In [1]:

```
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler
import pandas as pd

# Load the dataset
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)

# Check for missing values
print(X.isnull().sum())

# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
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
dtype: int64
```

Classification Algorithm Implementation

In [3]:

```
#1. logistic Regression
"""Logistic Regression is a linear model that estimates the probability of a binary outcome using the logistic function. It is su
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(max_iter=10000)
log_reg.fit(X_scaled, y)
```

Out[3]:

```
▾       LogisticRegression
LogisticRegression(max_iter=10000)
```

In [4]:

```
# 2.Decision Tree Classifier
"""Decision Trees split the data based on feature values to make predictions. They work well on datasets where relationships betw
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier()
tree.fit(X_scaled, y)
```

Out[4]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [5]:
```
#3.Random Forest Classifier
"""Random Forest is an ensemble method that combines multiple decision trees to improve classification accuracy. It reduces overf
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_scaled, y)
```

Out[5]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [6]:
```
# 4.Support Vector Machine (SVM)
"""SVM finds the optimal hyperplane that maximizes the margin between two classes. It is effective in high-dimensional spaces and
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_scaled, y)
```

Out[6]:
```
▾ SVC
SVC()
```

In [7]:
```
# 5.k-Nearest Neighbors (k-NN)
"""k-NN is a distance-based classifier where the class of a data point is determined by the majority class of its k nearest neigh
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_scaled, y)
```

Out[7]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

Model Comparison

In [8]:
```
"""Model Evaluation: Use accuracy, precision, recall, and F1-score as performance metrics. You can also use cross-validation to e
from sklearn.model_selection import cross_val_score

# Define classifiers
classifiers = {
    'Logistic Regression': log_reg,
    'Decision Tree': tree,
    'Random Forest': rf,
    'SVM': svm,
    'k-NN': knn
}

# Evaluate each classifier
for name, clf in classifiers.items():
    scores = cross_val_score(clf, X_scaled, y, cv=5)
    print(f"{name}: {scores.mean():.3f} (+/- {scores.std():.3f})")
```

```
Logistic Regression: 0.981 (+/- 0.007)
Decision Tree: 0.912 (+/- 0.021)
Random Forest: 0.956 (+/- 0.018)
SVM: 0.974 (+/- 0.015)
k-NN: 0.965 (+/- 0.010)
```

In [ ]:
```
Random Forest is the best performer
K-NN is the worst performer
```