VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT
on

Machine Learning
(20CS6PCMAL)

Submitted by

Ravi Sajjanar (1BM19CS127)

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

CERTIFICATE

This is to certify that the Lab work entitled "**Machine Learning**" carried out by **Ravi Sajjanar (1BM19CS127),** 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 a **Machine Learning (20CS6PCMAL)** work prescribed for the said degree.

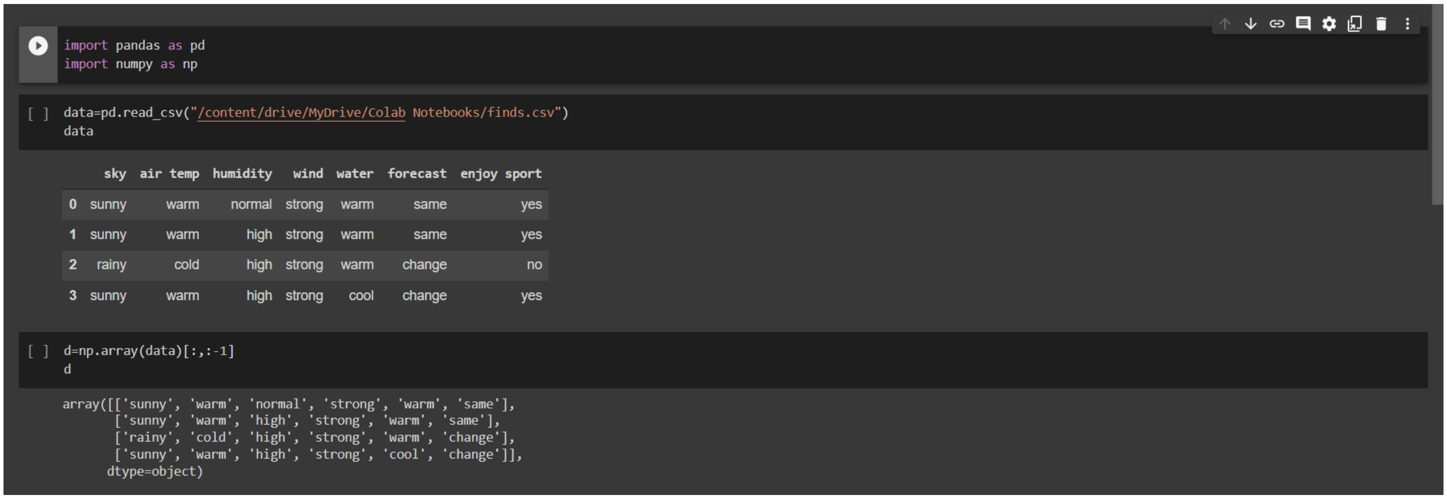
Dr. Asha G R
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

`

Index Sheet

1. Find S Algorithm



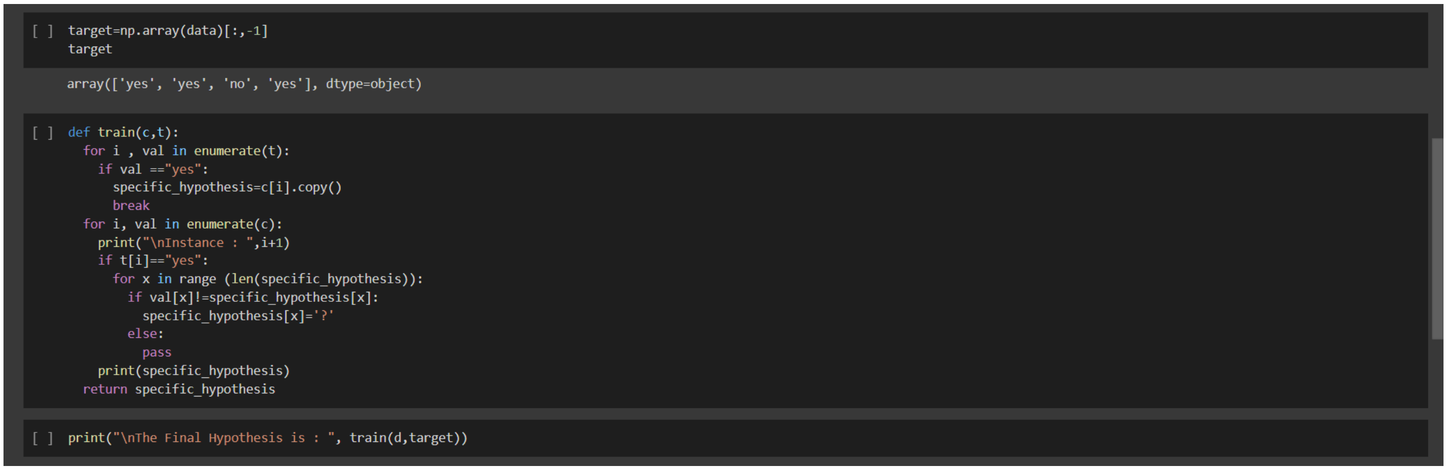
```
import pandas as pd
import numpy as np
```

```
[ ] data=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/finds.csv")
    data
```

	sky	air temp	humidity	wind	water	forecast	enjoy sport
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

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

```
array([['sunny', 'warm', 'normal', 'strong', 'warm', 'same'],
       ['sunny', 'warm', 'high', 'strong', 'warm', 'same'],
       ['rainy', 'cold', 'high', 'strong', 'warm', 'change'],
       ['sunny', 'warm', 'high', 'strong', 'cool', 'change']],
      dtype=object)
```

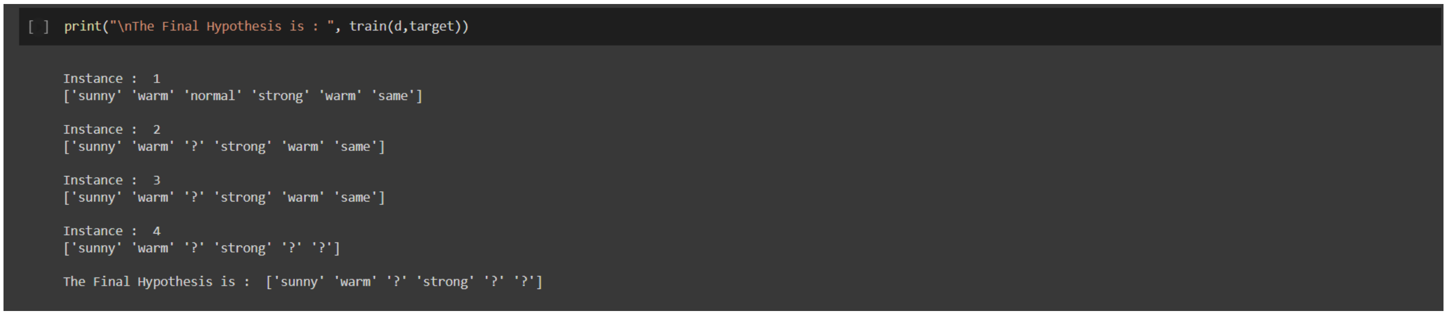


```
[ ] target=np.array(data)[:,-1]
    target
```

```
array(['yes', 'yes', 'no', 'yes'], dtype=object)
```

```
[ ] def train(c,t):
      for i , val in enumerate(t):
        if val =="yes":
          specific_hypothesis=c[i].copy()
          break
      for i, val in enumerate(c):
        print("\nInstance : ",i+1)
        if t[i]=="yes":
          for x in range (len(specific_hypothesis)):
            if val[x]!=specific_hypothesis[x]:
              specific_hypothesis[x]='?'
            else:
              pass
        print(specific_hypothesis)
      return specific_hypothesis
```

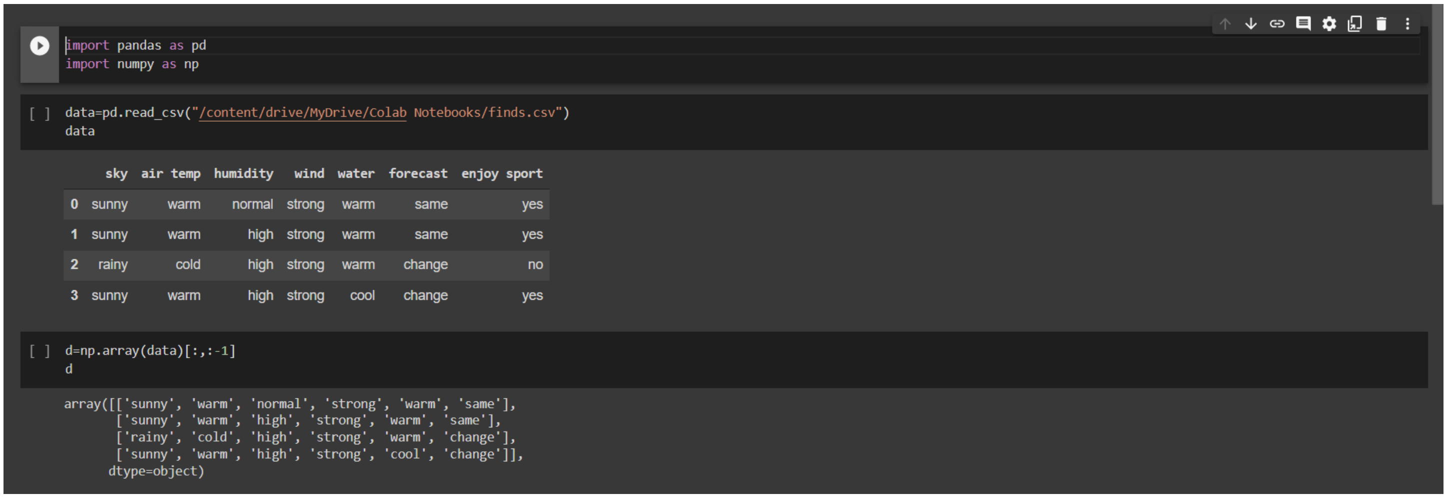
```
[ ] print("\nThe Final Hypothesis is : ", train(d,target))
```



```
[ ] print("\nThe Final Hypothesis is : ", train(d,target))
```

```

Instance :  1
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

Instance :  2
['sunny' 'warm' '?' 'strong' 'warm' 'same']

Instance :  3
['sunny' 'warm' '?' 'strong' 'warm' 'same']

Instance :  4
['sunny' 'warm' '?' 'strong' '?' '?']

The Final Hypothesis is :  ['sunny' 'warm' '?' 'strong' '?' '?']
```

2.Candidate Elimination Algorithm



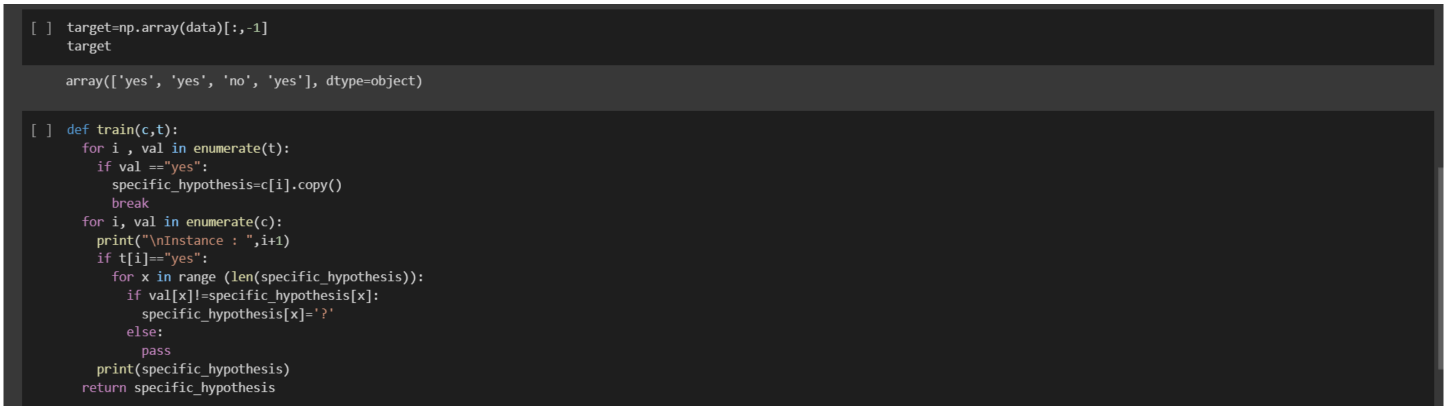
```
import pandas as pd
import numpy as np
```

```
data=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/finds.csv")
data
```

	sky	air temp	humidity	wind	water	forecast	enjoy sport
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

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

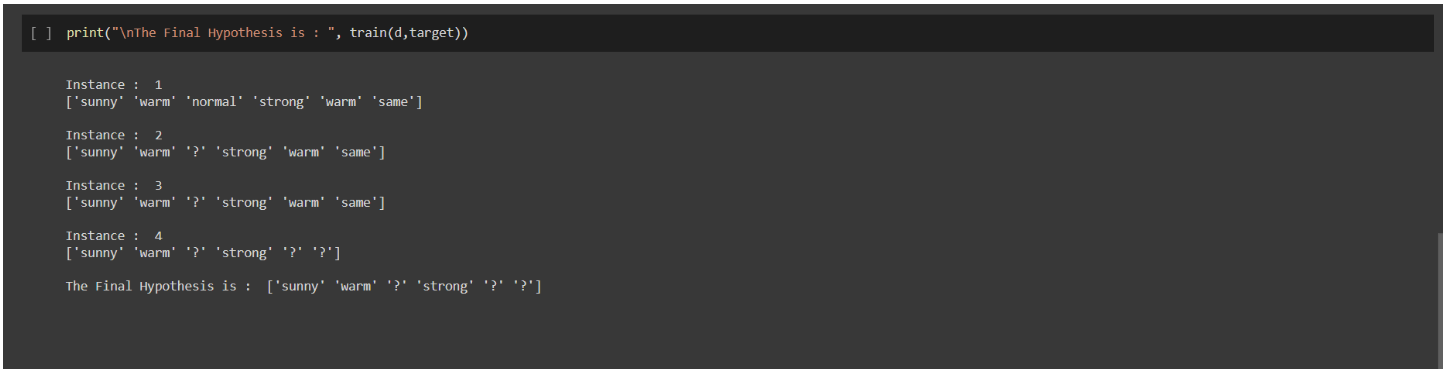
```
array([['sunny', 'warm', 'normal', 'strong', 'warm', 'same'],
       ['sunny', 'warm', 'high', 'strong', 'warm', 'same'],
       ['rainy', 'cold', 'high', 'strong', 'warm', 'change'],
       ['sunny', 'warm', 'high', 'strong', 'cool', 'change']],
      dtype=object)
```



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

```
array(['yes', 'yes', 'no', 'yes'], dtype=object)
```

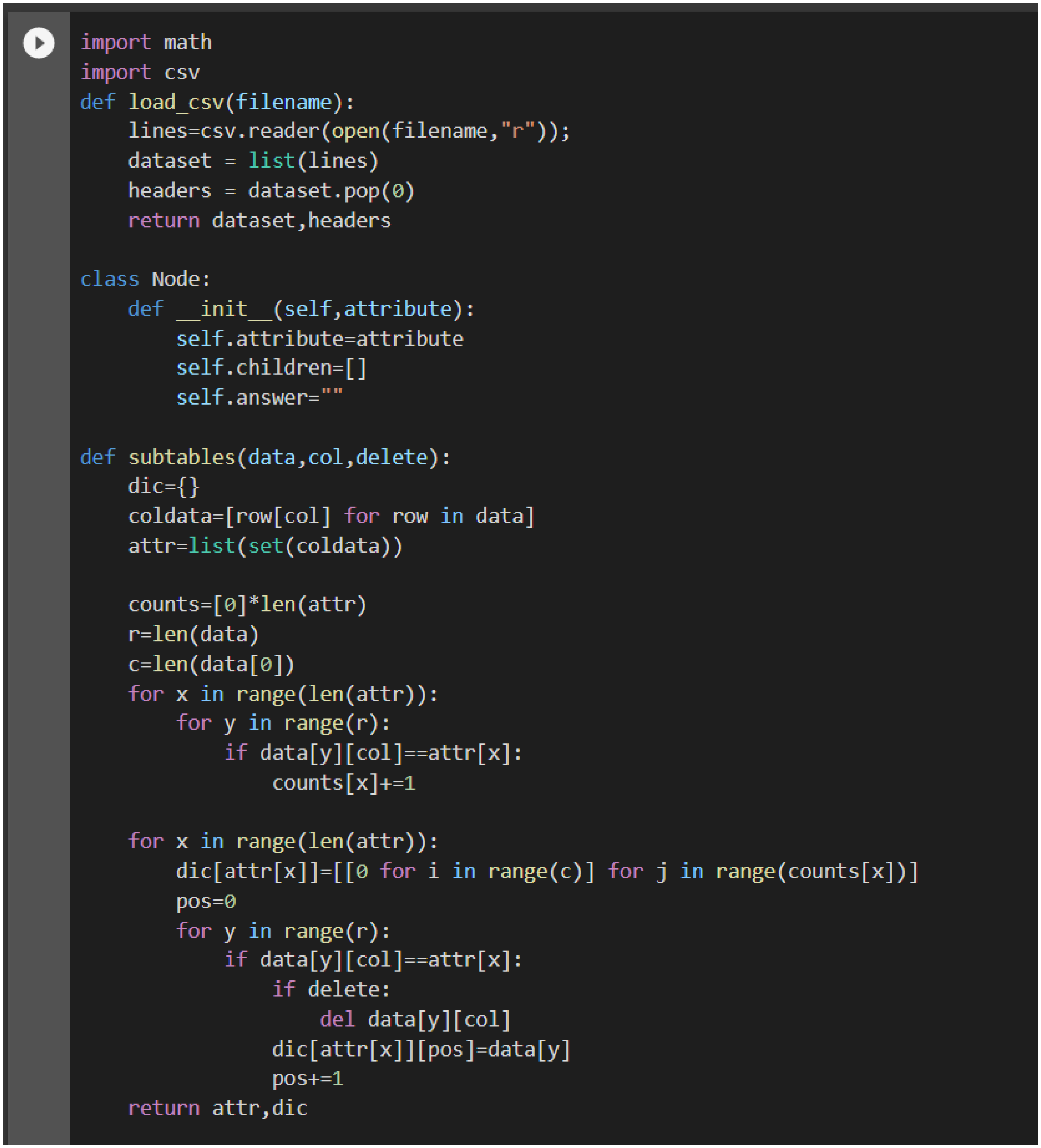
```
def train(c,t):
  for i , val in enumerate(t):
    if val =="yes":
      specific_hypothesis=c[i].copy()
      break
  for i, val in enumerate(c):
    print("\nInstance : ",i+1)
    if t[i]=="yes":
      for x in range (len(specific_hypothesis)):
        if val[x]!=specific_hypothesis[x]:
          specific_hypothesis[x]='?'
        else:
          pass
    print(specific_hypothesis)
  return specific_hypothesis
```



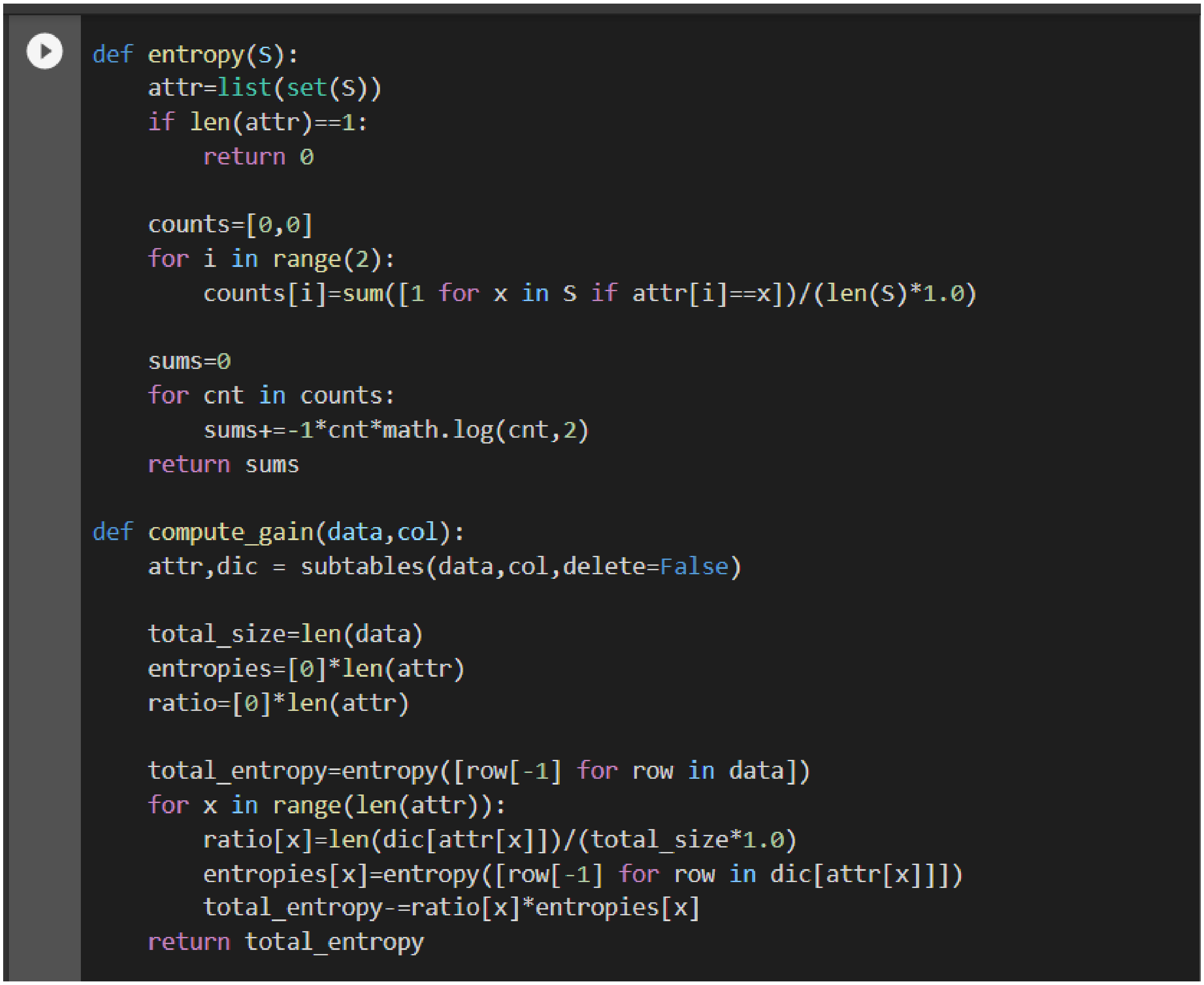
```
print("\nThe Final Hypothesis is : ", train(d,target))
```

```

Instance :  1
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

Instance :  2
['sunny' 'warm' '?' 'strong' 'warm' 'same']

Instance :  3
['sunny' 'warm' '?' 'strong' 'warm' 'same']

Instance :  4
['sunny' 'warm' '?' 'strong' '?' '?']

The Final Hypothesis is :  ['sunny' 'warm' '?' 'strong' '?' '?']
```

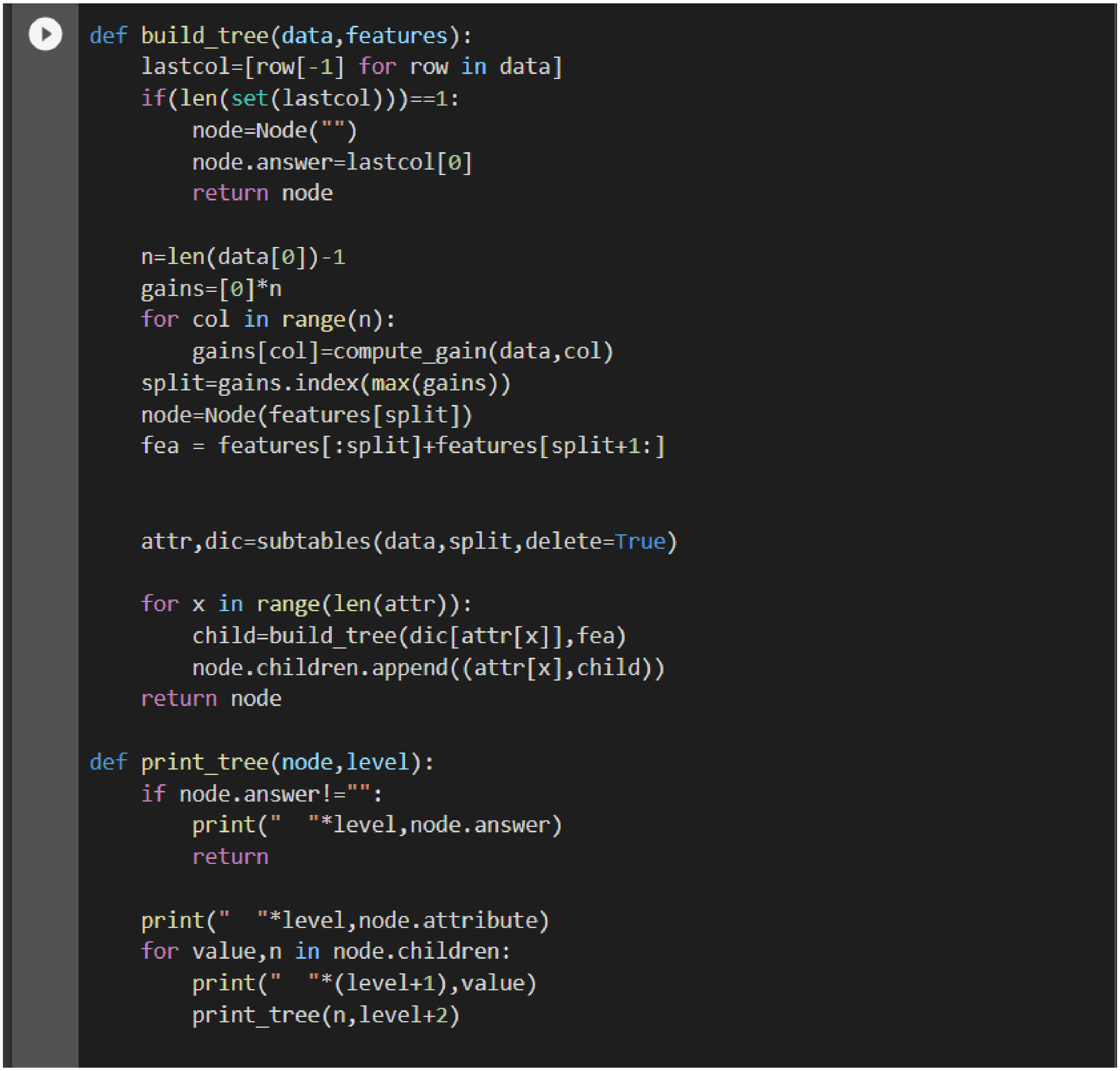
3.ID3 Algorithm (Decision Tree)



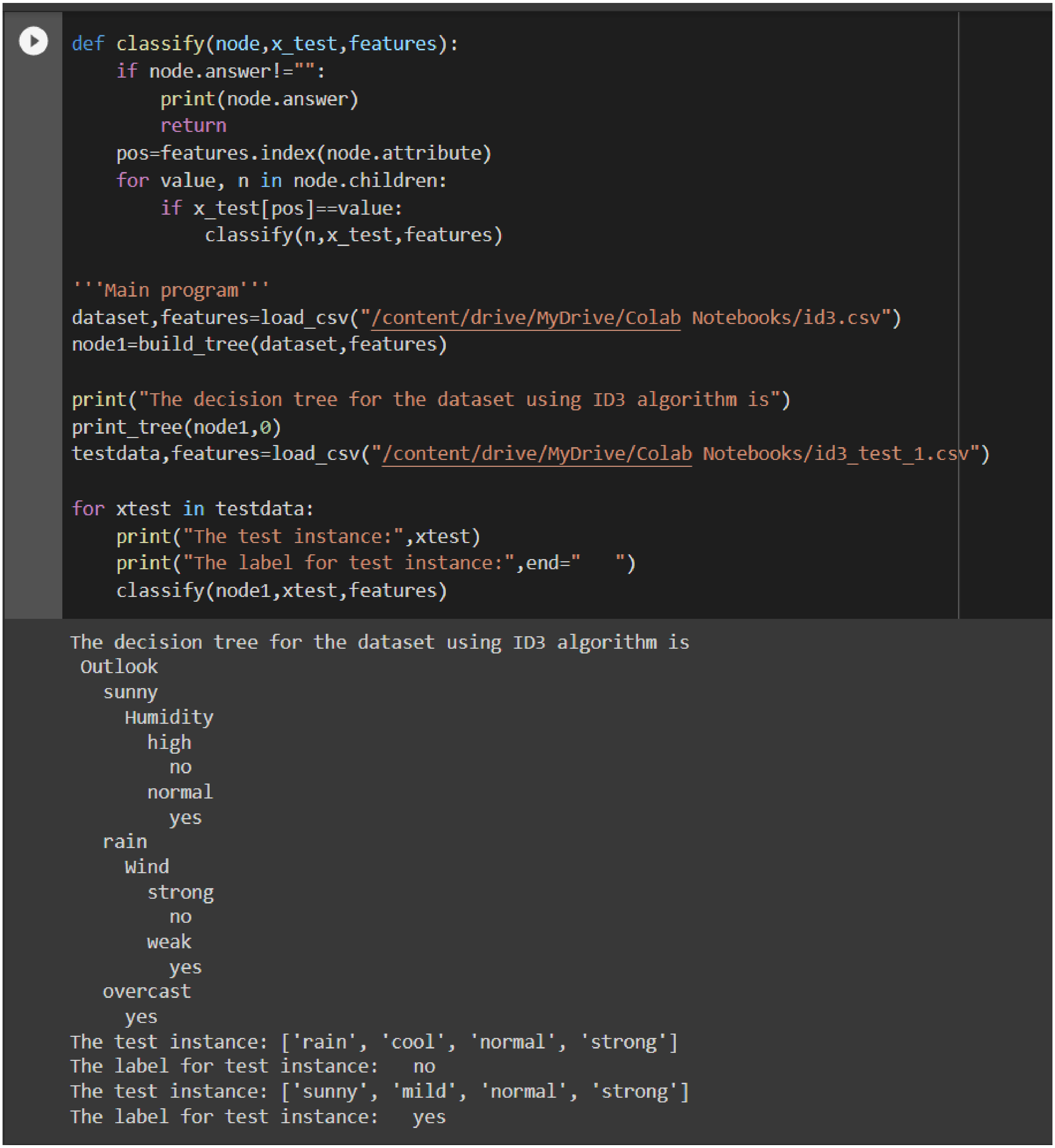
```
import math
import csv
def load_csv(filename):
    lines=csv.reader(open(filename,"r"));
    dataset = list(lines)
    headers = dataset.pop(0)
    return dataset,headers

class Node:
    def __init__(self,attribute):
        self.attribute=attribute
        self.children=[]
        self.answer=""

def subtables(data,col,delete):
    dic={}
    coldata=[row[col] for row in data]
    attr=list(set(coldata))

    counts=[0]*len(attr)
    r=len(data)
    c=len(data[0])
    for x in range(len(attr)):
        for y in range(r):
            if data[y][col]==attr[x]:
                counts[x]+=1

    for x in range(len(attr)):
        dic[attr[x]]=[[0 for i in range(c)] for j in range(counts[x])]
        pos=0
        for y in range(r):
            if data[y][col]==attr[x]:
                if delete:
                    del data[y][col]
                dic[attr[x]][pos]=data[y]
                pos+=1
    return attr,dic
```



```
def entropy(S):
    attr=list(set(S))
    if len(attr)==1:
        return 0

    counts=[0,0]
    for i in range(2):
        counts[i]=sum([1 for x in S if attr[i]==x])/(len(S)*1.0)

    sums=0
    for cnt in counts:
        sums+=-1*cnt*math.log(cnt,2)
    return sums

def compute_gain(data,col):
    attr,dic = subtables(data,col,delete=False)

    total_size=len(data)
    entropies=[0]*len(attr)
    ratio=[0]*len(attr)

    total_entropy=entropy([row[-1] for row in data])
    for x in range(len(attr)):
        ratio[x]=len(dic[attr[x]])/(total_size*1.0)
        entropies[x]=entropy([row[-1] for row in dic[attr[x]]])
        total_entropy-=ratio[x]*entropies[x]
    return total_entropy
```



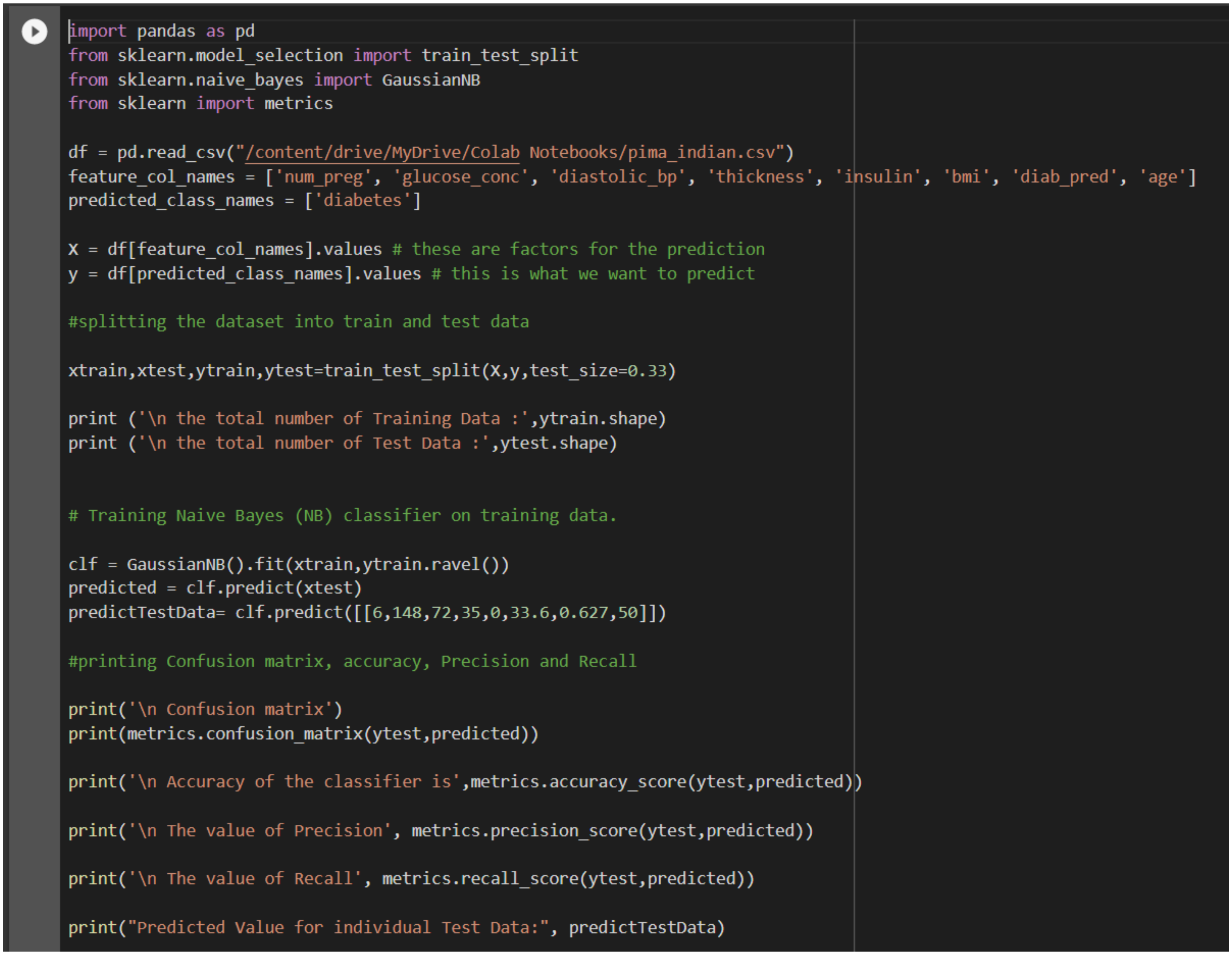
```python
def build_tree(data,features):
    lastcol=[row[-1] for row in data]
    if(len(set(lastcol)))==1:
        node=Node("")
        node.answer=lastcol[0]
        return node

    n=len(data[0])-1
    gains=[0]*n
    for col in range(n):
        gains[col]=compute_gain(data,col)
    split=gains.index(max(gains))
    node=Node(features[split])
    fea = features[:split]+features[split+1:]


    attr,dic=subtables(data,split,delete=True)

    for x in range(len(attr)):
        child=build_tree(dic[attr[x]],fea)
        node.children.append((attr[x],child))
    return node

def print_tree(node,level):
    if node.answer!="":
        print("  "*level,node.answer)
        return

    print("  "*level,node.attribute)
    for value,n in node.children:
        print("  "*(level+1),value)
        print_tree(n,level+2)
```



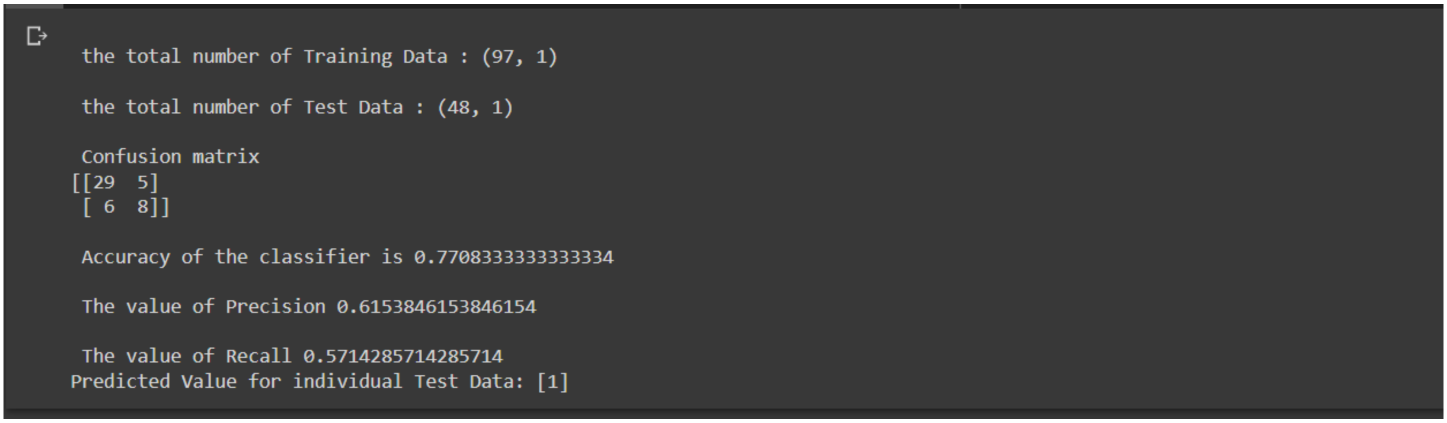
```
def classify(node,x_test,features):
    if node.answer!="":
        print(node.answer)
        return
    pos=features.index(node.attribute)
    for value, n in node.children:
        if x_test[pos]==value:
            classify(n,x_test,features)

'''Main program'''
dataset,features=load_csv("/content/drive/MyDrive/Colab Notebooks/id3.csv")
node1=build_tree(dataset,features)

print("The decision tree for the dataset using ID3 algorithm is")
print_tree(node1,0)
testdata,features=load_csv("/content/drive/MyDrive/Colab Notebooks/id3_test_1.csv")

for xtest in testdata:
    print("The test instance:",xtest)
    print("The label for test instance:",end="   ")
    classify(node1,xtest,features)
```

```
The decision tree for the dataset using ID3 algorithm is
 Outlook
   sunny
     Humidity
       high
         no
       normal
         yes
   rain
     Wind
       strong
         no
       weak
         yes
   overcast
     yes
The test instance: ['rain', 'cool', 'normal', 'strong']
The label for test instance:   no
The test instance: ['sunny', 'mild', 'normal', 'strong']
The label for test instance:   yes
```

4. Naïve Bayesian Classifier

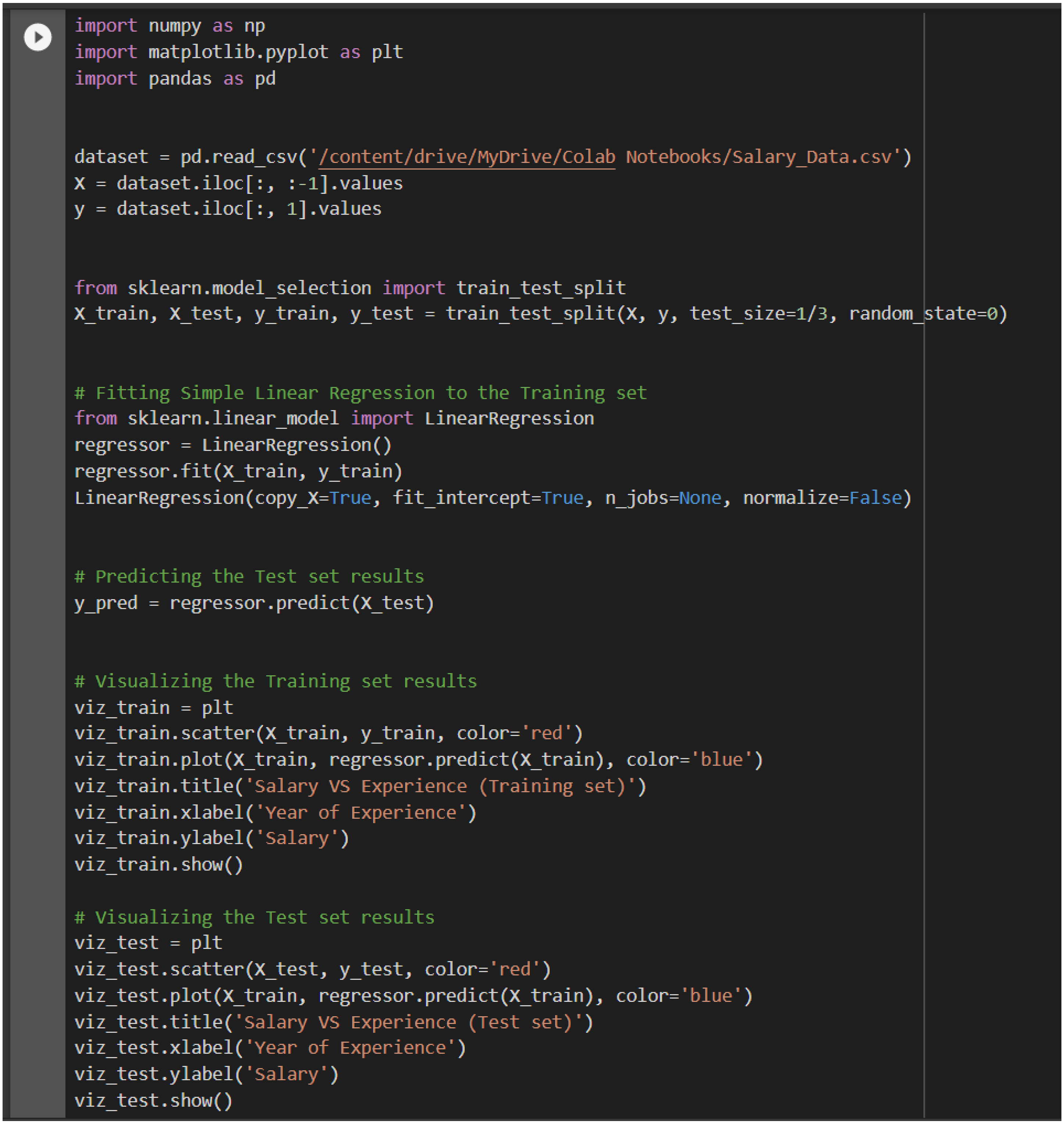


```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics

df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/pima_indian.csv")
feature_col_names = ['num_preg', 'glucose_conc', 'diastolic_bp', 'thickness', 'insulin', 'bmi', 'diab_pred', 'age']
predicted_class_names = ['diabetes']

X = df[feature_col_names].values # these are factors for the prediction
y = df[predicted_class_names].values # this is what we want to predict

#splitting the dataset into train and test data

xtrain,xtest,ytrain,ytest=train_test_split(X,y,test_size=0.33)

print ('\n the total number of Training Data :',ytrain.shape)
print ('\n the total number of Test Data :',ytest.shape)


# Training Naive Bayes (NB) classifier on training data.

clf = GaussianNB().fit(xtrain,ytrain.ravel())
predicted = clf.predict(xtest)
predictTestData= clf.predict([[6,148,72,35,0,33.6,0.627,50]])

#printing Confusion matrix, accuracy, Precision and Recall

print('\n Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))

print('\n Accuracy of the classifier is',metrics.accuracy_score(ytest,predicted))

print('\n The value of Precision', metrics.precision_score(ytest,predicted))

print('\n The value of Recall', metrics.recall_score(ytest,predicted))

print("Predicted Value for individual Test Data:", predictTestData)
```

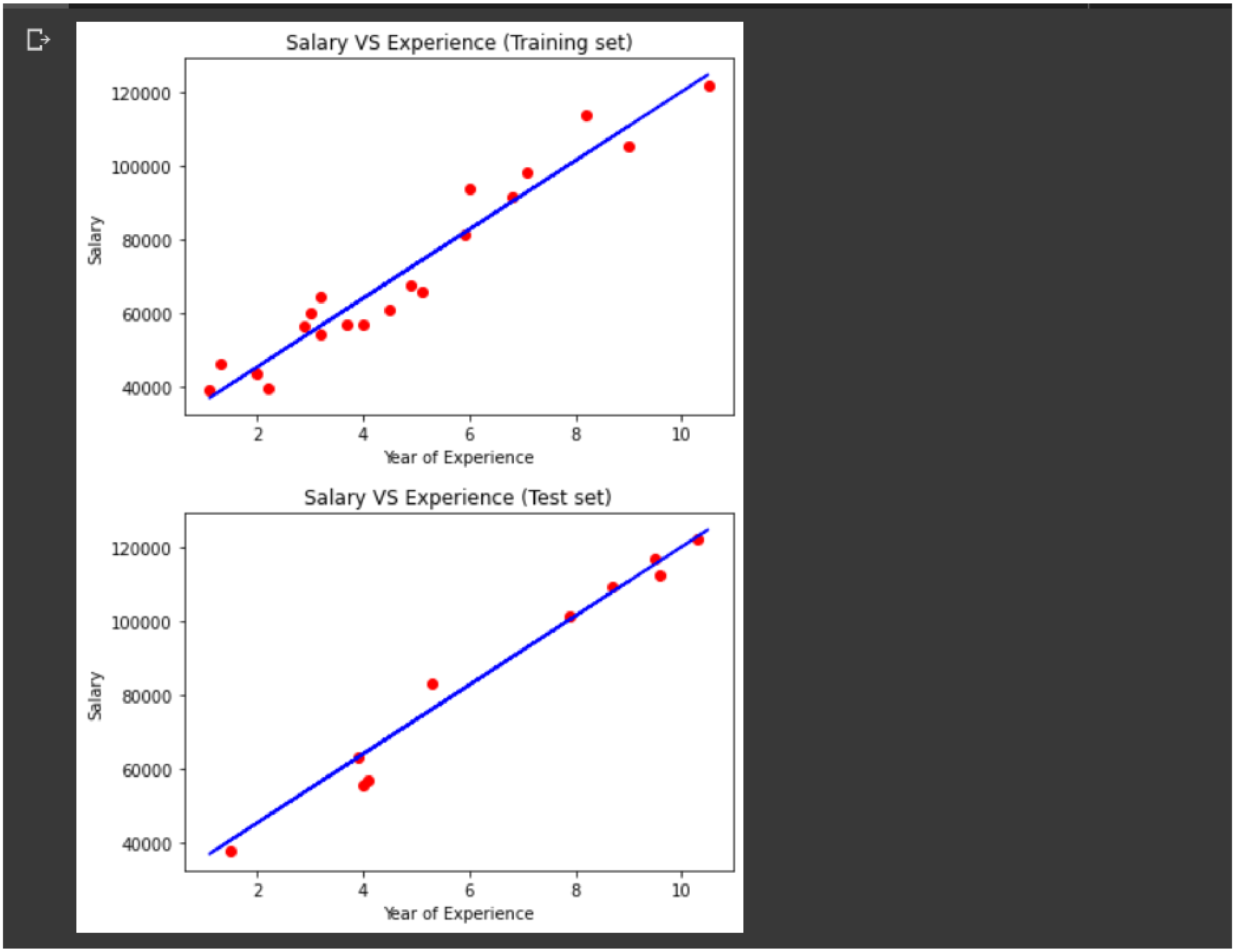


```

 the total number of Training Data : (97, 1)

 the total number of Test Data : (48, 1)

 Confusion matrix
[[29  5]
 [ 6  8]]

 Accuracy of the classifier is 0.7708333333333334

 The value of Precision 0.6153846153846154

 The value of Recall 0.5714285714285714
Predicted Value for individual Test Data: [1]
```

5. Linear Regression



```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd


dataset = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Salary_Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values


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)


# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)


# Predicting the Test set results
y_pred = regressor.predict(X_test)


# Visualizing the Training set results
viz_train = plt
viz_train.scatter(X_train, y_train, color='red')
viz_train.plot(X_train, regressor.predict(X_train), color='blue')
viz_train.title('Salary VS Experience (Training set)')
viz_train.xlabel('Year of Experience')
viz_train.ylabel('Salary')
viz_train.show()

# Visualizing the Test set results
viz_test = plt
viz_test.scatter(X_test, y_test, color='red')
viz_test.plot(X_train, regressor.predict(X_train), color='blue')
viz_test.title('Salary VS Experience (Test set)')
viz_test.xlabel('Year of Experience')
viz_test.ylabel('Salary')
viz_test.show()
```



Salary VS Experience (Training set)

Salary VS Experience (Test set)