**Implementation of Bayesian algorithm in python**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.datasets import load\_iris

from sklearn.preprocessing import LabelEncoder

iris = load\_iris()

X = iris.data

y = iris.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

model = GaussianNB()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

print("Classification Report:")

print(classification\_report(y\_test, y\_pred))

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

sample = [[5.1, 3.5, 1.4, 0.2]]

prediction = model.predict(sample)

print("Sample prediction:", iris.target\_names[prediction[0]])

probabilities = model.predict\_proba(sample)

print("Prediction probabilities:", probabilities)

data = pd.DataFrame({

'Weather': ['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy', 'Overcast',

'Sunny', 'Sunny', 'Rainy', 'Sunny', 'Overcast', 'Overcast', 'Rainy'],

'Temperature': ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool',

'Mild', 'Cool', 'Mild', 'Mild', 'Mild', 'Hot', 'Mild'],

'Play': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes',

'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']

})

le\_weather = LabelEncoder()

le\_temp = LabelEncoder()

le\_play = LabelEncoder()

data['Weather\_encoded'] = le\_weather.fit\_transform(data['Weather'])

data['Temp\_encoded'] = le\_temp.fit\_transform(data['Temperature'])

data['Play\_encoded'] = le\_play.fit\_transform(data['Play'])

X\_custom = data[['Weather\_encoded', 'Temp\_encoded']]

y\_custom = data['Play\_encoded']

model\_custom = GaussianNB()

model\_custom.fit(X\_custom, y\_custom)

test\_weather = le\_weather.transform(['Sunny'])[0]

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

custom\_pred = model\_custom.predict([[test\_weather, test\_temp]])

print("Custom prediction:", le\_play.inverse\_transform(custom\_pred)[0])