VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Machine Learning
(20CS6PCMAL)

Submitted by

KHUSHIL M SINDHWAD
(1BM19CS072)

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 **KHUSHIL M SINDHWAD (1BM19CS072),** 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.

Name of the Lab-Incharge
Designation
Department of CSE
BMSCE, Bengaluru

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

`

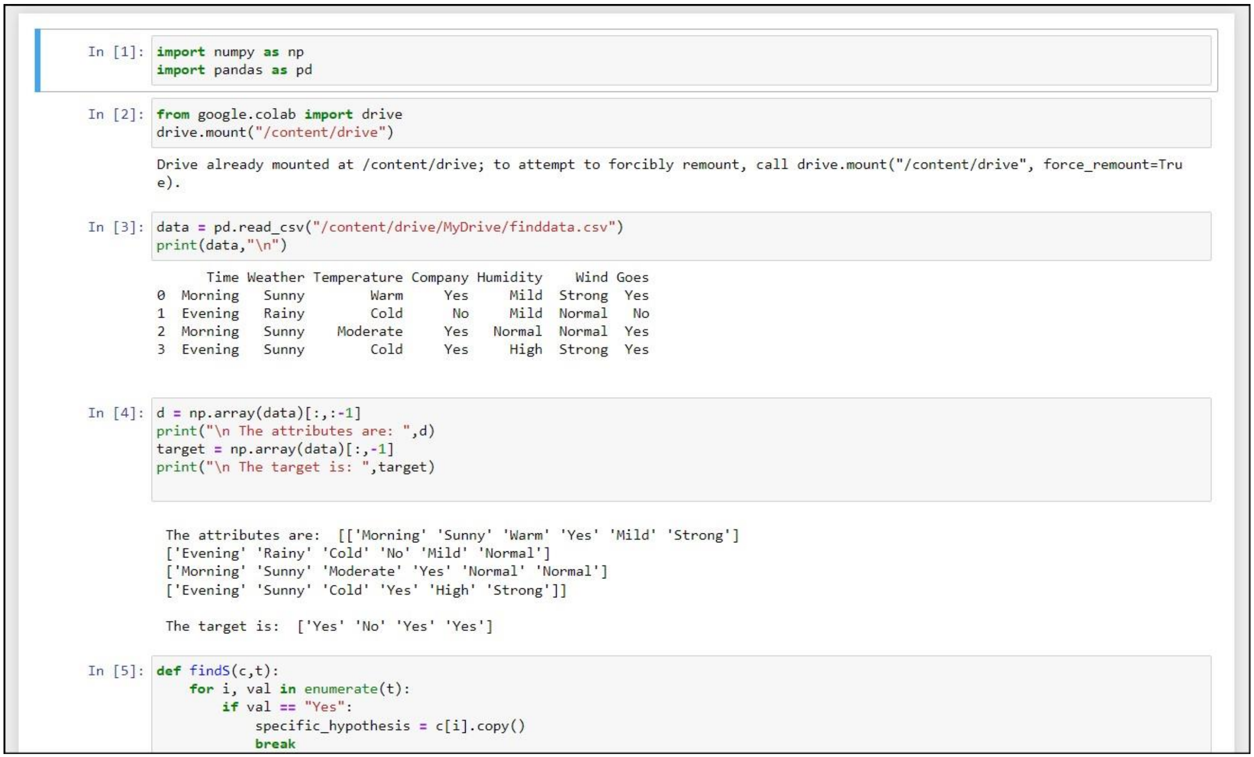
Index Sheet

Course Outcome

CO1	Ability to **apply** the different learning algorithms.
CO2	Ability to **analyse** the learning techniques for given dataset.
CO3	Ability to **design** a model using machine learning to solve a problem.
CO4	Ability to **conduct** practical experiments to solve problems using appropriate machine learning techniques.

1. Find S Algorithm

Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples



```
In [1]: import numpy as np
        import pandas as pd
```

```
In [2]: from google.colab import drive
        drive.mount("/content/drive")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=Tru
e).
```

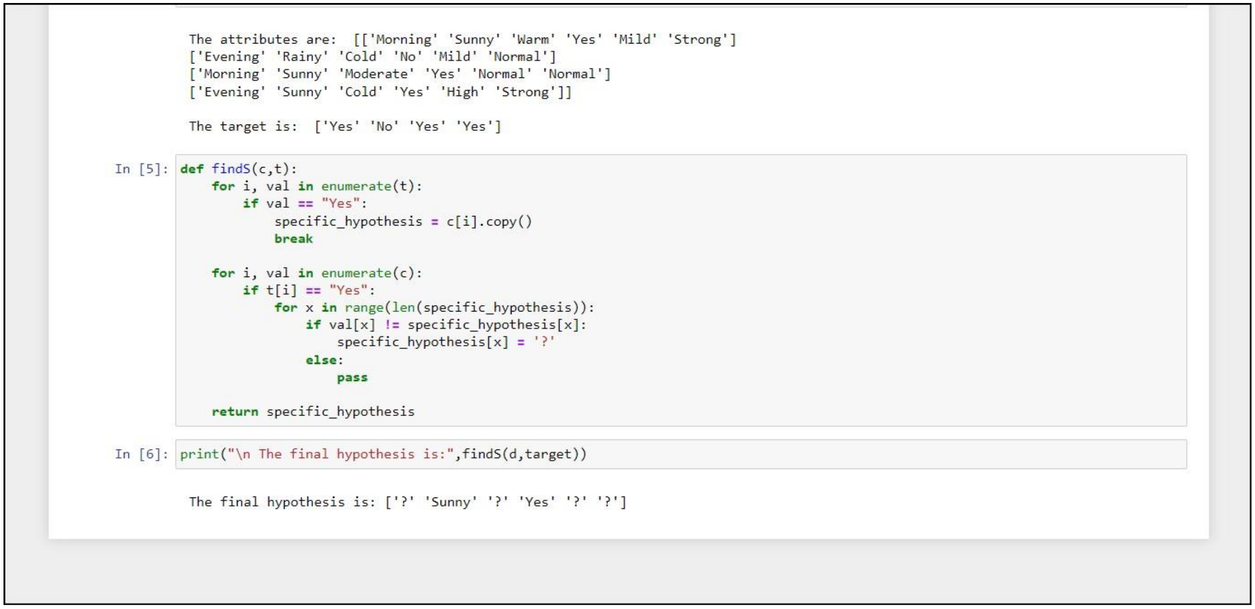
```
In [3]: data = pd.read_csv("/content/drive/MyDrive/finddata.csv")
        print(data,"\n")
```

```
      Time Weather Temperature Company Humidity    Wind Goes
0  Morning   Sunny        Warm     Yes     Mild  Strong  Yes
1  Evening   Rainy        Cold      No     Mild  Normal   No
2  Morning   Sunny    Moderate     Yes   Normal  Normal  Yes
3  Evening   Sunny        Cold     Yes     High  Strong  Yes
```

```
In [4]: d = np.array(data)[:,:-1]
        print("\n The attributes are: ",d)
        target = np.array(data)[:,-1]
        print("\n The target is: ",target)
```

```
 The attributes are:  [['Morning' 'Sunny' 'Warm' 'Yes' 'Mild' 'Strong']
 ['Evening' 'Rainy' 'Cold' 'No' 'Mild' 'Normal']
 ['Morning' 'Sunny' 'Moderate' 'Yes' 'Normal' 'Normal']
 ['Evening' 'Sunny' 'Cold' 'Yes' 'High' 'Strong']]

 The target is:  ['Yes' 'No' 'Yes' 'Yes']
```

```
In [5]: def findS(c,t):
            for i, val in enumerate(t):
                if val == "Yes":
                    specific_hypothesis = c[i].copy()
                    break
```



```
 The attributes are:  [['Morning' 'Sunny' 'Warm' 'Yes' 'Mild' 'Strong']
 ['Evening' 'Rainy' 'Cold' 'No' 'Mild' 'Normal']
 ['Morning' 'Sunny' 'Moderate' 'Yes' 'Normal' 'Normal']
 ['Evening' 'Sunny' 'Cold' 'Yes' 'High' 'Strong']]

 The target is:  ['Yes' 'No' 'Yes' 'Yes']
```

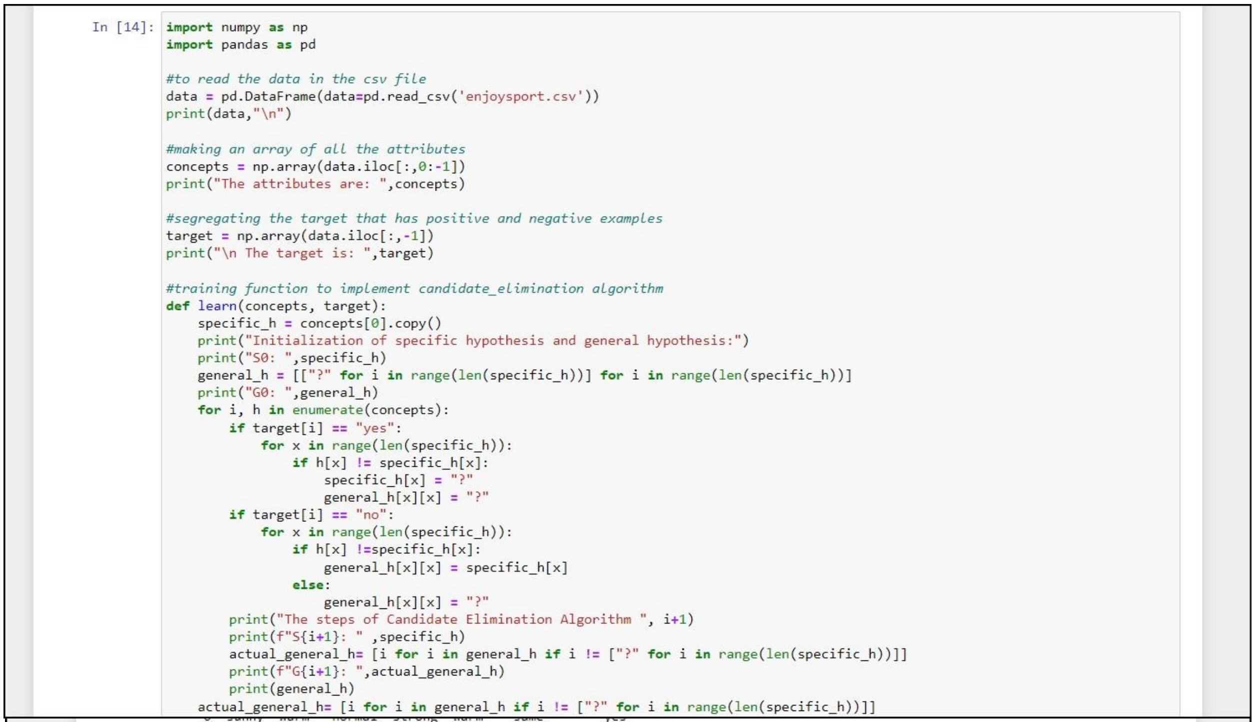
```
In [5]: def findS(c,t):
            for i, val in enumerate(t):
                if val == "Yes":
                    specific_hypothesis = c[i].copy()
                    break

            for i, val in enumerate(c):
                if t[i] == "Yes":
                    for x in range(len(specific_hypothesis)):
                        if val[x] != specific_hypothesis[x]:
                            specific_hypothesis[x] = '?'
                        else:
                            pass

            return specific_hypothesis
```

```
In [6]: print("\n The final hypothesis is:",findS(d,target))
```

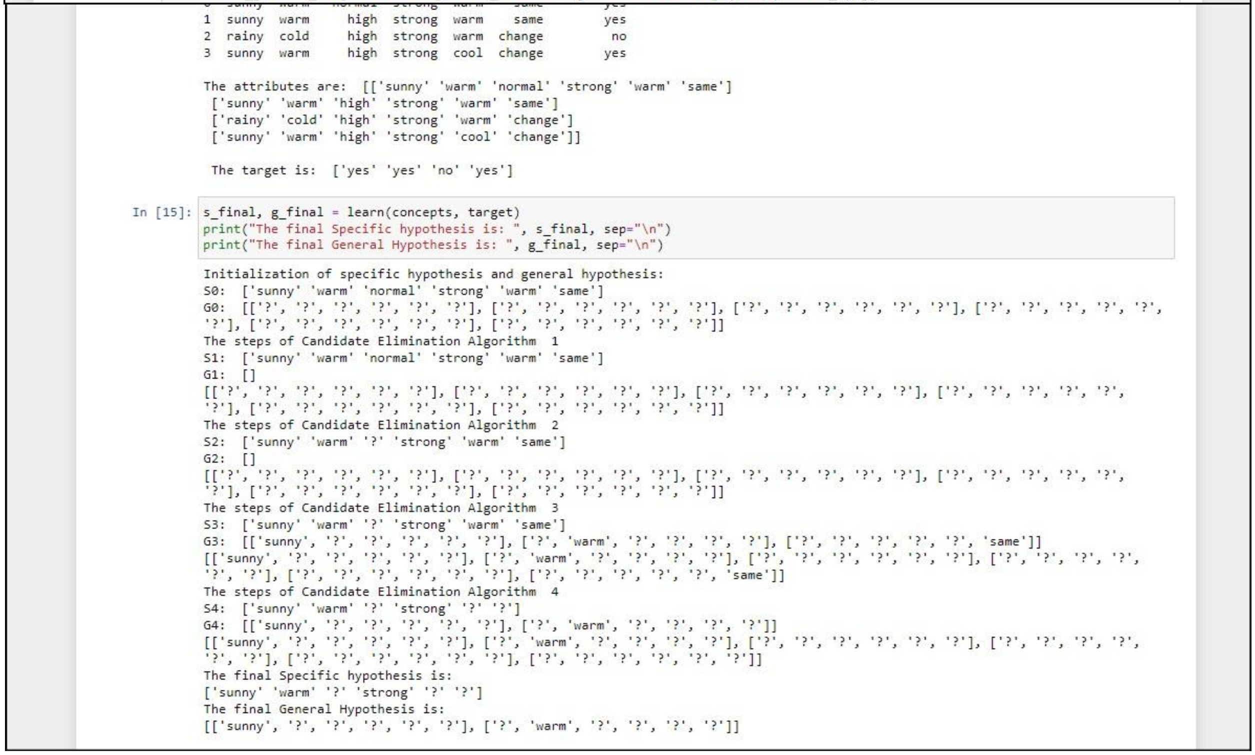
```
 The final hypothesis is: ['?' 'Sunny' '?' 'Yes' '?' '?']
```

2. Candidate Elimination Algorithm

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.



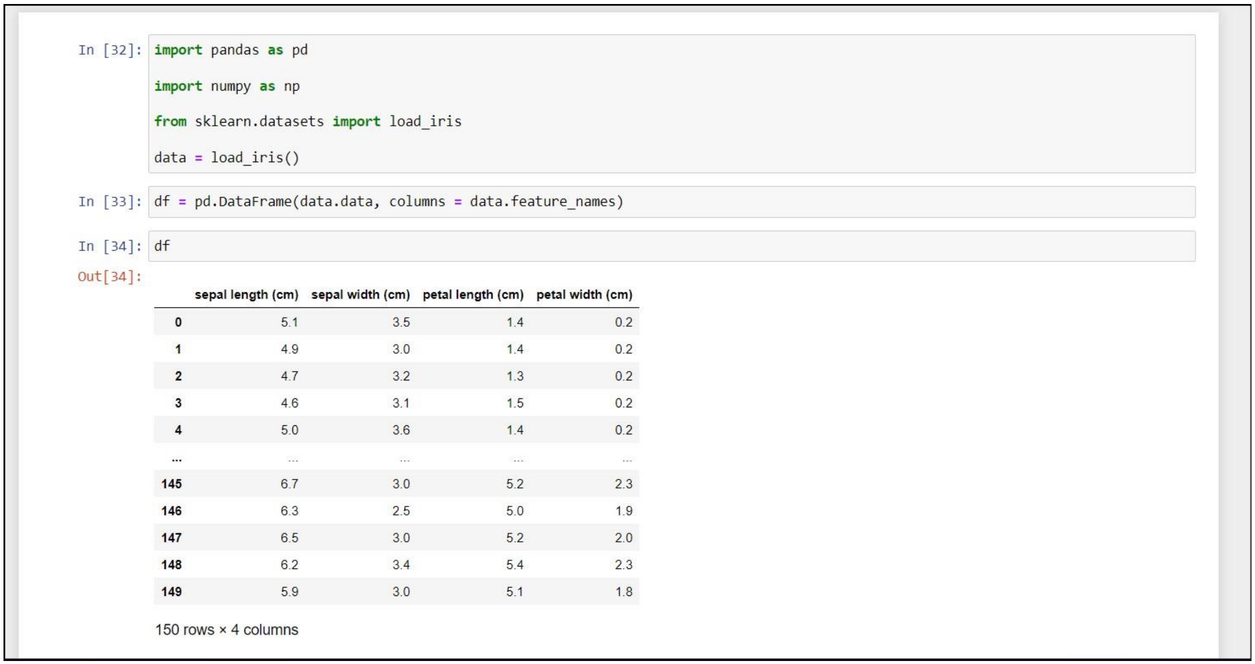
```
In [14]: import numpy as np
         import pandas as pd

         #to read the data in the csv file
         data = pd.DataFrame(data=pd.read_csv('enjoysport.csv'))
         print(data,"\n")

         #making an array of all the attributes
         concepts = np.array(data.iloc[:,0:-1])
         print("The attributes are: ",concepts)

         #segregating the target that has positive and negative examples
         target = np.array(data.iloc[:,-1])
         print("\n The target is: ",target)

         #training function to implement candidate_elimination algorithm
         def learn(concepts, target):
             specific_h = concepts[0].copy()
             print("Initialization of specific hypothesis and general hypothesis:")
             print("S0: ",specific_h)
             general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_h))]
             print("G0: ",general_h)
             for i, h in enumerate(concepts):
                 if target[i] == "yes":
                     for x in range(len(specific_h)):
                         if h[x] != specific_h[x]:
                             specific_h[x] = "?"
                             general_h[x][x] = "?"
                 if target[i] == "no":
                     for x in range(len(specific_h)):
                         if h[x] !=specific_h[x]:
                             general_h[x][x] = specific_h[x]
                         else:
                             general_h[x][x] = "?"
                 print("The steps of Candidate Elimination Algorithm ", i+1)
                 print(f"S{i+1}: " ,specific_h)
                 actual_general_h= [i for i in general_h if i != ["?" for i in range(len(specific_h))]]
                 print(f"G{i+1}: ",actual_general_h)
                 print(general_h)
             actual_general_h= [i for i in general_h if i != ["?" for i in range(len(specific_h))]]
```



```
1  sunny  warm     high  strong  warm    same        yes
2  rainy  cold     high  strong  warm  change         no
3  sunny  warm     high  strong  cool  change        yes

The attributes are:  [['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
 ['sunny' 'warm' 'high' 'strong' 'warm' 'same']
 ['rainy' 'cold' 'high' 'strong' 'warm' 'change']
 ['sunny' 'warm' 'high' 'strong' 'cool' 'change']]

 The target is:  ['yes' 'yes' 'no' 'yes']

In [15]: s_final, g_final = learn(concepts, target)
         print("The final Specific hypothesis is: ", s_final, sep="\n")
         print("The final General Hypothesis is: ", g_final, sep="\n")

Initialization of specific hypothesis and general hypothesis:
S0:  ['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
G0:  [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
The steps of Candidate Elimination Algorithm  1
S1:  ['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
G1:  []
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
The steps of Candidate Elimination Algorithm  2
S2:  ['sunny' 'warm' '?' 'strong' 'warm' 'same']
G2:  []
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
The steps of Candidate Elimination Algorithm  3
S3:  ['sunny' 'warm' '?' 'strong' 'warm' 'same']
G3:  [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'same']]
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'same']]
The steps of Candidate Elimination Algorithm  4
S4:  ['sunny' 'warm' '?' 'strong' '?' '?']
G4:  [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
The final Specific hypothesis is:
['sunny' 'warm' '?' 'strong' '?' '?']
The final General Hypothesis is:
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]
```

3. ID3 Algorithm

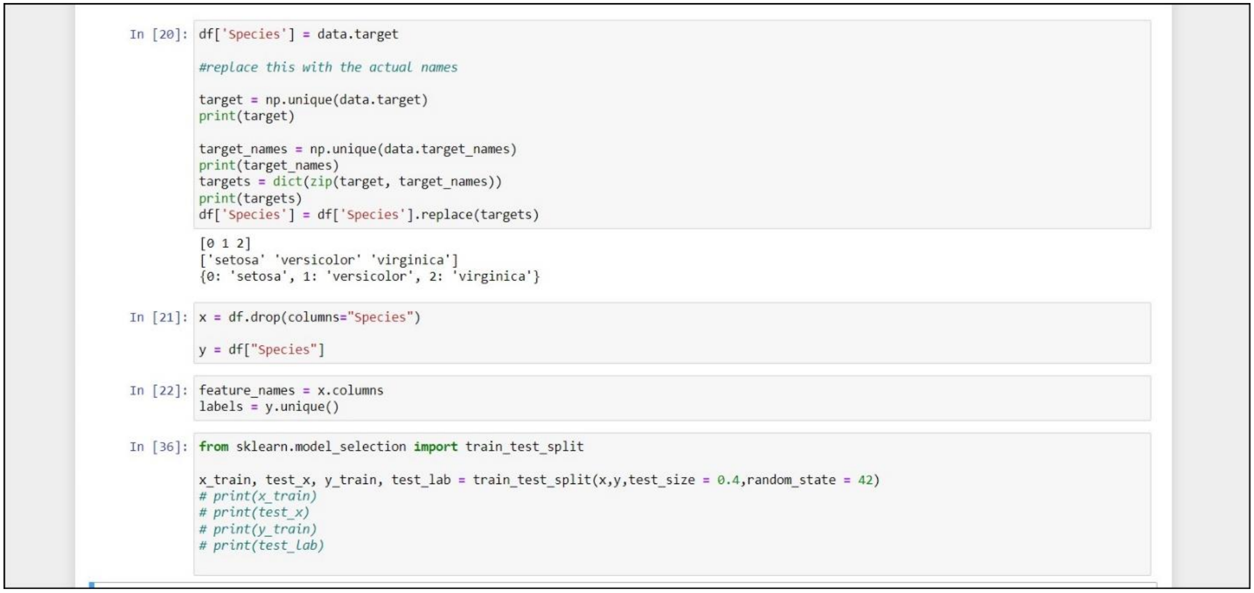
Write a program to demonstrate the working of the decision tree based ID3algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.



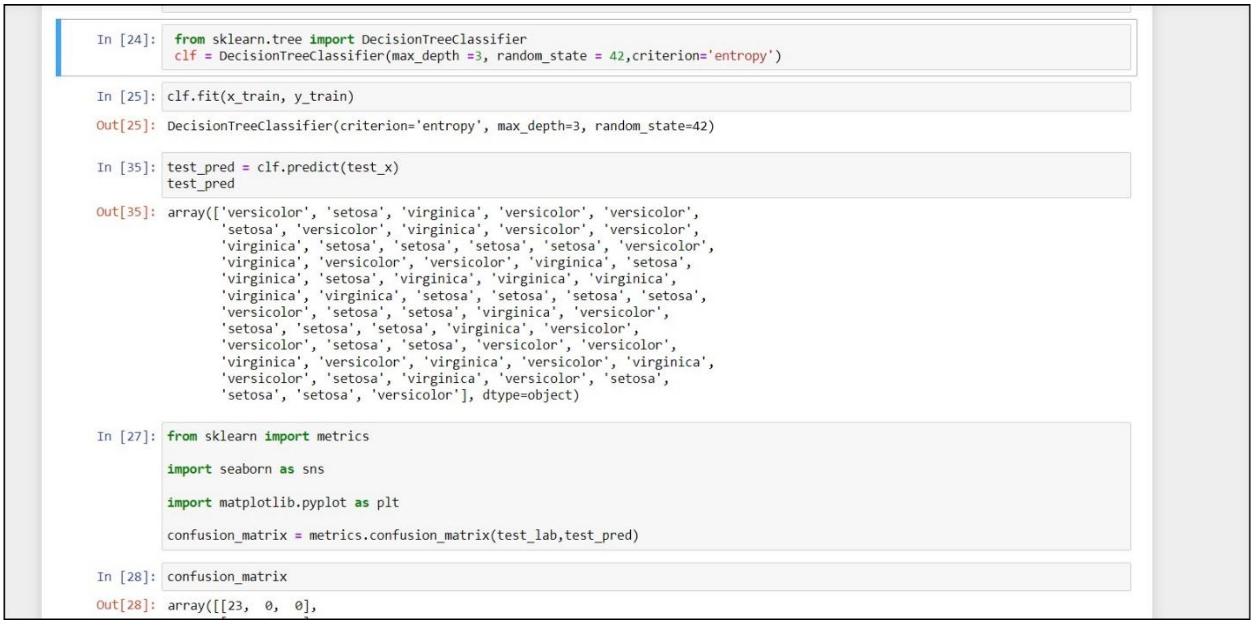
```
In [32]: import pandas as pd

         import numpy as np

         from sklearn.datasets import load_iris

         data = load_iris()

In [33]: df = pd.DataFrame(data.data, columns = data.feature_names)

In [34]: df
Out[34]:
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

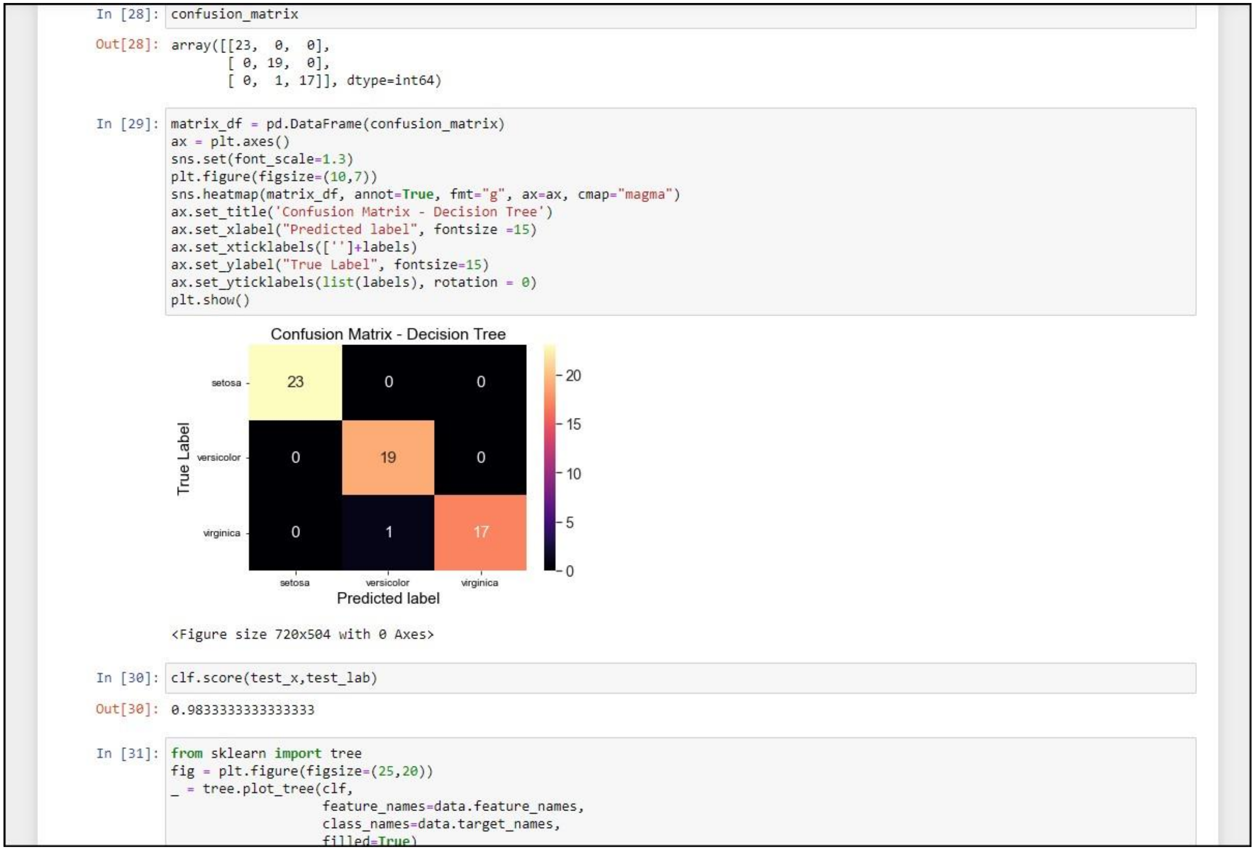
150 rows × 4 columns



```
In [20]: df['Species'] = data.target

         #replace this with the actual names

         target = np.unique(data.target)
         print(target)

         target_names = np.unique(data.target_names)
         print(target_names)
         targets = dict(zip(target, target_names))
         print(targets)
         df['Species'] = df['Species'].replace(targets)

         [0 1 2]
         ['setosa' 'versicolor' 'virginica']
         {0: 'setosa', 1: 'versicolor', 2: 'virginica'}

In [21]: x = df.drop(columns="Species")

         y = df["Species"]

In [22]: feature_names = x.columns
         labels = y.unique()

In [36]: from sklearn.model_selection import train_test_split

         x_train, test_x, y_train, test_lab = train_test_split(x,y,test_size = 0.4,random_state = 42)
         # print(x_train)
         # print(test_x)
         # print(y_train)
         # print(test_lab)
```

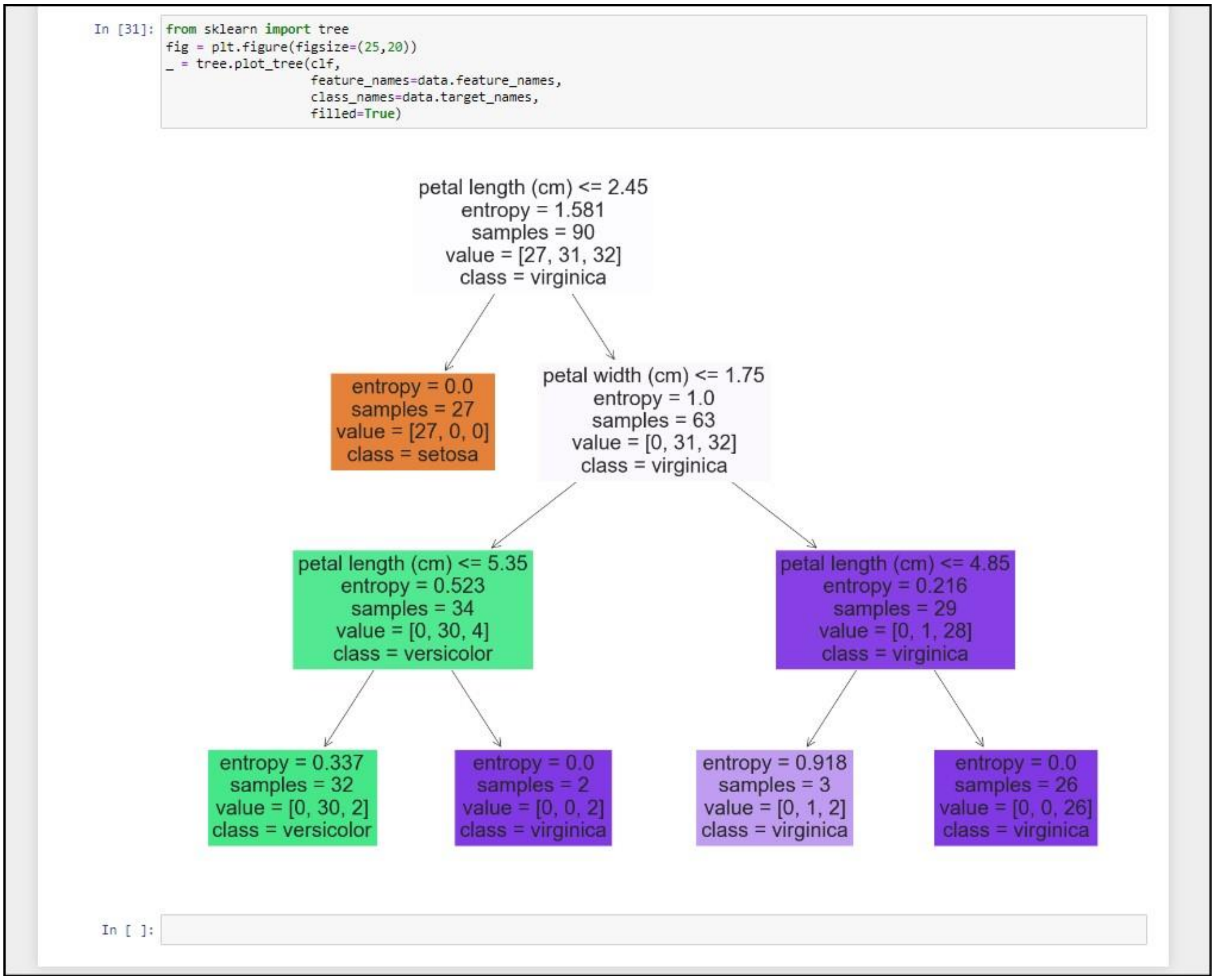


```
In [24]: from sklearn.tree import DecisionTreeClassifier
         clf = DecisionTreeClassifier(max_depth =3, random_state = 42,criterion='entropy')

In [25]: clf.fit(x_train, y_train)

Out[25]: DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=42)

In [35]: test_pred = clf.predict(test_x)
         test_pred

Out[35]: array(['versicolor', 'setosa', 'virginica', 'versicolor', 'versicolor',
                'setosa', 'versicolor', 'virginica', 'versicolor', 'versicolor',
                'virginica', 'setosa', 'setosa', 'setosa', 'setosa', 'versicolor',
                'virginica', 'versicolor', 'versicolor', 'virginica', 'setosa',
                'virginica', 'setosa', 'virginica', 'virginica', 'virginica',
                'virginica', 'virginica', 'setosa', 'setosa', 'setosa', 'setosa',
                'versicolor', 'setosa', 'setosa', 'virginica', 'versicolor',
                'setosa', 'setosa', 'setosa', 'virginica', 'versicolor',
                'versicolor', 'setosa', 'setosa', 'versicolor', 'versicolor',
                'virginica', 'versicolor', 'virginica', 'versicolor', 'virginica',
                'versicolor', 'setosa', 'virginica', 'versicolor', 'setosa',
                'setosa', 'setosa', 'versicolor'], dtype=object)

In [27]: from sklearn import metrics

         import seaborn as sns

         import matplotlib.pyplot as plt

         confusion_matrix = metrics.confusion_matrix(test_lab,test_pred)

In [28]: confusion_matrix

Out[28]: array([[23,  0,  0],
```



```
In [28]: confusion_matrix

Out[28]: array([[23,  0,  0],
                [ 0, 19,  0],
                [ 0,  1, 17]], dtype=int64)

In [29]: matrix_df = pd.DataFrame(confusion_matrix)
         ax = plt.axes()
         sns.set(font_scale=1.3)
         plt.figure(figsize=(10,7))
         sns.heatmap(matrix_df, annot=True, fmt="g", ax=ax, cmap="magma")
         ax.set_title('Confusion Matrix - Decision Tree')
         ax.set_xlabel("Predicted label", fontsize =15)
         ax.set_xticklabels(['']+labels)
         ax.set_ylabel("True Label", fontsize=15)
         ax.set_yticklabels(list(labels), rotation = 0)
         plt.show()
```

```
<Figure size 720x504 with 0 Axes>

In [30]: clf.score(test_x,test_lab)

Out[30]: 0.9833333333333333

In [31]: from sklearn import tree
         fig = plt.figure(figsize=(25,20))
         _ = tree.plot_tree(clf,
                            feature_names=data.feature_names,
                            class_names=data.target_names,
                            filled=True)
```

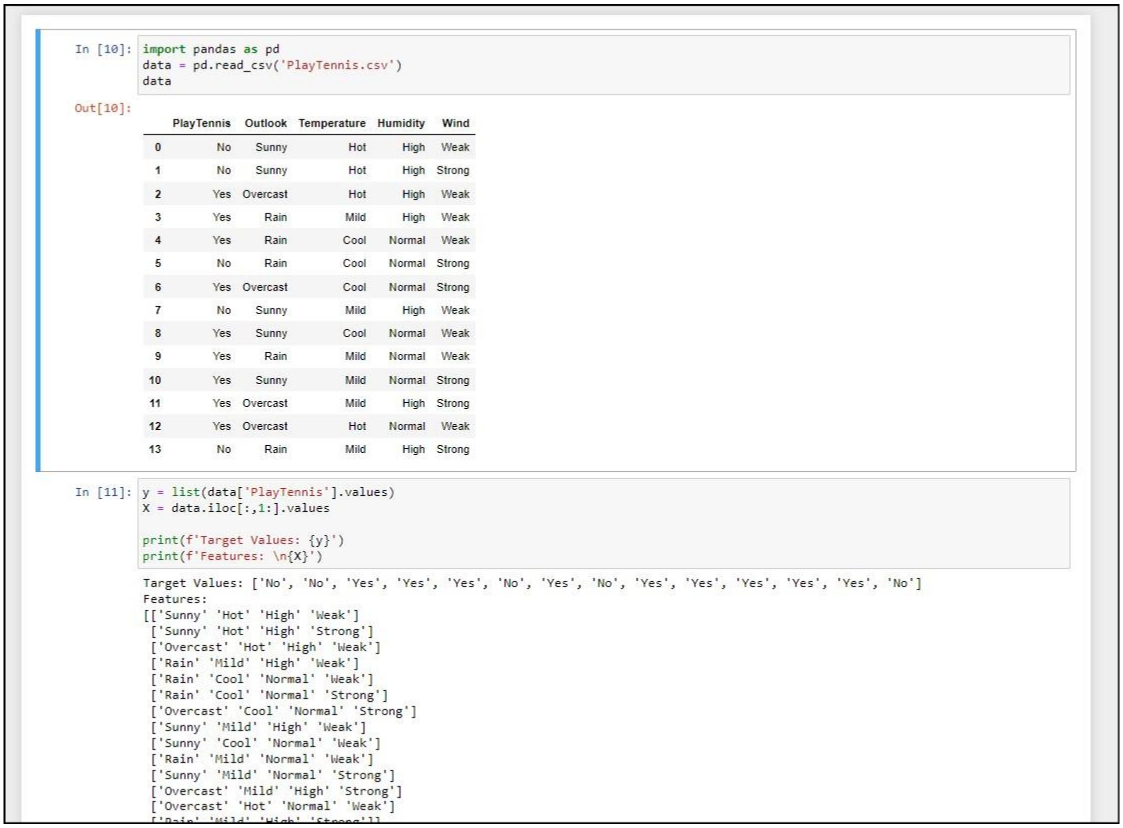


```
In [31]: from sklearn import tree
         fig = plt.figure(figsize=(25,20))
         _ = tree.plot_tree(clf,
                            feature_names=data.feature_names,
                            class_names=data.target_names,
                            filled=True)
```

```
In [ ]:
```

4. Naïve Bayes

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



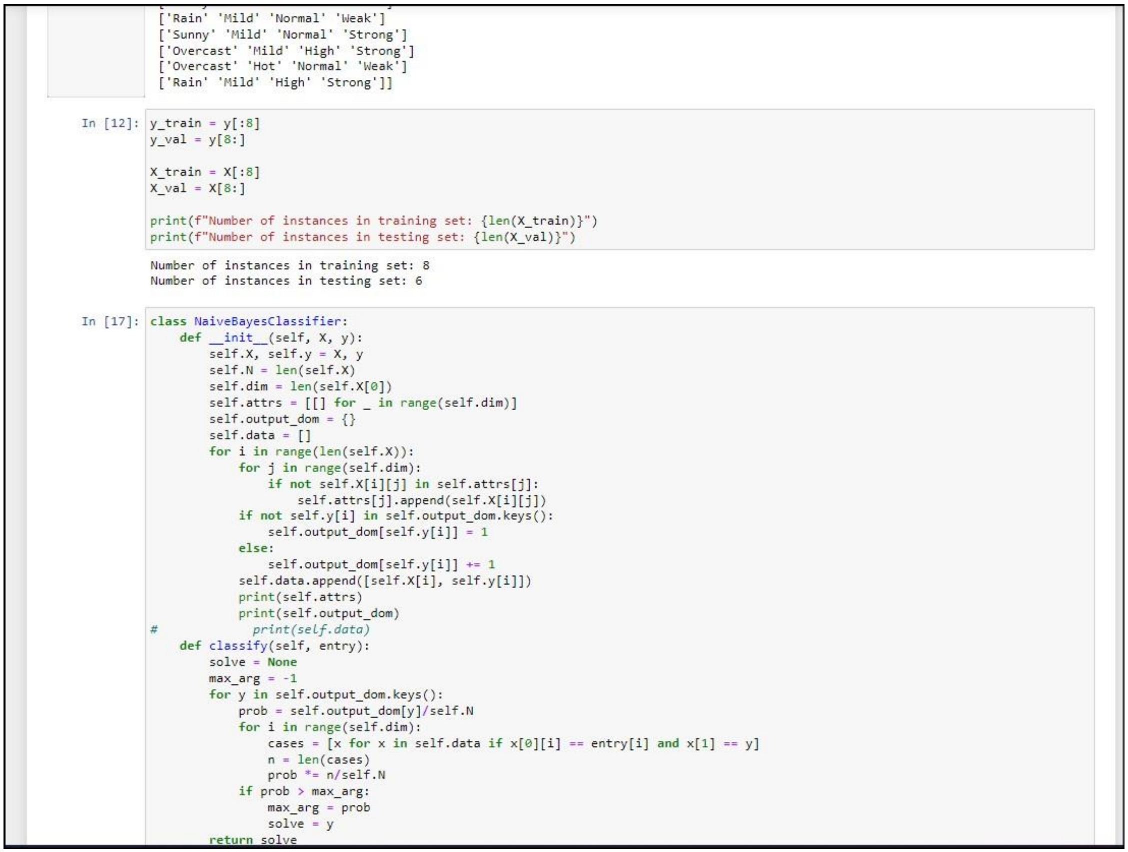
```
In [10]: import pandas as pd
         data = pd.read_csv('PlayTennis.csv')
         data
```

Out[10]:

	PlayTennis	Outlook	Temperature	Humidity	Wind
0	No	Sunny	Hot	High	Weak
1	No	Sunny	Hot	High	Strong
2	Yes	Overcast	Hot	High	Weak
3	Yes	Rain	Mild	High	Weak
4	Yes	Rain	Cool	Normal	Weak
5	No	Rain	Cool	Normal	Strong
6	Yes	Overcast	Cool	Normal	Strong
7	No	Sunny	Mild	High	Weak
8	Yes	Sunny	Cool	Normal	Weak
9	Yes	Rain	Mild	Normal	Weak
10	Yes	Sunny	Mild	Normal	Strong
11	Yes	Overcast	Mild	High	Strong
12	Yes	Overcast	Hot	Normal	Weak
13	No	Rain	Mild	High	Strong

```
In [11]: y = list(data['PlayTennis'].values)
         X = data.iloc[:,1:].values

         print(f'Target Values: {y}')
         print(f'Features: \n{X}')
```

```
Target Values: ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
Features:
[['Sunny' 'Hot' 'High' 'Weak']
 ['Sunny' 'Hot' 'High' 'Strong']
 ['Overcast' 'Hot' 'High' 'Weak']
 ['Rain' 'Mild' 'High' 'Weak']
 ['Rain' 'Cool' 'Normal' 'Weak']
 ['Rain' 'Cool' 'Normal' 'Strong']
 ['Overcast' 'Cool' 'Normal' 'Strong']
 ['Sunny' 'Mild' 'High' 'Weak']
 ['Sunny' 'Cool' 'Normal' 'Weak']
 ['Rain' 'Mild' 'Normal' 'Weak']
 ['Sunny' 'Mild' 'Normal' 'Strong']
 ['Overcast' 'Mild' 'High' 'Strong']
 ['Overcast' 'Hot' 'Normal' 'Weak']
 ['Rain' 'Mild' 'High' 'Strong']]
```

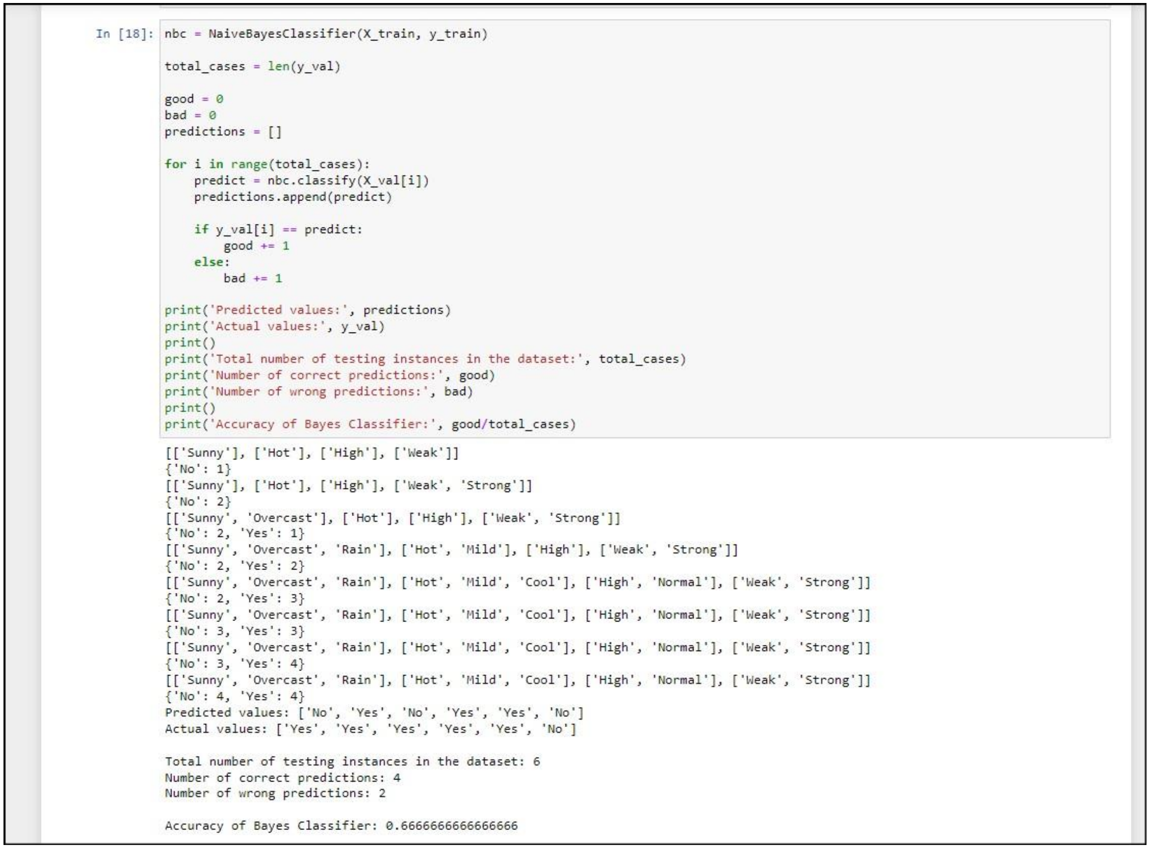


```
 ['Rain' 'Mild' 'Normal' 'Weak']
 ['Sunny' 'Mild' 'Normal' 'Strong']
 ['Overcast' 'Mild' 'High' 'Strong']
 ['Overcast' 'Hot' 'Normal' 'Weak']
 ['Rain' 'Mild' 'High' 'Strong']]
```

In [12]:
```
y_train = y[:8]
y_val = y[8:]

X_train = X[:8]
X_val = X[8:]

print(f"Number of instances in training set: {len(X_train)}")
print(f"Number of instances in testing set: {len(X_val)}")
```

```
Number of instances in training set: 8
Number of instances in testing set: 6
```

In [17]:
```
class NaiveBayesClassifier:
    def __init__(self, X, y):
        self.X, self.y = X, y
        self.N = len(self.X)
        self.dim = len(self.X[0])
        self.attrs = [[] for _ in range(self.dim)]
        self.output_dom = {}
        self.data = []
        for i in range(len(self.X)):
            for j in range(self.dim):
                if not self.X[i][j] in self.attrs[j]:
                    self.attrs[j].append(self.X[i][j])
            if not self.y[i] in self.output_dom.keys():
                self.output_dom[self.y[i]] = 1
            else:
                self.output_dom[self.y[i]] += 1
            self.data.append([self.X[i], self.y[i]])
            print(self.attrs)
            print(self.output_dom)
#             print(self.data)
    def classify(self, entry):
        solve = None
        max_arg = -1
        for y in self.output_dom.keys():
            prob = self.output_dom[y]/self.N
            for i in range(self.dim):
                cases = [x for x in self.data if x[0][i] == entry[i] and x[1] == y]
                n = len(cases)
                prob *= n/self.N
            if prob > max_arg:
                max_arg = prob
                solve = y
        return solve
```

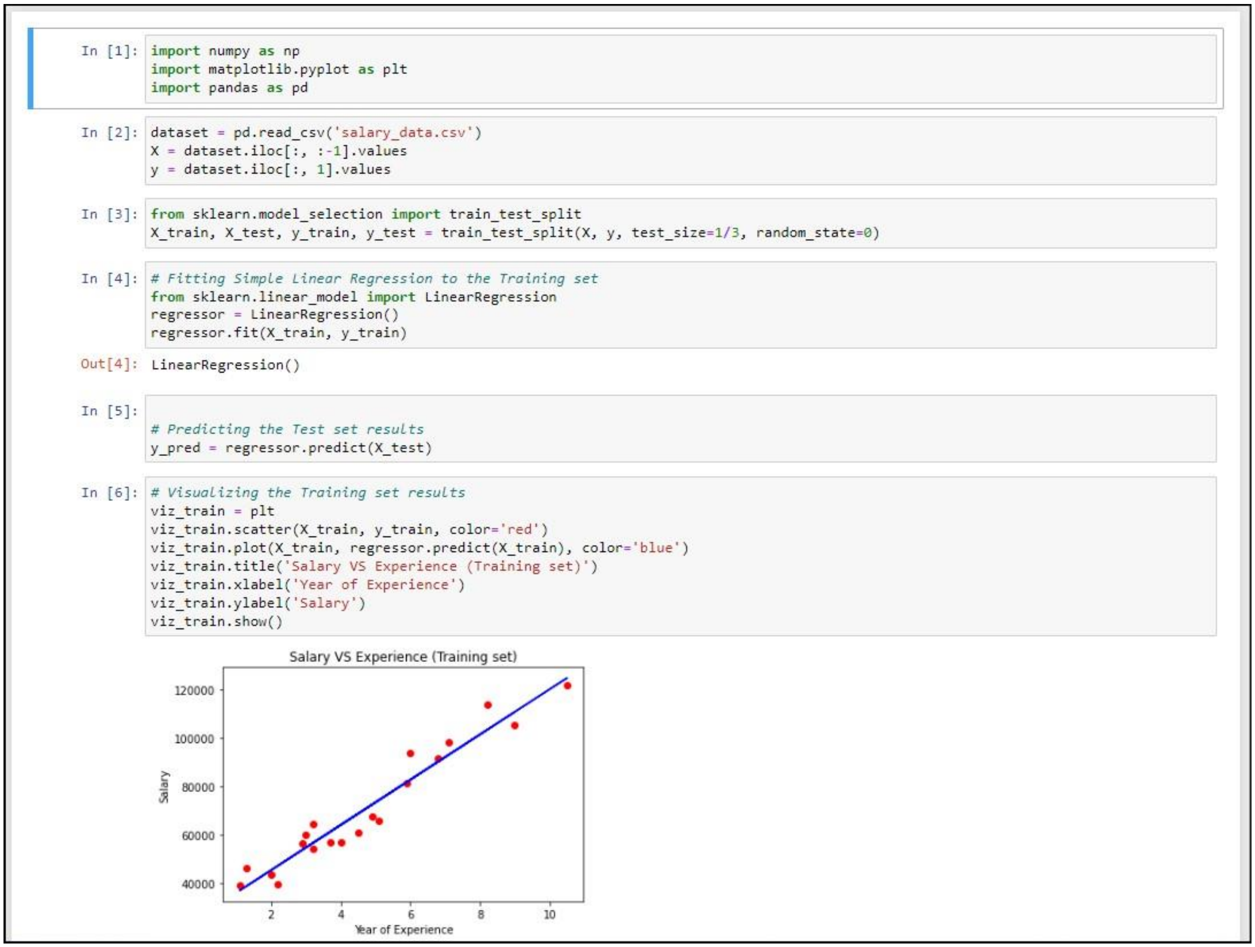


In [18]:
```
nbc = NaiveBayesClassifier(X_train, y_train)

total_cases = len(y_val)

good = 0
bad = 0
predictions = []

for i in range(total_cases):
    predict = nbc.classify(X_val[i])
    predictions.append(predict)

    if y_val[i] == predict:
        good += 1
    else:
        bad += 1

print('Predicted values:', predictions)
print('Actual values:', y_val)
print()
print('Total number of testing instances in the dataset:', total_cases)
print('Number of correct predictions:', good)
print('Number of wrong predictions:', bad)
print()
print('Accuracy of Bayes Classifier:', good/total_cases)
```

```
[['Sunny'], ['Hot'], ['High'], ['Weak']]
{'No': 1}
[['Sunny'], ['Hot'], ['High'], ['Weak', 'Strong']]
{'No': 2}
[['Sunny', 'Overcast'], ['Hot'], ['High'], ['Weak', 'Strong']]
{'No': 2, 'Yes': 1}
[['Sunny', 'Overcast', 'Rain'], ['Hot', 'Mild'], ['High'], ['Weak', 'Strong']]
{'No': 2, 'Yes': 2}
[['Sunny', 'Overcast', 'Rain'], ['Hot', 'Mild', 'Cool'], ['High', 'Normal'], ['Weak', 'Strong']]
{'No': 2, 'Yes': 3}
[['Sunny', 'Overcast', 'Rain'], ['Hot', 'Mild', 'Cool'], ['High', 'Normal'], ['Weak', 'Strong']]
{'No': 3, 'Yes': 3}
[['Sunny', 'Overcast', 'Rain'], ['Hot', 'Mild', 'Cool'], ['High', 'Normal'], ['Weak', 'Strong']]
{'No': 3, 'Yes': 4}
[['Sunny', 'Overcast', 'Rain'], ['Hot', 'Mild', 'Cool'], ['High', 'Normal'], ['Weak', 'Strong']]
{'No': 4, 'Yes': 4}
Predicted values: ['No', 'Yes', 'No', 'Yes', 'Yes', 'No']
Actual values: ['Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']

Total number of testing instances in the dataset: 6
Number of correct predictions: 4
Number of wrong predictions: 2

Accuracy of Bayes Classifier: 0.6666666666666666
```

5. Linear Regression

Implement the Linear Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.



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

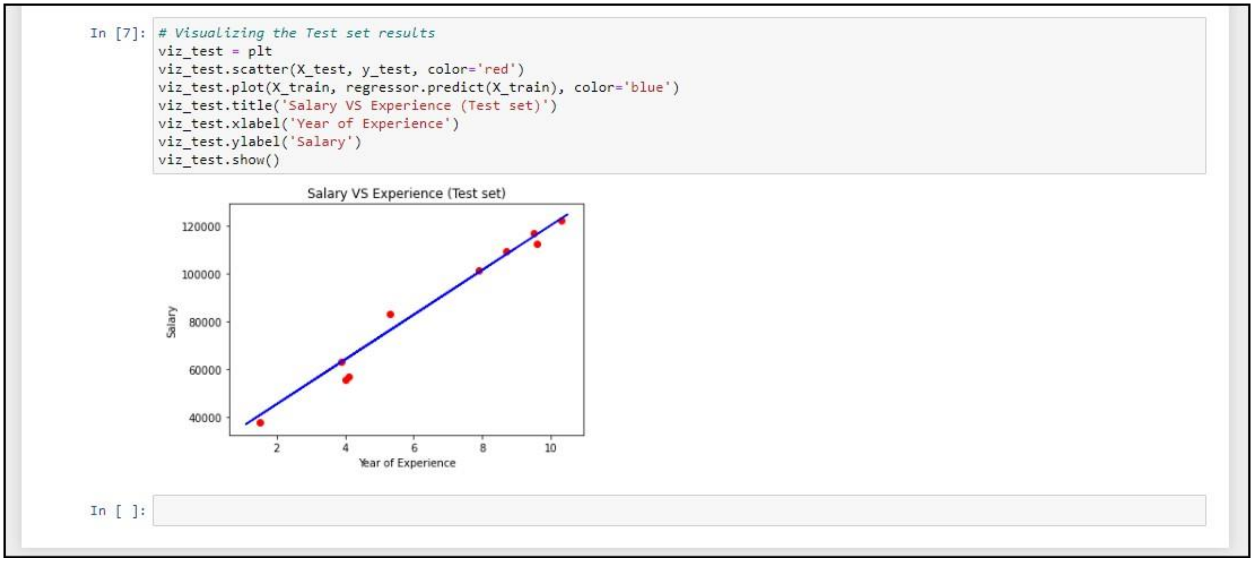
```
In [2]: dataset = pd.read_csv('salary_data.csv')
        X = dataset.iloc[:, :-1].values
        y = dataset.iloc[:, 1].values
```

```
In [3]: 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)
```

```
In [4]: # Fitting Simple Linear Regression to the Training set
        from sklearn.linear_model import LinearRegression
        regressor = LinearRegression()
        regressor.fit(X_train, y_train)

Out[4]: LinearRegression()
```

```
In [5]:
        # Predicting the Test set results
        y_pred = regressor.predict(X_test)
```

```
In [6]: # 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()
```



```
In [7]: # 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()
```

```
In [ ]:
```