VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT
on

MACHINE LEARNING (20CS6PCMAL)

Submitted by

Revanth.S(1BM19CS128)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)

BENGALURU-560019 May-2022 to July-2022

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

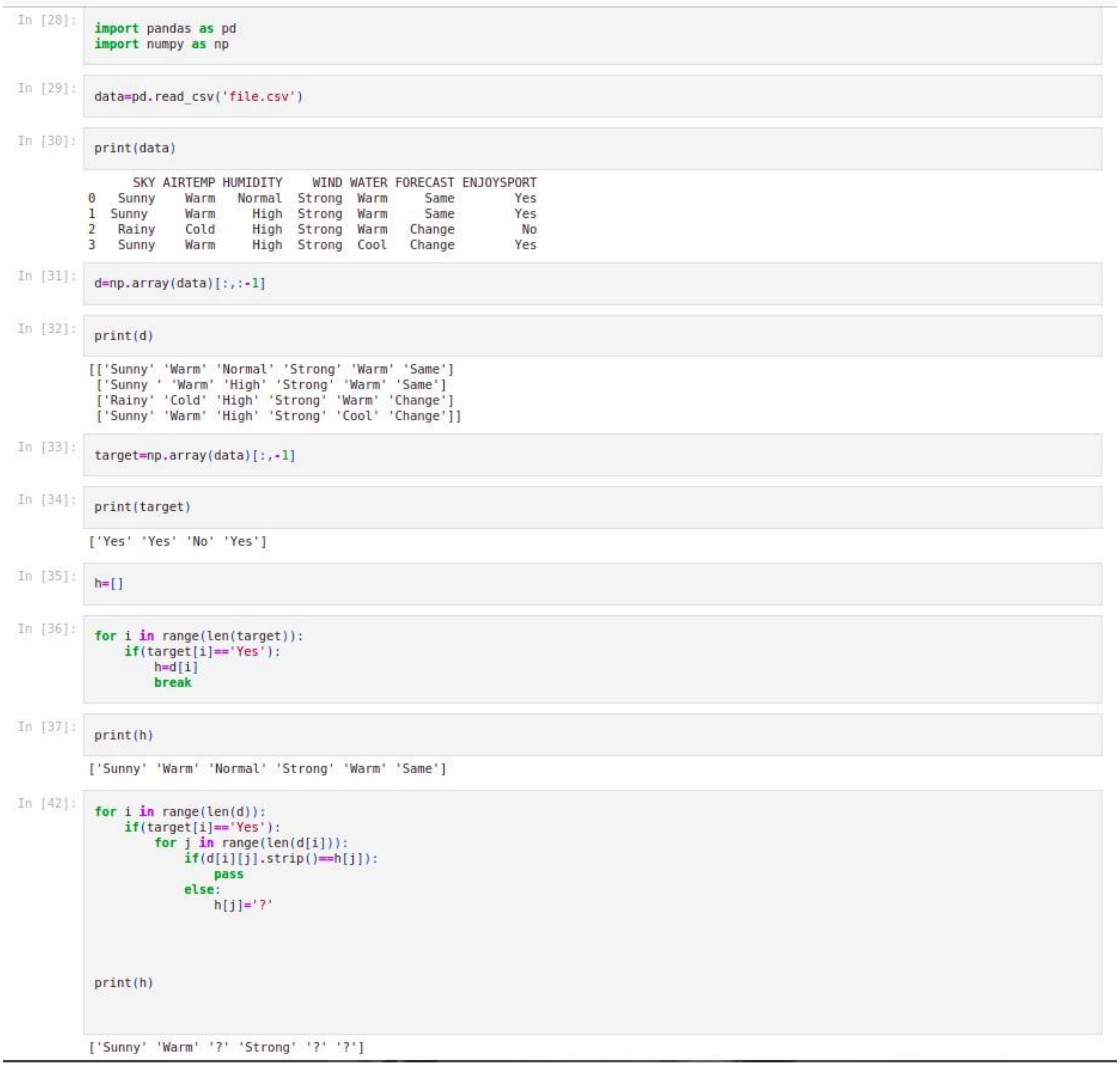
C ERTIFICATE

This is to certify that the Lab work entitled "**MACHINE LEARNING**" carried out by **Revanth.S(1BM19CS128),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of **Machine Learning - (20CS6PCMAL)**work prescribed for the said degree.

Saritha A N	
Assistant Professor	
Nameof the Lab-Incharge	**Dr. Jyothi S Nayak**
Designation	Professor and Head
Department of CSE	Department of CSE
BMSCE, Bengaluru	BMSCE, Bengaluru

`

PROGRAM TO IMPLEMENT FIND S ALGORITHM



In [28]:

```
import pandas as pd
import numpy as np
```

In [29]:

```
data=pd.read_csv('file.csv')
```

In [30]:

```
print(data)
```

```
      SKY AIRTEMP HUMIDITY    WIND WATER FORECAST ENJOYSPORT
0   Sunny    Warm   Normal  Strong  Warm     Same        Yes
1  Sunny     Warm     High  Strong  Warm     Same        Yes
2   Rainy    Cold     High  Strong  Warm   Change         No
3   Sunny    Warm     High  Strong  Cool   Change        Yes
```

In [31]:

```
d=np.array(data)[:,:-1]
```

In [32]:

```
print(d)
```

```
[['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
 ['Sunny ' 'Warm' 'High' 'Strong' 'Warm' 'Same']
 ['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
 ['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]
```

In [33]:

```
target=np.array(data)[:,-1]
```

In [34]:

```
print(target)
```

```
['Yes' 'Yes' 'No' 'Yes']
```

In [35]:

```
h=[]
```

In [36]:

```
for i in range(len(target)):
    if(target[i]=='Yes'):
        h=d[i]
        break
```

In [37]:

```
print(h)
```

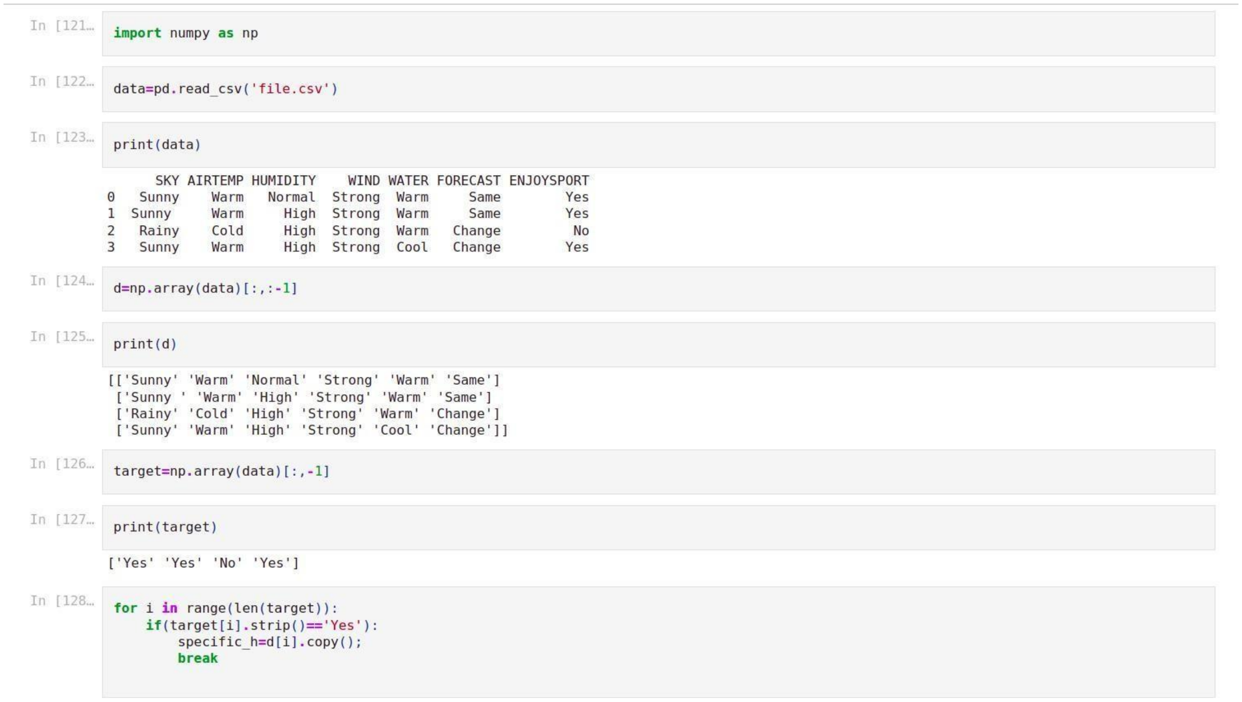
```
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
```

In [42]:

```
for i in range(len(d)):
    if(target[i]=='Yes'):
        for j in range(len(d[i])):
            if(d[i][j].strip()==h[j]):
                pass
            else:
                h[j]='?'



print(h)
```

```
['Sunny' 'Warm' '?' 'Strong' '?' '?']
```

PROGRAM TO IMPLEMENT CANDIDATE ELIMINATION ALGORITHM



In [121…

```
import numpy as np
```

In [122…

```
data=pd.read_csv('file.csv')
```

In [123…

```
print(data)
```

```
      SKY AIRTEMP HUMIDITY    WIND WATER FORECAST ENJOYSPORT
0   Sunny    Warm   Normal  Strong  Warm     Same        Yes
1  Sunny     Warm     High  Strong  Warm     Same        Yes
2   Rainy    Cold     High  Strong  Warm   Change         No
3   Sunny    Warm     High  Strong  Cool   Change        Yes
```

In [124…

```
d=np.array(data)[:,:-1]
```

In [125…

```
print(d)
```

```
[['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
 ['Sunny ' 'Warm' 'High' 'Strong' 'Warm' 'Same']
 ['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
 ['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]
```

In [126…

```
target=np.array(data)[:,-1]
```

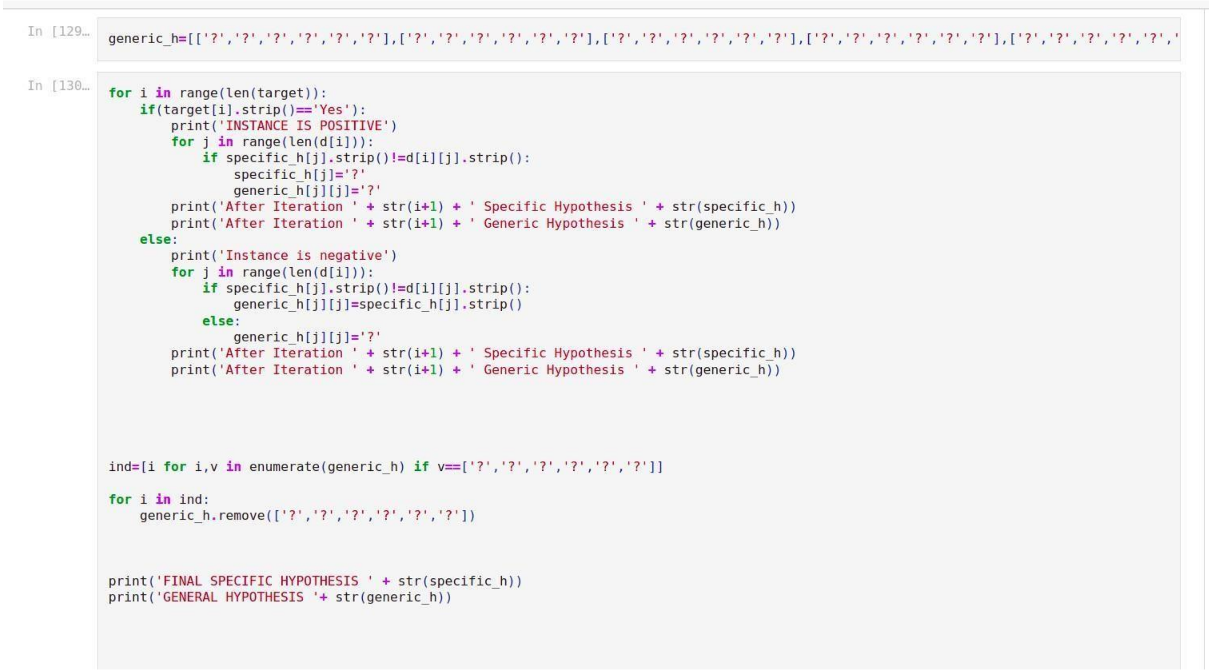
In [127…

```
print(target)
```

```
['Yes' 'Yes' 'No' 'Yes']
```

In [128…

```
for i in range(len(target)):
    if(target[i].strip()=='Yes'):
        specific_h=d[i].copy();
        break
```



In [129…

```
generic_h=[['?','?','?','?','?','?'],['?','?','?','?','?','?'],['?','?','?','?','?','?'],['?','?','?','?','?','?'],['?','?','?','?','?','
```

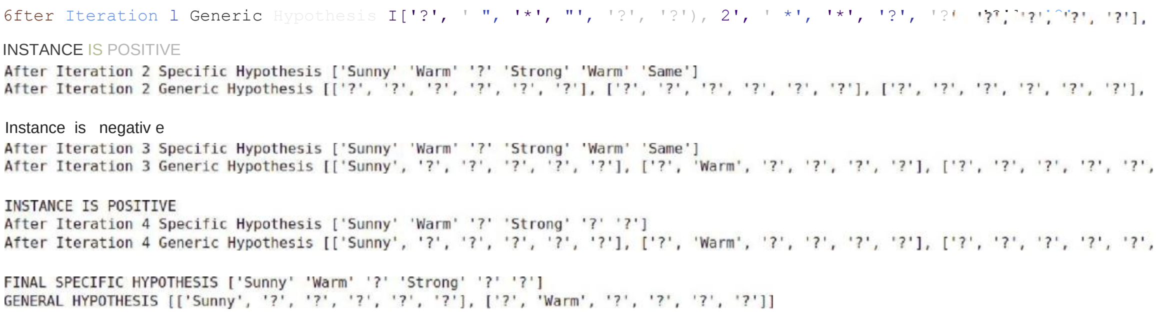
In [130…

```
for i in range(len(target)):
    if(target[i].strip()=='Yes'):
        print('INSTANCE IS POSITIVE')
        for j in range(len(d[i])):
            if specific_h[j].strip()!=d[i][j].strip():
                specific_h[j]='?'
                generic_h[j][j]='?'
        print('After Iteration ' + str(i+1) + ' Specific Hypothesis ' + str(specific_h))
        print('After Iteration ' + str(i+1) + ' Generic Hypothesis ' + str(generic_h))
    else:
        print('Instance is negative')
        for j in range(len(d[i])):
            if specific_h[j].strip()!=d[i][j].strip():
                generic_h[j][j]=specific_h[j].strip()
            else:
                generic_h[j][j]='?'
        print('After Iteration ' + str(i+1) + ' Specific Hypothesis ' + str(specific_h))
        print('After Iteration ' + str(i+1) + ' Generic Hypothesis ' + str(generic_h))




ind=[i for i,v in enumerate(generic_h) if v==['?','?','?','?','?','?']]

for i in ind:
    generic_h.remove(['?','?','?','?','?','?'])



print('FINAL SPECIFIC HYPOTHESIS ' + str(specific_h))
print('GENERAL HYPOTHESIS '+ str(generic_h))
```

print (FZNAL SPECZ FIC HYPOTHES IS ' + st r\
specTfic_h)) print (GENERAL HYPOTHESIS '+ str
(generic h))

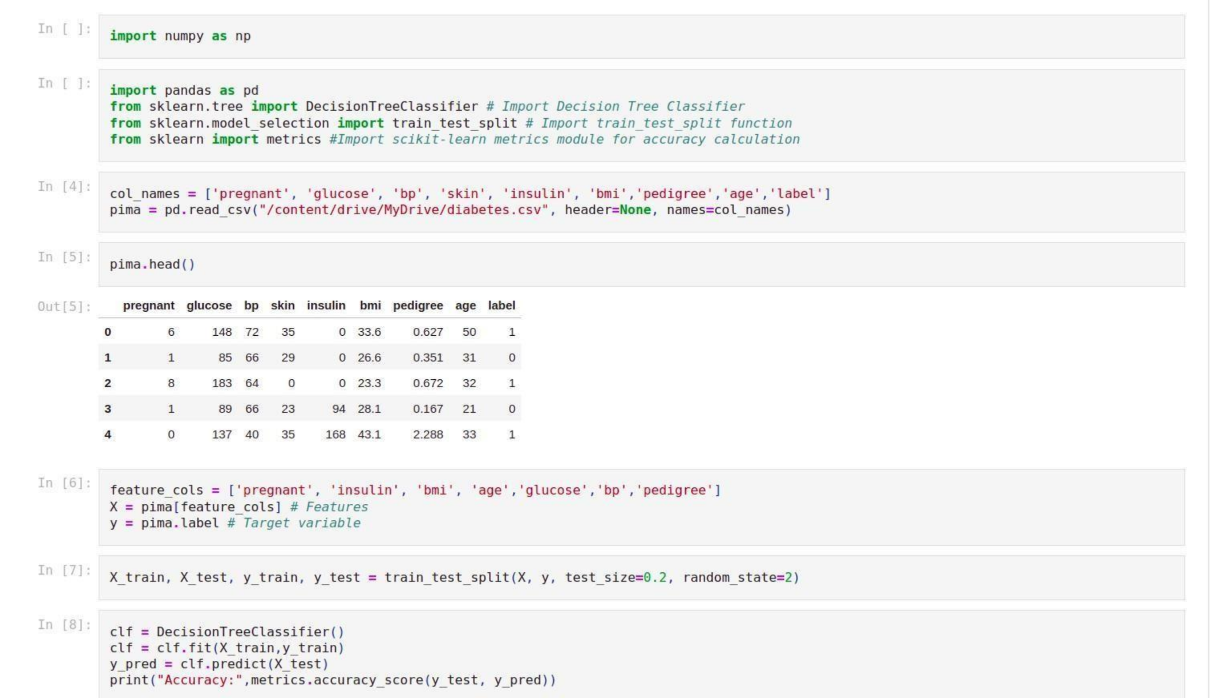
INSTANCE IS POSITIVE

```
After Iteration 1 Specific Hypothesis ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
```



```
6fter Iteration l Generic Hypothesis I['?', ' ", '*', "', '?', '?'), 2', ' *', '*', '?', '?' '?', '?', '?', '?'],

INSTANCE IS POSITIVE
After Iteration 2 Specific Hypothesis ['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
After Iteration 2 Generic Hypothesis [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],

Instance  is   negativ e
After Iteration 3 Specific Hypothesis ['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
After Iteration 3 Generic Hypothesis [['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',

INSTANCE IS POSITIVE
After Iteration 4 Specific Hypothesis ['Sunny' 'Warm' '?' 'Strong' '?' '?']
After Iteration 4 Generic Hypothesis [['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',

FINAL SPECIFIC HYPOTHESIS ['Sunny' 'Warm' '?' 'Strong' '?' '?']
GENERAL HYPOTHESIS [['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?']]
```

PROGRAM TO IMPLEMENT ID-3 ALGORITHM



In [ ]:
```
import numpy as np
```

In [ ]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
```

In [4]:
```
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi','pedigree','age','label']
pima = pd.read_csv("/content/drive/MyDrive/diabetes.csv", header=None, names=col_names)
```

In [5]:
```
pima.head()
```

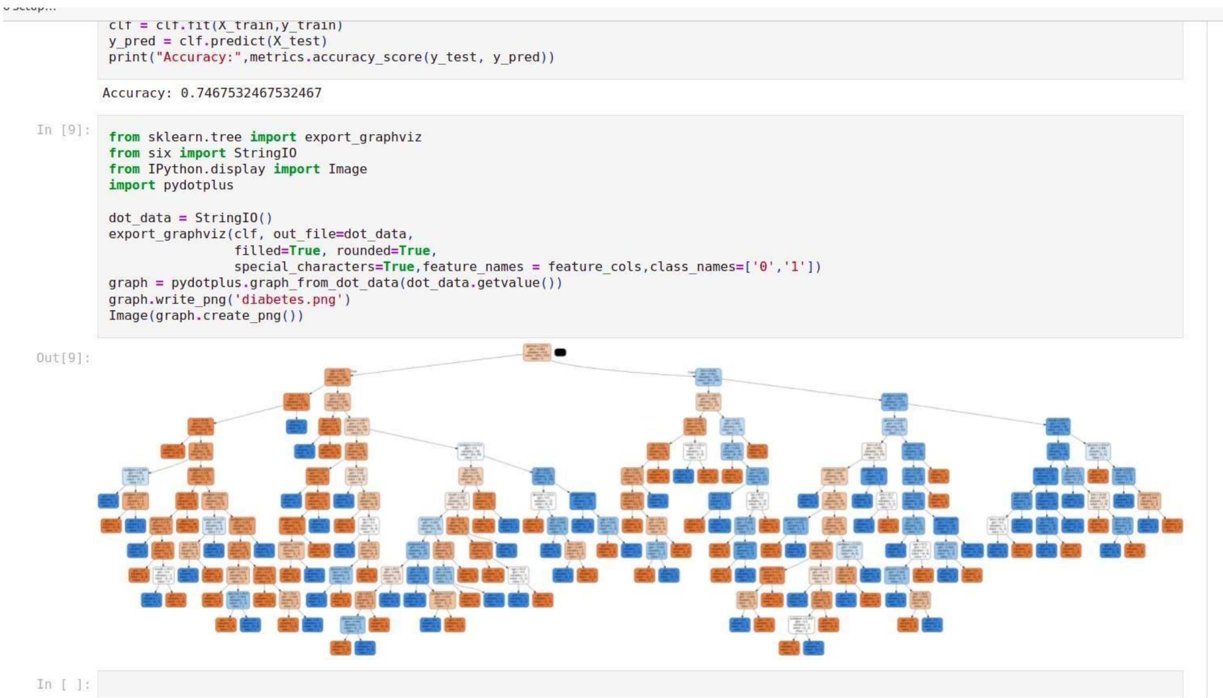
Out[5]:

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	label
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [6]:
```
feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']
X = pima[feature_cols] # Features
y = pima.label # Target variable
```

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

In [8]:
```
clf = DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```



```
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

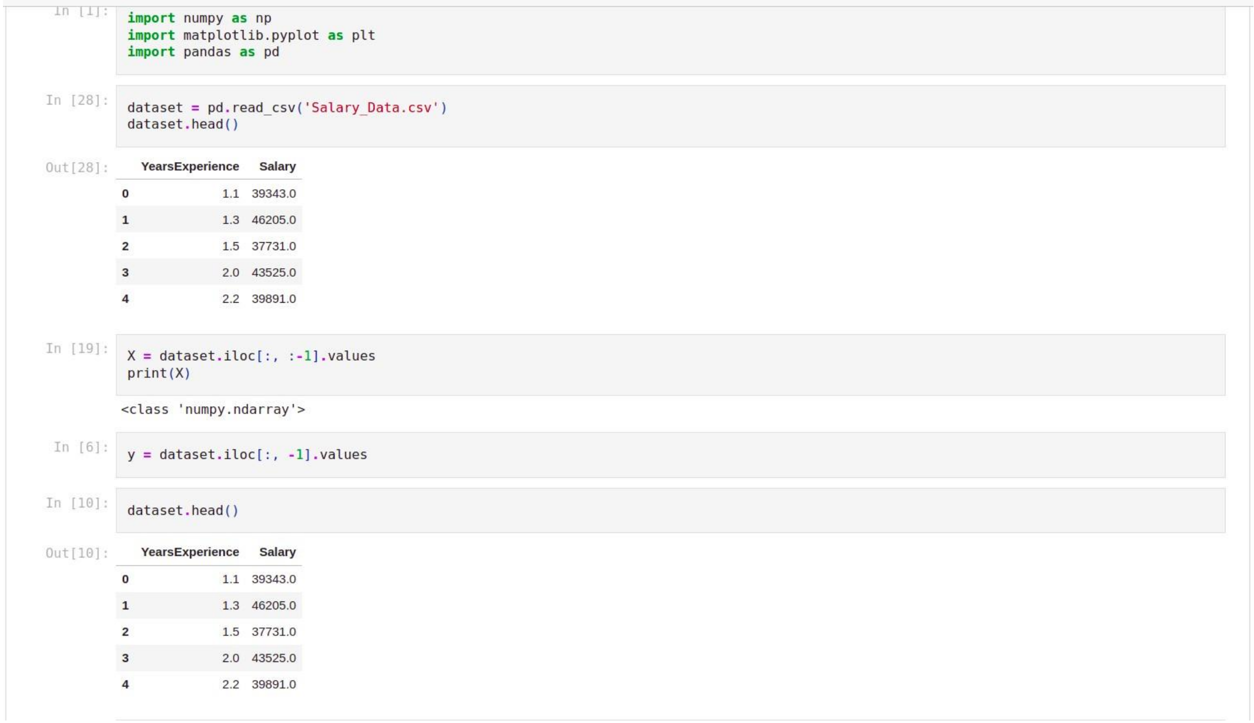
```
Accuracy: 0.7467532467532467
```

In [9]:
```
from sklearn.tree import export_graphviz
from six import StringIO
from IPython.display import Image
import pydotplus

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

Out[9]:

In [ ]:

PROGRAM TO IMPLEMENT LINEAR REGRESSION



```
In [1]:  import numpy as np
         import matplotlib.pyplot as plt
         import pandas as pd
```

```
In [28]: dataset = pd.read_csv('Salary_Data.csv')
         dataset.head()
```

Out[28]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

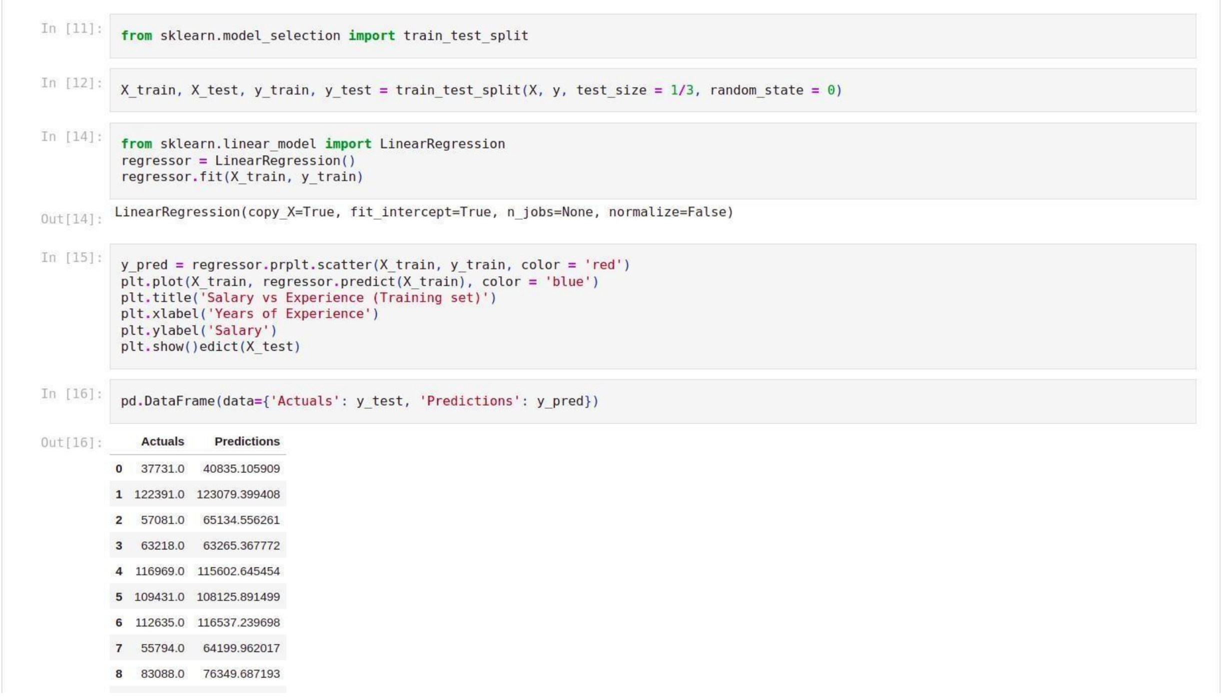
```
In [19]: X = dataset.iloc[:, :-1].values
         print(X)

         <class 'numpy.ndarray'>
```

```
In [6]:  y = dataset.iloc[:, -1].values
```

```
In [10]: dataset.head()
```

Out[10]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0



```
In [11]: from sklearn.model_selection import train_test_split
```

```
In [12]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
```

```
In [14]: from sklearn.linear_model import LinearRegression
         regressor = LinearRegression()
         regressor.fit(X_train, y_train)

Out[14]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
In [15]: y_pred = regressor.prplt.scatter(X_train, y_train, color = 'red')
         plt.plot(X_train, regressor.predict(X_train), color = 'blue')
         plt.title('Salary vs Experience (Training set)')
         plt.xlabel('Years of Experience')
         plt.ylabel('Salary')
         plt.show()edict(X_test)
```

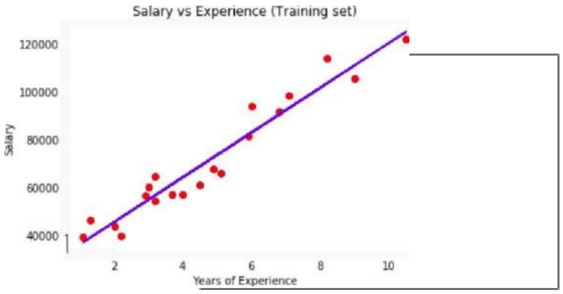
```
In [16]: pd.DataFrame(data={'Actuals': y_test, 'Predictions': y_pred})
```

Out[16]:

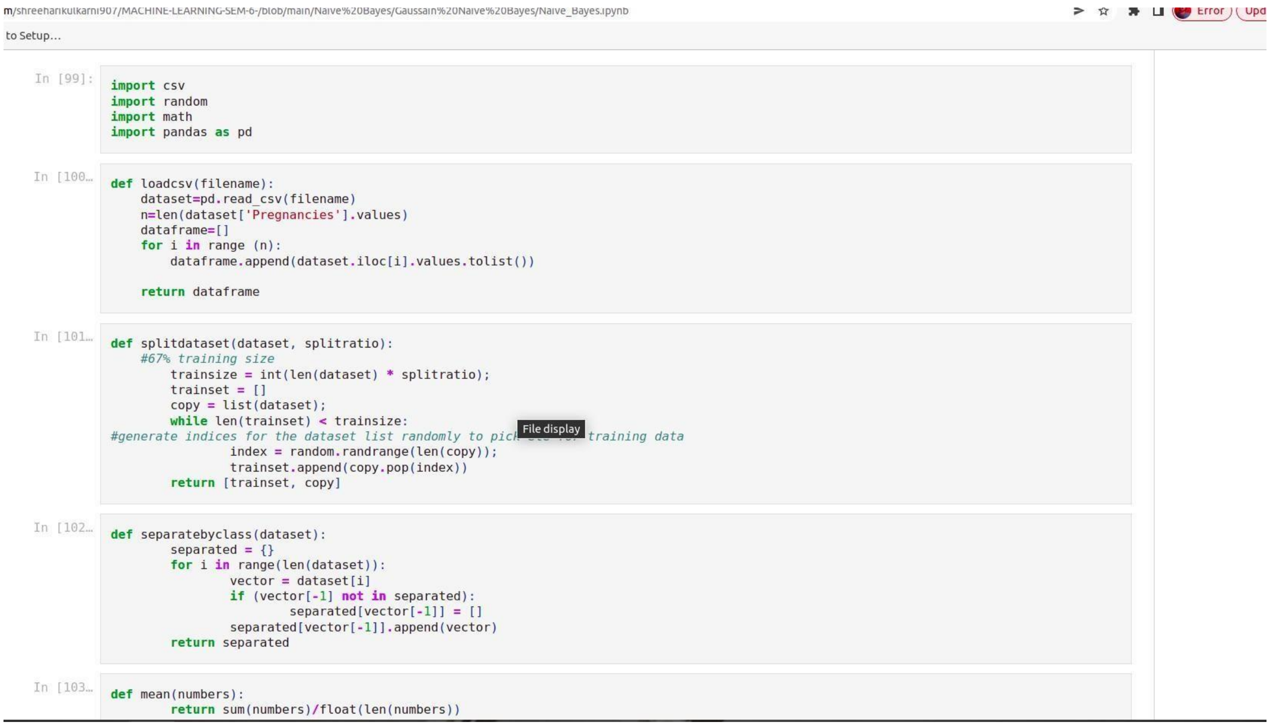
	Actuals	Predictions
0	37731.0	40835.105909
1	122391.0	123079.399408
2	57081.0	65134.556261
3	63218.0	63265.367772
4	116969.0	115602.645454
5	109431.0	108125.891499
6	112635.0	116537.239698
7	55794.0	64199.962017
8	83088.0	76349.687193

```
T

64199T6?0I7 8

8308110 76849 687193
```

```
pit.scatter(X train, y train, color: 're0')

pit.title('Salary vs Experience (Training
set)') pit.xtabel('Years oT Experience')
```


Salary vs Experience (Training set)
Salary
120000
100000
80000
60000
40000
2
4
6
8
10
Years of Experience

PROGRAM TO IMPLEMENT NAIVE BAYES


m/shreeharikulkarni907/MACHINE-LEARNING-SEM-6-/blob/main/Naive%20Bayes/Gaussain%20Naive%20Bayes/Naive_Bayes.ipynb
Error
Upd
to Setup...
File display

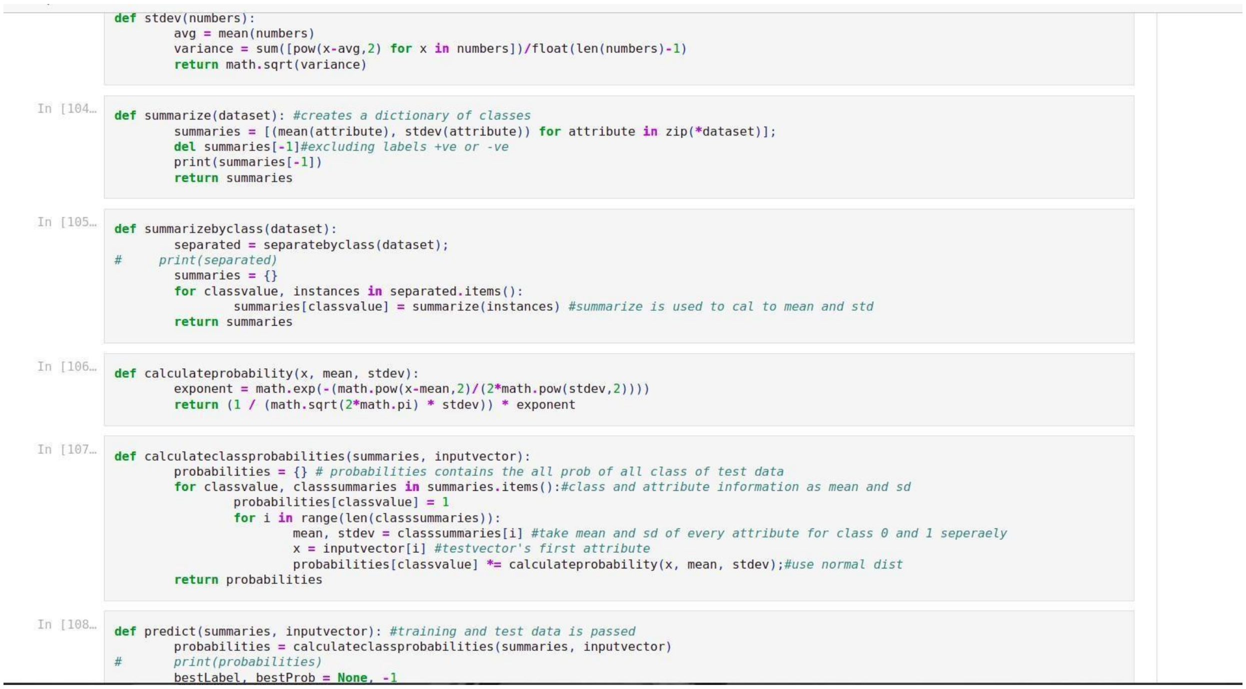
```
In [99]:
import csv
import random
import math
import pandas as pd
```

```
In [100…
def loadcsv(filename):
    dataset=pd.read_csv(filename)
    n=len(dataset['Pregnancies'].values)
    dataframe=[]
    for i in range (n):
        dataframe.append(dataset.iloc[i].values.tolist())

    return dataframe
```

```
In [101…
def splitdataset(dataset, splitratio):
    #67% training size
        trainsize = int(len(dataset) * splitratio);
        trainset = []
        copy = list(dataset);
        while len(trainset) < trainsize:
#generate indices for the dataset list randomly to pick ele for training data
                index = random.randrange(len(copy));
                trainset.append(copy.pop(index))
        return [trainset, copy]
```

```
In [102…
def separatebyclass(dataset):
        separated = {}
        for i in range(len(dataset)):
                vector = dataset[i]
                if (vector[-1] not in separated):
                        separated[vector[-1]] = []
                separated[vector[-1]].append(vector)
        return separated
```

```
In [103…
def mean(numbers):
        return sum(numbers)/float(len(numbers))
```



```
def stdev(numbers):
        avg = mean(numbers)
        variance = sum([pow(x-avg,2) for x in numbers])/float(len(numbers)-1)
        return math.sqrt(variance)
```

```
In [104…
def summarize(dataset): #creates a dictionary of classes
        summaries = [(mean(attribute), stdev(attribute)) for attribute in zip(*dataset)];
        del summaries[-1]#excluding labels +ve or -ve
        print(summaries[-1])
        return summaries
```

```
In [105…
def summarizebyclass(dataset):
        separated = separatebyclass(dataset);
#     print(separated)
        summaries = {}
        for classvalue, instances in separated.items():
                summaries[classvalue] = summarize(instances) #summarize is used to cal to mean and std
        return summaries
```

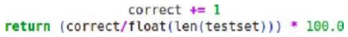
```
In [106…
def calculateprobability(x, mean, stdev):
        exponent = math.exp(-(math.pow(x-mean,2)/(2*math.pow(stdev,2))))
        return (1 / (math.sqrt(2*math.pi) * stdev)) * exponent
```

```
In [107…
def calculateclassprobabilities(summaries, inputvector):
        probabilities = {} # probabilities contains the all prob of all class of test data
        for classvalue, classsummaries in summaries.items():#class and attribute information as mean and sd
                probabilities[classvalue] = 1
                for i in range(len(classsummaries)):
                        mean, stdev = classsummaries[i] #take mean and sd of every attribute for class 0 and 1 seperaely
                        x = inputvector[i] #testvector's first attribute
                        probabilities[classvalue] *= calculateprobability(x, mean, stdev);#use normal dist
        return probabilities
```

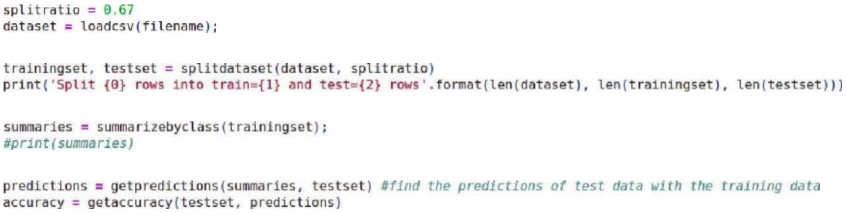
```
In [108…
def predict(summaries, inputvector): #training and test data is passed
        probabilities = calculateclassprobabilities(summaries, inputvector)
#       print(probabilities)
        bestLabel, bestProb = None, -1
```

```
                if bestLabel is None or' probability > bestProb:

                        bestLabel = classvalue



'""" del getpredictions(summaries, testset):
        for i in range(len(testset)):
                result = predict(summaries, testset[i])
                predictions.append(result)
```

```
’"”” dev getaccuracy(testset,        :
          for i in range(len(testset)):
```



```
                     correct += 1
          return (correct/float(len(testset))) * 100.0
```



```
splitratio = 0.67
dataset = loadcsv(filename);

trainingset, testset = splitdataset(dataset, splitratio)
print('Split {0} rows into train={1} and test={2} rows'.format(len(dataset), len(trainingset), len(testset)))

summaries = summarizebyclass(trainingset);
#print(summaries)

predictions = getpredictions(summaries, testset) #find the predictions of test data with the training data
accuracy = getaccuracy(testset, predictions)
```

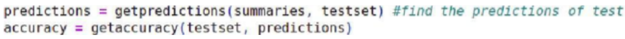
' ' ' “' ' del" get a ecu racy (te st set , p red Action s) :

```
                 i
          ”’  "if’testset[i][-1l’=='predictions[il: returu

          (‹orrert/Iloat(len(testset))) * 100.0 dataset =

          loadcsv(filename),
```

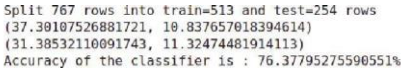
pr 1 p 0} rows*int o't ra in={1} and'test=(2} rows ' . lornat (len (dataset) , ten (I rain ingset) , ten t test set))) *re’* *'6'r*

s unna ries = sumrna rizebyct ass (I rainings et) ;
#p *r:i* n t (sommari est

```
" "' ’ mainC
```



```
predictions = getpredictions(summaries, testset) #find the predictions of test
accuracy = getaccuracy(testset, predictions)
```



```
Split 767 rows into train=513 and test=254 rows
(37.30107526881721, 10.837657018394614)
(31.38532110091743, 11.32474481914113)
Accuracy of the classifier is : 76.37795275590551%
```