicted
_class = kmeans.predict([[1, 10,
15]])print(predicted_class)
```

OUTPUT:

Unnamed=0 unnamed=1 flow report sorted unnamed U by station

U	OBS	STATION	SHIFT	EMPLOYEE	0+	NO.OF ROWS
0	1	Amberst	2	Hyme	1	4
1	2	Goshen	2	Peth	2	4
2	3	Hadley	2	John	3	3
3	4	Holyorce	1	Woxter	4	0
4	5	Holyorce	1	Barb	5	3
5	6	Orange	2	Card	6	5
6	7	Otis	1	Bey	7	0
7	8	Pledom	2	Mike	8	4
8	9	Standard	1	Sam	9	1
9	10	Suttled	2	Lisa	10	1
11	NAN	NAN	NAN	NAN	11	NAN

RESULT: Hence the program has been Succesfully Implemented