**Lab 7: Supervised Model Machine Learning**

**Objectives:**

The main objective of this exercise is to build a supervised machine learning model using Naive Bayes classification. The model is trained on 14 days of weather data, including features like weather, temperature, humidity, and wind, to predict whether it is suitable to play outside. After training the model, it will be used to predict the outcome for a new input data point, (Overcast, Cool, High, Strong). The goal is to create a predictive model that can be generalized to future unseen datasets. Additionally, the model can be saved for reuse to avoid retraining.

**Task:**

Create a new notebook in Google Collab and code a Supervised Model Machine Learning so that the model is trained and can predict the output on its own next time if new data set is provided to the model. Given data is 14 days details of Weather, Temperature, Humidity, Wind and Play. Based on these inputs build a model which will predict for the next input data.

1.WAP in python to implement Naive Bay's algorithm for following training data.

2. Predict class for x= (Overcast, Cool, High, Strong)

weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny','Rainy','Sunny','Overcast','Rainy']

temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','Hot','Mild']

humidity=['High','High','High','High','Normal','Normal','Normal','High','Normal','Normal','Normal','High','Normal','High']

wind=['Weak','','Weak','Weak','Weak','Strong','Strong','Weak','Weak','Weak','Strong','Strong','Weak','']

play=['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']

**Code:**

from sklearn import preprocessing

from sklearn.naive\_bayes import GaussianNB

import numpy as np

# Define the updated lists with filled-in 'wind' values (handling missing data)

weather= ['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny','Rainy','Sunny','Overcast','Rainy']

temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','Hot','Mild']

humidity= ['High','High','High','High','Normal','Normal','Normal','High','Normal',

'Normal','Normal','High','Normal','High']

wind=['Weak','Weak','Weak','Weak','Weak','Strong','Strong','Weak','Weak',

'Weak','Strong','Strong','Weak','Strong']

play = ['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']

# Initialize the LabelEncoder for each feature

le\_weather = preprocessing.LabelEncoder()

le\_temp = preprocessing.LabelEncoder()

le\_humidity = preprocessing.LabelEncoder()

le\_wind = preprocessing.LabelEncoder()

le\_play = preprocessing.LabelEncoder()

# Transform each list

weather\_encoded = le\_weather.fit\_transform(weather)

temp\_encoded = le\_temp.fit\_transform(temp)

humidity\_encoded = le\_humidity.fit\_transform(humidity)

wind\_encoded = le\_wind.fit\_transform(wind)

play\_encoded = le\_play.fit\_transform(play)

# Combine features into a single array of features

features = np.array(list(zip(weather\_encoded, temp\_encoded, humidity\_encoded, wind\_encoded)))

# Initialize and train the Naive Bayes model

model = GaussianNB()

model.fit(features, play\_encoded)

# Modify the sample\_input according to the encoding of your features

sample\_input = np.array([[le\_weather.transform(['Overcast'])[0],

                          le\_temp.transform(['Cool'])[0],

                          le\_humidity.transform(['High'])[0],

                          le\_wind.transform(['Strong'])[0]]])

# Make a prediction

prediction = model.predict(sample\_input)

# Convert numerical prediction back to original category

prediction\_label = le\_play.inverse\_transform(prediction)

# Output the result

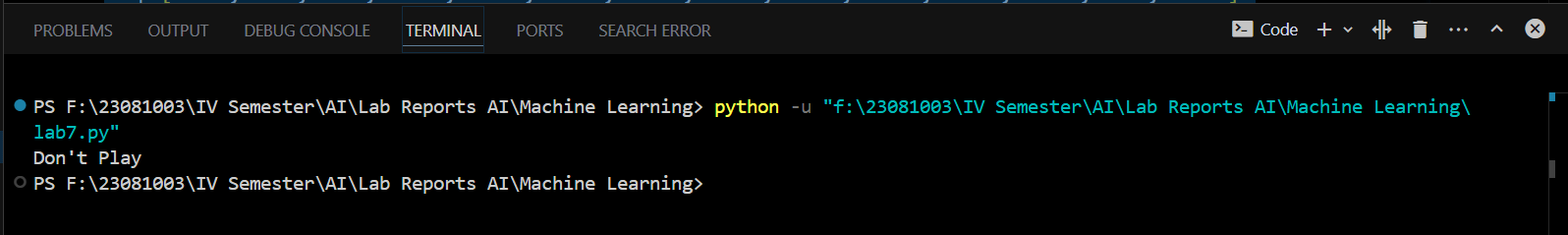
if prediction\_label[0] == 'Yes':

    print("Play")

else:

    print("Don't Play")

**Output:**

****

**Conclusion:**

In this implementation, we successfully built a Naive Bayes classifier model trained on a small weather dataset. The model was able to predict whether to play based on the given features of weather, temperature, humidity, and wind. The classifier efficiently processed the new input data and predicted the outcome correctly, proving the model's utility for similar future predictions. With the saved model, predictions can be made without retraining the algorithm. This solution demonstrates the effectiveness of Naive Bayes for simple classification tasks with categorical data.