|  |  |
| --- | --- |
| **Ex. No: 2**  **Date: 03.01.24** | **Naïve Bayes Classifier** |

**Name: S Sowmithaa Sri**

**Reg no: 22011101096**

**Aim:**

To implement a supervised Machine learning algorithm

1. Naïve Bayes Classifier

In mathematical version and using the built-in function from sklearn.

**Dataset description:**

The dataset consists of observations on various weather conditions, each described by four features: 'Outlook' representing the current weather state (e.g., 'Sunny', 'Overcast', 'Rainy'), 'Temperature' denoting the temperature level ('Hot', 'Mild', 'Cool'), 'Humidity' indicating humidity status ('High', 'Normal'), and 'Wind' signifying wind strength ('Weak', 'Strong'). The target variable 'PlayTennis' signifies whether or not tennis was played under those conditions ('Yes' or 'No').

**Code:**

1. **NBC- Mathematical Version**

import numpy as np

import pandas as pd

# Sample dataset

data = {

'Outlook': ['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy', 'Overcast', 'Sunny', 'Sunny', 'Rainy'],

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

'Humidity': ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'Normal'],

'Wind': ['Weak', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak', 'Weak', 'Weak'],

'PlayTennis': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes']

}

df = pd.DataFrame(data)

# Function to calculate probabilities

def calculate\_probabilities(df, feature, target, value):

total\_count = df.shape[0]

target\_count = df[df[target] == value].shape[0]

feature\_count = df[(df[target] == value) & (df[feature] == value)].shape[0]

prob\_target = target\_count / total\_count

prob\_feature\_given\_target = feature\_count / target\_count if target\_count != 0 else 0

return prob\_target, prob\_feature\_given\_target

# Function to predict using Naive Bayes

def predict\_naive\_bayes(df, new\_data):

target\_values = df['PlayTennis'].unique()

probabilities = []

for target\_value in target\_values:

prob\_target = 1.0

for feature, value in new\_data.items():

prob\_target \*= calculate\_probabilities(df, feature, 'PlayTennis', target\_value)[1]

probabilities.append((target\_value, prob\_target))

# Return the class with the highest probability

return max(probabilities, key=lambda x: x[1])[0]

# Example usage

new\_data = {'Outlook': 'Sunny', 'Temperature': 'Cool', 'Humidity': 'Normal', 'Wind': 'Weak'}

prediction = predict\_naive\_bayes(df, new\_data)

print(f"The predicted class for the new data is: {prediction}")

**Result:**

The predicted class for the new data is: No

1. **NBC- Built-in Version**

**Code:**

from sklearn.naive\_bayes import CategoricalNB

from sklearn.preprocessing import OrdinalEncoder

# Sample dataset

data = {

'Outlook': ['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy', 'Overcast', 'Sunny', 'Sunny', 'Rainy'],

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

'Humidity': ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'Normal'],

'Wind': ['Weak', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak', 'Weak', 'Weak'],

'PlayTennis': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes']

}

df = pd.DataFrame(data)

# Encode categorical features

encoder = OrdinalEncoder()

X\_encoded = encoder.fit\_transform(df[['Outlook', 'Temperature', 'Humidity', 'Wind']])

y = df['PlayTennis']

# Create and train Naive Bayes classifier

model = CategoricalNB()

model.fit(X\_encoded, y)

# Example usage for prediction

new\_data = np.array(encoder.transform([['Sunny', 'Cool', 'Normal', 'Weak']]))

prediction = model.predict(new\_data)

print(f"The predicted class for the new data is: {prediction[0]}")

**Result:**

The predicted class for the new data is: Yes

**Result Description:**

In the mathematical implementation, a custom Naive Bayes classifier is created and applied to predict the "PlayTennis" class for a set of new weather conditions. The result is determined by calculating probabilities based on the training dataset. In the built-in library implementation using scikit-learn, a Categorical Naive Bayes classifier is trained on the encoded features, and its prediction is made for a new set of weather conditions. The result is obtained using the model's predict method. Both implementations aim to predict whether tennis will be played or not based on the provided weather features, showcasing the flexibility of implementing Naive Bayes either from scratch or using a library.

**Conclusion:**

Thus the supervised ML algorithm-Naïve Bayes Classifier is successfully implemented and executed.