

## ▼ The Breast Cancer Dataset in Python: Binary Naive Bayes Classifier

### ▼ 1. Importing necessary libraries

First, we need to import the libraries required for our analysis: `pandas` for data manipulation, `sklearn` for loading the dataset, creating the Naive Bayes Classifier, and evaluating the performance of the model, and `seaborn` for creating a confusion matrix heatmap.

```
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
```

### ▼ 2. Loading the dataset

Here, we load the Breast Cancer dataset from `sklearn`. This dataset contains various features describing cell characteristics, with the goal of predicting whether a tumor is benign (harmless) or malignant (cancerous).


```
data = load_breast_cancer()
```

### ▼ 3. Creating a DataFrame

We convert the loaded dataset into a `pandas` `DataFrame` for easier manipulation. The `data.data` contains the feature values (cell characteristics), and `data.feature_names` gives the corresponding names of these features. We also add a column named `Target` to the `DataFrame`, which contains the labels (0 for benign, 1 for malignant).


```
df = pd.DataFrame(data=data.data, columns=data.feature_names)
df['Target'] = data.target
```

```
# Displaying the first few rows of the DataFrame
df.head()
```



	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1061
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1616
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1753
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2013
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1713

5 rows × 31 columns



### ▼ 4. Separating features and target variable

Here, `x` contains all columns except the `Target` column, which are the input features. `y` contains only the `Target` column, which indicates whether the tumor is benign or malignant.

```
X = df.drop('Target', axis=1)
y = df['Target']
```

### ▼ 5. Splitting the data into training and testing sets

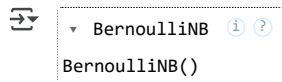
We now split the dataset into training and testing sets using the `train_test_split` function. We specify `test_size=0.2` to reserve 20% of the data for testing and `random_state=42` to ensure the data is split consistently every time we run the code.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

## 6. Creating and training the Naive Bayes Classifier

In this section, we create an instance of the Bernoulli Naive Bayes model and train it. The model learns the relationship between the features (`X_train`) and the target labels (`y_train`).

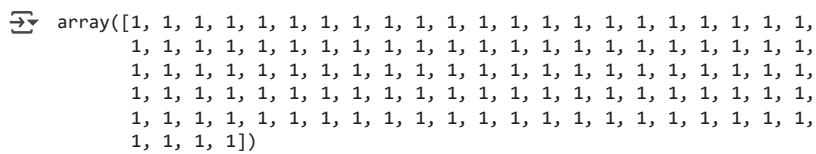
```
model = BernoulliNB()
model.fit(X_train, y_train)
```



## 7. Making predictions

Once the model is trained, we use it to make predictions on the test data (`X_test`). The model generates predicted labels (`y_pred`) for the test set, which we will compare with the actual labels (`y_test`) to evaluate its performance.

```
y_pred = model.predict(X_test)  
  
# Displaying the predicted labels  
y_pred
```



## 8. Calculating accuracy

Next, we calculate the accuracy of the model, i.e., the proportion of correct predictions made by the model out of all predictions.

```
accuracy = accuracy_score(y_test, y_pred)  
accuracy
```



## 9. Generating the confusion matrix

Finally, we compute the confusion matrix to better visualise the performance of the model. This matrix shows how well the model performed in classifying the tumor types:

1. True positives (correctly predicted malignant tumors).
2. True negatives (correctly predicted benign tumors).
3. False positives (benign tumors predicted as malignant).
4. False negatives (malignant tumors predicted as benign).

```
conf_matrix = confusion_matrix(y_test, y_pred)  
  
import matplotlib.pyplot as plt  
plt.figure(figsize=(4,4))  
sns.heatmap(conf_matrix,  
            annot=True,  
            fmt='g',  
            cmap='Blues',  
            xticklabels=['Benign', 'Malignant'],  
            yticklabels=['Benign', 'Malignant'],  
            cbar=False,  
            annot_kws={"size": 10},  
            linewidths=1,  
            linecolor='black'  
            )  
  
plt.title('Confusion Matrix: Naive Bayes Classifier', fontsize=12)  
plt.xlabel('Predicted', fontsize=10)  
plt.ylabel('True', fontsize=10)  
plt.tight_layout()  
  
plt.show()
```

Confusion Matrix: Naive Bayes Classifier

