PYTHON PROGRAMMING LAB MANUAL

Department of Computer Science & Engineering

VEMU INSTITUTE OF TECHNOLOGY::P.KOTHAKOTA

NEAR PAKALA, CHITTOOR-517112

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuramu)

PYTHON PROGRAMMING LAB MANUAL

Name:_____

H.T.No:_____

Year/Semester:_____

Department of Computer Science & Engineering

VEMU INSTITUTE OF TECHNOLOGY::P.KOTHAKOTA

NEAR PAKALA, CHITTOOR-517112

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuramu)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

Syllabus for (R20 Regulations)

PYTHON PROGRAMMING LAB (20A05101P)

S.NO	NAME OF THE PROGRAM
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 compute summary statistics such as mean, median, mode, standard deviation and variance of the given different types of data.
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.
8	Write a program to demonstrate Regression analysis with residual plots on a given data set.
9	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.
10	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.
11	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.
12	Write a program to implement k-Means clustering algorithm to cluster the set of data stored in.CSV file. Compare the results of various “k” values for the quality of clustering.

PROBLEM SOLVING AND PROGRAMMINGINDEX

GENERAL INSTRUCTIONS FOR LABORATORY CLASSES

DO'S

1. Without Prior permission do not enter into the Laboratory.
2. While entering into the LAB students should wear their ID cards.
3. The Students should come with proper uniform.
4. Students should sign in the LOGIN REGISTER before entering into the laboratory.
5. Students should come with observation and record note book to the laboratory.
6. Students should maintain silence inside the laboratory.
7. After completing the laboratory exercise, make sure to shutdown the system properly

DONT'S

8. Students bringing the bags inside the laboratory..
9. Students wearing slippers/shoes insides the laboratory.
10. Students using the computers in an improper way.
11. Students scribbling on the desk and mishandling the chairs.
12. Students using mobile phones inside the laboratory.
13. Students making noise inside the laboratory.

Program Educational Objectives

The program educational objectives of the Computer Science and Engineering program describe accomplishments that graduates are expected to attain after 3 to 5 years of graduation.

- **PEO1:**To exhibit strong fundamental concepts of Computer Science & Engineering along with advanced knowledge on emerging technologies so that they can devise solutions for real time & social issues.
- **PEO2:**To be employed, to pursue higher studies, to become entrepreneurs and also to have an excellent aptitude for research.
- **PEO3:**To be technically sound, socially acceptable and ethical professionals with global competence.
- **PEO4**:To be young leaders with the capability to lead teams with good communication skills and excellence in social awareness.

Program Specific Outcomes

- **PSO1:**An ability to get an employment in Computer Science and Engineering field and related software industries.
- **PSO2:**An ability to participate in competitive examinations and enable them to pursue higher education.

Program Outcomes

- **PO1:Engineering knowledge:** Apply knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems.
- **PO2:Problem analysis:** Identify, formulate, research literature, and analyses complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- **PO3:Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.
- **PO4:Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and

interpretation of data, and synthesis of the information to provide valid conclusions.

- **PO5:Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- **PO6:The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- **PO7:Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- **PO8:Ethics:**Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- **PO9:Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **PO10:Communication:** Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions
- **PO11:Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- **PO12:Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Lab Programs

1(A).Experiment

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

1(B).Experiment

```
n1=int(input("Enter first number:"))
n2=int(input("Enter second number:"))
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)
```

3.Experiment

#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)
```

4.Experiment

```
#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)
```

5(A).Experiment

```
#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())
```

5(B).Experiment

```
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)
```

6.Experiment

```
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:])
```

```
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))
```

7.Experiment

```
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()
```

8.Experiment

```
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()
```

9.Experiment

```
# 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()
```

10.Experiment

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

a .CSV file.

```
class NaiveBayesClassifier:
def __init__(self, X, y):
"'X and y denotes the features and the target labels respectively"'
self.X, self.y = X, y
self.N = len(self.X) # Length of the training set
self.dim = len(self.X[0]) # Dimension of the vector of features
self.attrs = [[] for _ in range(self.dim)] # Here we'll store the columns of the training set
self.output_dom = {} # Output classes with the number of ocurrences in the training set. In this
case we have only 2 classes
self.data = [] # To store every row [Xi, yi]
for i in range(len(self.X)):
for j in range(self.dim):
# if we have never seen this value for this attr before,
# then we add it to the attrs array in the corresponding position
if not self.X[i][j] in self.attrs[j]:
self.attrs[j].append(self.X[i][j])
# if we have never seen this output class before,
# then we add it to the output_dom and count one occurrence for now
if not self.y[i] in self.output_dom.keys():
self.output_dom[self.y[i]] = 1
# otherwise, we increment the occurrence of this output in the training set by 1
else:
self.output_dom[self.y[i]] += 1
```

```
# store the row
self.data.append([self.X[i], self.y[i]])
def classify(self, entry):
solve = None # Final result
max_arg = -1 # partial maximum
for y in self.output_dom.keys():

prob = self.output_dom[y]/self.N # P(y)
for i in range(self.dim):
cases = [x for x in self.data if x[0][i] == entry[i] and x[1] == y] # all rows with Xi = xi
n = len(cases)
prob *= n/self.N # P *= P(Xi = xi)
# if we have a greater prob for this output than the partial maximum...
if prob > max_arg:
max_arg = prob
solve = y
```

11.Experiment

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

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']
# Data generation
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)', 'Stem diameter(cm)', 'type'])
index = 0
# Natural
for p in config:
    for i in range(int(train_num/3-3)):
        row = []
        for  j, [min_val, max_val] in enumerate(config[p]):
            v = round(np.random.rand()*(max_val-min_val)+min_val, 2)
            while v in dataset[dataset.columns[j]]:
                v = round(np.random.rand()*(max_val-min_val)+min_val, 2)
            row.append(v)
        row.append(p)
        dataset.loc[index] = row
        index +=  1
# Wrong data
for i in range(train_num - index):
    k = np.random.randint(3)
    p = plants[k]
    row = []
    for  j, [min_val, max_val] in enumerate(config[p]):
        v = round(np.random.rand()*(max_val-min_val)+min_val, 2)
        while v in dataset[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] = row
    index+=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)

def visualize(dataset, labels, features, classes, fig_size=(10, 10), layout=None):
    plt.figure(figsize=fig_size)
    index = 1
    if layout == None:
        layout = [len(features), 1]
    for i in range(len(features)):
        for j in range(i+1, len(features)):
            p = plt.subplot(layout[0], layout[1], index)
            plt.subplots_adjust(hspace=0.4)
            p.set_title(features[i]+'&'+features[j])
            p.set_xlabel(features[i])
            p.set_ylabel(features[j])
            for k in range(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')
features = list(dataset.keys())
classes = np.array(['Corn', 'Potato', 'grass'])
for i in range(3):
    labels.loc[labels['type']==classes[i], 'type'] = i
```

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

12.Experiment

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

```
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np
import pickle

# read csv input file
input_data = pd.read_csv("input_data.txt", sep="\t")

# initialize KMeans object specifying the number of desired clusters
kmeans = KMeans(n_clusters=4)

# learning the clustering from the input date
kmeans.fit(input_data.values)
# output the labels for the input data
print(kmeans.labels_)

# predict the classification for given data sample
predicted_class = kmeans.predict([[1, 10, 15]])
print(predicted_class)
```