

PRG – 01

```
import pandas as pd
import numpy as np

data = pd.read_csv(r"C:\Users\adith\Downloads\enjoySport.csv")
data

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

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

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

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

PRG – 02

```
import pandas as pd
import numpy as np
data = pd.read_csv(r"C:\Users\adith\Downloads\enjoySport.csv")
data
concepts = np.array(data.iloc[:, 0:-1])
concepts
target = np.array(data.iloc[:, -1])
target
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 j in range(len(specific_h))]
    print("Initialization of general_h\n", general_h)
    for i, h in enumerate(concepts):
        if target[i] == "yes":
            print("\nIf instance is Positive:")
            for x in range(len(specific_h)):
                if h[x] != specific_h[x]:
                    specific_h[x] = '?'
        elif target[i] == "no":
            print("\nIf 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("\nStep {}".format(i + 1))
    print("specific_h:", specific_h)
    print("general_h:", general_h)
indices = [i for i, val in enumerate(general_h)
           if val == ['?', '?', '?', '?', '?', '?']]
for i in indices:
    general_h.remove(['?', '?', '?', '?', '?', '?'])
```

PRG – 03

```
import numpy as np
import pandas as pd
from sklearn import metrics
df = pd.read_csv(r"C:\Users\HPR\Desktop\ML Syllabus\Play Tennis.csv")
value = ['Outlook','Temprature','Humidity','Wind']
len(df)
df.shape
df.head()
df.tail()
df.describe()
from sklearn import preprocessing
string_to_int = preprocessing.LabelEncoder()
df = df.apply(string_to_int.fit_transform)
df
feature_cols = ['Outlook','Temprature','Humidity','Wind']
X = df[feature_cols]
y = df.Play_Tennis
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)
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion="entropy", random_state=100)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import accuracy_score
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
data_p = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
data_p
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

PRG – 04

```
import numpy as np
import pandas as pd
from sklearn import metrics
df = pd.read_csv(r"C:\Users\HPR\Desktop\ML Syllabus\Play Tennis.csv")
value = ['Outlook','Temprature','Humidity','Wind']
len(df)
df.shape
df.head()
df.tail()
df.describe()
from sklearn import preprocessing
string_to_int = preprocessing.LabelEncoder()
df = df.apply(string_to_int.fit_transform)
df
feature_cols = ['Outlook','Temprature','Humidity','Wind']
X = df[feature_cols]
y = df.Play_Tennis
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)
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion="gini", random_state=100)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
data_p = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
data_p
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

PRG – 05

```
import numpy as np
import pandas as pd
from sklearn import metrics
df = pd.read_csv(r"C:\Users\HPR\Desktop\ML Syllabus\Play_Tennis_reg.csv")
len(df)
df.shape
X = df.drop("Golf Players", axis=1)
y = df['Golf Players']
from sklearn.preprocessing import LabelEncoder
string_to_int = LabelEncoder()
X = X.apply(string_to_int.fit_transform)
X
from sklearn.tree import DecisionTreeRegressor
reg = DecisionTreeRegressor()
reg = reg.fit(X, y)
y_pred = reg.predict([[2,1,0,1]])
print("Result is: ", y_pred)
y_pred = reg.predict([[2,1,0,0]])
print("Result is: ", y_pred)
y_pred = reg.predict([[1,2,0,0]])
print("Result is: ", y_pred)
```

PRG - 06

```
import numpy as np
print("Testing perceptron for AND gate:")
inputs = np.array([
    [0, 0],
    [0, 1],
    [1, 0],
    [1, 1]])
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]}')

expected_outputs = np.array([0, 0, 0, 1])
w1, w2 = 1.2, 0.6
bias = -1.0
threshold = 1
learning_rate = 0.5

def activation_function(net_input):
    return 1 if net_input >= threshold else 0

epochs = 0
while True:
    error_count = 0
    for i in range(len(inputs)):
        net_input = w1 * inputs[i][0] + w2 * inputs[i][1] + bias
        output = activation_function(net_input)
        error = expected_outputs[i] - output
        if error != 0:
            w1 += learning_rate * error * inputs[i][0]
            w2 += learning_rate * error * inputs[i][1]
            bias += learning_rate * error
            error_count += 1
    epochs += 1
    if error_count == 0:
        break
print(f'Training completed in {epochs} epochs')
print(f'Final weights: w1 = {w1}, w2 = {w2}, bias = {bias}')
```

PRG – 07

```
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

data = pd.read_csv("Iris.csv")

X = data.drop("Species", axis=1)

y = data['Species']

le = LabelEncoder()

y = le.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

gnb = GaussianNB()

gnb.fit(X_train, y_train)

y_pred = gnb.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f"The Accuracy of Prediction on Iris Flower is: {accuracy}")

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

print(df)
```

PRG - 08

```
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

sms_data = pd.read_csv("spam.csv", encoding='latin-1')

sms_data = sms_data[['v1', 'v2']]

sms_data = sms_data.rename(columns={'v1': 'label', 'v2': 'text'})

X = sms_data['text']

y = sms_data['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

vectorizer = CountVectorizer()

X_train_vec = vectorizer.fit_transform(X_train)

X_test_vec = vectorizer.transform(X_test)

mnb = MultinomialNB(alpha=0.8, fit_prior=True, force_alpha=True)

mnb.fit(X_train_vec, y_train)

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')

gnb = GaussianNB()

gnb.fit(X_train_vec.toarray(), y_train)

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("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)
```

PRG – 09

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
import pandas as pd
iris = load_iris()
X = iris.data
y = iris.target
num_samples = X.shape[0]
print(f'Number of samples in the Iris dataset: {num_samples}')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
train_samples = X_train.shape[0]
test_samples = X_test.shape[0]
print(f'Number of samples in the training set: {train_samples}')
print(f'Number of samples in the testing set: {test_samples}')
rf = RandomForestClassifier(n_estimators = 100)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
print("Random Forest model accuracy(in %):", metrics.accuracy_score(y_test, y_pred)*100)
print("Actual values:", y_test)
print("Predicted values:", y_pred)
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(df)
label_mapping = {0: "iris-setosa", 1: "iris-versicolor", 2: "iris-virginica"}
y_pred = rf.predict([[3, 3, 2, 2]])
print("Result is:", label_mapping[y_pred[0]])
```

PRG - 10

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
data = pd.read_csv("Iris.csv")
print(data.shape)
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))
total_classes = y.unique()
print("Number of unique species in dataset are: ", total_classes)
distribution = y.value_counts()
print(distribution)
X_train, X_val, Y_train, Y_val = train_test_split(X, y, test_size=0.25, random_state=42)
adb = AdaBoostClassifier()
adb_model = adb.fit(X_train, Y_train)
print("The accuracy of the model on validation set is", adb_model.score(X_val, Y_val))
from sklearn.metrics import accuracy_score
y_pred = adb_model.predict(X_val)
accuracy = accuracy_score(Y_val, y_pred)
print(f"The Accuracy of Prediction on Iris Flower is: {accuracy}")
df = pd.DataFrame({'Actual': Y_val, 'Predicted': y_pred})
print(df)
```