# Name: TULIKA GUPTA

MCA/10019/22

# Question 1: Dataset Preparation and Exploration

1. Obtain a dataset suitable for classification, such as the Iris dataset or a text classification dataset. Describe the dataset.
2. Explore the dataset to understand its features and labels. Are there any missing values or outliers that need preprocessing?

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
```

In [3]:

```
iris=load_iris()
data = iris.data
iris_df = pd.DataFrame(data, columns=iris.feature_names)
iris_df
```

Out[3]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 |
| **...** | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 |

150 rows × 4 columns

In [4]:

```
iris_df.describe()
```

Out[4]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|
| **count** | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| **mean** | 5.843333 | 3.057333 | 3.758000 | 1.199333 |

In [4]:
```
iris_df.describe()
```

Out[4]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|
| **count** | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| **mean** | 5.843333 | 3.057333 | 3.758000 | 1.199333 |
| **std** | 0.828066 | 0.435866 | 1.765298 | 0.762238 |
| **min** | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| **25%** | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| **50%** | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| **75%** | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| **max** | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

In [5]:
```
missing_values = iris_df.isnull().sum()
missing_values_df = pd.DataFrame({'Missing Values': missing_values})
missing_values_df
```

Out[5]:

| | Missing Values |
|---|---|
| **sepal length (cm)** | 0 |
| **sepal width (cm)** | 0 |
| **petal length (cm)** | 0 |
| **petal width (cm)** | 0 |

# Question 2: Data Preprocessing

1. Perform data preprocessing tasks, such as handling missing values, dealing with outliers, and encoding categorical variables if necessary. Explain the steps you took and why they were important

In [6]:
```
Q1 = iris_df.quantile(0.25)
Q3 = iris_df.quantile(0.75)
IQR = Q3 - Q1
```

In [7]:
```
outliers = ((iris_df < (Q1 - 1.5 * IQR)) | (iris_df > (Q3 + 1.5 * IQR))).any(a
```

In [8]:
```
outliers_df = iris_df[outliers]
print("Rows with outliers:")
outliers_df
```

```
Rows with outliers:
```

Out[8]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|
| **15** | 5.7 | 4.4 | 1.5 | 0.4 |
| **32** | 5.2 | 4.1 | 1.5 | 0.1 |
| **33** | 5.5 | 4.2 | 1.4 | 0.2 |
| **60** | 5.0 | 2.0 | 3.5 | 1.0 |

In [9]:
```
iris_df = iris_df[~outliers]
print("After removing Outliers:")
iris_df
```

```
After removing Outliers:
```

Out[9]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|

| 60 | 5.0 | 2.0 | 3.5 | 1.0 |
|---|---|---|---|---|

In [9]:
```
iris_df = iris_df[~outliers]
print("After removing Outliers:")
iris_df
```

```
After removing Outliers:
```

Out[9]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 |
| **...** | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 |

146 rows × 4 columns

# Question 3: Feature Selection If applicable, choose a subset of features to use in the

classification task. Discuss your feature selection process and rationale

In [10]:
```
X = iris_df
y = iris.target[~outliers]
```

In [11]:
```

n_components = 2
pca = PCA(n_components=n_components)
X_pca = pca.fit_transform(X)
```

In [14]:
```
pca_df = pd.DataFrame(data=X_pca, columns=[f'PCA_{i+1}' for i in range(n_compo
```

In [15]:
```

pca_df['target'] = y
```

In [16]:
```

print("PCA Components:")
print(pca_df.head())
```

```
PCA Components:
      PCA_1     PCA_2  target
0 -2.735303  0.369462       0
1 -2.772500 -0.114016       0
2 -2.946195 -0.093716       0
3 -2.804664 -0.270769       0
4 -2.779388  0.369842       0
```

In [17]:
```

explained_variance = pca.explained_variance_ratio_
print("Explained Variance Ratio:")
print(explained_variance)
```

```
Explained Variance Ratio:
[0.93035174 0.04705254]
```

In [17]:

```
explained_variance = pca.explained_variance_ratio_
print("Explained Variance Ratio:")
print(explained_variance)
```

```
Explained Variance Ratio:
[0.93035174 0.04705254]
```

In [18]:

```
total_variance = np.sum(explained_variance)
print(f"Total Variance Explained: {total_variance:.2f}")
```

```
Total Variance Explained: 0.98
```

# Question 4: Model Building Implement a Naive Bayesian classifier

In [19]:

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
```

In [20]:

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

In [21]:

```
naive_bayes = GaussianNB()
naive_bayes.fit(X_train, y_train)
y_pred = naive_bayes.predict(X_test)
```

In [22]:

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9333333333333333
```

In [23]:

```
report = classification_report(y_test, y_pred, target_names=iris.target_names,


report_df = pd.DataFrame(report).transpose()


print("Classification Report:")
print(report_df)
```

```
Classification Report:
              precision    recall  f1-score    support
setosa         1.000000  1.000000  1.000000  12.000000
versicolor     0.875000  0.875000  0.875000   8.000000
virginica      0.900000  0.900000  0.900000  10.000000
accuracy       0.933333  0.933333  0.933333   0.933333
macro avg      0.925000  0.925000  0.925000  30.000000
weighted avg   0.933333  0.933333  0.933333  30.000000
```

# Question 5: Model Training Train the Naive Bayesian classifier on the training data. Discuss the model training process and any hyperparameters you set

In [24]:

```
from sklearn.naive_bayes import GaussianNB
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
```

Out[24]:

```
▾ GaussianNB
GaussianNB()
```

Discuss

the model training process and any hyperparameters you set

In [24]:
```python
from sklearn.naive_bayes import GaussianNB
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
```

Out[24]:
```
▾ GaussianNB
GaussianNB()
```

In [25]:
```python
print("Class Priors:")
print(nb_classifier.class_prior_)
print("Mean Values:")
print(nb_classifier.theta_)
shared_variance = np.var(X_train)
print("Shared Variance Value:", shared_variance)
```

```
Class Priors:
[0.30172414 0.35344828 0.34482759]
Mean Values:
[[4.95428571 3.39714286 1.44571429 0.25714286]
 [5.95121951 2.77804878 4.27560976 1.3195122 ]
 [6.5675     2.9875     5.5325     2.005     ]]
Shared Variance Value: sepal length (cm)    0.698022
sepal width (cm)     0.164402
petal length (cm)    2.962473
petal width (cm)     0.536078
dtype: float64
```

# Question 6: Model Evaluation Evaluate the model's performance on the testing data. Use

appropriate evaluation metrics such as accuracy, precision, recall, and F1-score. Interpret the results

In [26]:
```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_
y_pred = nb_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.93
```

In [28]:
```python
precision = precision_score(y_test, y_pred, average='weighted')
print(f"Precision: {precision:.2f}")
```

```
Precision: 0.93
```

In [29]:
```python
recall = recall_score(y_test, y_pred, average='weighted')
print(f"Recall: {recall:.2f}")
```

```
Recall: 0.93
```

In [30]:
```python
f1 = f1_score(y_test, y_pred, average='weighted')
print(f"F1-score: {f1:.2f}")
```

```
F1-score: 0.93
```

robustness. Discuss the benefits of cross- validation and any changes in performance compared to a single train-test split

# Question 8: Cross-Validation Perform k-fold cross-validation to assess the model's

In [31]:
```python
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
nb_classifier = GaussianNB()
```

```
F1-score: 0.93
```

robustness. Discuss the benefits of cross- validation and any changes in performance compared to a single train-test split

## Question 8: Cross-Validation Perform k-fold cross-validation to assess the model's

In [31]:
```python
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
nb_classifier = GaussianNB()
```

In [32]:
```python
n_folds = 5
cross_val = StratifiedKFold(n_splits=n_folds, shuffle=True, random_state=42)
cross_val_accuracy = cross_val_score(nb_classifier, X, y, cv=cross_val, scorin
```

In [33]:
```python
print("Cross-Validated Accuracy:")
print(cross_val_accuracy)
print(f"Mean Accuracy: {cross_val_accuracy.mean():.2f}")
```

```
Cross-Validated Accuracy:
[0.96666667 0.96551724 0.93103448 1.         0.89655172]
Mean Accuracy: 0.95
```

## Question 10: Comparison with Other Models Compare the performance of the Naive

Bayesian classifier with other classification algorithms like decision trees, logistic regression, or support vector machines. What are the trade-offs between different algorithms?

In [34]:
```python
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

In [35]:
```python
classifiers = [
("Naive Bayes", GaussianNB()),
("Decision Tree", DecisionTreeClassifier(random_state=42)),
("Logistic Regression", LogisticRegression(max_iter=1000, random_state=42)),
("Support Vector Machine", SVC(random_state=42))
]
```

In [38]:
```python

n_folds = 5

for classifier_name, classifier in classifiers:
    cross_val = StratifiedKFold(n_splits=n_folds, shuffle=True, random_state=4
    cross_val_accuracy = cross_val_score(classifier, X, y, cv=cross_val, scori
    mean_accuracy = cross_val_accuracy.mean()
```

In [38]:

```
n_folds = 5

for classifier_name, classifier in classifiers:
    cross_val = StratifiedKFold(n_splits=n_folds, shuffle=True, random_state=4
    cross_val_accuracy = cross_val_score(classifier, X, y, cv=cross_val, scori
    mean_accuracy = cross_val_accuracy.mean()
    print(f"{classifier_name}:")
    print("Cross-Validated Accuracy:")
    print(cross_val_accuracy)
    print(f"Mean Accuracy: {mean_accuracy:.2f}")
    print()
    print(f"Mean Accuracy: {mean_accuracy:.2f}")
```

```
Naive Bayes:
Cross-Validated Accuracy:
[0.96666667 0.96551724 0.93103448 1.         0.89655172]
Mean Accuracy: 0.95

Mean Accuracy: 0.95
Decision Tree:
Cross-Validated Accuracy:
[1.         0.93103448 0.89655172 0.93103448 0.89655172]
Mean Accuracy: 0.93

Mean Accuracy: 0.93
Logistic Regression:
Cross-Validated Accuracy:
[1.         0.96551724 0.93103448 0.96551724 0.93103448]
Mean Accuracy: 0.96

Mean Accuracy: 0.96
Support Vector Machine:
Cross-Validated Accuracy:
[1.         0.96551724 0.93103448 0.96551724 0.93103448]
Mean Accuracy: 0.96

Mean Accuracy: 0.96
```

In [ ]: