Implementation of Naive Bayes Classification

We demonstrate how a Naive Bayes classifier works using a simple dataset related to playing tennis. Here, we manually compute the prior and conditional probabilities involved in the classification process step by step.

1. Preparing the dataset

First, we create a dataset that contains information about weather conditions and whether or not a game of tennis was played. The dataset includes four features: `Outlook`, `Temperature`, `Humidity`, and `Wind`. We then load this dataset into a Pandas DataFrame for further analysis.

```
import pandas as pd

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'],
    'Play_Tennis': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes',
                    'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
}
df = pd.DataFrame(data)

# Displaying the dataset
df
```

	Outlook	Temperature	Humidity	Wind	Play_Tennis
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

Next steps: Generate code with df | View recommended plots | New interactive sheet

2. Computing prior probabilities

We calculate the prior probabilities for the two classes in our dataset: `Play_Tennis = Yes` and `Play_Tennis = No`. The prior probability gives us the likelihood of each class occurring without any knowledge of the features. This is computed by counting the occurrences of each class in the dataset.

```
prior_yes = len(df[df['Play_Tennis'] == 'Yes']) / len(df)
prior_no = len(df[df['Play_Tennis'] == 'No']) / len(df)

print(f"P(Play_Tennis = Yes) = {prior_yes:.4f}")
print(f"P(Play_Tennis = No) = {prior_no:.4f}")
```

```
P(Play_Tennis = Yes) = 0.6429
P(Play_Tennis = No) = 0.3571
```

3. Computing conditional probabilities

Next, we compute the conditional probabilities for each feature given the class labels. This helps us understand how each feature contributes to the probability of playing tennis. We create separate tables for the conditional probabilities of each feature.

```
def conditional_probabilities(df, feature, target):
    return df.groupby([feature, target]).size().unstack(fill_value=0).div(df[target].value_counts(), axis=1)

features = ['Outlook', 'Temperature', 'Humidity', 'Wind']
conditional_probs = {feature: conditional_probabilities(df, feature, 'Play_Tennis') for feature in features}

for feature, table in conditional_probs.items():
    print(f"\nConditional probabilities for {feature}:")
    print(table)
```

```
Conditional probabilities for Outlook:
Play_Tennis   No       Yes
Outlook
Overcast     0.0  0.444444
Rain         0.4  0.333333
Sunny        0.6  0.222222

Conditional probabilities for Temperature:
Play_Tennis   No       Yes
Temperature
Cool         0.2  0.333333
Hot          0.4  0.222222
Mild         0.4  0.444444

Conditional probabilities for Humidity:
Play_Tennis   No       Yes
Humidity
High         0.8  0.333333
Normal       0.2  0.666667

Conditional probabilities for Wind:
Play_Tennis   No       Yes
Wind
Strong       0.6  0.333333
Weak         0.4  0.666667
```

4. User input for prediction

Finally, we take an unlabeled sample from the user. Based on the input features, we calculate the probabilities of the two classes (`Yes` and `No`) using the prior and conditional probabilities computed earlier. We determine which probability is greater and assign a label accordingly.

```
user_input = {
    'Outlook': 'Sunny',
    'Temperature': 'Cool',
    'Humidity': 'High',
    'Wind': 'Strong'
}

def calculate_probabilities(user_input, prior_yes, prior_no, conditional_probs):
    prob_yes = prior_yes
    prob_no = prior_no

    for feature, value in user_input.items():
        prob_yes *= conditional_probs[feature].loc[value, 'Yes']
        prob_no *= conditional_probs[feature].loc[value, 'No']

    return prob_yes, prob_no

prob_yes, prob_no = calculate_probabilities(user_input, prior_yes, prior_no, conditional_probs)

print(f"\nP(Play_Tennis = Yes | features) = {prob_yes:.4f}")
print(f"P(Play_Tennis = No | features) = {prob_no:.4f}")

if prob_yes > prob_no:
    label = 'Yes'
else:
    label = 'No'

print(f"The predicted label for the input is: {label}")
```

```
P(Play_Tennis = Yes | features) = 0.0053
P(Play_Tennis = No | features) = 0.0206
The predicted label for the input is: No
```