MACHINE LEARNING (Integrated) – P21CS704

[As per Choice Based Credit System (CBCS) & OBE Scheme]

SEMESTER – VII

Laboratory Exercise

1.	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. (enjoySport Dataset)
2.	For a given set of training data examples stored in a .CSV file (enjoySport Dataset), implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.
3.	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 the knowledge to classify a new sample.(Play Tennis Dataset)
4.	Write a program to demonstrate the working of the decision tree based CART algorithm. (Play Tennis Dataset)
5.	Write a program to demonstrate Decision tree regression for a given dataset.(Play_Tennis_reg Dataset)
6.	Implement a Perceptron Algorithm for AND Logic Gate with 2-bit Binary Input. Test for different Hyper parameters.
7.	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. (Iris Dataset)
8.	Implement Naive Bayes Classifier for text classification task.(spam Dataset)
9.	Write a program to demonstrate Random Forest for classification task on a given dataset.(Iris Dataset)
10.	Implement AdaBoost ensemble method on a given dataset.(Iris Dataset)

In [ ]:

```
#Program 1
"""
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. (enjoySport Dataset)

"""
```

In [1]:

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

In [2]:

```
data = pd.read_csv(r"C:\Users\HPR\Desktop\ML Syllabus\2.csv")
```

In [3]:

```
data
```

Out[3]:

	sky	airtemp	humidity	wind	water	forcast	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 [4]:

```
concepts = np.array(data)[:,:-1] #The first colon (:) indicates that all rows of the array are selected.
                                 #The second part (:-1) specifies that all columns except the last one are selected.
```

In [5]:

```
concepts
```

Out[5]:

```
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)
```

In [6]:

```
target = np.array(data)[:,-1] #First colon (:): Selects all rows in the array.
                              #-1: refers to the last column
```

In [7]:

```
target
```

Out[7]:

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

In [8]:

```
def train(con, tar):
    for i, val in enumerate(tar):
        if val == 'yes':
            specific_h = con[i].copy()
            break

    for i, val in enumerate(con):
        if tar[i] == 'yes':
            for x in range(len(specific_h)):
                if val[x] != specific_h[x]:
                    specific_h[x] = '?'
                else:
                    pass
    return specific_h
```

In [9]:

```
print(train(concepts, target))
```

```
['sunny' 'warm' '?' 'strong' '?' '?']
```

In [ ]:

```
#Program-2
""""
For a given set of training data examples stored in a .CSV file (enjoySport Dataset),
implement and demonstrate the Candidate-Elimination algorithm to output a description of the
set of all hypotheses consistent with the training examples.

"""""
```

In [1]:

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

In [2]:

```
data = pd.read_csv(r"C:\Users\HPR\Desktop\ML Syllabus\2.csv")
concepts = np.array(data.iloc[:,0:-1])
target = np.array(data.iloc[:,-1])
```

In [4]:

```
def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("initialization of specific_h \n",specific_h)
    general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_h))]
    print("initialization of general_h \n", general_h)

    for i, h in enumerate(concepts):
        if target[i] == "yes":
            print("If instance is Positive ")
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'

        if target[i] == "no":
            print("If instance is Negative ")
            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(" step {}".format(i+1))
        print(specific_h)
        print(general_h)
        print("\n")
        print("\n")

    indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?', '?', '?']]
    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?', '?'])
    return specific_h, general_h
```

In [5]:

```
s_final, g_final = learn(concepts, target)

print("Final Specific_h:", s_final, sep="\n")
print("Final General_h:", g_final, sep="\n")
```

```
initialization of specific_h
 ['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
initialization of general_h
 [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
If instance is Negative
 step 1
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]




If instance is Positive
 step 2
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]




If instance is Negative
 step 3
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'same']]




If instance is Positive
 step 4
['sunny' 'warm' '?' 'strong' '?' '?']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]




Final Specific_h:
['sunny' 'warm' '?' 'strong' '?' '?']
Final General_h:
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]
```

In [1]:

```
#Program-3
""""
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 the knowledge to classify
a new sample.(Play Tennis Dataset)
""""
```

In [ ]:

```
# Load libraries
import numpy as np
import pandas as pd
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
```

In [2]:

```
df=pd.read_csv(r"C:\Users\HPR\Desktop\ML Syllabus\Play Tennis.csv")
value=['Outlook','Temprature','Humidity','Wind']
df
```

Out[2]:

	Day	Outlook	Temprature	Humidity	Wind	Play_Tennis
0	D1	Sunny	Hot	High	Weak	No
1	D2	Sunny	Hot	High	Strong	No
2	D3	Overcast	Hot	High	Weak	Yes
3	D4	Rain	Mild	High	Weak	Yes
4	D5	Rain	Cool	Normal	Weak	Yes
5	D6	Rain	Cool	Normal	Strong	No
6	D7	Overcast	Cool	Normal	Strong	Yes
7	D8	Sunny	Mild	High	Weak	No
8	D9	Sunny	Cool	Normal	Weak	Yes
9	D10	Rain	Mild	Normal	Weak	Yes
10	D11	Sunny	Mild	Normal	Strong	Yes
11	D12	Overcast	Mild	High	Strong	Yes
12	D13	Overcast	Hot	Normal	Weak	Yes
13	D14	Rain	Mild	High	Strong	No

In [3]:
```
len(df)
```

Out[3]:
```
14
```

In [4]:
```
df.shape  #To see the number of rows and columns in our dataset:
```

Out[4]:
```
(14, 6)
```

In [5]:
```
df.head() #prints first five samples
```

Out[5]:

	Day	Outlook	Temprature	Humidity	Wind	Play_Tennis
0	D1	Sunny	Hot	High	Weak	No
1	D2	Sunny	Hot	High	Strong	No
2	D3	Overcast	Hot	High	Weak	Yes
3	D4	Rain	Mild	High	Weak	Yes
4	D5	Rain	Cool	Normal	Weak	Yes

In [6]:
```
df.tail() #prints last five samples
```

Out[6]:

	Day	Outlook	Temprature	Humidity	Wind	Play_Tennis
9	D10	Rain	Mild	Normal	Weak	Yes
10	D11	Sunny	Mild	Normal	Strong	Yes
11	D12	Overcast	Mild	High	Strong	Yes
12	D13	Overcast	Hot	Normal	Weak	Yes
13	D14	Rain	Mild	High	Strong	No

In [7]:
```
df.describe()     #To see statistical details of the dataset:
```

Out[7]:

	Day	Outlook	Temprature	Humidity	Wind	Play_Tennis
count	14	14	14	14	14	14
unique	14	3	3	2	2	2
top	D1	Sunny	Mild	High	Weak	Yes
freq	1	5	6	7	8	9

In [8]:
```
#machine learning algorithms can only learn from numbers (int, float, doubles .. )
#so let us encode it to int
from sklearn import preprocessing
string_to_int= preprocessing.LabelEncoder()                          #encode your data
df=df.apply(string_to_int.fit_transform) #fit and transform it
df
```

Out[8]:

	Day	Outlook	Temprature	Humidity	Wind	Play_Tennis
0	0	2	1	0	1	0
1	6	2	1	0	0	0
2	7	0	1	0	1	1
3	8	1	2	0	1	1
4	9	1	0	1	1	1
5	10	1	0	1	0	0
6	11	0	0	1	0	1
7	12	2	2	0	1	0
8	13	2	0	1	1	1
9	1	1	2	1	1	1
10	2	2	2	1	0	1
11	3	0	2	0	0	1
12	4	0	1	1	1	1
13	5	1	2	0	0	0

In [9]:

```
#To divide our data into attribute set and Label:
feature_cols = ['Outlook','Temprature','Humidity','Wind']
X = df[feature_cols ]                               #contains the attribute
y = df.Play_Tennis
```

In [10]:

```
#To divide our data into training and test sets:
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

In [11]:

```
# perform training
from sklearn.tree import DecisionTreeClassifier                             # import the classifier
classifier =DecisionTreeClassifier(criterion="entropy", random_state=100)     # create a classifier object
classifier.fit(X_train, y_train)                                              # fit the classifier with X and Y data
```

Out[11]:

```
DecisionTreeClassifier(criterion='entropy', random_state=100)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [12]:

```
#Predict the response for test dataset
y_pred= classifier.predict(X_test)
```

In [13]:

```
# Model Accuracy, how often is the classifier correct?
from sklearn.metrics import accuracy_score
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.6
```

In [14]:

```
data_p=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
data_p
```

Out[14]:

	Actual	Predicted
4	1	1
10	1	0
12	1	1
9	1	1
5	0	1

In [15]:

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[0 1]
 [1 3]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.75      0.75      0.75         4

    accuracy                           0.60         5
   macro avg       0.38      0.38      0.38         5
weighted avg       0.60      0.60      0.60         5
```

In [ ]:

```
#Program-4
""""
Write a program to demonstrate the working of the decision tree based CART algorithm.
(Play Tennis Dataset)
""""
```

In [1]:

```
# Load libraries
import numpy as np
import pandas as pd
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
```

In [2]:

```
df=pd.read_csv(r"C:\Users\HPR\Desktop\ML Syllabus\Play Tennis.csv")
value=['Outlook','Temprature','Humidity','Wind']
df
```

Out[2]:

	Day	Outlook	Temprature	Humidity	Wind	Play_Tennis
0	D1	Sunny	Hot	High	Weak	No
1	D2	Sunny	Hot	High	Strong	No
2	D3	Overcast	Hot	High	Weak	Yes
3	D4	Rain	Mild	High	Weak	Yes
4	D5	Rain	Cool	Normal	Weak	Yes
5	D6	Rain	Cool	Normal	Strong	No
6	D7	Overcast	Cool	Normal	Strong	Yes
7	D8	Sunny	Mild	High	Weak	No
8	D9	Sunny	Cool	Normal	Weak	Yes
9	D10	Rain	Mild	Normal	Weak	Yes
10	D11	Sunny	Mild	Normal	Strong	Yes
11	D12	Overcast	Mild	High	Strong	Yes
12	D13	Overcast	Hot	Normal	Weak	Yes
13	D14	Rain	Mild	High	Strong	No

In [3]:
```
len(df)
```

Out[3]:
```
14
```

In [4]:
```
df.shape  #To see the number of rows and columns in our dataset:
```

Out[4]:
```
(14, 6)
```

In [5]:
```
df.head() #prints first five samples
```

Out[5]:

	Day	Outlook	Temprature	Humidity	Wind	Play_Tennis
0	D1	Sunny	Hot	High	Weak	No
1	D2	Sunny	Hot	High	Strong	No
2	D3	Overcast	Hot	High	Weak	Yes
3	D4	Rain	Mild	High	Weak	Yes
4	D5	Rain	Cool	Normal	Weak	Yes

In [6]:
```
df.tail() #prints last five samples
```

Out[6]:

	Day	Outlook	Temprature	Humidity	Wind	Play_Tennis
9	D10	Rain	Mild	Normal	Weak	Yes
10	D11	Sunny	Mild	Normal	Strong	Yes
11	D12	Overcast	Mild	High	Strong	Yes
12	D13	Overcast	Hot	Normal	Weak	Yes
13	D14	Rain	Mild	High	Strong	No

In [7]:
```
df.describe()      #To see statistical details of the dataset:
```

Out[7]:

	Day	Outlook	Temprature	Humidity	Wind	Play_Tennis
count	14	14	14	14	14	14
unique	14	3	3	2	2	2
top	D1	Sunny	Mild	High	Weak	Yes
freq	1	5	6	7	8	9

In [8]:

```
#machine learning algorithms can only learn from numbers (int, float, doubles .. )
#so let us encode it to int
from sklearn import preprocessing
string_to_int= preprocessing.LabelEncoder()                    #encode your data
df=df.apply(string_to_int.fit_transform) #fit and transform it
df
```

Out[8]:

	Day	Outlook	Temprature	Humidity	Wind	Play_Tennis
0	0	2	1	0	1	0
1	6	2	1	0	0	0
2	7	0	1	0	1	1
3	8	1	2	0	1	1
4	9	1	0	1	1	1
5	10	1	0	1	0	0
6	11	0	0	1	0	1
7	12	2	2	0	1	0
8	13	2	0	1	1	1
9	1	1	2	1	1	1
10	2	2	2	1	0	1
11	3	0	2	0	0	1
12	4	0	1	1	1	1
13	5	1	2	0	0	0

In [9]:

```
#To divide our data into attribute set and Label:
feature_cols = ['Outlook','Temprature','Humidity','Wind']
X = df[feature_cols ]                                   #contains the attribute
y = df.Play_Tennis
```

In [10]:

```
#To divide our data into training and test sets:
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

In [11]:
```
# perform training
from sklearn.tree import DecisionTreeClassifier                              # import the classifier
classifier =DecisionTreeClassifier(criterion="gini", random_state=100)      # create a classifier object
classifier.fit(X_train, y_train)                                               # fit the classifier with X and Y data
```

Out[11]:
```
DecisionTreeClassifier(random_state=100)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [12]:
```
#Predict the response for test dataset
y_pred= classifier.predict(X_test)
```

In [13]:
```
# Model Accuracy, how often is the classifier correct?
from sklearn.metrics import accuracy_score
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.2
```

In [14]:
```
data_p=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
data_p
```

Out[14]:

	Actual	Predicted
1	0	0
11	1	1
12	1	1
6	1	0
13	0	1

In [15]:

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[1 1]
 [1 2]]
              precision    recall  f1-score   support

           0       0.50      0.50      0.50         2
           1       0.67      0.67      0.67         3

    accuracy                           0.60         5
   macro avg       0.58      0.58      0.58         5
weighted avg       0.60      0.60      0.60         5
```

In [ ]:
```
#Program-5
""""
Write a program to demonstrate Decision tree regression for a given dataset.(Play_Tennis_reg)
""""
```

In [1]:
```
# Load libraries
import numpy as np
import pandas as pd
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
```

In [2]:
```
df=pd.read_csv(r"C:\Users\HPR\Desktop\ML Syllabus\Play_Tennis_reg.csv")
```

In [3]:
```
len(df)
```

Out[3]:
```
14
```

In [4]:
```
df.shape  #To see the number of rows and columns in our dataset:
```

Out[4]:
```
(14, 5)
```

In [5]:
```
# Select features and target
X = df.drop("Golf Players", axis=1)
y = df['Golf Players']
```

In [6]:
```
X
```

Out[6]:

	Outlook	Temprature	Humidity	Wind
0	Sunny	Hot	High	Weak
1	Sunny	Hot	High	Strong
2	Overcast	Hot	High	Weak
3	Rain	Mild	High	Weak
4	Rain	Cool	Normal	Weak
5	Rain	Cool	Normal	Strong
6	Overcast	Cool	Normal	Strong
7	Sunny	Mild	High	Weak
8	Sunny	Cool	Normal	Weak
9	Rain	Mild	Normal	Weak
10	Sunny	Mild	Normal	Strong
11	Overcast	Mild	High	Strong
12	Overcast	Hot	Normal	Weak
13	Rain	Mild	High	Strong

In [7]:
```
y
```

Out[7]:
```
0     25
1     30
2     46
3     45
4     52
5     23
6     43
7     35
8     38
9     46
10    48
11    52
12    44
13    30
Name: Golf Players, dtype: int64
```

In [8]:
```
from sklearn.preprocessing import LabelEncoder
```

In [9]:

```
from sklearn import preprocessing
string_to_int= preprocessing.LabelEncoder()                     #encode your data
X=X.apply(string_to_int.fit_transform) #fit and transform it
X
```

Out[9]:

	Outlook	Temprature	Humidity	Wind
0	2	1	0	1
1	2	1	0	0
2	0	1	0	1
3	1	2	0	1
4	1	0	1	1
5	1	0	1	0
6	0	0	1	0
7	2	2	0	1
8	2	0	1	1
9	1	2	1	1
10	2	2	1	0
11	0	2	0	0
12	0	1	1	1
13	1	2	0	0

In [10]:

```
from sklearn.tree import DecisionTreeRegressor
reg = DecisionTreeRegressor()
reg = reg.fit(X, y)
```

In [13]:

```
y_pred = reg.predict([[2,1,0,1]])
```

```
C:\Users\HPR\AppData\Roaming\Python\Python38\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but DecisionTree
Regressor was fitted with feature names
  warnings.warn(
```

In [14]:

```
# print the Result
print("Result is: ", y_pred)
```

```
Result is: % d
 [25.]
```

In [15]:

```
y_pred = reg.predict([[2,1,0,0]])
# print the Result
print("Result is: ", y_pred)
```

```
Result is: % d
 [30.]

C:\Users\HPR\AppData\Roaming\Python\Python38\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but DecisionTree
Regressor was fitted with feature names
  warnings.warn(
```

In [16]:

```
y_pred = reg.predict([[1,2,0,0]])
# print the Result
print("Result is: ", y_pred)
```

```
Result is: % d
 [30.]

C:\Users\HPR\AppData\Roaming\Python\Python38\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but DecisionTree
Regressor was fitted with feature names
  warnings.warn(
```

In [ ]:

```
""""
#Implement a Perceptron Algorithm for AND Logic Gate with 2-bit Binary Input.
#Test for the following Hyper parameters:
    -->w1=1.2, w2=0.6, bias =0, threshold = 1, learning_rate = 0.5
    -->w1=1.2, w2=0.6, bias =0.5, threshold = 1, learning_rate = 0.5
    -->w1=1.2, w2=0.6, bias =1.0, threshold = 1, learning_rate = 0.5
    -->w1=1.2, w2=0.6, bias =-1.0, threshold = 1, learning_rate = 0.5
"""""
```

In [1]:

```
import numpy as np
```

In [2]:

```
# Define inputs and expected outputs for an AND gate
inputs = np.array([
    [0, 0],
    [0, 1],
    [1, 0],
    [1, 1]
])
expected_outputs = np.array([0, 0, 0, 1])
```

In [7]:

```
# Initialize weights, bias, threshold, and learning rate
w1, w2 = 1.2, 0.6
bias =-1.0
threshold = 1
learning_rate = 0.5
```

In [8]:

```
# Activation function
def activation_function(net_input):
    return 1 if net_input >= threshold else 0
```

In [9]:

```
# Training loop
epochs = 0
while True:
    error_count = 0  # Track the number of misclassifications

    for i in range(len(inputs)):
        # Calculate weighted sum including the bias
        net_input = w1 * inputs[i][0] + w2 * inputs[i][1] + bias

        # Apply activation function
        output = activation_function(net_input)

        # Calculate error
        error = expected_outputs[i] - output

        # Update weights and bias if there is an error
        if error != 0:
            w1 += learning_rate * error * inputs[i][0]
            w2 += learning_rate * error * inputs[i][1]
            bias += learning_rate * error  # Update bias as well
            error_count += 1

    epochs += 1

    # Break if there are no errors
    if error_count == 0:
        break
```

In [ ]:

```python
# Display results
print(f"Training completed in {epochs} epochs")
print(f"Final weights: w1 = {w1}, w2 = {w2}, bias = {bias}")

# Test the perceptron on all input cases
print("Testing perceptron for AND gate:")
for i in range(len(inputs)):
    net_input = w1 * inputs[i][0] + w2 * inputs[i][1] + bias
    output = activation_function(net_input)
    print(f"Input: {inputs[i]}, Output: {output}, Expected: {expected_outputs[i]}")
```

```
Training completed in 3 epochs
Final weights: w1 = 1.2, w2 = 1.1, bias = -1.0
Testing perceptron for AND gate:
Input: [0 0], Output: 0, Expected: 0
Input: [0 1], Output: 0, Expected: 0
Input: [1 0], Output: 0, Expected: 0
Input: [1 1], Output: 1, Expected: 1
```

In [ ]:

```
#Program-7
""""
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. (Iris Dataset)
""""
```

In [1]:

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

In [2]:

```
# Load the Iris dataset
data = pd.read_csv("Iris.csv")

# Select features and target
X = data.drop("Species", axis=1)
y = data['Species']
```

In [3]:
```
X
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	1	5.1	3.5	1.4	0.2
1	2	4.9	3.0	1.4	0.2
2	3	4.7	3.2	1.3	0.2
3	4	4.6	3.1	1.5	0.2
4	5	5.0	3.6	1.4	0.2
...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3
146	147	6.3	2.5	5.0	1.9
147	148	6.5	3.0	5.2	2.0
148	149	6.2	3.4	5.4	2.3
149	150	5.9	3.0	5.1	1.8

150 rows × 5 columns

In [4]:
```
y
```

Out[4]:
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
             ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

In [5]:
```
# Encoding the Species column to get numerical class
le = LabelEncoder()
y = le.fit_transform(y)
```

In [6]:
```
y
```

Out[6]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [7]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [8]:
```
# Gaussian Naive Bayes classifier
gnb = GaussianNB()

# Train the classifier on the training data
gnb.fit(X_train, y_train)
```

Out[8]:
```
GaussianNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [9]:
```
# Make predictions on the testing data
y_pred = gnb.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print(f"The Accuracy of Prediction on Iris Flower is: {accuracy}")
```

```
The Accuracy of Prediction on Iris Flower is: 1.0
```

In [10]:

```
# Create a DataFrame to display actual and predicted values
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

# Print the table
print(df)
```

```
    Actual  Predicted
0        1          1
1        0          0
2        2          2
3        1          1
4        1          1
5        0          0
6        1          1
7        2          2
8        1          1
9        1          1
10       2          2
11       0          0
12       0          0
13       0          0
14       0          0
15       1          1
16       2          2
17       1          1
18       1          1
19       2          2
20       0          0
21       2          2
22       0          0
23       2          2
24       2          2
25       2          2
26       2          2
27       2          2
28       0          0
29       0          0
30       0          0
31       0          0
32       1          1
33       0          0
34       0          0
35       2          2
36       1          1
37       0          0
38       0          0
39       0          0
40       2          2
41       1          1
42       1          1
```

In [ ]:

```
#Program-8
""""
Implement Naive Bayes Classifier for text classification task.
url: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset
""""
```

In [1]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB, GaussianNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, f1_score
import matplotlib.pyplot as plt
```

In [2]:

```
# Load the SMS Spam Collection Dataset
sms_data = pd.read_csv("spam.csv", encoding='latin-1') # url: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset
```

In [3]:

```
# Preprocess the data
sms_data = sms_data[['v1', 'v2']]
sms_data = sms_data.rename(columns={'v1': 'label', 'v2': 'text'})
```

In [4]:
```
sms_data
```

Out[4]:

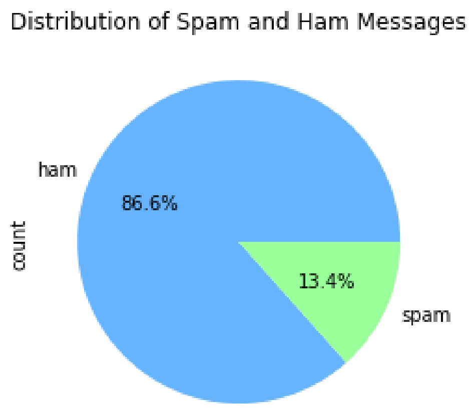
	label	text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will Ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

In [5]:
```
# Split the data into features and labels
X = sms_data['text']
y = sms_data['label']
```

In [6]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [7]:
```
# EDA 1: Distribution of Classes
class_distribution = sms_data['label'].value_counts()
class_distribution.plot(kind='pie', autopct='%1.1f%%', colors=['#66b3ff','#99ff99'])
plt.title('Distribution of Spam and Ham Messages')
plt.show()
```



In [8]:
```
# Create a CountVectorizer to convert text data into numerical features
vectorizer = CountVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

In [9]:
```
X_train_vec
```

Out[9]:
```
<4457x7735 sparse matrix of type '<class 'numpy.int64'>'
	with 58978 stored elements in Compressed Sparse Row format>
```

In [10]:
```
# Train a Multinomial Naive Bayes classifier
mnb = MultinomialNB(alpha=0.8, fit_prior=True, force_alpha=True)
mnb.fit(X_train_vec, y_train)
```

Out[10]:
```
MultinomialNB(alpha=0.8, force_alpha=True)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [11]:

```
# Train a Gaussian Naive Bayes classifier
gnb = GaussianNB()
gnb.fit(X_train_vec.toarray(), y_train)
```

Out[11]:

```
GaussianNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [12]:

```
# Evaluate the models using accuracy and F1-score
y_pred_mnb = mnb.predict(X_test_vec)
accuracy_mnb = accuracy_score(y_test, y_pred_mnb)
f1_mnb = f1_score(y_test, y_pred_mnb, pos_label='spam')

y_pred_gnb = gnb.predict(X_test_vec.toarray())
accuracy_gnb = accuracy_score(y_test, y_pred_gnb)
f1_gnb = f1_score(y_test, y_pred_gnb, pos_label='spam')

# Print the results
print("Multinomial Naive Bayes - Accuracy:", accuracy_mnb)
print("Multinomial Naive Bayes - F1-score for 'spam' class:", f1_mnb)

print("Gaussian Naive Bayes - Accuracy:", accuracy_gnb)
print("Gaussian Naive Bayes - F1-score for 'spam' class:", f1_gnb)
```

```
Multinomial Naive Bayes - Accuracy: 0.9838565022421525
Multinomial Naive Bayes - F1-score for 'spam' class: 0.9370629370629371
Gaussian Naive Bayes - Accuracy: 0.9004484304932735
Gaussian Naive Bayes - F1-score for 'spam' class: 0.7131782945736436
```

In [ ]:

```
#Program-9
""""
Write a program to demonstrate Random Forest for classification task on a given dataset.(Iris Dataset)
""""
```

In [1]:

```
# Load the iris dataset
from sklearn.datasets import load_iris
```

In [2]:

```
iris = load_iris()
```

In [3]:

```
# store the feature matrix (X) and response vector (y)
X = iris.data
y = iris.target
```

In [4]:

```
# Count the number of samples
num_samples = X.shape[0]  # The number of rows represents the number of samples

print(f'Number of samples in the Iris dataset: {num_samples}')
```

```
Number of samples in the Iris dataset: 150
```

In [5]:

```
# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

In [6]:

```
# Count the number of samples in the training and testing sets
train_samples = X_train.shape[0]  # Number of rows in X_train
test_samples = X_test.shape[0]    # Number of rows in X_test

print(f'Number of samples in the training set: {train_samples}')
print(f'Number of samples in the testing set: {test_samples}')
```

```
Number of samples in the training set: 105
Number of samples in the testing set: 45
```

In [7]:

```
# importing random forest classifier from assemble module
from sklearn.ensemble import RandomForestClassifier
# creating a RF classifier
rf = RandomForestClassifier(n_estimators = 100)

# Training the model on the training dataset
# fit function is used to train the model using the training sets as parameters
rf.fit(X_train, y_train)
```

Out[7]:

```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [8]:

```
# performing predictions on the test dataset
y_pred = rf.predict(X_test)
```

In [9]:

```
# comparing actual response values (y_test) with predicted response values (y_pred)
from sklearn import metrics
print("Random Forest model accuracy(in %):", metrics.accuracy_score(y_test, y_pred)*100)
```

```
Random Forest model accuracy(in %): 95.55555555555556
```

In [10]:

```
# Print the actual and predicted values
print("Actual values:", y_test)
print("Predicted values:", y_pred)
```

```
Actual values: [0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 1 0 2 1 0 0 1 2 1 2 1 2 2 0 1
 0 1 2 2 0 2 2 1]
Predicted values: [0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 2 0 2 1 0 0 1 2 1 2 1 2 2 0 1
 0 1 2 2 0 1 2 1]
```

In [11]:

```
import pandas as pd
# Create a DataFrame to display actual and predicted values
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

# Print the table
print(df)
```

```
    Actual  Predicted
0        0          0
1        1          1
2        1          1
3        0          0
4        2          2
5        1          1
6        2          2
7        0          0
8        0          0
9        2          2
10       1          1
11       0          0
12       2          2
13       1          1
14       1          1
15       0          0
16       1          1
17       1          1
18       0          0
19       0          0
20       1          1
21       1          1
22       1          2
23       0          0
24       2          2
25       1          1
26       0          0
27       0          0
28       1          1
29       2          2
30       1          1
31       2          2
32       1          1
33       2          2
34       2          2
35       0          0
36       1          1
37       0          0
38       1          1
39       2          2
40       2          2
41       0          0
42       2          1
43       2          2
44       1          1
```

In [14]:

```
# Assuming the classes are as follows:
label_mapping = {0: "iris-setosa", 1: "iris-versicolor", 2: "iris-virginica"}
```

In [15]:

```
y_pred=rf.predict([[3, 3, 2, 2]])
print("Result is:", label_mapping[y_pred[0]])
```

```
Result is: iris-setosa
```

In [ ]:

```
#Program-10
""""
Implement AdaBoost ensemble method on a given dataset.(Iris dataset)
""""
```

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
#import warnings warnings.filterwarnings("ignore")
```

In [2]:

```
# Reading the dataset from the csv file # separator is a vertical line, as seen in the dataset
data = pd.read_csv("Iris.csv")
# Printing the shape of the dataset
print(data.shape)
```

```
(150, 6)
```

In [3]:

```
data.head()
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [4]:

```
data = data.drop('Id',axis=1)
X = data.iloc[:,:-1]
y = data.iloc[:,-1]
print("Shape of X is %s and shape of y is %s"%(X.shape,y.shape))
```

```
Shape of X is (150, 4) and shape of y is (150,)
```

In [5]:

```
total_classes = y.nunique()
print("Number of unique species in dataset are: ",total_classes)
```

```
Number of unique species in dataset are:  3
```

In [6]:

```
distribution = y.value_counts()
print(distribution)
```

```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

In [13]:

```
X_train, X_val, Y_train, Y_val = train_test_split( X, y, test_size=0.25, random_state=42)
```

In [14]:

```
# Creating adaboost classifier model
adb = AdaBoostClassifier()
adb_model = adb.fit(X_train,Y_train)
```

In [15]:

```
print("The accuracy of the model on validation set is", adb_model.score(X_val,Y_val))
```

```
The accuracy of the model on validation set is 1.0
```

In [18]:

```
from sklearn.metrics import accuracy_score
```

In [19]:

```
# Make predictions on the testing data
y_pred = adb_model.predict(X_val)

# Calculate the accuracy of the model
accuracy = accuracy_score(Y_val, y_pred)
print(f"The Accuracy of Prediction on Iris Flower is: {accuracy}")
```

```
The Accuracy of Prediction on Iris Flower is: 1.0
```

In [21]:

```
# Create a DataFrame to display actual and predicted values
df = pd.DataFrame({'Actual': Y_val, 'Predicted': y_pred})

# Print the table
print(df)
```

```
              Actual        Predicted
73   Iris-versicolor  Iris-versicolor
18       Iris-setosa      Iris-setosa
118   Iris-virginica   Iris-virginica
78   Iris-versicolor  Iris-versicolor
76   Iris-versicolor  Iris-versicolor
31       Iris-setosa      Iris-setosa
64   Iris-versicolor  Iris-versicolor
141   Iris-virginica   Iris-virginica
68   Iris-versicolor  Iris-versicolor
82   Iris-versicolor  Iris-versicolor
110   Iris-virginica   Iris-virginica
12       Iris-setosa      Iris-setosa
36       Iris-setosa      Iris-setosa
9        Iris-setosa      Iris-setosa
19       Iris-setosa      Iris-setosa
56   Iris-versicolor  Iris-versicolor
104   Iris-virginica   Iris-virginica
69   Iris-versicolor  Iris-versicolor
55   Iris-versicolor  Iris-versicolor
132   Iris-virginica   Iris-virginica
29       Iris-setosa      Iris-setosa
127   Iris-virginica   Iris-virginica
26       Iris-setosa      Iris-setosa
128   Iris-virginica   Iris-virginica
131   Iris-virginica   Iris-virginica
145   Iris-virginica   Iris-virginica
108   Iris-virginica   Iris-virginica
143   Iris-virginica   Iris-virginica
45       Iris-setosa      Iris-setosa
30       Iris-setosa      Iris-setosa
22       Iris-setosa      Iris-setosa
15       Iris-setosa      Iris-setosa
65   Iris-versicolor  Iris-versicolor
11       Iris-setosa      Iris-setosa
42       Iris-setosa      Iris-setosa
146   Iris-virginica   Iris-virginica
51   Iris-versicolor  Iris-versicolor
27       Iris-setosa      Iris-setosa
```