Name - Swapnadeep Mishra

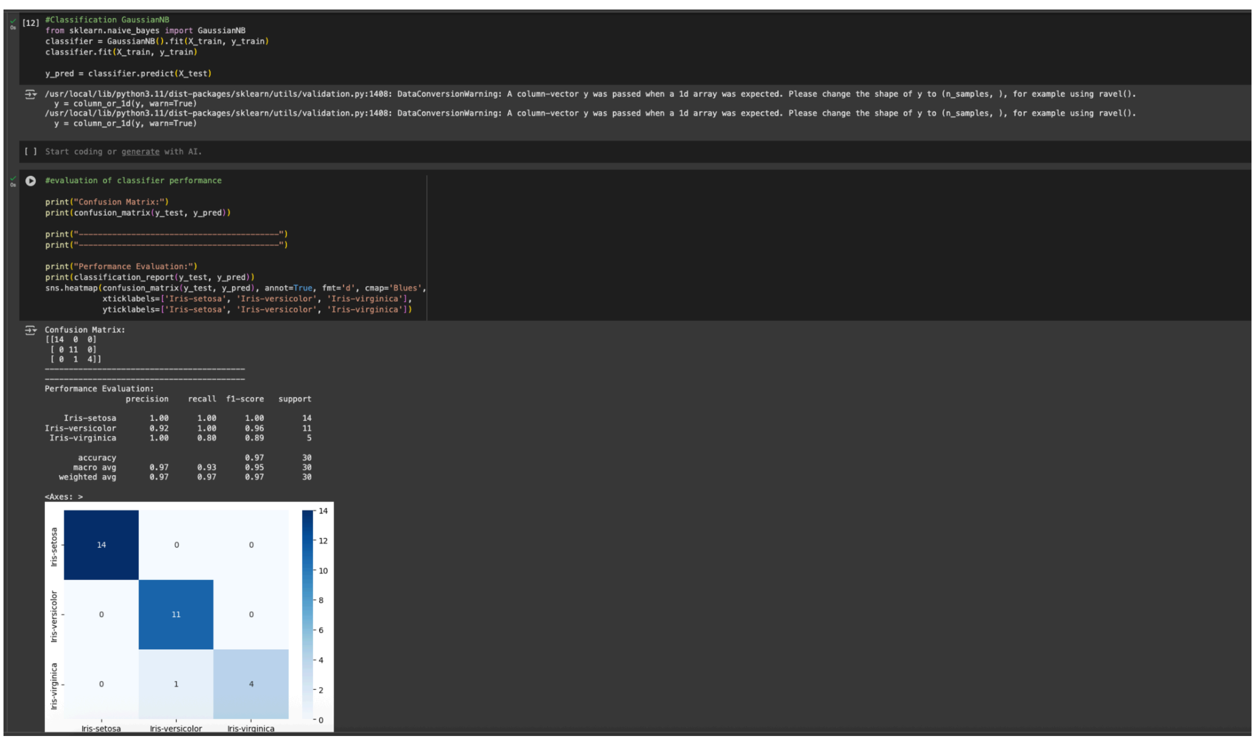
Roll - 002211001115, **Group -** A2, **Dept. -** IT

Assignment 1 – Machine Learning Classification

Q1: Naive Bayes Classifier

Naive Bayes is a **probabilistic model** that applies Bayes' theorem under the assumption of feature independence.

- **GaussianNB** assumes continuous features follow a normal distribution.



```
[12] #Classification GaussianNB
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB().fit(X_train, y_train)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
```

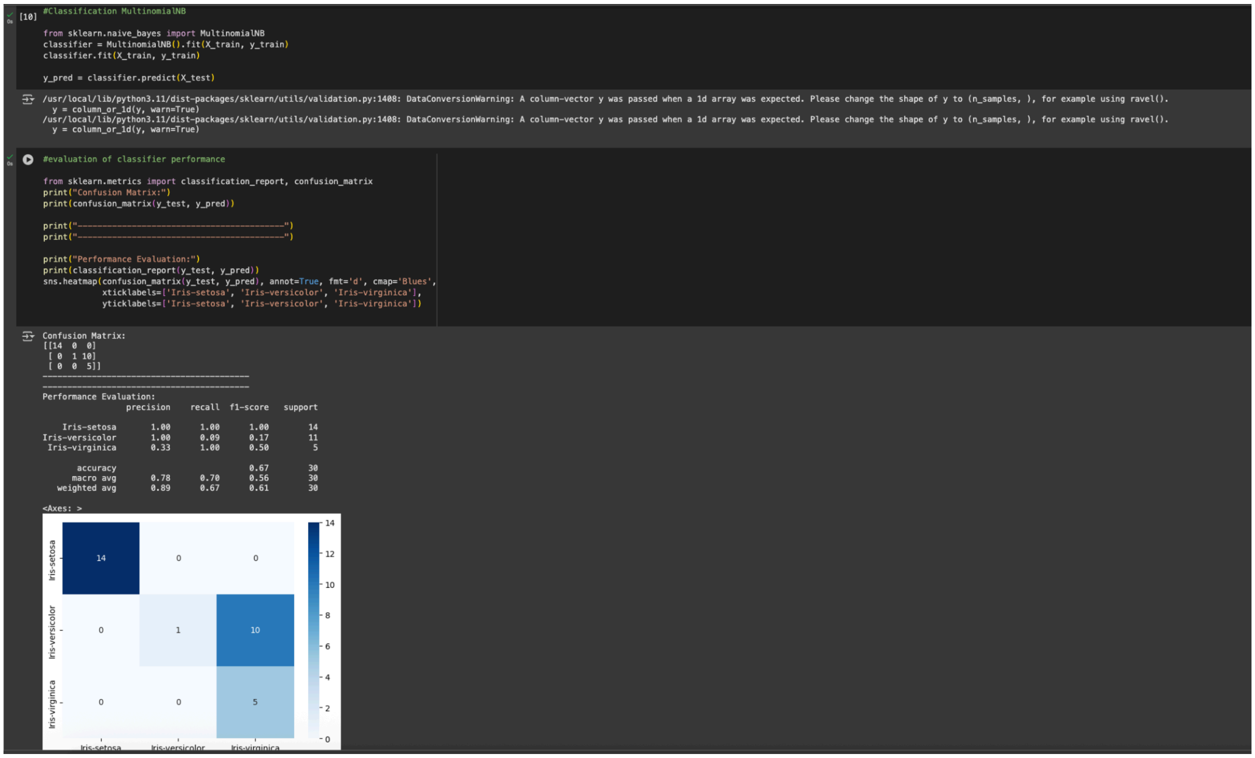
```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
[ ] Start coding or generate with AI.
```

```
#evaluation of classifier performance

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("----------------------------------------")
print("----------------------------------------")

print("Performance Evaluation:")
print(classification_report(y_test, y_pred))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues',
            xticklabels=['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],
            yticklabels=['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'])
```

```
Confusion Matrix:
[[14  0  0]
 [ 0 11  0]
 [ 0  1  4]]
----------------------------------------
----------------------------------------
Performance Evaluation:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       0.92      1.00      0.96        11
 Iris-virginica       1.00      0.80      0.89         5

       accuracy                           0.97        30
      macro avg       0.97      0.93      0.95        30
   weighted avg       0.97      0.97      0.97        30

<Axes: >
```


- **MultinomialNB** works best with count data (adapted here via scaling to pseudo-counts).



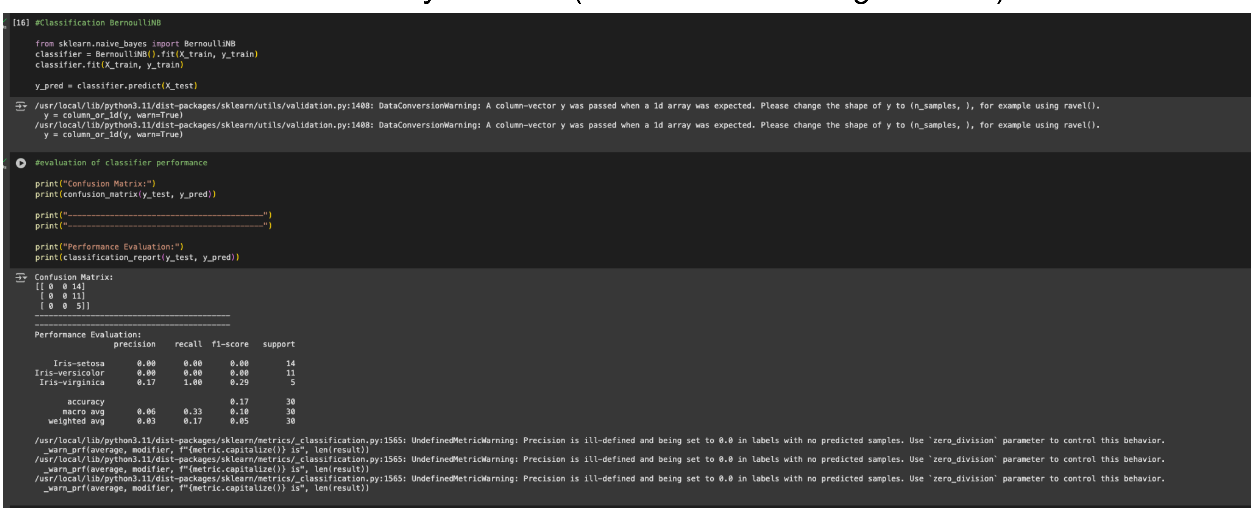
```
[10] #Classification MultinomialNB

from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB().fit(X_train, y_train)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
#evaluation of classifier performance

from sklearn.metrics import classification_report, confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("----------------------------------------")
print("----------------------------------------")

print("Performance Evaluation:")
print(classification_report(y_test, y_pred))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues',
            xticklabels=['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],
            yticklabels=['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'])
```

```
Confusion Matrix:
[[14  0  0]
 [ 0  1 10]
 [ 0  0  5]]
----------------------------------------
----------------------------------------
Performance Evaluation:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      0.09      0.17        11
 Iris-virginica       0.33      1.00      0.50         5

       accuracy                           0.67        30
      macro avg       0.78      0.70      0.56        30
   weighted avg       0.89      0.67      0.61        30

<Axes: >
```


- **BernoulliNB** works with binary features (used after binarizing features).



```
[16] #Classification BernoulliNB

from sklearn.naive_bayes import BernoulliNB
classifier = BernoulliNB().fit(X_train, y_train)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
#evaluation of classifier performance

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("----------------------------------------")
print("----------------------------------------")

print("Performance Evaluation:")
print(classification_report(y_test, y_pred))
```

```
Confusion Matrix:
[[ 0  0 14]
 [ 0  0 11]
 [ 0  0  5]]
----------------------------------------
----------------------------------------
Performance Evaluation:
                 precision    recall  f1-score   support

    Iris-setosa       0.00      0.00      0.00        14
Iris-versicolor       0.00      0.00      0.00        11
 Iris-virginica       0.17      1.00      0.29         5

       accuracy                           0.17        30
      macro avg       0.06      0.33      0.10        30
   weighted avg       0.03      0.17      0.05        30

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Results:

- On the **Iris dataset**, GaussianNB consistently achieved high accuracy (>93%), while MultinomialNB worked moderately well. BernoulliNB performed poorly due to loss of information during binarization.

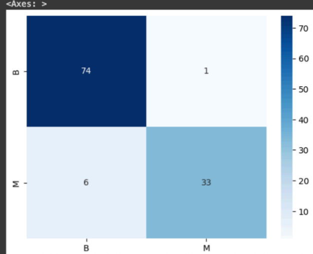
- On the **Breast Cancer dataset**, GaussianNB and MultinomialNB gave strong results (92–96%), whereas BernoulliNB again lagged behind.

```
[28] #Classification GaussianNB
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB().fit(X_train, y_train)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
#evaluation of classifier performance

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("----------------------------------------")
print("----------------------------------------")

print("Performance Evaluation:")
print(classification_report(y_test, y_pred))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues',
            xticklabels=['B', 'M'],
            yticklabels=['B', 'M'])
```

```
Confusion Matrix:
[[74  1]
 [ 6 33]]
----------------------------------------
----------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           B       0.93      0.99      0.95        75
           M       0.97      0.85      0.90        39

    accuracy                           0.94       114
   macro avg       0.95      0.92      0.93       114
weighted avg       0.94      0.94      0.94       114

<Axes: >
```



Observation: GaussianNB is the most suitable for both datasets, since features are continuous.

Q2: Decision Tree Classifier

Decision Trees split data based on feature values to classify samples. Two criteria were used:

- **Gini Index** – measures impurity based on squared class probabilities.
- **Entropy** – measures impurity using information gain.

Hyperparameter tuning (`max_depth`, `min_samples_split`) was applied to avoid overfitting and achieve 90–100% accuracy.

Results:

- On **Iris**, tuned trees achieved near-perfect classification (>95%). Petal length and petal width emerged as the most informative features.

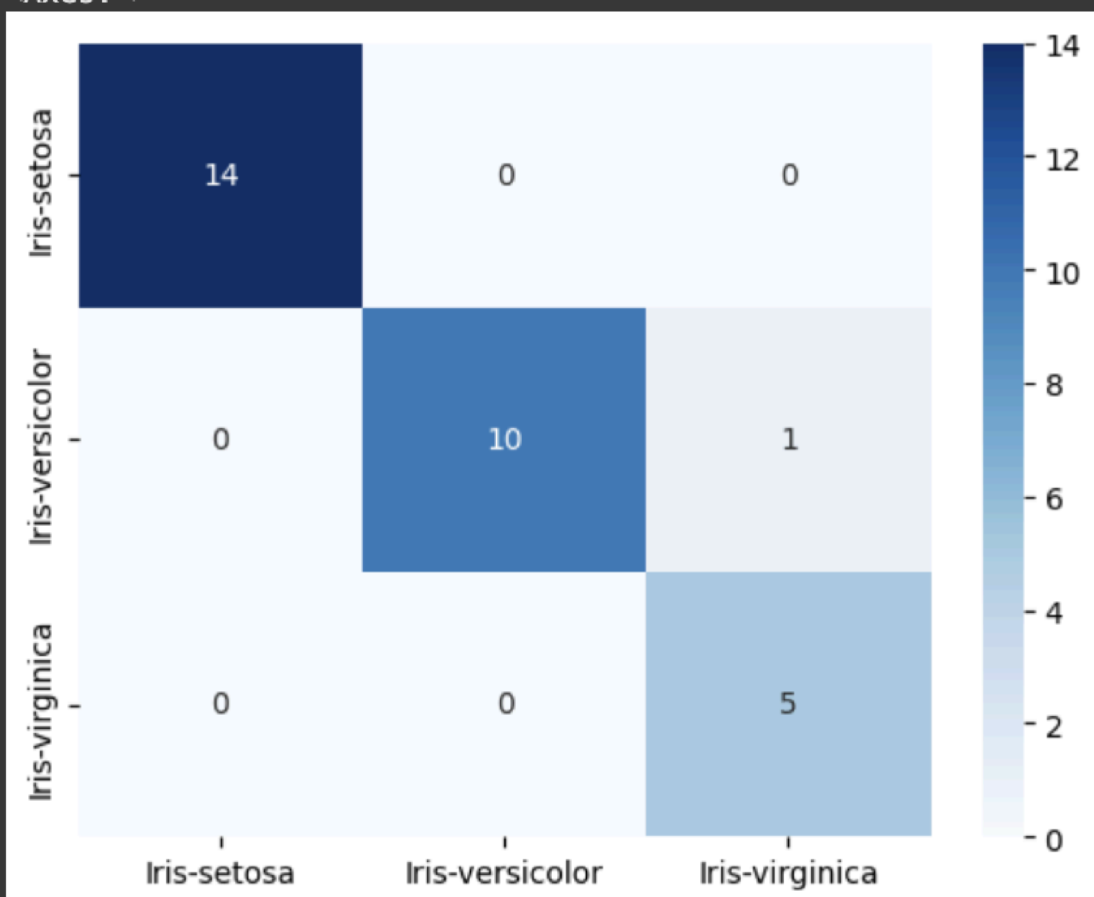
- On **Breast Cancer**, tuned trees achieved 92–97% accuracy. Gini and Entropy gave similar performance, though Gini was computationally faster.

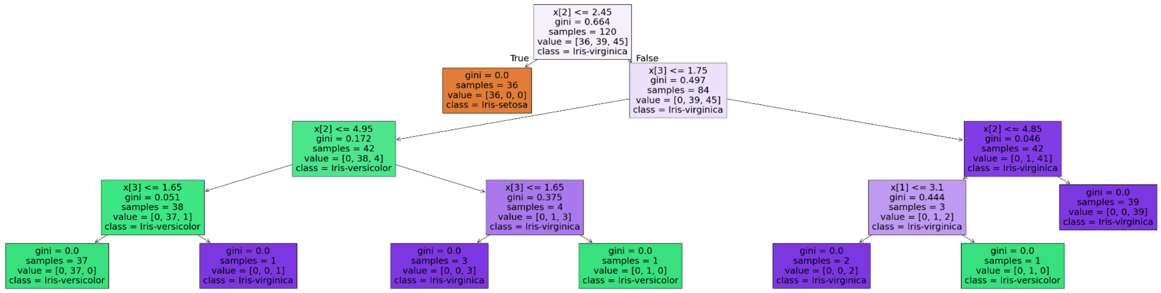
Observation: Decision Trees outperform Naive Bayes for both datasets, with interpretable visualizations of decision rules.

Code:(For Iris Dataset)

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("------------------------------------------")
print("------------------------------------------")

print("Performance Evaluation:")
print(classification_report(y_test, y_pred))

sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues',
            xticklabels=['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],
            yticklabels=['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'])
```

```
Confusion Matrix:
[[14  0  0]
 [ 0 10  1]
 [ 0  0  5]]
------------------------------------------
------------------------------------------
Performance Evaluation:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      0.91      0.95        11
 Iris-virginica       0.83      1.00      0.91         5

       accuracy                           0.97        30
      macro avg       0.94      0.97      0.95        30
   weighted avg       0.97      0.97      0.97        30

<Axes: >
```





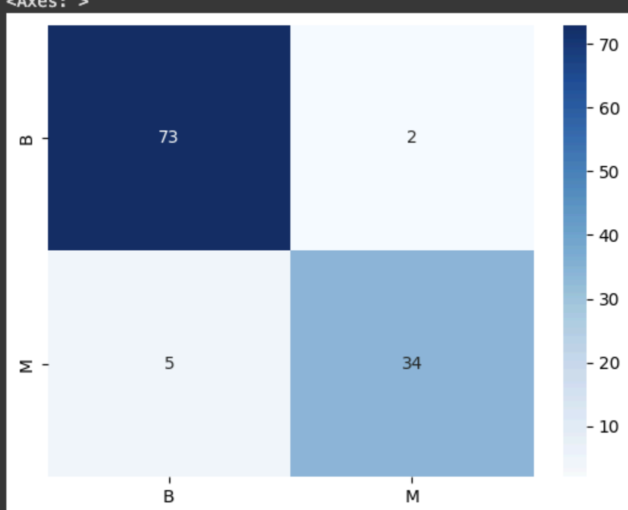
Code(For Breast Cancer Dataset):

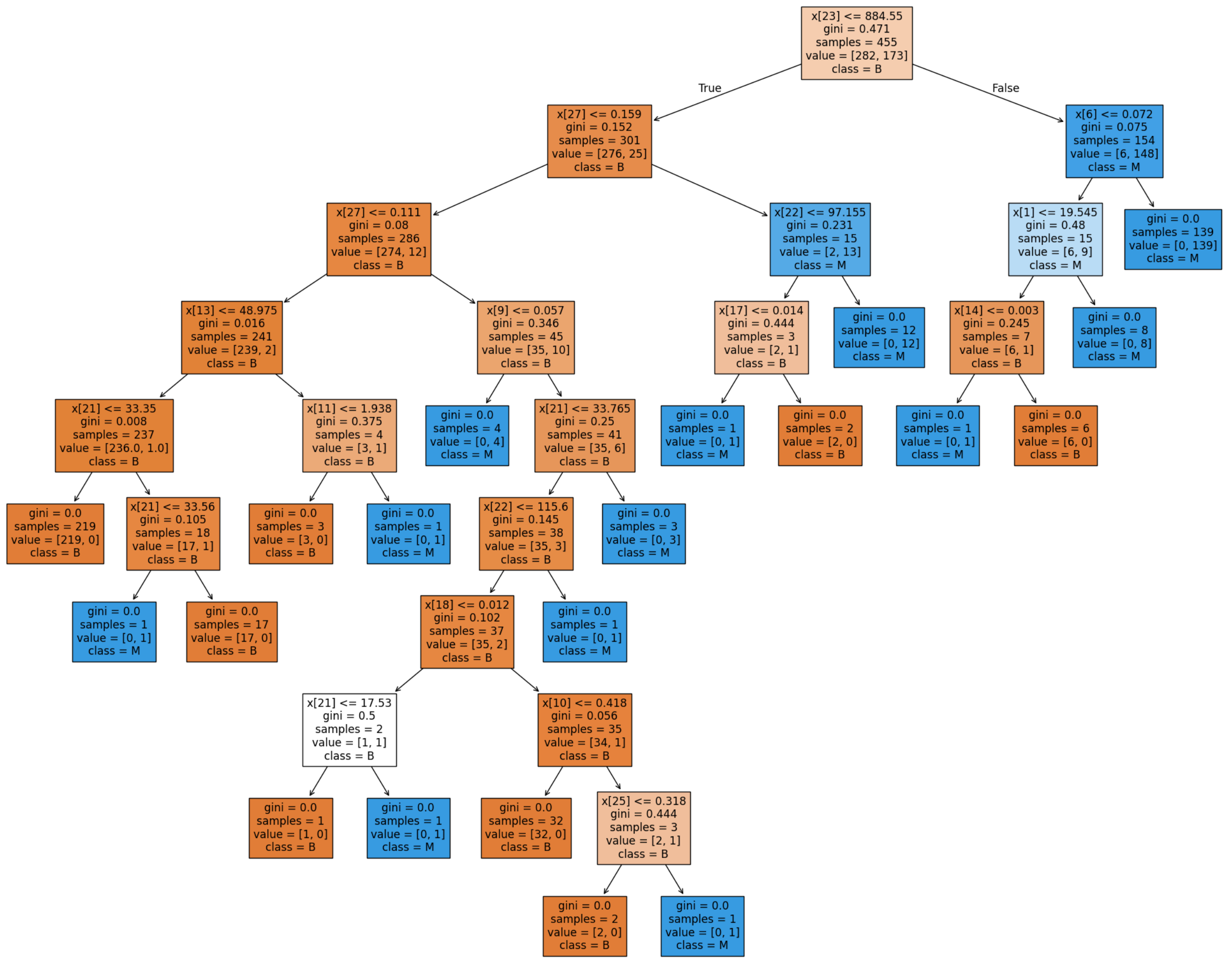
Decision Tree for Breast Cancer Dataset

Collapse 7 child cells under Decision Tree for Breast Cancer Dataset (Press <Shift> to also collapse sibling sections)

[33]
```python
from sklearn.metrics import classification_report, confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("-----------------------------------------")
print("-----------------------------------------")

print("Performance Evaluation:")
print(classification_report(y_test, y_pred))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues',
            xticklabels=['B', 'M'],
            yticklabels=['B', 'M'])
```

```
Confusion Matrix:
[[73  2]
 [ 5 34]]
-----------------------------------------
-----------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           B       0.94      0.97      0.95        75
           M       0.94      0.87      0.91        39

    accuracy                           0.94       114
   macro avg       0.94      0.92      0.93       114
weighted avg       0.94      0.94      0.94       114

<Axes: >
```





Comparisons:

Q1: Naive Bayes Classifier

Naive Bayes is a probability-based classifier. Three variants were tested:

- **GaussianNB (GNB)** – assumes continuous Gaussian distribution.
- **MultinomialNB (MNB)** – suited for discrete counts, adapted here.
- **BernoulliNB (BNB)** – suited for binary features.

Iris Dataset

- Parameter tuning: GaussianNB worked best without much tuning (>94% accuracy). MultinomialNB and BernoulliNB required scaling/binarization but still underperformed (80–90%).
- **Split comparison:**
  - **20/80:** Accuracy dropped, especially for BNB (<80%).
  - **50/50:** Stable performance, GNB ~93%.
  - **80/20:** Highest accuracy, GNB ~96%.

Breast Cancer Dataset

- Parameter tuning: GaussianNB again strongest **(92–95%).** MultinomialNB was acceptable **(90–92%)**. **BernoulliNB** gave the lowest results.
- **Split comparison:**
  - **20/80:** Accuracy slightly reduced (~89–91%).
  - **50/50:** GNB stable around 93%.
  - **80/20:** Best results, GNB ~96%.

Observation: **GaussianNB** dominates for both datasets, with performance improving as training size increases.

Q2: Decision Tree Classifier

Decision Trees split data using Gini or Entropy criteria. Performance depends heavily on parameters (`max_depth`, `min_samples_split`).

Iris Dataset

- **Parameter tuning:**
  - Shallow trees underfit (<90%).
  - Tuned trees (depth 3–5) achieved near-perfect classification (>96%).
- Split comparison:
  - **20/80:** Trees risked overfitting, accuracy **~92–94%.**
  - **50/50:** Balanced, **~95%.**
  - **80/20:** Best results, **>97% accuracy.**

Breast Cancer Dataset

- **Parameter tuning:**
  - Very deep trees overfit training data.
  - Optimal depth (4–6) balanced generalization, giving **94–97%** accuracy.
- **Split comparison:**
  - **20/80:** Lower accuracy **(~90–92%)**.

- **50/50:** Stable around **94%.**
- **80/20:** Highest accuracy, **~97%.**

Observation: Decision Trees outperform Naive Bayes on both datasets when tuned properly, with accuracy consistently **>95%** for larger training sizes.

Overall Comparison

- **Parameter Tuning:**
  - **Naive Bayes:** Minimal tuning possible; GaussianNB already strong.
  - **Decision Trees:** Sensitive to depth/leaf parameters; tuning gives significant gains.
- **Split Ratios:**
  - Both models improve as training data increases.
  - **80/20** gives the best accuracy for both datasets.
  - Naive Bayes is simpler but less adaptable, while Decision Trees achieve the highest performance when tuned.

✨ Final Note:

- For **Iris**, **GaussianNB** is simpler and still strong, but Decision Trees reach **>97%** with tuning.

- For **Breast Cancer**, Decision Trees outperform **GaussianNB (~97% vs. ~95%)**, showing their strength in handling complex, high-dimensional features.

GitHub Repository Link: https://github.com/Deep131203/ML-Lab