PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

ptovide code from the example of naive bayes classifier for the example : play tennis (attached) image

Absolutely. The image shows the classic **Play Tennis** dataset, which is a good example for demonstrating a **Naive Bayes classifier with categorical variables**.

Below is a complete Python example using `CategoricalNB` from `scikit-learn`.

Naive Bayes — Play Tennis

Python Ejecutar

```python
import pandas as pd
from sklearn.naive_bayes import CategoricalNB
from sklearn.preprocessing import OrdinalEncoder

# ------------------------------------------------
# 1. Create the Play Tennis dataset
# ------------------------------------------------

data = {
    'Outlook': [
        'Sunny', 'Sunny', 'Overcast', 'Rain',
        'Rain', 'Rain', 'Overcast', 'Sunny',
        'Sunny', 'Rain', 'Sunny', 'Overcast',
        'Overcast', 'Rain'
    ],

    'Temperature': [
```

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

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

df = pd.DataFrame(data)

print("Dataset:")
print(df)


# -------------------------------------------------
# 2. Separate features (X) and target (y)
# -------------------------------------------------

X = df[['Outlook', 'Temperature', 'Humidity', 'Wind']]
y = df['PlayTennis']


# -------------------------------------------------
# 3. Convert categorical values to numbers
# -------------------------------------------------

encoder = OrdinalEncoder()

X_encoded = encoder.fit_transform(X)
```

```
print("\nEncoded Features:")
print(X_encoded)


# Convert Yes/No to 1/0
y_encoded = y.map({'No': 0, 'Yes': 1})


# ------------------------------------------------
# 4. Create and train the Naive Bayes classifier
# ------------------------------------------------

model = CategoricalNB()

model.fit(X_encoded, y_encoded)


# ------------------------------------------------
# 5. Make a prediction
# ------------------------------------------------

# Example:
# Outlook = Sunny
# Temperature = Cool
# Humidity = Normal
# Wind = Strong

new_data = pd.DataFrame({
    'Outlook': ['Sunny'],
    'Temperature': ['Cool'],
    'Humidity': ['Normal'],
    'Wind': ['Strong']
})

new_data_encoded = encoder.transform(new_data)

prediction = model.predict(new_data_encoded)

# Convert prediction back to Yes/No
result = 'Yes' if prediction[0] == 1 else 'No'

print("\nPrediction:")
print("Outlook = Sunny")
print("Temperature = Cool")
print("Humidity = Normal")
print("Wind = Strong")
```

```
print("\nWill we play tennis?", result)
```

Expected result

For:

Outlook	Temperature	Humidity	Wind
Sunny	Cool	Normal	Strong

the classifier predicts:

```
Will we play tennis? Yes
```

6. Get the probabilities

One of the advantages of Naive Bayes is that we can see the probability of each class:

Python Ejecutar

```
probabilities = model.predict_proba(new_data_encoded)

print("\nProbabilities:")
print("P(No)  =", probabilities[0][0])
print("P(Yes) =", probabilities[0][1])
```

You can also write:

Python Ejecutar

```
print(f"P(No)  = {probabilities[0][0]:.4f}")
print(f"P(Yes) = {probabilities[0][1]:.4f}")
```

This gives students a concrete connection between the **Bayes theorem calculation** and the Python implementation.

The Naive Bayes idea behind the code

For the example

Sunny, Cool, Normal, Strong

Naive Bayes compares:

$$P(Yes|Sunny, Cool, Normal, Strong)$$

with

$$P(No|Sunny, Cool, Normal, Strong)$$

Using the Naive Bayes assumption:

$$P(Yes|X) \propto P(Yes)P(Sunny|Yes)P(Cool|Yes)P(Normal|Yes)P(Strong|$$

and

$$P(No|X) \propto P(No)P(Sunny|No)P(Cool|No)P(Normal|No)P(Strong|No)$$

The class with the larger probability is selected as the prediction.

Important: `CategoricalNB` is appropriate here because all four input variables—Outlook, Temperature, Humidity, and Wind—are **categorical**, not continuous numerical variables.