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

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

# Create a DataFrame for easier manipulation
df = pd.DataFrame(X, columns=data.feature_names)
df['target'] = y

# Check for missing values
print(df.isnull().sum())
```

```
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
```

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

# Apply feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Initialize and train the model
log_reg = LogisticRegression(max_iter=10000)
log_reg.fit(X_train_scaled, y_train)

# Predict and evaluate
y_pred_log_reg = log_reg.predict(X_test_scaled)
accuracy_log_reg = accuracy_score(y_test, y_pred_log_reg)
```

```
from sklearn.tree import DecisionTreeClassifier

# Initialize and train the model
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train_scaled, y_train)

# Predict and evaluate
y_pred_tree = decision_tree.predict(X_test_scaled)
accuracy_tree = accuracy_score(y_test, y_pred_tree)
```

```
from sklearn.ensemble import RandomForestClassifier

# Initialize and train the model
random_forest = RandomForestClassifier()
random_forest.fit(X_train_scaled, y_train)

# Predict and evaluate
y_pred_forest = random_forest.predict(X_test_scaled)
accuracy_forest = accuracy_score(y_test, y_pred_forest)
```

```
from sklearn.svm import SVC

# Initialize and train the model
svm = SVC()
svm.fit(X_train_scaled, y_train)
```

```
# Predict and evaluate
y_pred_svm = svm.predict(X_test_scaled)
accuracy_svm = accuracy_score(y_test, y_pred_svm)


from sklearn.neighbors import KNeighborsClassifier

# Initialize and train the model
knn = KNeighborsClassifier()
knn.fit(X_train_scaled, y_train)

# Predict and evaluate
y_pred_knn = knn.predict(X_test_scaled)
accuracy_knn = accuracy_score(y_test, y_pred_knn)



# Print accuracies of all models
print(f"Logistic Regression Accuracy: {accuracy_log_reg:.2f}")
print(f"Decision Tree Accuracy: {accuracy_tree:.2f}")
print(f"Random Forest Accuracy: {accuracy_forest:.2f}")
print(f"SVM Accuracy: {accuracy_svm:.2f}")
print(f"k-NN Accuracy: {accuracy_knn:.2f}")
```

```
Logistic Regression Accuracy: 0.98
Decision Tree Accuracy: 0.92
Random Forest Accuracy: 0.98
SVM Accuracy: 0.98
k-NN Accuracy: 0.96
```