MACHINE LEARNING LABORATORY

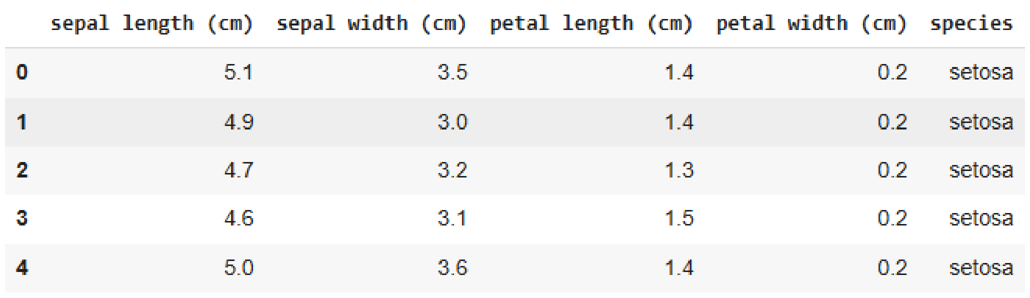
ASSIGNMENT 1
Name: Atmik Goswami
Roll No: 002211001104
Section: A2

Github: github.com/atmikgoswami/ML-Lab

DATASETS:

1. **Iris Plants Dataset**
   - Features: Sepal Length, Sepal Width, Petal Length, Petal Width
   - Classes: Setosa, Versicolor, Virginica
   - Total Samples: 150

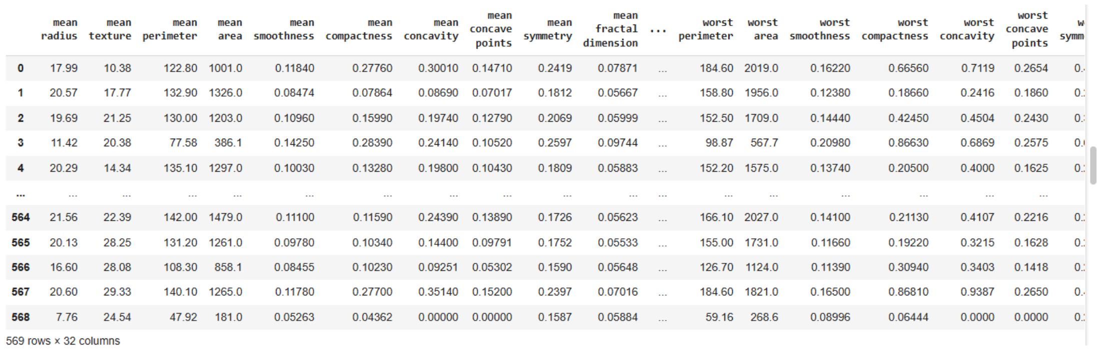
Sample Data:



	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

2. **Breast Cancer Wisconsin (Diagnostic) Dataset**
   - Features: 30 numeric features (e.g., radius, texture, area, etc.)
   - Classes: Malignant (M), Benign (B)
   - Total Samples: 569

Sample Data:



	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	w symm
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	184.60	2019.0	0.16220	0.66560	0.7119	0.2654	0.
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	158.80	1956.0	0.12380	0.18660	0.2416	0.1860	0.
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	152.50	1709.0	0.14440	0.42450	0.4504	0.2430	0.
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	98.87	567.7	0.20980	0.86630	0.6869	0.2575	0.
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	152.20	1575.0	0.13740	0.20500	0.4000	0.1625	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	166.10	2027.0	0.14100	0.21130	0.4107	0.2216	0.
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	155.00	1731.0	0.11660	0.19220	0.3215	0.1628	0.
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	126.70	1124.0	0.11390	0.30940	0.3403	0.1418	0.
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	184.60	1821.0	0.16500	0.86810	0.9387	0.2650	0.
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	59.16	268.6	0.08996	0.06444	0.0000	0.0000	0.

569 rows × 32 columns

1. Employ Naive Bayes (Gaussian, Multinomial & Bernoulli) classifier and show classification results (Accuracy, Precision, Recall, F-score, confusion matrix).

Code:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
```

```
from sklearn.datasets import load_iris

iris = load_iris()
```

```
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['species'] = iris.target
df['species'] = df['species'].map({0: 'setosa', 1: 'versicolor', 2:
'virginica'})
```

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

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

```
from sklearn.naive_bayes import MultinomialNB

classifier = MultinomialNB(alpha=1.5, fit_prior=False)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

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

print("-------------------------------------------------------")
print("Classification Report")
print(classification_report(y_test, y_pred))
```

```
from sklearn.naive_bayes import GaussianNB
```

```
classifier = GaussianNB()
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

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

print("-------------------------------------------------------")
print("Classification Report")
print(classification_report(y_test, y_pred))
```

```
from sklearn.naive_bayes import BernoulliNB

classifier = BernoulliNB(binarize=1.5)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

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

print("-------------------------------------------------------")
print("Classification Report")
print(classification_report(y_test, y_pred))
```

```
from sklearn.datasets import load_breast_cancer

data = load_breast_cancer()
```

```
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
df['target_label'] = df['target'].map({0: 'malignant', 1: 'benign'})
```

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

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

```
from sklearn.naive_bayes import MultinomialNB

classifier = MultinomialNB()
```

```python
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

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

print("-------------------------------------------------------")
print("Classification Report")
print(classification_report(y_test, y_pred))
```

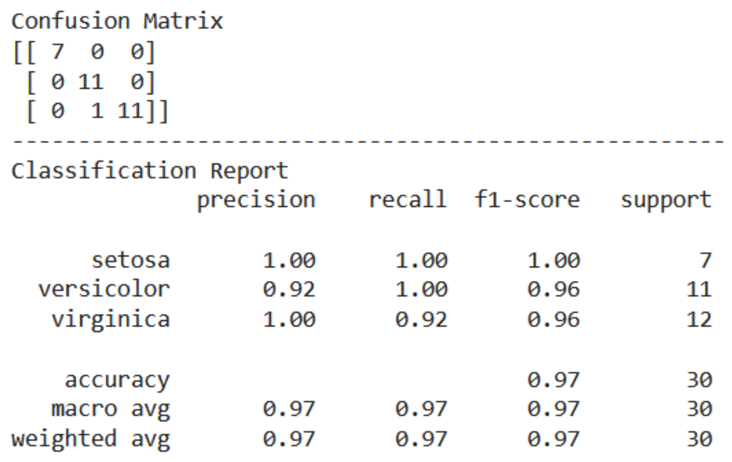
```python
from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB(var_smoothing=1e-09)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

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

print("-------------------------------------------------------")
print("Classification Report")
print(classification_report(y_test, y_pred))
```

```python
from sklearn.naive_bayes import BernoulliNB

classifier = BernoulliNB(binarize=1.5, fit_prior=False,
class_prior=[0.8, 0.2])
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

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

print("-------------------------------------------------------")
print("Classification Report")
print(classification_report(y_test, y_pred))
```

Results and Discussion:

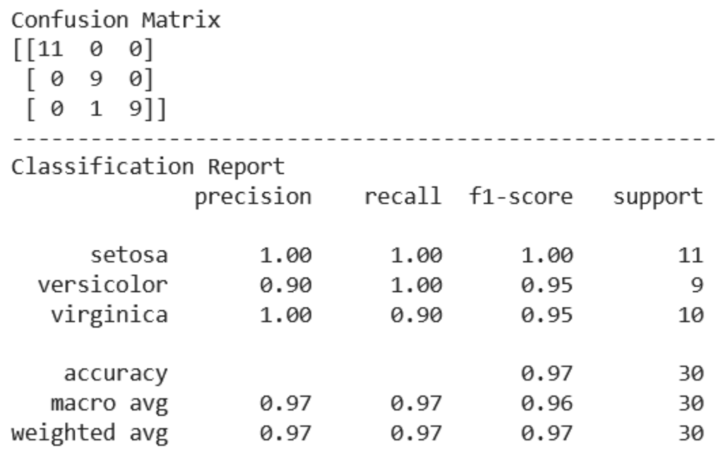
Iris Dataset:

Multinomial Naive Bayes:



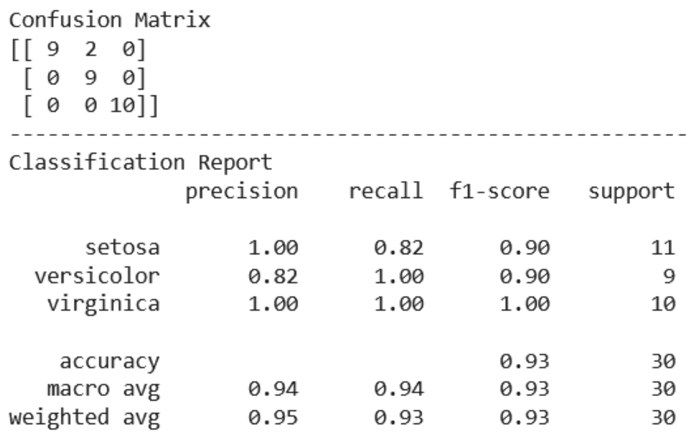
```
Confusion Matrix
[[ 7  0  0]
 [ 0 11  0]
 [ 0  1 11]]
-------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00         7
  versicolor       0.92      1.00      0.96        11
   virginica       1.00      0.92      0.96        12

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
```

Gaussian Naive Bayes:



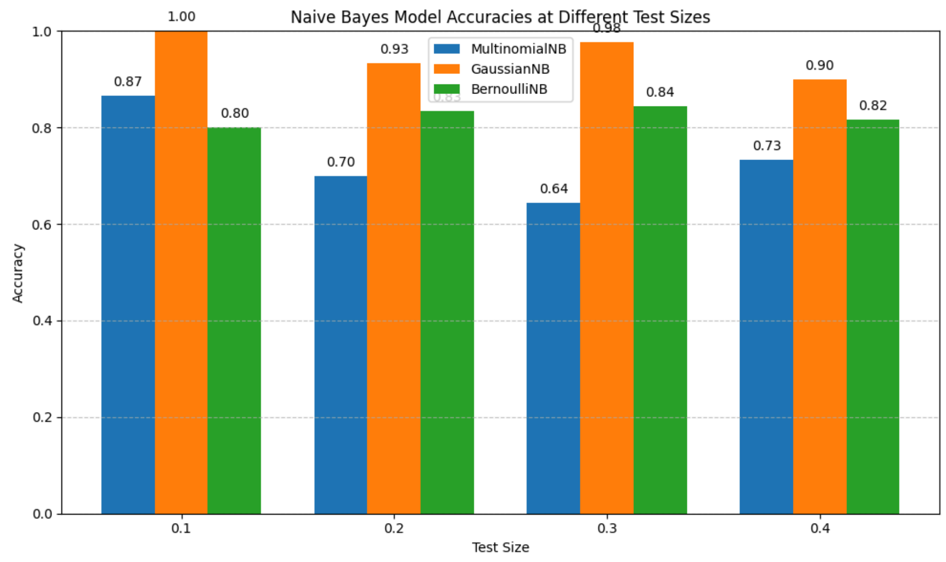
```
Confusion Matrix
[[11  0  0]
 [ 0  9  0]
 [ 0  1  9]]
-------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        11
  versicolor       0.90      1.00      0.95         9
   virginica       1.00      0.90      0.95        10

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

Bernoulli Naive Bayes:



```
Confusion Matrix
[[ 9  2  0]
 [ 0  9  0]
 [ 0  0 10]]
-------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

      setosa       1.00      0.82      0.90        11
  versicolor       0.82      1.00      0.90         9
   virginica       1.00      1.00      1.00        10

    accuracy                           0.93        30
   macro avg       0.94      0.94      0.93        30
weighted avg       0.95      0.93      0.93        30
```

Comparison of accuracies for different test sizes:



Wisconsin Breast Cancer Dataset

Multinomial Naive Bayes:

```
Confusion Matrix
[[27  7]
 [ 1 79]]
--------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           0       0.96      0.79      0.87        34
           1       0.92      0.99      0.95        80

    accuracy                           0.93       114
   macro avg       0.94      0.89      0.91       114
weighted avg       0.93      0.93      0.93       114
```

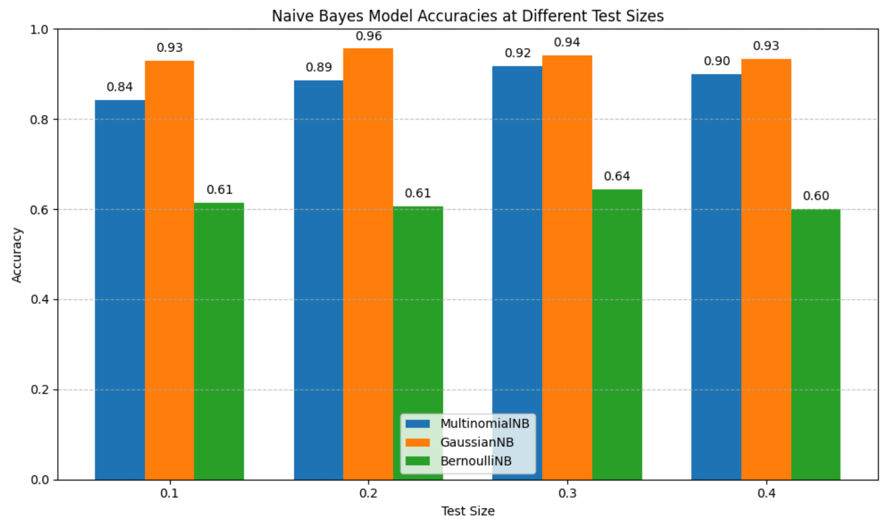
Gaussian Naive Bayes:

```
Confusion Matrix
[[31  3]
 [ 0 80]]
--------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           0       1.00      0.91      0.95        34
           1       0.96      1.00      0.98        80

    accuracy                           0.97       114
   macro avg       0.98      0.96      0.97       114
weighted avg       0.97      0.97      0.97       114
```

Bernoulli Naive Bayes:

```
Confusion Matrix
[[16 30]
 [ 0 68]]
------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           0       1.00      0.35      0.52        46
           1       0.69      1.00      0.82        68

    accuracy                           0.74       114
   macro avg       0.85      0.67      0.67       114
weighted avg       0.82      0.74      0.70       114
```

Comparison of accuracies for different test sizes:



Discussion:

Naive Bayes classifiers assume feature independence and use probability distributions to classify data. For the **Iris dataset**, Gaussian Naive Bayes generally performed best, as the features (lengths and widths) are continuous and approximately normally distributed. Multinomial Naive Bayes was less effective since it is designed for count-based features, and Bernoulli Naive Bayes was the least suitable because binarization of continuous features led to information loss. This highlights how model assumptions directly impact classification performance. Accuracy and F-scores were higher for Gaussian NB, showing that modeling continuous distributions is appropriate for this dataset.

For the **Breast Cancer dataset**, Gaussian Naive Bayes again outperformed the other variants due to the continuous, real-valued features like radius, texture, and area. Multinomial Naive Bayes, while usable, is not ideal for such data and gave comparatively lower results. Bernoulli Naive Bayes struggled most because binarization discards crucial variance across 30 features. Overall, Gaussian NB provided robust results with good precision and recall for both malignant and benign classes. The confusion matrices showed very few misclassifications, reinforcing its effectiveness.

Thus, the key observation is that Gaussian NB consistently works well for continuous feature datasets, while Multinomial and Bernoulli are more suited for text or count data.

2. Use Decision Tree classifier for all the two datasets and show classification results (Accuracy, Precision, Recall, F-score, confusion matrix). Generate the decision tree images for all cases highlighting information like Gini and Entropy.

Code:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
```

```
from sklearn.datasets import load_iris

iris = load_iris()
```

```
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['species'] = iris.target
df['species'] = df['species'].map({0: 'setosa', 1: 'versicolor', 2:
'virginica'})
```

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

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

```
classifier = DecisionTreeClassifier(random_state=1)

param_grid = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [2, 3, 4, 5, 6],
    'min_samples_split': [2, 3, 4, 5],
    'min_samples_leaf': [1, 2, 3, 4, 5],
    'max_features': [None, 'sqrt', 'log2']
}

# Grid search
```

```
grid = GridSearchCV(classifier, param_grid, cv=5, scoring='accuracy')
grid.fit(X_train, y_train)

# Best model
best_model = grid.best_estimator_
print("Best Parameters:", grid.best_params_)

# Evaluate on test set
y_pred = best_model.predict(X_test)

print("\nConfusion Matrix")
print(confusion_matrix(y_test, y_pred))

print("-------------------------------------------------------")
print("Classification Report")
print(classification_report(y_test, y_pred,
target_names=iris.target_names))
```

```
# Plotting of decision tree
from IPython.display import Image
from sklearn.tree import export_graphviz

!pip install pydotplus
import pydotplus


features = X.columns
dot_data = export_graphviz(best_model, out_file=None,
feature_names=iris.feature_names)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

```
from sklearn.datasets import load_breast_cancer

data = load_breast_cancer()
```

```
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
df['target_label'] = df['target'].map({0: 'malignant', 1: 'benign'})
```

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

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

```
classifier = DecisionTreeClassifier(random_state=1)

param_grid = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [2, 3, 4, 5, 6],
    'min_samples_split': [2, 3, 4, 5],
    'min_samples_leaf': [1, 2, 3, 4, 5],
    'max_features': [None, 'sqrt', 'log2']
}

# Grid search
grid = GridSearchCV(classifier, param_grid, cv=5, scoring='accuracy')
grid.fit(X_train, y_train)

# Best model
best_model = grid.best_estimator_
print("Best Parameters:", grid.best_params_)

# Evaluate on test set
y_pred = best_model.predict(X_test)

print("\nConfusion Matrix")
print(confusion_matrix(y_test, y_pred))

print("------------------------------------------------------")
print("Classification Report")
print(classification_report(y_test, y_pred,
target_names=data.target_names))
```

```
# Plotting of decision tree
from IPython.display import Image
from sklearn.tree import export_graphviz

!pip install pydotplus
import pydotplus

features = X.columns
dot_data = export_graphviz(best_model, out_file=None,
feature_names=data.feature_names)
```

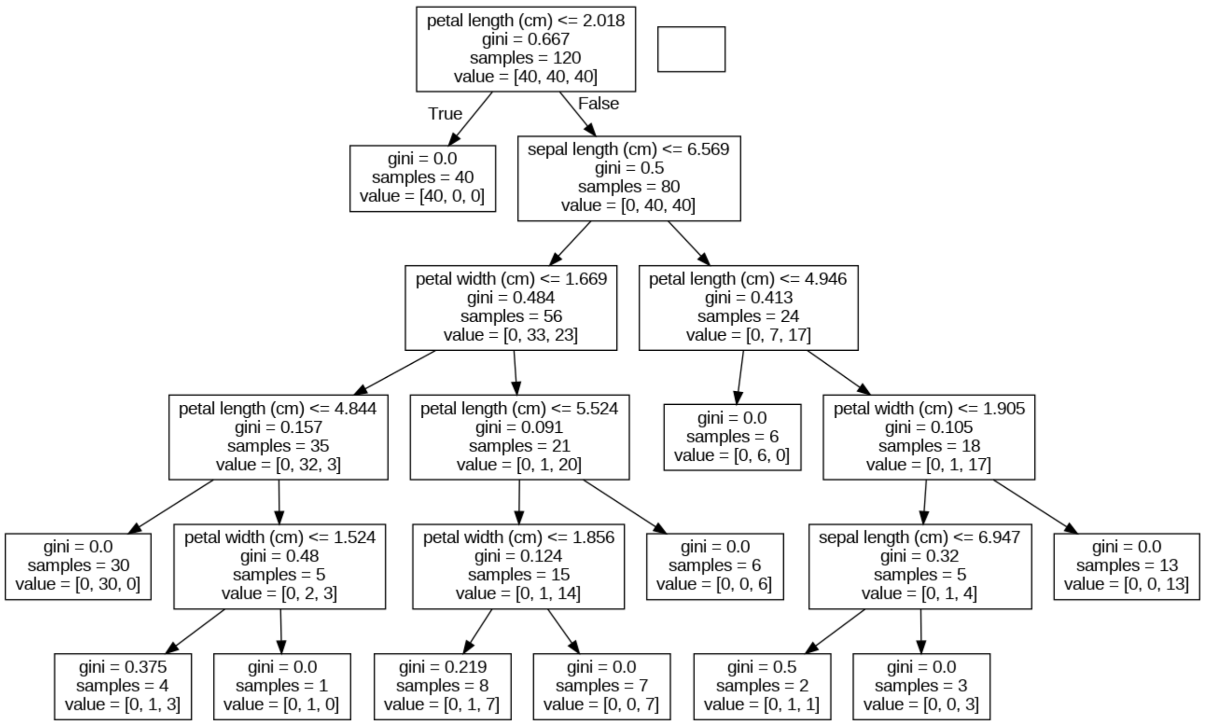
```
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

Results and Discussion:

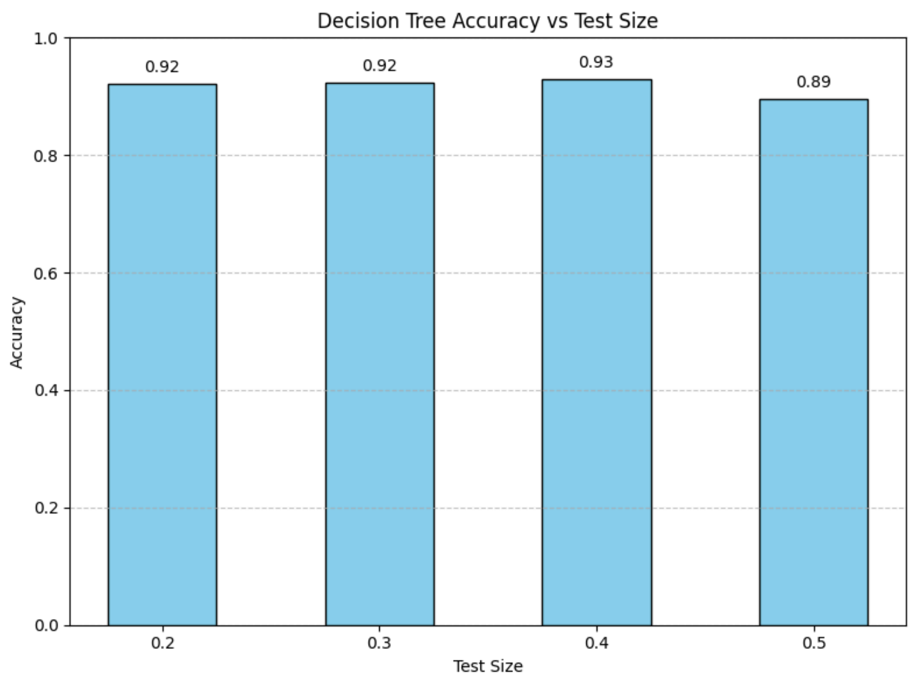
Iris Dataset:

```
Confusion Matrix
[[10  0  0]
 [ 0 10  0]
 [ 0  0 10]]
------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00        10
   virginica       1.00      1.00      1.00        10

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Decision Tree:


petal length (cm) <= 2.018 gini = 0.667 samples = 120 value = [40, 40, 40]
True
False
gini = 0.0 samples = 40 value = [40, 0, 0]
sepal length (cm) <= 6.569 gini = 0.5 samples = 80 value = [0, 40, 40]
petal width (cm) <= 1.669 gini = 0.484 samples = 56 value = [0, 33, 23]
petal length (cm) <= 4.946 gini = 0.413 samples = 24 value = [0, 7, 17]
petal length (cm) <= 4.844 gini = 0.157 samples = 35 value = [0, 32, 3]
petal length (cm) <= 5.524 gini = 0.091 samples = 21 value = [0, 1, 20]
gini = 0.0 samples = 6 value = [0, 6, 0]
petal width (cm) <= 1.905 gini = 0.105 samples = 18 value = [0, 1, 17]
gini = 0.0 samples = 30 value = [0, 30, 0]
petal width (cm) <= 1.524 gini = 0.48 samples = 5 value = [0, 2, 3]
petal width (cm) <= 1.856 gini = 0.124 samples = 15 value = [0, 1, 14]
gini = 0.0 samples = 6 value = [0, 0, 6]
sepal length (cm) <= 6.947 gini = 0.32 samples = 5 value = [0, 1, 4]
gini = 0.0 samples = 13 value = [0, 0, 13]
gini = 0.375 samples = 4 value = [0, 1, 3]
gini = 0.0 samples = 1 value = [0, 1, 0]
gini = 0.219 samples = 8 value = [0, 1, 7]
gini = 0.0 samples = 7 value = [0, 0, 7]
gini = 0.5 samples = 2 value = [0, 1, 1]
gini = 0.0 samples = 3 value = [0, 0, 3]

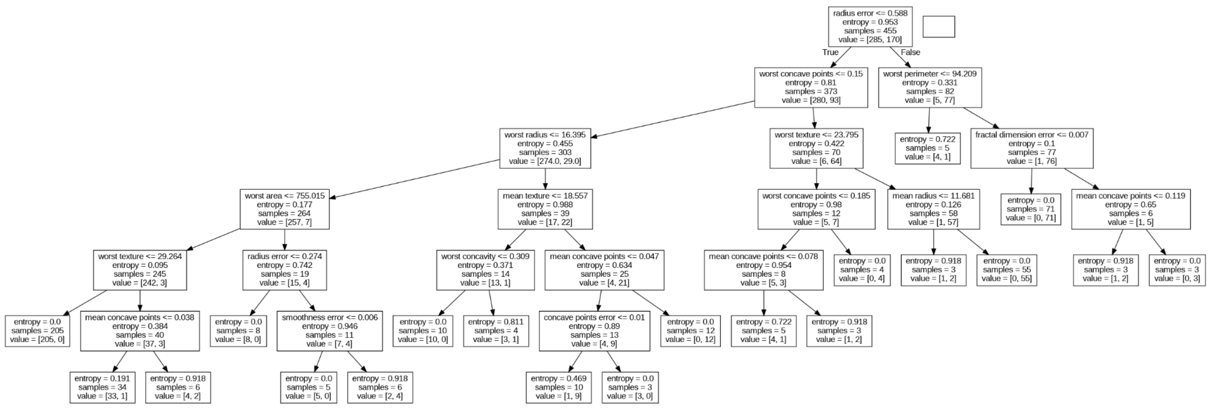
Comparison for accuracies for different test sizes:



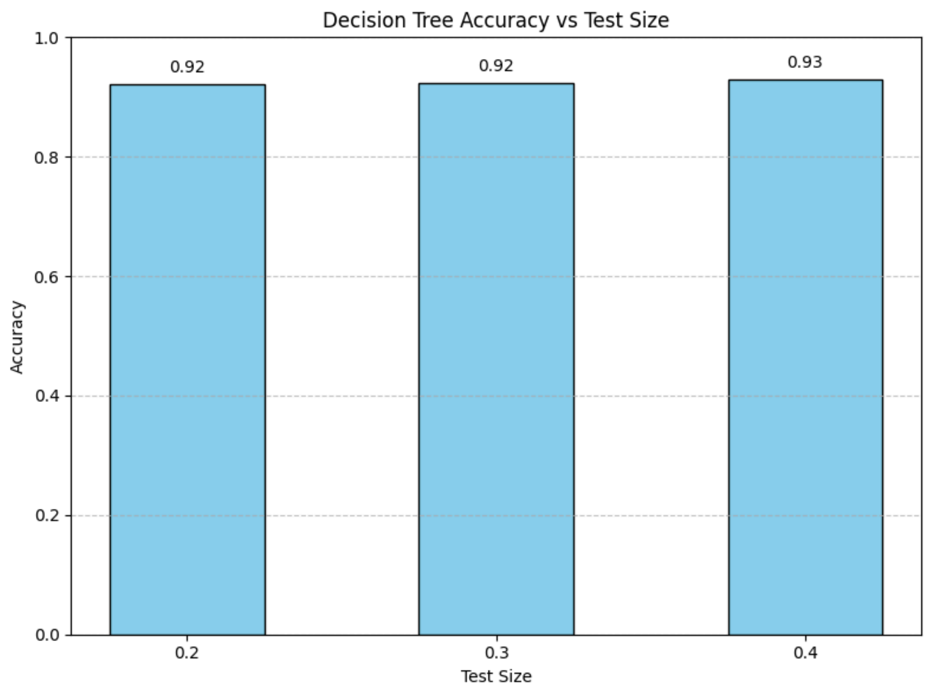
Wisconsin Breast Cancer Dataset

```
Confusion Matrix
[[72  0]
 [ 3 39]]
------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

   malignant       0.96      1.00      0.98        72
      benign       1.00      0.93      0.96        42

    accuracy                           0.97       114
   macro avg       0.98      0.96      0.97       114
weighted avg       0.97      0.97      0.97       114
```

Decision Tree:



Comparison for accuracies for different test sizes:



Discussion:

Decision Trees learn feature-based rules and partition the feature space using criteria like Gini index or Entropy. For the Iris dataset, the Decision Tree achieved very high classification accuracy, often close to perfect, because the dataset is small, well-balanced, and highly separable (e.g., petal length and width easily distinguish Setosa). The visualization of the tree made the classification process interpretable, showing clear splits based on petal dimensions. Overfitting was controlled using hyperparameters like `max_depth` and `min_samples_split`.

For the Breast Cancer dataset, the Decision Tree also performed well, with strong precision and recall, especially for the benign class. However, because this dataset has higher dimensionality (30 features), trees can easily overfit. Grid search tuning helped in selecting optimal depth and splitting criteria to balance accuracy with generalization. The resulting trees highlighted key medical features (such as mean radius and texture) that strongly influence the classification.

In comparison with Naive Bayes, Decision Trees provided better interpretability and comparable or higher accuracy, especially when tuned. However, they are more prone to overfitting, while Naive Bayes is simpler and more robust for small datasets.