Dataset Information

Dataset 1: Iris Dataset

Dataset 2: Breast Cancer Wisconsin Diagnostic Dataset

Note on Dataset Source:

Although the assignment specifies using the UCI datasets, I have chosen to use the versions provided in the scikit-learn (sklearn.datasets) library.

Reason: The sklearn versions include proper *feature_names*, *target_names*, and *metadata* directly, which makes things more convenient.

Implementation Details

1. Mount Storage

```
## Store the images directly in drive or locally

## Comment this out if not required
# drive.mount('/content/drive')

## Comment out one of the below two lines
#folder_path = '/content/drive/My Drive/ML Lab/Ass 1/assets'
folder_path = 'assets'

if not os.path.exists(folder_path):
    os.makedirs(folder_path)
```

2. **evaluate_model() function**

This function automates the model evaluation process by:

- Predicting labels on the test dataset.
- Computing performance metrics: **Accuracy**, **Precision**, **Recall**, and **F1-score**.
- Displaying a detailed **classification report** for each class.
- Generating and visualizing a **confusion matrix**.
- Saving the confusion matrix plot to a file.

```
 def evaluate_model(model, X_test, y_test, class_names=None,
folder=None):

  if class_names is None:
      class_names = np.unique(y_test)

  # Predictions
  y_pred = model.predict(X_test)

  # Metrics
  acc = accuracy_score(y_test, y_pred)
  prec = precision_score(y_test, y_pred, average='macro',
zero_division=0)
```

```
  rec = recall_score(y_test, y_pred, average='macro',
zero_division=0)
  f1 = f1_score(y_test, y_pred, average='macro',
zero_division=0)

  # Print scores
  print(f"Accuracy:  {acc:.4f}")
  print(f"Precision: {prec:.4f}")
  print(f"Recall:    {rec:.4f}")
  print(f"F1-score:  {f1:.4f}")
  print("\nClassification Report:")
  print(classification_report(y_test, y_pred,
target_names=class_names, zero_division=0))

  # Confusion matrix
  cm = confusion_matrix(y_test, y_pred)
  disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=class_names)
  disp.plot(cmap=plt.cm.Blues)
  plt.title('Confusion Matrix')
  plt.xlabel("Predicted")
  plt.ylabel("True")


  # Save Fig
  timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
  if folder is None:
    folder = f"plot_{timestamp}.png"
  save_path = os.path.join(folder_path, folder)
  plt.savefig(save_path)
  print(f"Plot saved to {save_path}")

  plt.show()
```

3. **plot_decision_tree() function**

   This function creates a visual representation of a trained **Decision Tree** model by:

   - Displaying the tree structure with **feature names**, **class labels**, and split criteria.
   - Highlighting **Gini impurity** or **Entropy** values, depending on the chosen criterion.
   - Saving the decision tree diagram as an image.

```
def plot_decision_tree(model, X_test, y_test,
criterion='gini', feature_names=None, class_names=None,
folder=None):
  # Plot decision tree
  plt.figure(figsize=(25, 10))
  plot_tree(
      dt,
      filled=True,
      feature_names=feature_names,
      class_names=class_names,
      rounded=True
  )
  plt.title(f"Decision Tree ({criterion})")

  # Save Fig
  timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
  if folder is None:
    folder = f"decisiontree_{criterion}_{timestamp}.png"
  save_path = os.path.join(folder_path, folder)
  plt.savefig(save_path)
  print(f"Plot saved to {save_path}")

  plt.show()
```

Naive Bayes Classification

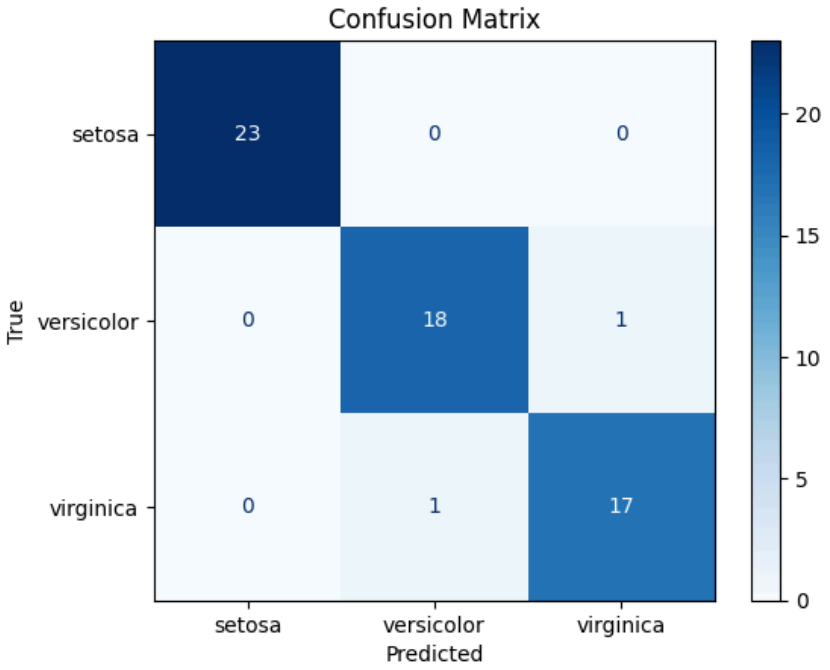
1. Iris Dataset – Naive Bayes Results

```
iris = load_iris()


X = iris.data
y = iris.target

print("Feature names:", iris.feature_names)
print("Target names:", iris.target_names)
print("Description:\n", iris.DESCR[:250], "...")

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.40, random_state=42
)
```

1.1 Gaussian

```
gnb = GaussianNB(var_smoothing=1e-05)

gnb.fit(X_train, y_train)

print("--- Naive Bayes: Gaussian ---")
evaluate_model(gnb, X_test, y_test,
class_names=iris.target_names,
folder="iris_confusion_matrix_gaussian.png")
```

```
--- Naive Bayes: Gaussian ---
Accuracy:  0.9667
Precision: 0.9639
Recall:    0.9639
F1-score:  0.9639

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        23
  versicolor       0.95      0.95      0.95        19
   virginica       0.94      0.94      0.94        18

    accuracy                           0.97        60
   macro avg       0.96      0.96      0.96        60
weighted avg       0.97      0.97      0.97        60

Plot saved to assets/iris_confusion_matrix_gaussian.png
```

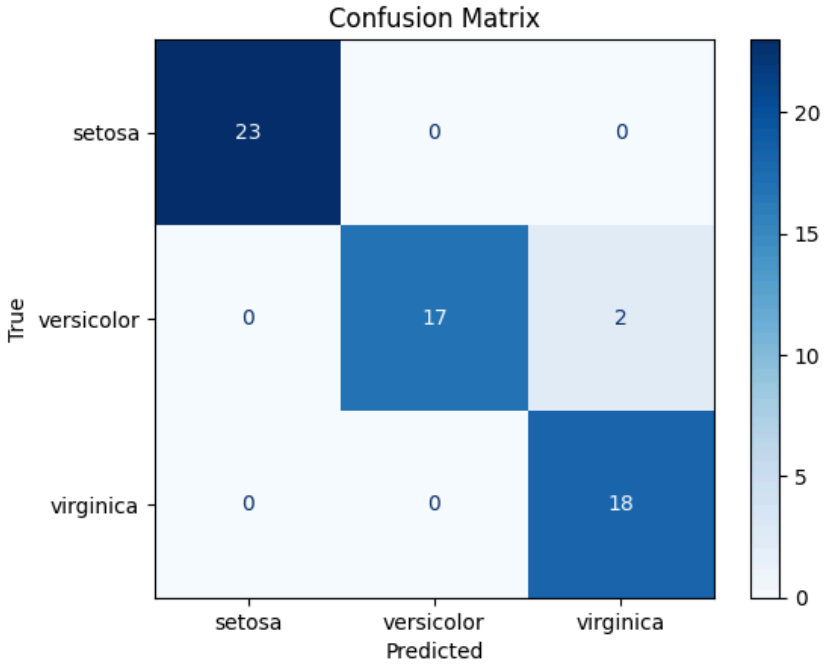


1.2 Multinomial

```
mnb = MultinomialNB(alpha=1.0)

mnb.fit(X_train, y_train)

print("--- Naive Bayes: Multinomial ---")
evaluate_model(mnb, X_test, y_test,
class_names=iris.target_names,
folder="iris_confusion_matrix_multinomial.png")
```

```
--- Naive Bayes: Multinomial ---
Accuracy:  0.9667
Precision: 0.9667
Recall:    0.9649
F1-score:  0.9639

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        23
  versicolor       1.00      0.89      0.94        19
   virginica       0.90      1.00      0.95        18

    accuracy                           0.97        60
   macro avg       0.97      0.96      0.96        60
weighted avg       0.97      0.97      0.97        60
```

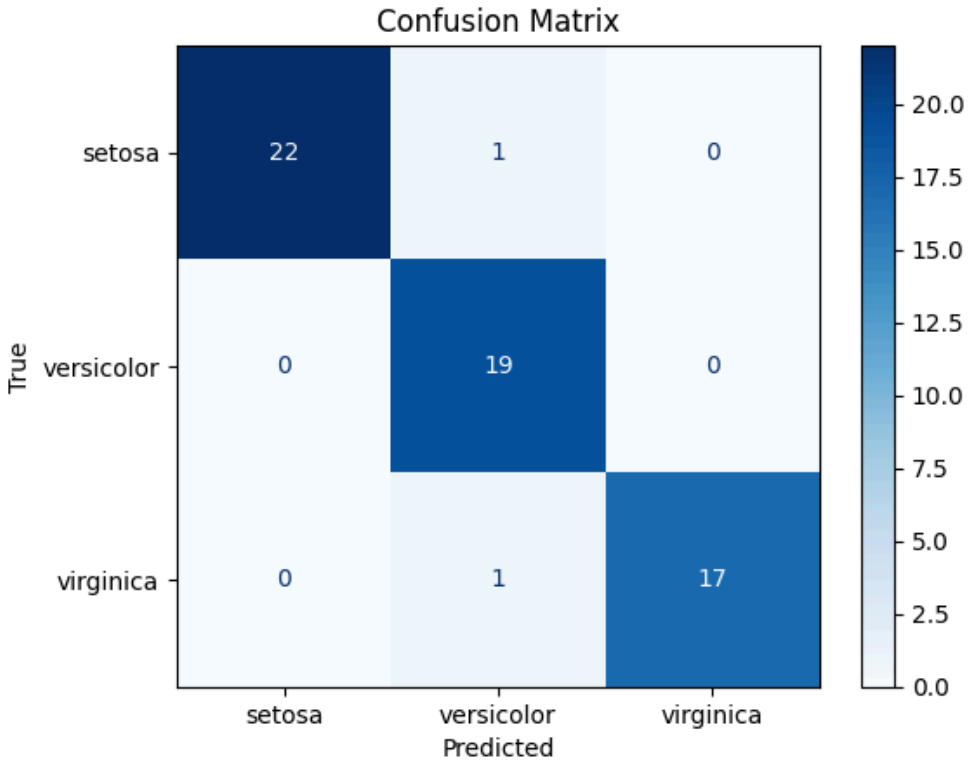
```
Plot saved to assets/iris_confusion_matrix_multinomial.png
```



1.3 Bernoulli

```
bnb = BernoulliNB(alpha=1.0, binarize=1.7)
bnb.fit(X_train, y_train)

print("--- Naive Bayes: Bernoulli ---")
evaluate_model(bnb, X_test, y_test,
class_names=iris.target_names,
folder="iris_confusion_matrix_bernoulli.png")
```

```
--- Naive Bayes: Bernoulli ---
Accuracy:  0.9667
Precision: 0.9683
Recall:    0.9670
F1-score:  0.9664

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      0.96      0.98        23
  versicolor       0.90      1.00      0.95        19
   virginica       1.00      0.94      0.97        18

    accuracy                           0.97        60
   macro avg       0.97      0.97      0.97        60
weighted avg       0.97      0.97      0.97        60

Plot saved to assets/iris_confusion_matrix_bernoulli.png
```


Confusion Matrix
setosa
versicolor
virginica
True
22
1
0
0
19
0
0
1
17
setosa
versicolor
virginica
Predicted
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0

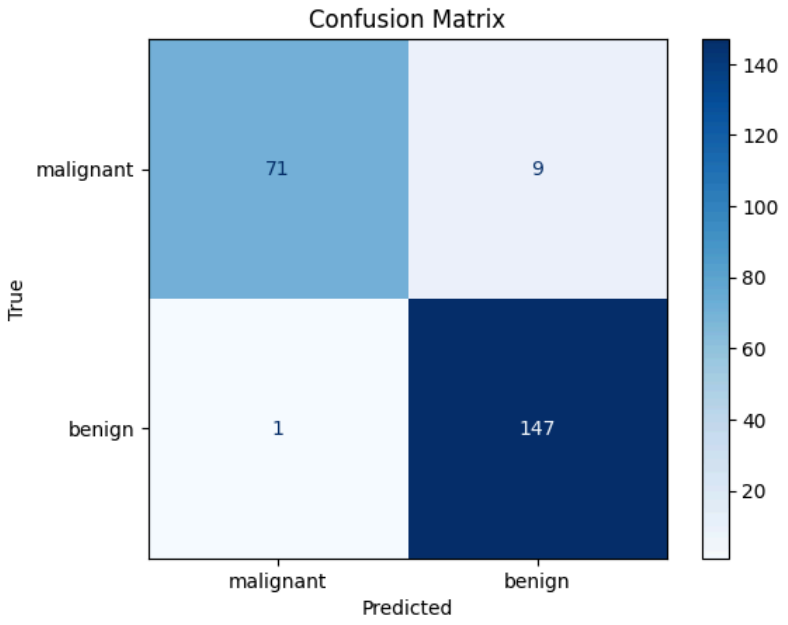
2. Breast Cancer Dataset – Naive Bayes Results

```
bc = load_breast_cancer()


X = bc.data
y = bc.target

print("Feature names:", bc.feature_names)
print("Target names:", bc.target_names)
print("Description:\n", bc.DESCR[:300], "...")

# print(X[:1])
# print(y[:1])

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.40, random_state=42
)
```

2.1 Gaussian

```
gnb = GaussianNB(var_smoothing=1e-05)
gnb.fit(X_train, y_train)

print("--- Naive Bayes: Gaussian ---")
evaluate_model(gnb, X_test, y_test,
class_names=bc.target_names,
folder="breast_cancer_confusion_matrix_gaussian.png")
```

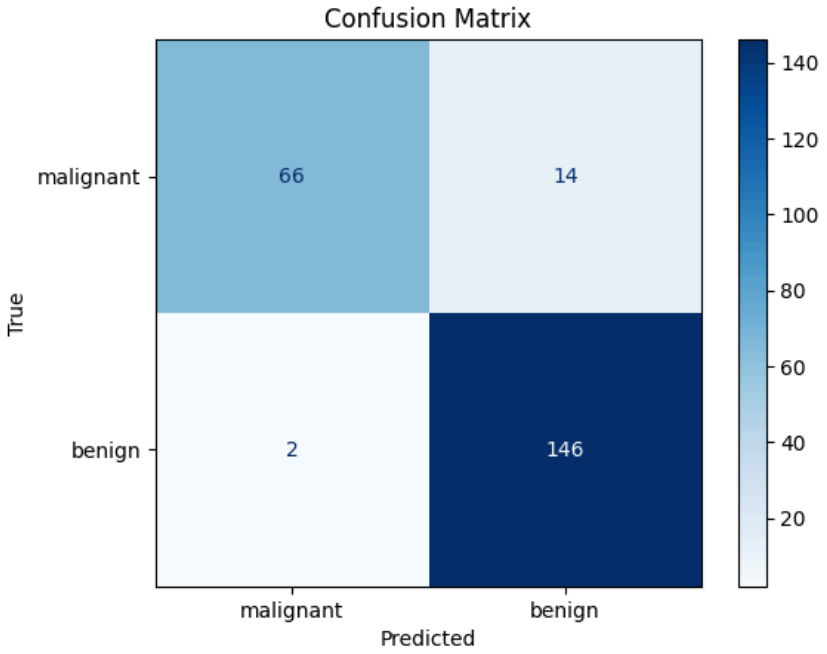
```
 --- Naive Bayes: Gaussian ---
Accuracy:  0.9561
Precision: 0.9642
Recall:    0.9404
F1-score:  0.9507

Classification Report:
              precision    recall  f1-score   support

   malignant       0.99      0.89      0.93        80
      benign       0.94      0.99      0.97       148

    accuracy                           0.96       228
   macro avg       0.96      0.94      0.95       228
weighted avg       0.96      0.96      0.96       228

Plot saved to assets/breast_cancer_confusion_matrix_gaussian.png
```



2.2 Multinomial

```
mnb = MultinomialNB(alpha=1)

mnb.fit(X_train, y_train)

print("--- Naive Bayes: Multinomial ---")
evaluate_model(mnb, X_test, y_test,
class_names=bc.target_names,
folder="breast_cancer_confusion_matrix_multinomial.png")
```

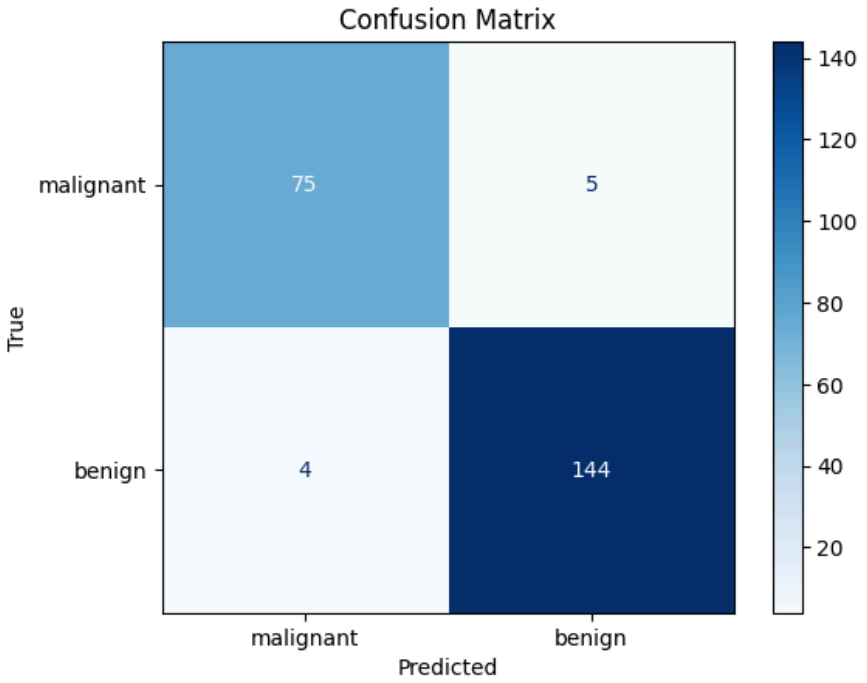
```
--- Naive Bayes: Multinomial ---
Accuracy:  0.9298
Precision: 0.9415
Recall:    0.9057
F1-score:  0.9200

Classification Report:
              precision    recall  f1-score   support

   malignant       0.97      0.82      0.89        80
      benign       0.91      0.99      0.95       148

    accuracy                           0.93       228
   macro avg       0.94      0.91      0.92       228
weighted avg       0.93      0.93      0.93       228

Plot saved to assets/breast_cancer_confusion_matrix_multinomial.png
```



2.3 Bernoulli

```
bnb = BernoulliNB(alpha=1.0, binarize=110)

bnb.fit(X_train, y_train)

print("--- Naive Bayes: Bernoulli ---")
evaluate_model(bnb, X_test, y_test,
class_names=bc.target_names,
folder="breast_cancer_confusion_matrix_bernoulli.png")
```

```
--- Naive Bayes: Bernoulli ---
Accuracy:  0.9605
Precision: 0.9579
Recall:    0.9552
F1-score:  0.9565

Classification Report:
              precision    recall  f1-score   support

   malignant       0.95      0.94      0.94        80
      benign       0.97      0.97      0.97       148

    accuracy                           0.96       228
   macro avg       0.96      0.96      0.96       228
weighted avg       0.96      0.96      0.96       228

Plot saved to assets/breast_cancer_confusion_matrix_bernoulli.png
```


Confusion Matrix
malignant
benign
True
75
5
4
144
malignant
benign
Predicted
140
120
100
80
60
40
20

Decision Tree Classification

1. Iris Dataset – Decision Tree Results

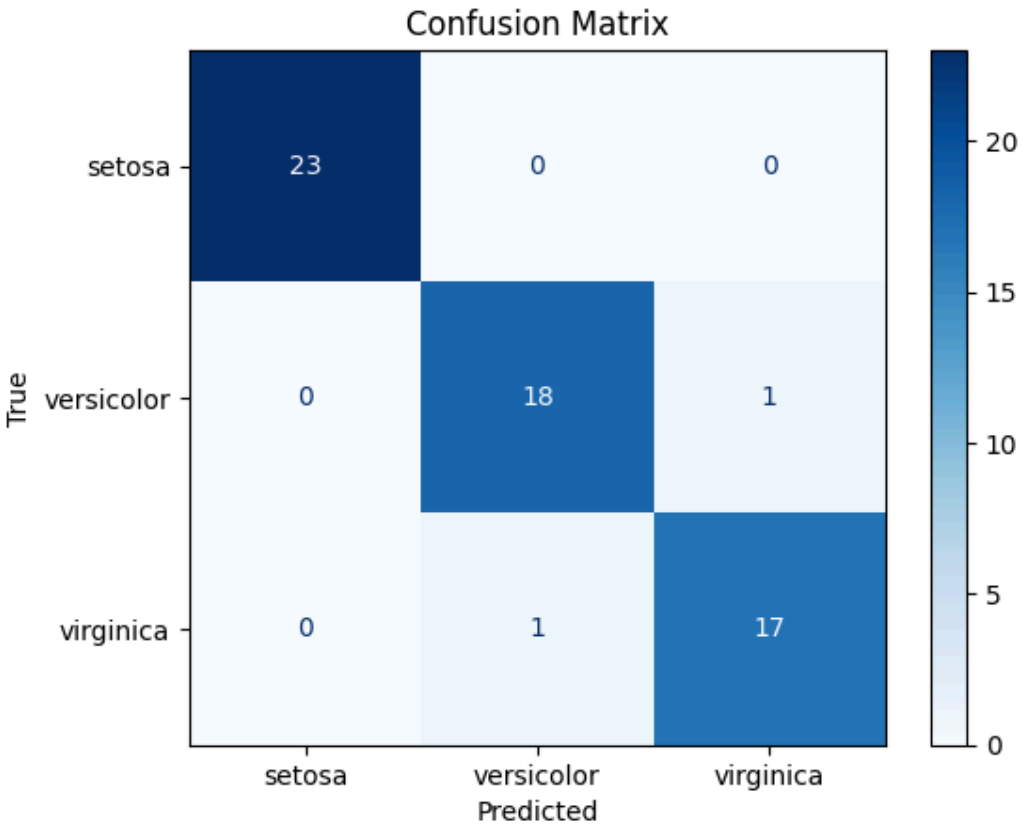
```
iris = load_iris()


X = iris.data
y = iris.target

print("Feature names:", iris.feature_names)
print("Target names:", iris.target_names)
print("Description:\n", iris.DESCR[:250], "...")

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.40, random_state=42
)
```

1.1 Gini

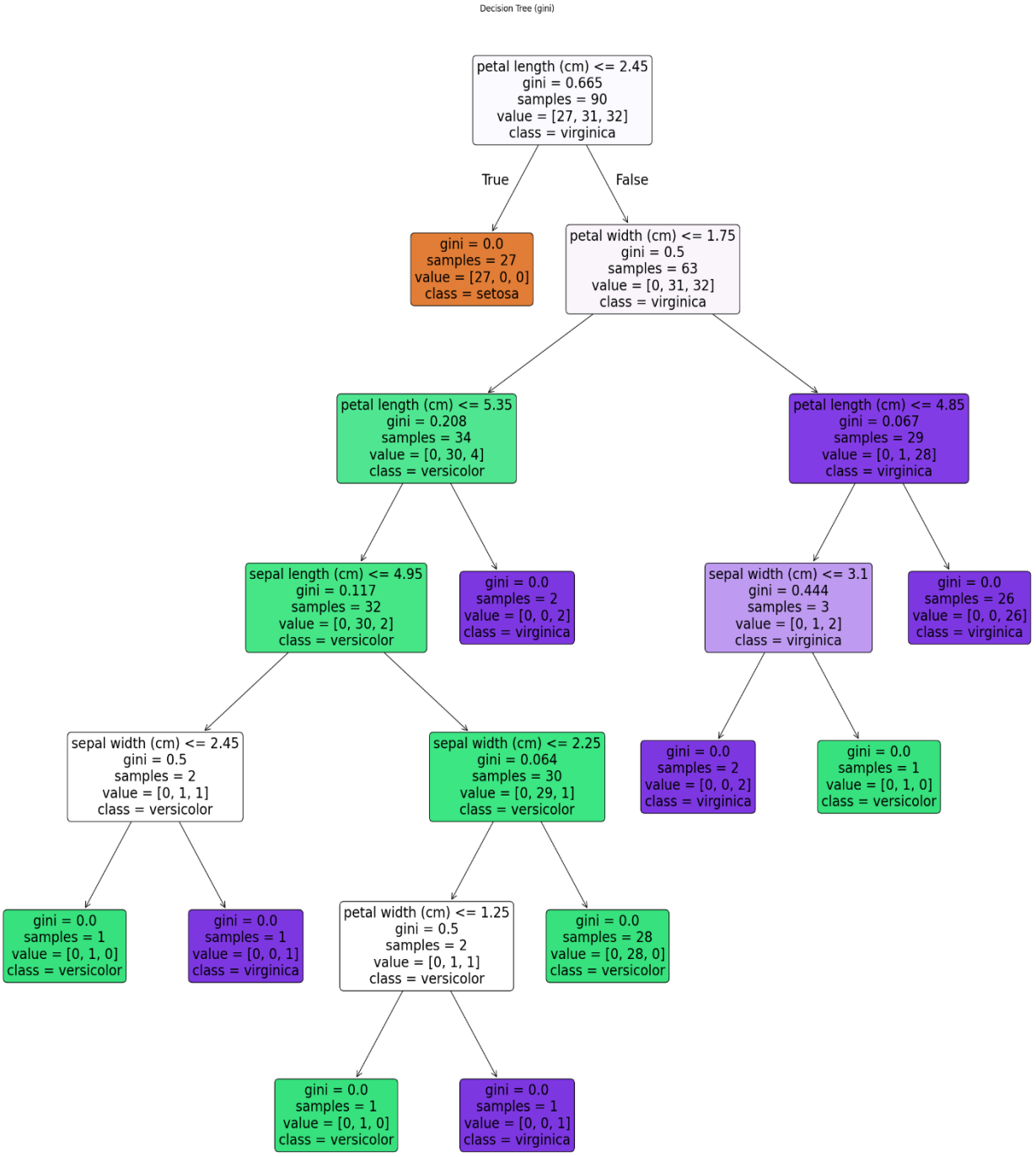
```
print(f"\n--- Decision Tree (gini) ---")


dt = DecisionTreeClassifier(
    criterion="gini",
    max_depth=None,
    random_state=42
)
dt.fit(X_train, y_train)
evaluate_model(dt, X_test, y_test,
class_names=iris.target_names,
folder="iris_confusion_matrix_gini.png")

plot_decision_tree(dt, X_test, y_test, criterion="gini",
feature_names=iris.feature_names,
class_names=iris.target_names,
folder="iris_decision_tree_gini.png")
```

```
--- Decision Tree (gini) ---
Accuracy:  0.9667
Precision: 0.9639
Recall:    0.9639
F1-score:  0.9639
```

```
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        23
  versicolor       0.95      0.95      0.95        19
   virginica       0.94      0.94      0.94        18

    accuracy                           0.97        60
   macro avg       0.96      0.96      0.96        60
weighted avg       0.97      0.97      0.97        60

Plot saved to assets/iris_confusion_matrix_gini.png
```



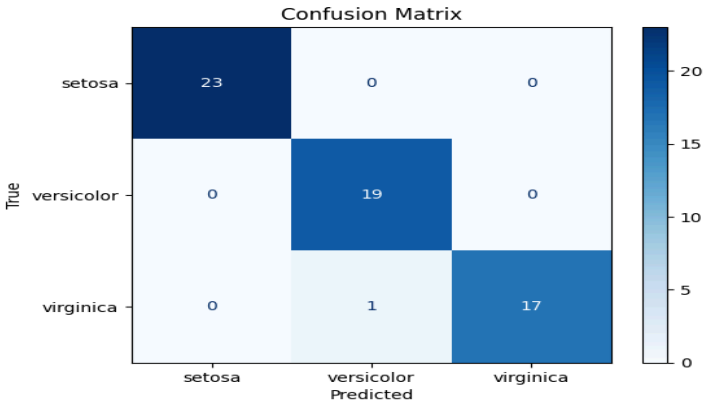

```
Plot saved to assets/iris_decision_tree_gini.png
```


Decision Tree (gini)
petal length (cm) <= 2.45
gini = 0.665
samples = 90
value = [27, 31, 32]
class = virginica
True
False
gini = 0.0
samples = 27
value = [27, 0, 0]
class = setosa
petal width (cm) <= 1.75
gini = 0.5
samples = 63
value = [0, 31, 32]
class = virginica
petal length (cm) <= 5.35
gini = 0.208
samples = 34
value = [0, 30, 4]
class = versicolor
petal length (cm) <= 4.85
gini = 0.067
samples = 29
value = [0, 1, 28]
class = virginica
sepal length (cm) <= 4.95
gini = 0.117
samples = 32
value = [0, 30, 2]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
sepal width (cm) <= 3.1
gini = 0.444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 26
value = [0, 0, 26]
class = virginica
sepal width (cm) <= 2.45
gini = 0.5
samples = 2
value = [0, 1, 1]
class = versicolor
sepal width (cm) <= 2.25
gini = 0.064
samples = 30
value = [0, 29, 1]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
petal width (cm) <= 1.25
gini = 0.5
samples = 2
value = [0, 1, 1]
class = versicolor
gini = 0.0
samples = 28
value = [0, 28, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica

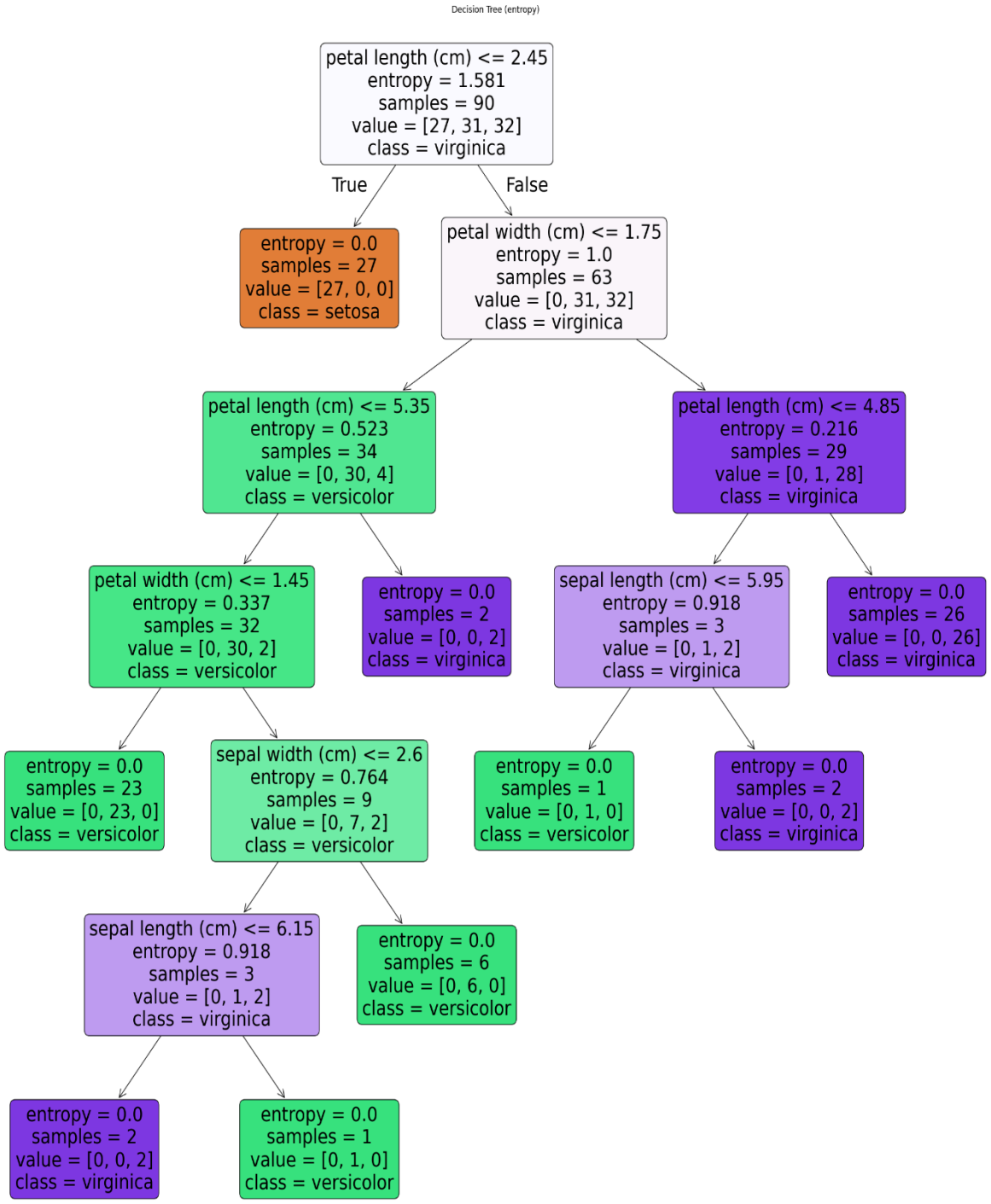
1.2 Entropy

```
print(f"\n--- Decision Tree (entropy) ---")


dt = DecisionTreeClassifier(
    criterion="entropy",
    max_depth=None,
    random_state=42
)
dt.fit(X_train, y_train)
evaluate_model(dt, X_test, y_test,
class_names=iris.target_names,
folder="iris_confusion_matrix_entropy.png")

plot_decision_tree(dt, X_test, y_test, criterion="entropy",
feature_names=iris.feature_names,
class_names=iris.target_names,
folder="iris_decision_tree_entropy.png")
```

```
--- Decision Tree (entropy) ---
Accuracy:  0.9833
Precision: 0.9833
Recall:    0.9815
F1-score:  0.9819

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        23
  versicolor       0.95      1.00      0.97        19
   virginica       1.00      0.94      0.97        18

    accuracy                           0.98        60
   macro avg       0.98      0.98      0.98        60
weighted avg       0.98      0.98      0.98        60

Plot saved to assets/iris_confusion_matrix_entropy.png
```




```
Plot saved to assets/iris_decision_tree_entropy.png
```


Decision Tree (entropy)
petal length (cm) <= 2.45
entropy = 1.581
samples = 90
value = [27, 31, 32]
class = virginica
True
False
entropy = 0.0
samples = 27
value = [27, 0, 0]
class = setosa
petal width (cm) <= 1.75
entropy = 1.0
samples = 63
value = [0, 31, 32]
class = virginica
petal length (cm) <= 5.35
entropy = 0.523
samples = 34
value = [0, 30, 4]
class = versicolor
petal length (cm) <= 4.85
entropy = 0.216
samples = 29
value = [0, 1, 28]
class = virginica
petal width (cm) <= 1.45
entropy = 0.337
samples = 32
value = [0, 30, 2]
class = versicolor
entropy = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
sepal length (cm) <= 5.95
entropy = 0.918
samples = 3
value = [0, 1, 2]
class = virginica
entropy = 0.0
samples = 26
value = [0, 0, 26]
class = virginica
entropy = 0.0
samples = 23
value = [0, 23, 0]
class = versicolor
sepal width (cm) <= 2.6
entropy = 0.764
samples = 9
value = [0, 7, 2]
class = versicolor
entropy = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
entropy = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
sepal length (cm) <= 6.15
entropy = 0.918
samples = 3
value = [0, 1, 2]
class = virginica
entropy = 0.0
samples = 6
value = [0, 6, 0]
class = versicolor
entropy = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
entropy = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor

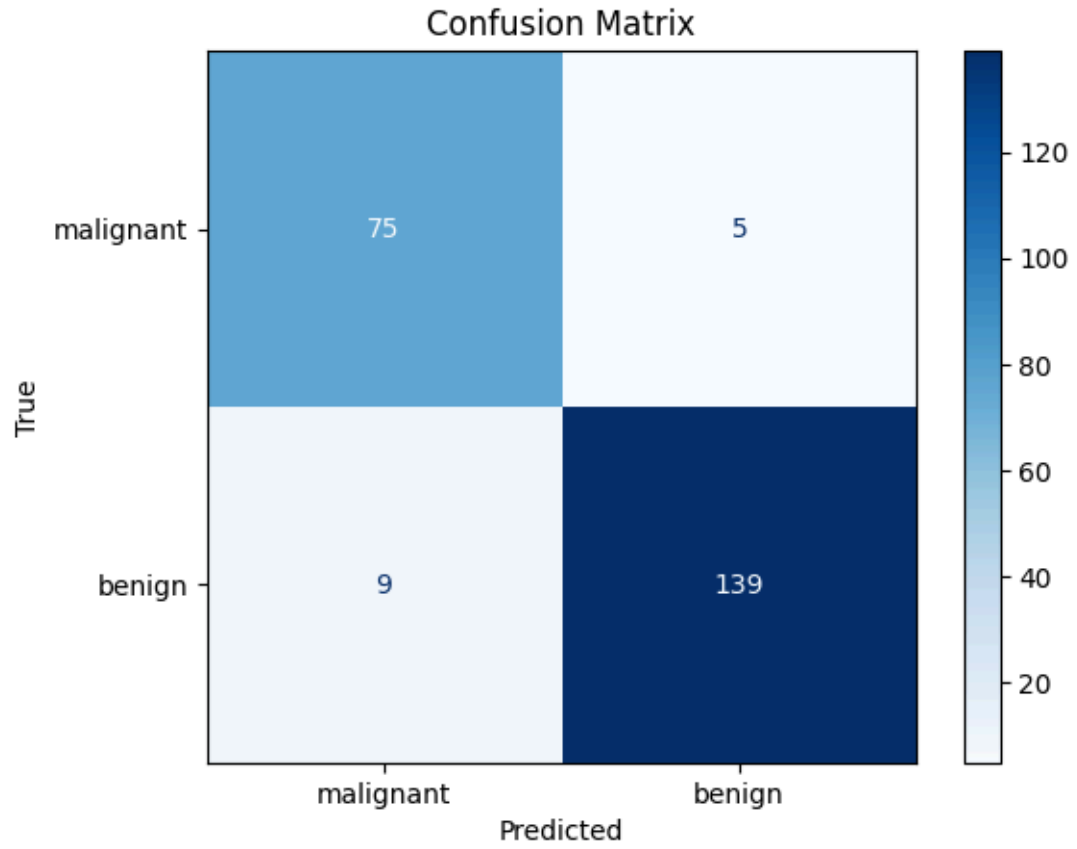
2. Breast Cancer Dataset – Decision Tree Results

```
bc = load_breast_cancer()


X = bc.data
y = bc.target

print("Feature names:", bc.feature_names)
print("Target names:", bc.target_names)
print("Description:\n", bc.DESCR[:300], "...")

# print(X[:1])
# print(y[:1])

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.40, random_state=42
)
```

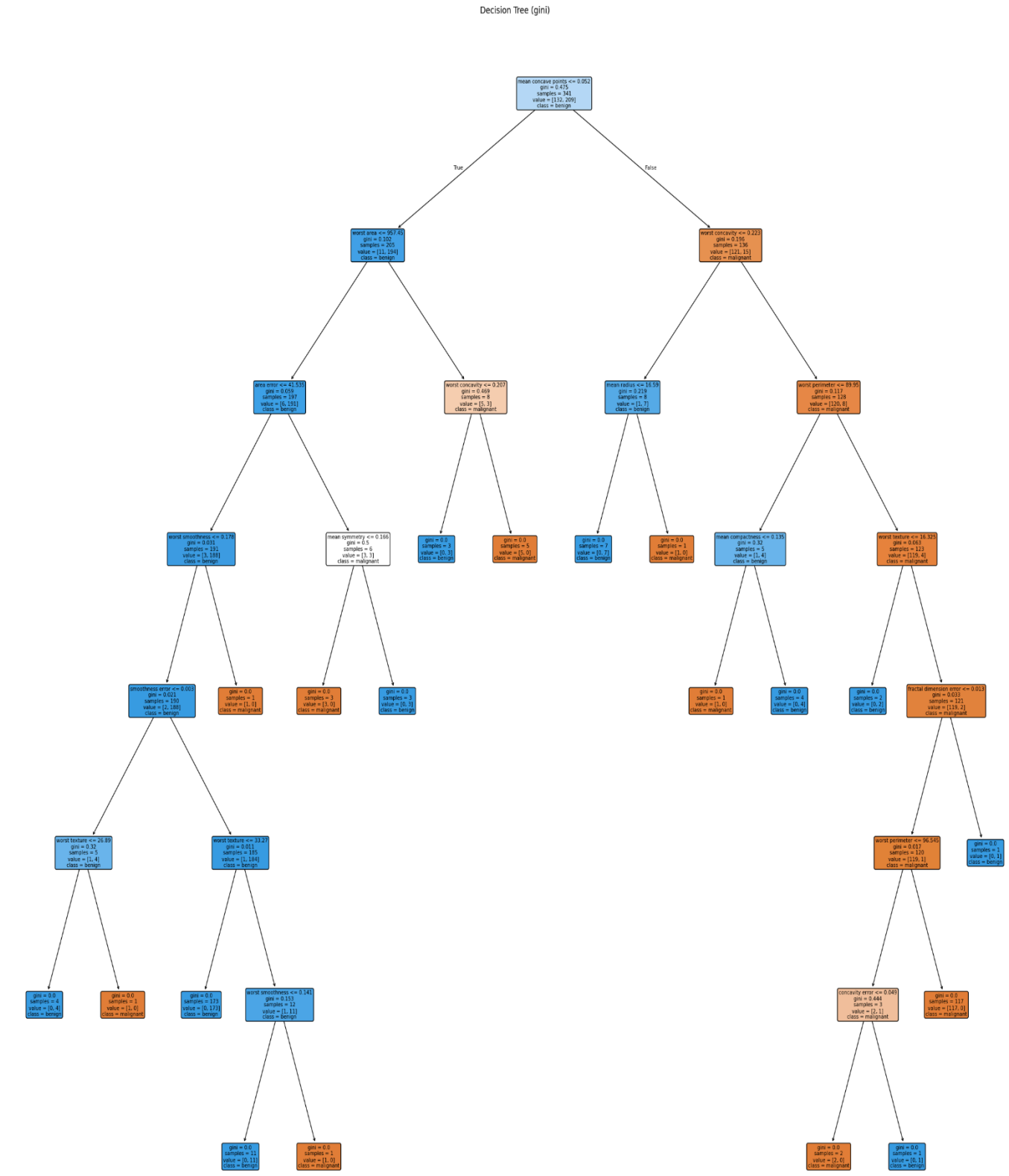
2.1 Gini

```
print(f"\n--- Decision Tree (gini) ---")

dt = DecisionTreeClassifier(
    criterion="gini",
    max_depth=None,
    random_state=42
)
dt.fit(X_train, y_train)
evaluate_model(dt, X_test, y_test,
class_names=bc.target_names,
folder="breast_cancer_confusion_matrix_gini.png")

plot_decision_tree(dt, X_test, y_test, criterion="gini",
feature_names=bc.feature_names, class_names=bc.target_names,
folder="breast_cancer_decision_tree_gini.png")
```

```
--- Decision Tree (gini) ---
Accuracy:  0.9386
Precision: 0.9291
Recall:    0.9383
F1-score:  0.9333

Classification Report:
              precision    recall  f1-score   support

   malignant       0.89      0.94      0.91        80
      benign       0.97      0.94      0.95       148

    accuracy                           0.94       228
   macro avg       0.93      0.94      0.93       228
weighted avg       0.94      0.94      0.94       228

Plot saved to assets/breast_cancer_confusion_matrix_gini.png
```



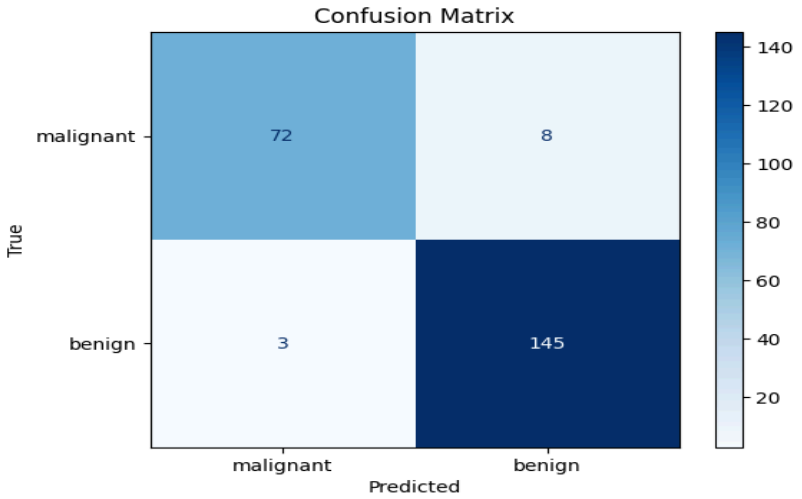

```
Plot saved to assets/breast_cancer_decision_tree_gini.png
```


Decision Tree (gini)
mean concave points <= 0.052
gini = 0.475
samples = 341
value = [132, 209]
class = benign
True
False
worst area <= 957.45
gini = 0.102
samples = 205
value = [11, 194]
class = benign
worst concavity <= 0.223
gini = 0.196
samples = 136
value = [121, 15]
class = malignant
area error <= 41.535
gini = 0.059
samples = 197
value = [6, 191]
class = benign
worst concavity <= 0.207
gini = 0.469
samples = 8
value = [5, 3]
class = malignant
mean radius <= 16.55
gini = 0.219
samples = 8
value = [1, 7]
class = benign
worst perimeter <= 89.95
gini = 0.117
samples = 128
value = [120, 8]
class = malignant
worst smoothness <= 0.178
gini = 0.031
samples = 191
value = [3, 188]
class = benign
mean symmetry <= 0.166
gini = 0.5
samples = 6
value = [3, 3]
class = malignant
gini = 0.0
samples = 3
value = [0, 3]
class = benign
gini = 0.0
samples = 5
value = [5, 0]
class = malignant
gini = 0.0
samples = 7
value = [0, 7]
class = benign
gini = 0.0
samples = 1
value = [1, 0]
class = malignant
mean compactness <= 0.135
gini = 0.32
samples = 5
value = [1, 4]
class = benign
worst texture <= 16.325
gini = 0.063
samples = 123
value = [119, 4]
class = malignant
smoothness error <= 0.003
gini = 0.021
samples = 190
value = [2, 188]
class = benign
gini = 0.0
samples = 1
value = [1, 0]
class = malignant
gini = 0.0
samples = 3
value = [3, 0]
class = malignant
gini = 0.0
samples = 3
value = [0, 3]
class = benign
gini = 0.0
samples = 1
value = [1, 0]
class = malignant
gini = 0.0
samples = 4
value = [0, 4]
class = benign
gini = 0.0
samples = 2
value = [0, 2]
class = benign
fractal dimension error <= 0.013
gini = 0.033
samples = 121
value = [119, 2]
class = malignant
worst texture <= 26.39
gini = 0.32
samples = 5
value = [1, 4]
class = benign
worst texture <= 33.27
gini = 0.01
samples = 185
value = [1, 184]
class = benign
worst perimeter <= 96.545
gini = 0.017
samples = 120
value = [119, 1]
class = malignant
gini = 0.0
samples = 1
value = [0, 1]
class = benign
gini = 0.0
samples = 4
value = [0, 4]
class = benign
gini = 0.0
samples = 1
value = [1, 0]
class = malignant
gini = 0.0
samples = 173
value = [0, 173]
class = benign
worst smoothness <= 0.141
gini = 0.153
samples = 12
value = [1, 11]
class = benign
concavity error <= 0.049
gini = 0.444
samples = 3
value = [2, 1]
class = malignant
gini = 0.0
samples = 117
value = [117, 0]
class = malignant
gini = 0.0
samples = 11
value = [0, 11]
class = benign
gini = 0.0
samples = 1
value = [1, 0]
class = malignant
gini = 0.0
samples = 2
value = [2, 0]
class = malignant
gini = 0.0
samples = 1
value = [0, 1]
class = benign

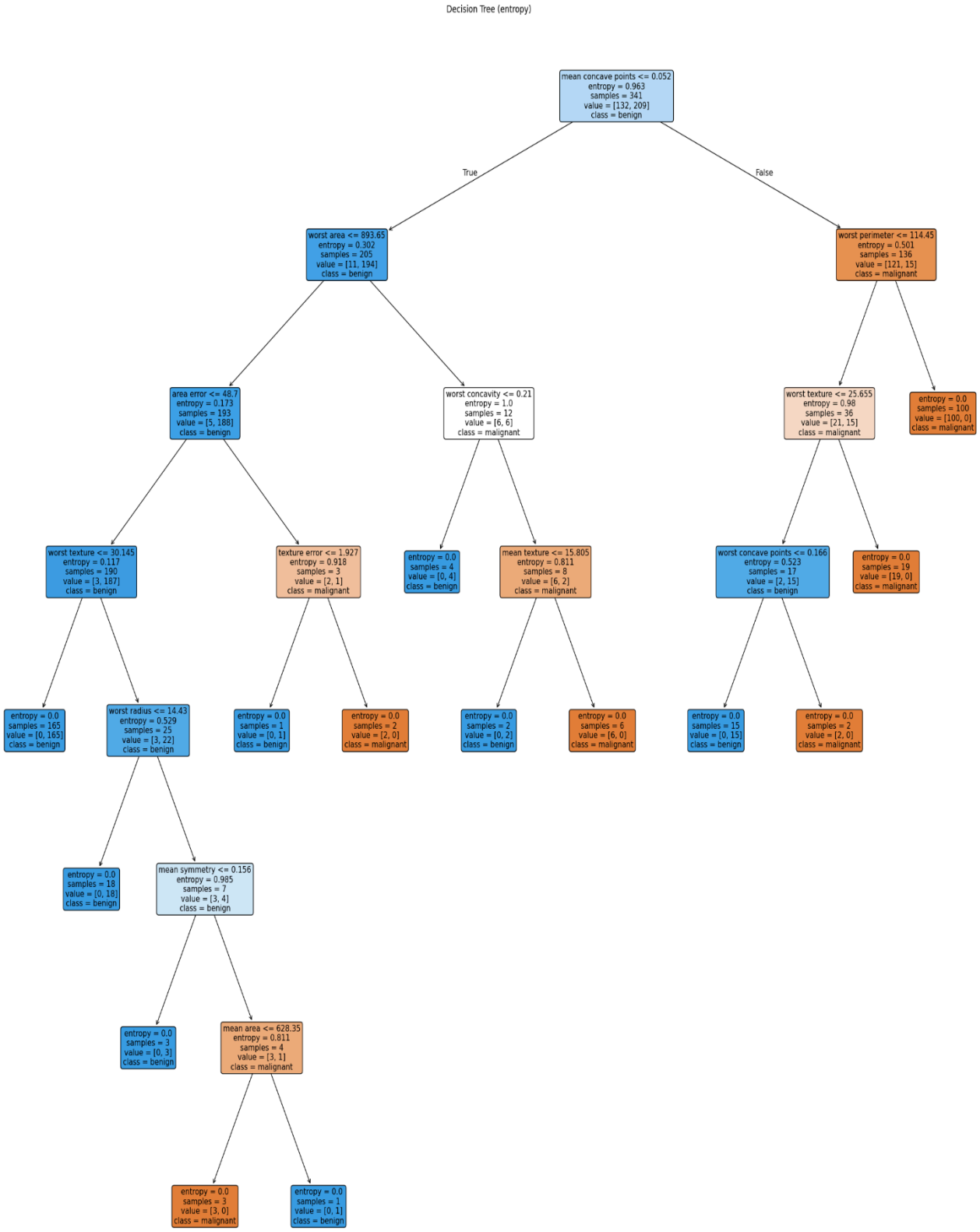
2.2 Entropy

```
print(f"\n--- Decision Tree (entropy) ---")

dt = DecisionTreeClassifier(
    criterion="entropy",
    max_depth=None,
    random_state=42
)
dt.fit(X_train, y_train)
evaluate_model(dt, X_test, y_test,
class_names=bc.target_names,
folder="breast_cancer_confusion_matrix_entropy.png")

plot_decision_tree(dt, X_test, y_test, criterion="entropy",
feature_names=bc.feature_names, class_names=bc.target_names,
folder="breast_cancer_decision_tree_entropy.png")
```

```
--- Decision Tree (entropy) ---
Accuracy:  0.9518
Precision: 0.9539
Recall:    0.9399
F1-score:  0.9462

Classification Report:
              precision    recall  f1-score   support

   malignant       0.96      0.90      0.93        80
      benign       0.95      0.98      0.96       148

    accuracy                           0.95       228
   macro avg       0.95      0.94      0.95       228
weighted avg       0.95      0.95      0.95       228

Plot saved to assets/breast_cancer_confusion_matrix_entropy.png
```




```
Plot saved to assets/breast_cancer_decision_tree_entropy.png
```


Decision Tree (entropy)
mean concave points <= 0.052
entropy = 0.963
samples = 341
value = [132, 209]
class = benign
True
False
worst area <= 893.65
entropy = 0.302
samples = 205
value = [11, 194]
class = benign
worst perimeter <= 114.45
entropy = 0.501
samples = 136
value = [121, 15]
class = malignant
area error <= 48.7
entropy = 0.173
samples = 193
value = [5, 188]
class = benign
worst concavity <= 0.21
entropy = 1.0
samples = 12
value = [6, 6]
class = malignant
worst texture <= 25.655
entropy = 0.98
samples = 36
value = [21, 15]
class = malignant
entropy = 0.0
samples = 100
value = [100, 0]
class = malignant
worst texture <= 30.145
entropy = 0.117
samples = 190
value = [3, 187]
class = benign
texture error <= 1.927
entropy = 0.918
samples = 3
value = [2, 1]
class = malignant
entropy = 0.0
samples = 4
value = [0, 4]
class = benign
mean texture <= 15.805
entropy = 0.811
samples = 8
value = [6, 2]
class = malignant
worst concave points <= 0.166
entropy = 0.523
samples = 17
value = [2, 15]
class = benign
entropy = 0.0
samples = 19
value = [19, 0]
class = malignant
entropy = 0.0
samples = 165
value = [0, 165]
class = benign
worst radius <= 14.43
entropy = 0.529
samples = 25
value = [3, 22]
class = benign
entropy = 0.0
samples = 1
value = [0, 1]
class = benign
entropy = 0.0
samples = 2
value = [2, 0]
class = malignant
entropy = 0.0
samples = 2
value = [0, 2]
class = benign
entropy = 0.0
samples = 6
value = [6, 0]
class = malignant
entropy = 0.0
samples = 15
value = [0, 15]
class = benign
entropy = 0.0
samples = 2
value = [2, 0]
class = malignant
entropy = 0.0
samples = 18
value = [0, 18]
class = benign
mean symmetry <= 0.156
entropy = 0.985
samples = 7
value = [3, 4]
class = benign
entropy = 0.0
samples = 3
value = [0, 3]
class = benign
mean area <= 628.35
entropy = 0.811
samples = 4
value = [3, 1]
class = malignant
entropy = 0.0
samples = 3
value = [3, 0]
class = malignant
entropy = 0.0
samples = 1
value = [0, 1]
class = benign

Observations

Naive Bayes Observations

1. Gaussian Naive Bayes
    - On both datasets, GaussianNB performed well - Iris: 96.67%, Breast Cancer: 95.61% - due to its ability to model continuous-valued features.
2. Multinomial Naive Bayes
    - Hyperparameters: alpha = 1.0 (Laplace smoothing)
    - Despite being designed for discrete counts, it was tested on continuous features for comparison and achieved performance comparable to GaussianNB – Iris: 96.67%, Breast Cancer: 92.98%.
3. Bernoulli Naive Bayes
    - Iris Hyperparameters: alpha = 1.0, binarize = 1.7
    - Breast Cancer Hyperparameters: alpha = 1.0, binarize = 110
    - The binarize parameter threshold converted continuous features into binary form (values above threshold set to 1, otherwise 0).
    - This approach still yielded competitive results — Iris: 96.67%, Breast Cancer: 96.05% — showing that binary transformation can be effective even for originally continuous features, though interpretability may be reduced.

Decision Tree Observations

1. Gini Criterion
    - Hyperparameters: criterion = 'gini', max_depth = None, random_state = 42
    - Fully grown trees achieved strong accuracy on both datasets (Iris: 96.67%, Breast Cancer: 93.86%).
    - Overfitting issues observed in cancer dataset.
2. Entropy Criterion
    - Hyperparameters: criterion = 'entropy', max_depth = None, random_state = 42
    - Performed slightly better than Gini in most cases (Iris: 98.33%, Breast Cancer: 95.18%), indicating that information gain sometimes produces better splits for these datasets.
    - Overfitting issues observed in cancer dataset.

General Findings

- GaussianNB consistently performed strongly.
- BernoulliNB’s binarization produced competitive accuracy, but its applicability depends on selecting an appropriate threshold.
- MultinomialNB was less effective in comparison for continuous datasets, as it is more suitable for count-based or categorical data.
- Decision Trees with the entropy criterion slightly outperformed Gini across both datasets.