**Ml LAB**

**1.The probability that it is Friday and that a student is absent is 3 %. Since there are 5 school days in a week, the probability that it is Friday is 20 %. What is the probability that a student is absent given that today is Friday? Apply Baye’s rule in python to get the result. (Ans: 15%)**

**Program:**

'''

**Procedure:**

Since we are given that

Probability that it is Friday and that a student is absent = 0.03

P(F∩A)= 0.03

Probability it is Friday = 0.2 = P(F)

We need to find the probability that a student is absent given that today is Friday.

So, We will use "Conditional Probability":

P(A/F)=P(F∩A)/P(F)

=0.03/0.2 =0.15

Hence, our required probability is 0.15.

'''

fridayandabsent=input("Probability that it is Friday and that a student is absent: ")

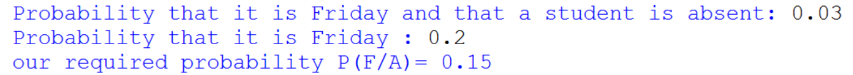
friday=input("Probability that it is Friday : ")

#We need to find the probability that a student is absent given that today is Friday

absentandfriday=float(fridayandabsent)/float(friday)

print("our required probability P(F/A)=",absentandfriday)

**Output:**

****

**2. Extract the data from a csv file using python**

import pandas as pd

data = pd.read\_csv('titanic.csv')

print(data)

**3. Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file .**

**FIND-S Algorithm :-**

1. Initialize h to the most specific hypothesis in H

2. For each positive training instance x

For each attribute constraint a in h

If the constraint ai is satisfied by x

Then do nothing

Else replace ai in h by the next more general constraint that is satisfied by x

3. Output hypothesis h

import pandas as pd

import numpy as np

data = pd.read\_csv('enjoysport.csv')

a = np.array(data.iloc[:])

print(a)

print("\nthe total number of traini9ng instances are : ",len(a))

print(len(a[0]))

num\_att = len(a[0])-1

print("\n The initial hypothesis is : ")

hypo = ['0']\*num\_att

print(hypo)

for i in range(0,len(a)):

  if a[i][num\_att] == 'yes':

    for j in range(0,num\_att):

      if hypo[j] == '0' or hypo[j] == a[i][j] :

        hypo[j] = a[i][j]

      else:

        hypo[j] = '?'

    print("\n The hypothesis for the training instance ",i," is :\n",hypo)

print("\nThe maximumly specific hypothesis for the training instance is :")

print(hypo)

**4. For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples. CANDIDATE-ELIMINATION Learning Algorithm**

**CANDIDATE-ELIMINATION Learning Algorithm :-**

Text

Description automatically generated

import pandas as pd

import numpy as np

data = pd.read\_csv('enjoysport.csv')

concept = np.array(data.iloc[:,0:-1])

target = np.array(data.iloc[:,-1])

print (concept)

print(target)

s\_f,g\_f = learn(concept,target)

print("Final specific hypothesis : ",s\_f,sep="\n")

print("Final general hypothesis : ",g\_f,sep="\n")

def learn(concept,target):

  specific\_h = concept[0].copy()

  print("\nInitilizing of specific \_h and general\_h")

  print(specific\_h)

  general\_h = [["?" for i in range(len(specific\_h))]for i in range(len(specific\_h))]

  print(general\_h)

  for i,h in enumerate(concept):

    print("\n H : ",h,"\n")

    print("for loop starts")

    if target[i] == "yes":

      print("\nif instance is positive")

      for j in range(0,len(specific\_h)):

        if h[j] != specific\_h[j]:

          specific\_h[j] = '?'

          general\_h[j][j] = '?'

    if target[i] == "no" :

       print("\nif instance is negitive")

       for j in range(0,len(specific\_h)):

         if h[j] != specific\_h[j]:

           general\_h[j][j] = specific\_h[j]

         else:

           general\_h[j][j] = '?'

    print("steps of candidate elimination algo ",i+1)

    print(specific\_h)

    print(general\_h)

    print("\n\n")

  indices = [i for i,val in enumerate(general\_h) if val==['?','?','?','?','?','?']]

  for i in indices:

    general\_h.remove(['?','?','?','?','?','?'])

  return specific\_h,general\_h

**5. Write a program to implement Decision Tree Classifier in python using an appropriate data set.**

**Algorithm**

**ID3(Examples, Target\_attribute, Attributes)**

Examples are the training examples. Target\_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given

Examples.

 Create a Root node for the tree

 If all Examples are positive, Return the single-node tree Root, with label = +

 If all Examples are negative, Return the single-node tree Root, with label = -

 If Attributes is empty, Return the single-node tree Root, with label = most common value of Target\_attribute in Examples

 Otherwise Begin

 A ← the attribute from Attributes that best\* classifies Examples

 The decision attribute for Root ← A

 For each possible value, vi, of A,

 Add a new tree branch below Root, corresponding to the test A = vi

 Let Examples vi, be the subset of Examples that have value vi for A

 If Examples vi , is empty

 Then below this new branch add a leaf node with label = most common value of Target\_attribute in Examples

 Else below this new branch add the subtree

ID3(Examples vi, Targe\_tattribute, Attributes – {A}))

 End

 Return Root

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

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

from sklearn import metrics

col\_names = ['pragnant','glucose','bp','skin','insulin','bmi','pedigree','age','label']

pima = pd.read\_csv('pima-indians-diabetes.csv',header=None,names=col\_names)

print(pima.head())

feature\_col = ['pragnant','glucose','bp','insulin','bmi','pedigree','age']

x = pima[feature\_col]

y = pima.label

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

clf = DecisionTreeClassifier()

clf = clf.fit(x\_train,y\_train)

y\_pred = clf.predict(x\_test)

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

print("\nTest values")

print(y\_test)

print("\npred values")

print(y\_pred)

print("Accuracy : ",metrics.accuracy\_score(y\_test,y\_pred))

**6.Write a program to implement k-nearest neighbours’ classification in python using an appropriate data set.**

Text

Description automatically generated

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn import datasets

iris = datasets.load\_iris()

x = iris.data

y = iris.target

print('sepal-length','sepal-width','petal-length','petal-width')

print(x)

print('class: 0-Iris-setosa, 1-Iris-versicolour, 2-Iris-Virginica')

print(y)

#to train the model and Nearest neighbour k=5

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

classifier = KNeighborsClassifier(n\_neighbors=5)

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

print(y\_test)

y\_pred = classifier.predict(x\_test)

print("prediction are")

print(y\_pred)

**7. Write a program to implement k-means clustering with 3 means (i.e., 3 centroids) in python using an appropriate data set.**

**Text

Description automatically generated with low confidence**

from sklearn.datasets import load\_iris

from sklearn.cluster import KMeans

iris = load\_iris()

print(iris.data)

print(iris.data)

kmeans = KMeans(n\_clusters=3)

KMmodel = kmeans.fit(iris.data)

print(KMmodel.labels\_)

print(KMmodel.cluster\_centers\_)

import pandas as pd

pd.crosstab(iris.target,KMmodel.labels\_)

**8. Write a program to implement linear regression in python using an appropriate data set.**

**Graphical user interface, text, application, email

Description automatically generated**

**Table

Description automatically generated**

**Letter

Description automatically generated with medium confidence**

**Text

Description automatically generated**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

dataset = pd.read\_csv('Salary\_Data.csv')

print(dataset.head())

X = dataset.iloc[:, :-1].values  #independent variable array

y = dataset.iloc[:,1].values  #dependent variable vector

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=1/3,random\_state=0)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train,y\_train)

y\_pred = regressor.predict(X\_test)

plt.scatter(X\_train, y\_train, color='red') # plotting the observation line

plt.plot(X\_train, regressor.predict(X\_train), color='blue') # plotting the regression line

plt.title("Salary vs Experience (Training set)") # stating the title of the graph

plt.xlabel("Years of experience") # adding the name of x-axis

plt.ylabel("Salaries") # adding the name of y-axis

plt.show() # specifies end of graph

plt.scatter(X\_test, y\_test, color='red')

plt.plot(X\_train, regressor.predict(X\_train), color='blue') # plotting the regression line

plt.title("Salary vs Experience (Testing set)")

plt.xlabel("Years of experience")

plt.ylabel("Salaries")

plt.show()