Objective: The objective of this assessment is to evaluate your understanding and ability to apply supervised learning techniques to a real-world dataset.

Dataset: Use the breast cancer dataset available in the sklearn library.

1. Loading and Preprocessing

In [ ]:

```

```

A)Load the breast cancer dataset from sklearn.

In [4]:

```
from sklearn.datasets import load_breast_cancer
import pandas as pd
import numpy as np

# Load the dataset
data = load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
```

B)Preprocess the data to handle any missing values and perform necessary feature scaling.

In [6]:

```
#1. Handle Missing Values:
print(df.isnull().sum())  # Check for missing values



df.fillna(df.mean(), inplace=True)
```

```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
target                     0
dtype: int64
```

C)Explain the preprocessing steps you performed and justify why they are necessary for this dataset.

the main preprocessing steps performed on the breast cancer dataset were:

1)Feature Scaling (Standardization) 2)Data Splitting

In [12]:

```
#1)feature scaling
from sklearn.preprocessing import StandardScaler
```

In [14]:

```
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)
# y contains the target variable (labels)
y = pd.Series(data.target)
```

In [16]:

```
#2)data spliting
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df_scaled, y, test_size=0.3,
```

1. Data Cleaning:

Removed duplicates and handled missing values (e.g., mean/median imputation) to ensure data consistency and completeness.

2. Data Transformation:

Encoded categorical variables and scaled numerical features to improve algorithm compatibility and performance.

3. Outlier Handling:

Identified and treated outliers using IQR or Z-scores to prevent skewed results.

4. Feature Selection/Engineering:

Removed irrelevant features and created new ones to enhance model interpretability and accuracy.

5. Data Splitting:

Split into training, validation, and test sets to evaluate model performance on unseen data.

6. Balancing (if applicable):

Addressed class imbalance using oversampling or class weighting to ...

2. Classification Algorithm Implementation

Implement the following five classification algorithms:

1. Logistic Regression

In [20]:

```
#Description: A linear model that predicts probabilities using a sigmoid functio
#features and the target.

#IMPLementation:


from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression().fit(X_train, y_train)
predictions = log_reg.predict(X_test)
```

2. Decision Tree Classifier

In [22]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Decision Tree Classifier Model
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)
dt_accuracy = accuracy_score(y_test, y_pred_dt)
print(f'Decision Tree Accuracy: {dt_accuracy}')
```

```
Decision Tree Accuracy: 1.0
```

3. Random Forest Classifier

In [24]:

```
from sklearn.ensemble import RandomForestClassifier

# Random Forest Classifier Model
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
rf_accuracy = accuracy_score(y_test, y_pred_rf)
print(f'Random Forest Accuracy: {rf_accuracy}')
```

```
Random Forest Accuracy: 1.0
```

4. Support Vector Machine (SVM)

In [26]:

```
from sklearn.svm import SVC

# Support Vector Machine Model
svm = SVC(random_state=42)
svm.fit(X_train, y_train)
y_pred_svm = svm.predict(X_test)
svm_accuracy = accuracy_score(y_test, y_pred_svm)
print(f'Support Vector Machine Accuracy: {svm_accuracy}')
```

```
Support Vector Machine Accuracy: 0.9941520467836257
```

5. k-Nearest Neighbors (k-NN)

For each algorithm, provide a brief description of how it works and why it might be suitable for this dataset.

In [28]:

```
from sklearn.neighbors import KNeighborsClassifier

# k-Nearest Neighbors Model
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)
knn_accuracy = accuracy_score(y_test, y_pred_knn)
print(f'k-Nearest Neighbors Accuracy: {knn_accuracy}')
```

```
k-Nearest Neighbors Accuracy: 1.0
```

3. Model Comparison

Compare the performance of the five classification algorithms. Which algorithm performed the best and which one performed the worst?

In [30]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```python
from sklearn.datasets import load_breast_cancer

# Load the dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Preprocess the data (Standardize the features)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3,

# Initialize models
models = [
    ("Logistic Regression", LogisticRegression(max_iter=10000)),
    ("Decision Tree", DecisionTreeClassifier()),
    ("Random Forest", RandomForestClassifier()),
    ("SVM", SVC()),
    ("k-NN", KNeighborsClassifier())
]

# Train models and calculate accuracy
accuracies = {}
for name, model in models:
    model.fit(X_train, y_train)  # Train the model
    y_pred = model.predict(X_test)  # Predict with the model
    accuracy = accuracy_score(y_test, y_pred)  # Calculate accuracy
    accuracies[name] = accuracy  # Save accuracy score

# Print the accuracy of each model
for name, accuracy in accuracies.items():
    print(f"{name} Accuracy: {accuracy:.4f}")

# Identify the best and worst performing models
best_model = max(accuracies, key=accuracies.get)
worst_model = min(accuracies, key=accuracies.get)

print(f"\nBest Performing Model: {best_model} with Accuracy: {accuracies[best_mo
print(f"Worst Performing Model: {worst_model} with Accuracy: {accuracies[worst_m
```

```
Logistic Regression Accuracy: 0.9825
Decision Tree Accuracy: 0.9181
Random Forest Accuracy: 0.9708
SVM Accuracy: 0.9708
k-NN Accuracy: 0.9591

Best Performing Model: Logistic Regression with Accuracy: 0.9825
Worst Performing Model: Decision Tree with Accuracy: 0.9181
```

```
In [ ]:
```