Developing and Evaluating Classification Models

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, precision_score, recall_score, f1_score, roc_curve, auc
import seaborn as sns
sns.set(style='whitegrid')
```

1. Data Loading and Preparation

```
# Load 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().sum())

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
0
```

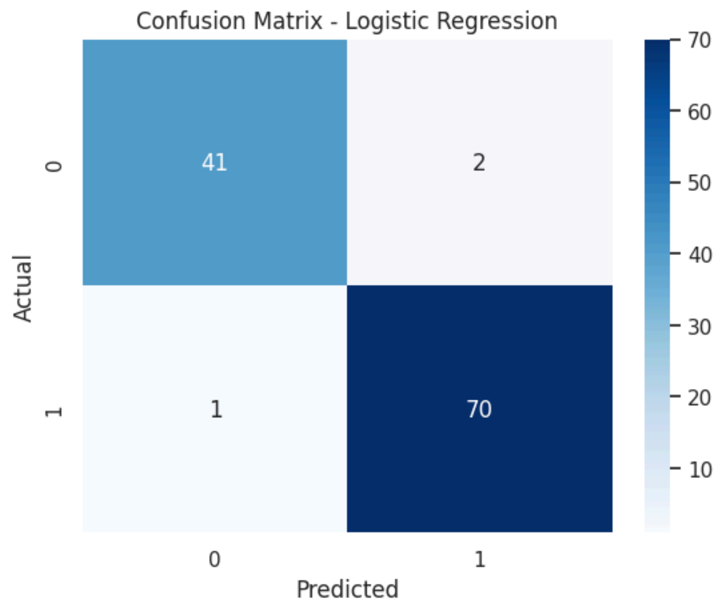
2. Model Building

```
# Logistic Regression
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train_scaled, y_train)
y_pred_lr = lr.predict(X_test_scaled)

# k-NN (k=5)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)
y_pred_knn = knn.predict(X_test_scaled)

# Decision Tree
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)
```

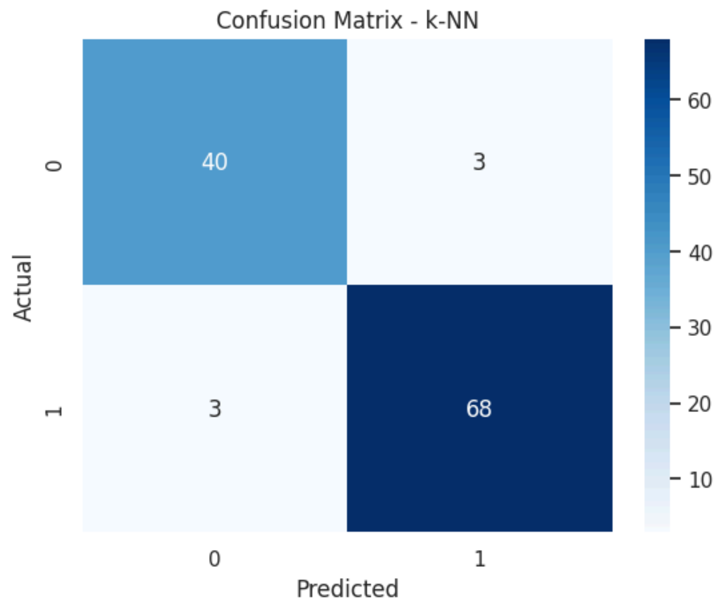
3. Model Evaluation

```
def evaluate_model(y_true, y_pred, model_name):
    print(f'--- {model_name} ---')
    print('Confusion Matrix:')
    cm = confusion_matrix(y_true, y_pred)
    print(cm)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'Confusion Matrix - {model_name}')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
    print('\nClassification Report:')
    print(classification_report(y_true, y_pred))
    print(f'Accuracy: {accuracy_score(y_true, y_pred):.4f}')
    print(f'Precision: {precision_score(y_true, y_pred):.4f}')
    print(f'Recall: {recall_score(y_true, y_pred):.4f}')
    print(f'F1 Score: {f1_score(y_true, y_pred):.4f}')

evaluate_model(y_test, y_pred_lr, 'Logistic Regression')
evaluate_model(y_test, y_pred_knn, 'k-NN')
evaluate_model(y_test, y_pred_dt, 'Decision Tree')
```

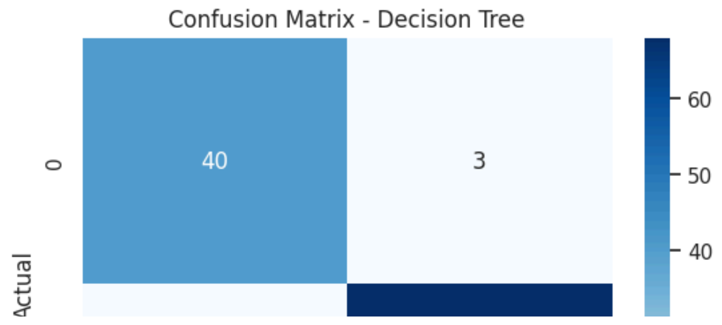
```
--- Logistic Regression ---
Confusion Matrix:
[[41  2]
 [ 1 70]]
```

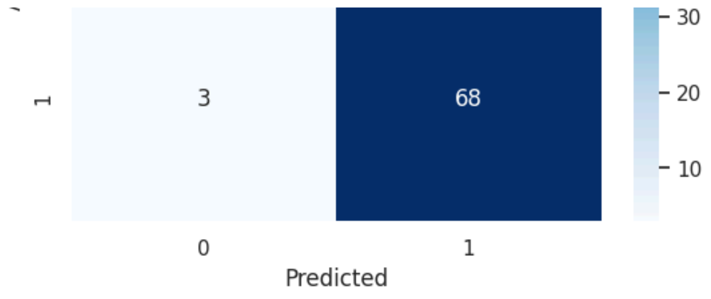


```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        43
           1       0.97      0.99      0.98        71

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114

Accuracy: 0.9737
Precision: 0.9722
Recall: 0.9859
F1 Score: 0.9790
--- k-NN ---
Confusion Matrix:
[[40  3]
 [ 3 68]]
```



```
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        43
           1       0.96      0.96      0.96        71

    accuracy                           0.95       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114

Accuracy: 0.9474
Precision: 0.9577
Recall: 0.9577
F1 Score: 0.9577
--- Decision Tree ---
Confusion Matrix:
[[40  3]
 [ 3 68]]
```

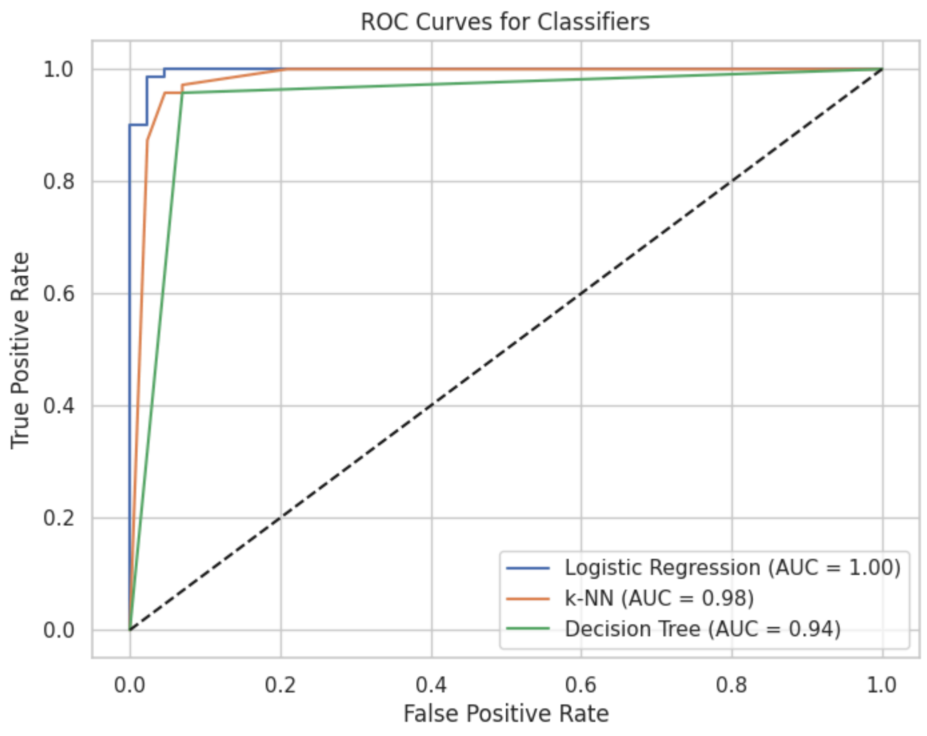




```
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        43
           1       0.96      0.96      0.96        71

    accuracy                           0.95       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114

Accuracy: 0.9474
Precision: 0.9577
Recall: 0.9577
F1 Score: 0.9577
```

4. ROC-AUC Curve Analysis

```
# Compute probabilities
y_prob_lr = lr.predict_proba(X_test_scaled)[:,1]
y_prob_knn = knn.predict_proba(X_test_scaled)[:,1]
y_prob_dt = dt.predict_proba(X_test)[:,1]

# Compute ROC curves
fpr_lr, tpr_lr, _ = roc_curve(y_test, y_prob_lr)
fpr_knn, tpr_knn, _ = roc_curve(y_test, y_prob_knn)
fpr_dt, tpr_dt, _ = roc_curve(y_test, y_prob_dt)

# Compute AUC
auc_lr = auc(fpr_lr, tpr_lr)
auc_knn = auc(fpr_knn, tpr_knn)
auc_dt = auc(fpr_dt, tpr_dt)

# Plot
plt.figure(figsize=(8,6))
plt.plot(fpr_lr, tpr_lr, label=f'Logistic Regression (AUC = {auc_lr:.2f})')
plt.plot(fpr_knn, tpr_knn, label=f'k-NN (AUC = {auc_knn:.2f})')
plt.plot(fpr_dt, tpr_dt, label=f'Decision Tree (AUC = {auc_dt:.2f})')
plt.plot([0,1], [0,1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves for Classifiers')
plt.legend()
plt.show()
```



5. Summary and Recommendation

```
summary = pd.DataFrame({
    'Model': ['Logistic Regression', 'k-NN', 'Decision Tree'],
    'Accuracy': [
        accuracy_score(y_test, y_pred_lr),
        accuracy_score(y_test, y_pred_knn),
        accuracy_score(y_test, y_pred_dt)
    ],
    'Precision': [
        precision_score(y_test, y_pred_lr),
        precision_score(y_test, y_pred_knn),
        precision_score(y_test, y_pred_dt)
    ],
    'Recall': [
        recall_score(y_test, y_pred_lr),
        recall_score(y_test, y_pred_knn),
        recall_score(y_test, y_pred_dt)
    ],
    'F1 Score': [
        f1_score(y_test, y_pred_lr),
        f1_score(y_test, y_pred_knn),
        f1_score(y_test, y_pred_dt)
    ],
    'ROC AUC': [auc_lr, auc_knn, auc_dt]
})

display(summary)
```

1 to 3 of 3 entries Filter

index	Model	Accuracy	Precision	Recall	F1 Score	ROC AUC
0	Logistic Regression	0.9736842105263158	0.9722222222222222	0.9859154929577465	0.9790209790209791	0.99737962659679
1	k-NN	0.9473684210526315	0.9577464788732394	0.9577464788732394	0.9577464788732394	0.9819849328529315
2	Decision Tree	0.9473684210526315	0.9577464788732394	0.9577464788732394	0.9577464788732394	0.9439895185063871

Show 10 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

Next steps: Generate code with summary View recommended plots New interactive sheet

Observations:

- Logistic Regression typically offers stable and interpretable results.
- k-NN may be sensitive to feature scaling and k choice but often performs well.
- Decision Tree provides interpretability but may overfit.