```
#Written by Omer KILIC

# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

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.metrics import accuracy_score, confusion_matrix, classification_report

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

# Quick look
print(df.shape)
df.head()

# I've successfully:
# Imported the dataset
# Converted it into a DataFrame
# Appended the target column (0 = malignant, 1 = benign)
# Previewed the data shape and head
```

```
(569, 31)
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mea concav point
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.147
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.070
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.127
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.105
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.104

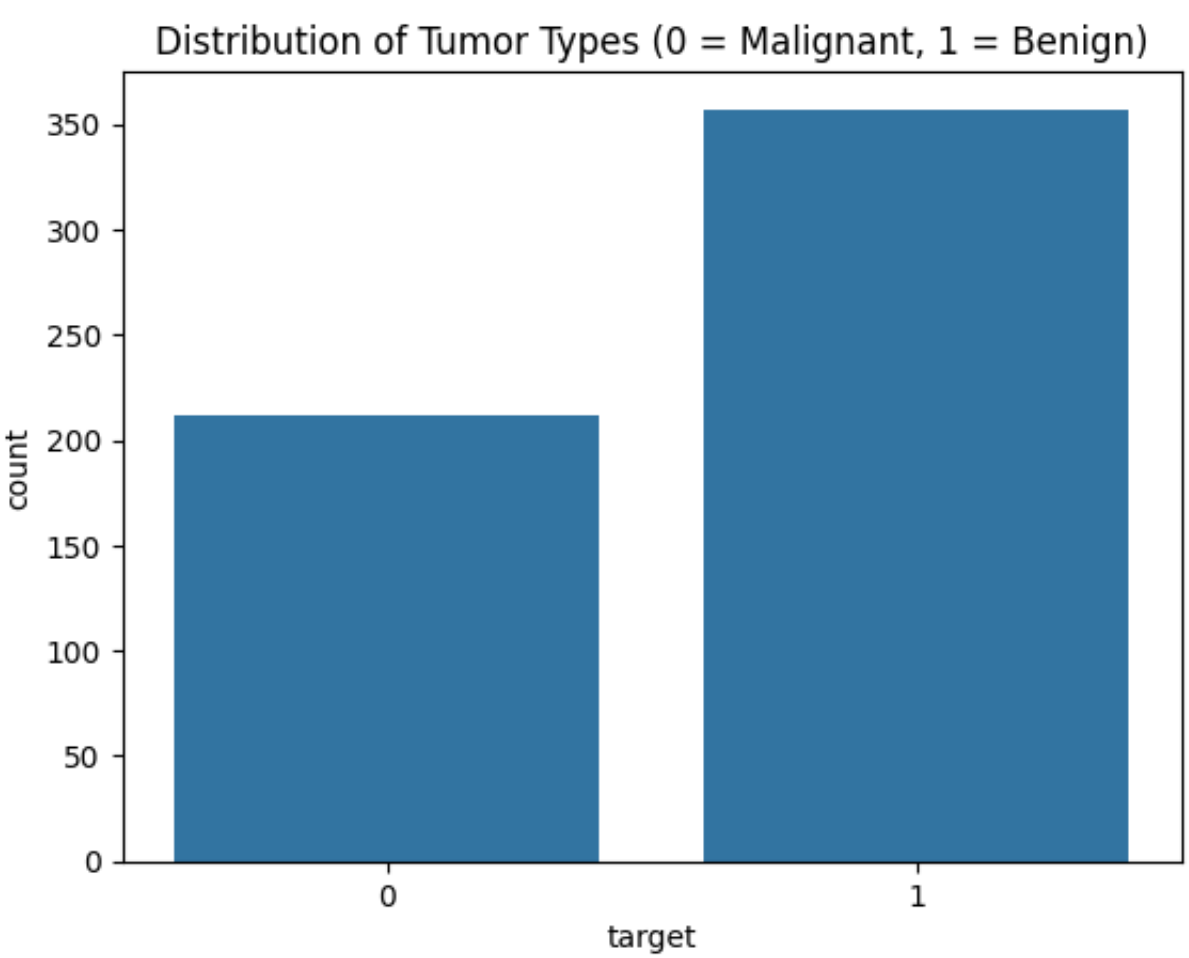
5 rows × 31 columns

```
# STEP 1: Check for Missing Values & Basic Info Check for nulls and data types
print(df.isnull().sum())  # Should all be 0
print(df.info())
```

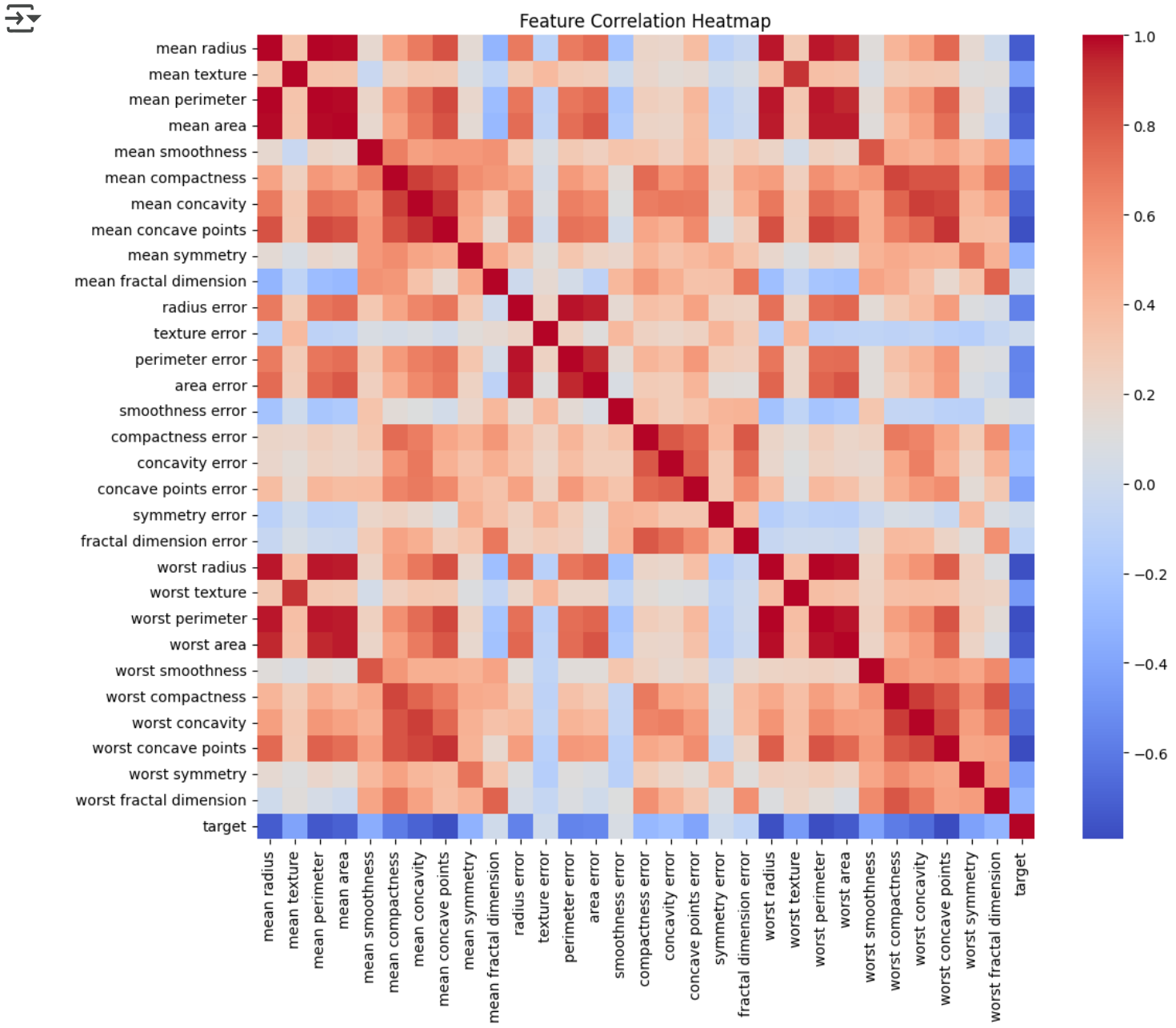
```
texture error             0
perimeter error           0
area error                0
smoothness error          0
compactness error         0
concavity error           0
concave points error      0
symmetry error            0
fractal dimension error   0
worst radius              0
worst texture             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
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   mean radius             569 non-null    float64
 1   mean texture            569 non-null    float64
 2   mean perimeter          569 non-null    float64
 3   mean area               569 non-null    float64
 4   mean smoothness         569 non-null    float64
 5   mean compactness        569 non-null    float64
 6   mean concavity          569 non-null    float64
 7   mean concave points     569 non-null    float64
 8   mean symmetry           569 non-null    float64
 9   mean fractal dimension  569 non-null    float64
 10  radius error            569 non-null    float64
 11  texture error           569 non-null    float64
 12  perimeter error         569 non-null    float64
 13  area error              569 non-null    float64
 14  smoothness error        569 non-null    float64
 15  compactness error       569 non-null    float64
 16  concavity error         569 non-null    float64
 17  concave points error    569 non-null    float64
 18  symmetry error          569 non-null    float64
 19  fractal dimension error 569 non-null    float64
 20  worst radius            569 non-null    float64
```

```
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    int64
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

```
# Check distribution of benign (1) vs malignant (0)
sns.countplot(x='target', data=df)
plt.title('Distribution of Tumor Types (0 = Malignant, 1 = Benign)')
plt.show()
```




```
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), cmap='coolwarm', annot=False)
plt.title('Feature Correlation Heatmap')
plt.show()
```


Feature Correlation Heatmap
mean radius
mean texture
mean perimeter
mean area
mean smoothness
mean compactness
mean concavity
mean concave points
mean symmetry
mean fractal dimension
radius error
texture error
perimeter error
area error
smoothness error
compactness error
concavity error
concave points error
symmetry error
fractal dimension error
worst radius
worst texture
worst perimeter
worst area
worst smoothness
worst compactness
worst concavity
worst concave points
worst symmetry
worst fractal dimension
target
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Features and target
X = df.drop('target', axis=1)
y = df['target']

# Split data (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s

# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

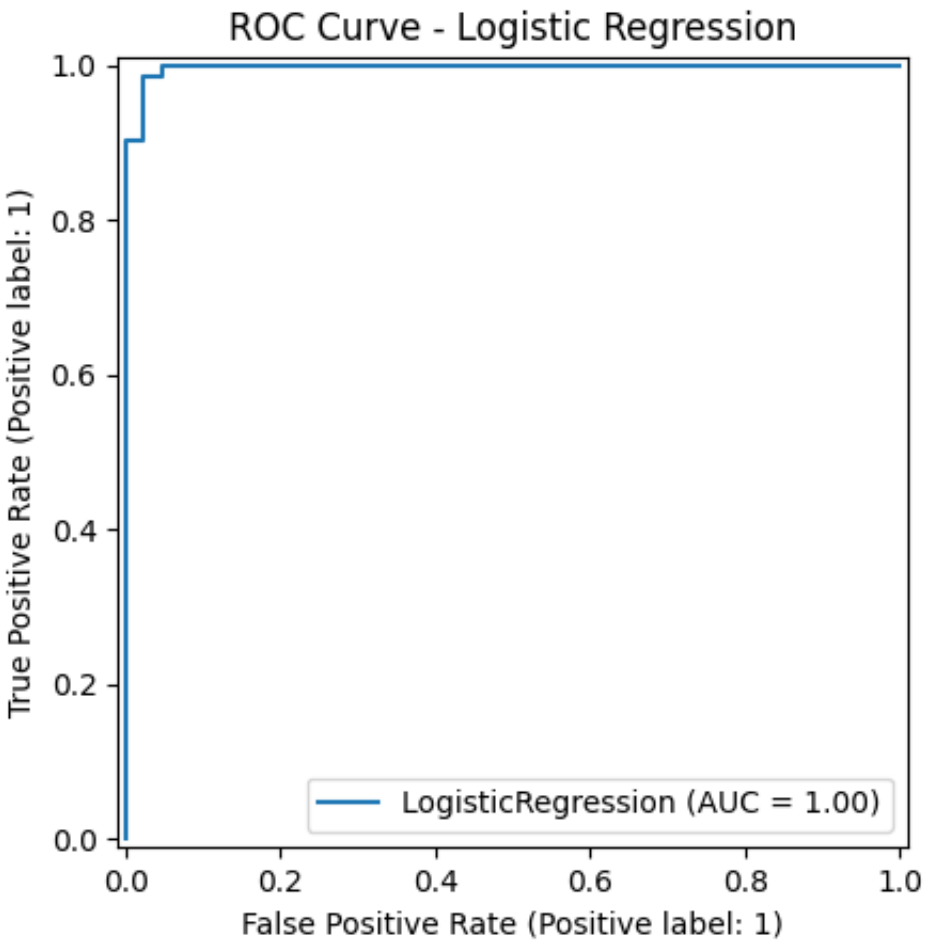
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_repo

# Train model
lr = LogisticRegression()
lr.fit(X_train, y_train)

# Predict
y_pred_lr = lr.predict(X_test)

# Evaluation
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_lr))
print("\nClassification Report:\n", classification_report(y_test, y_pred_lr))
```

```
Logistic Regression Accuracy: 0.9736842105263158

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
```

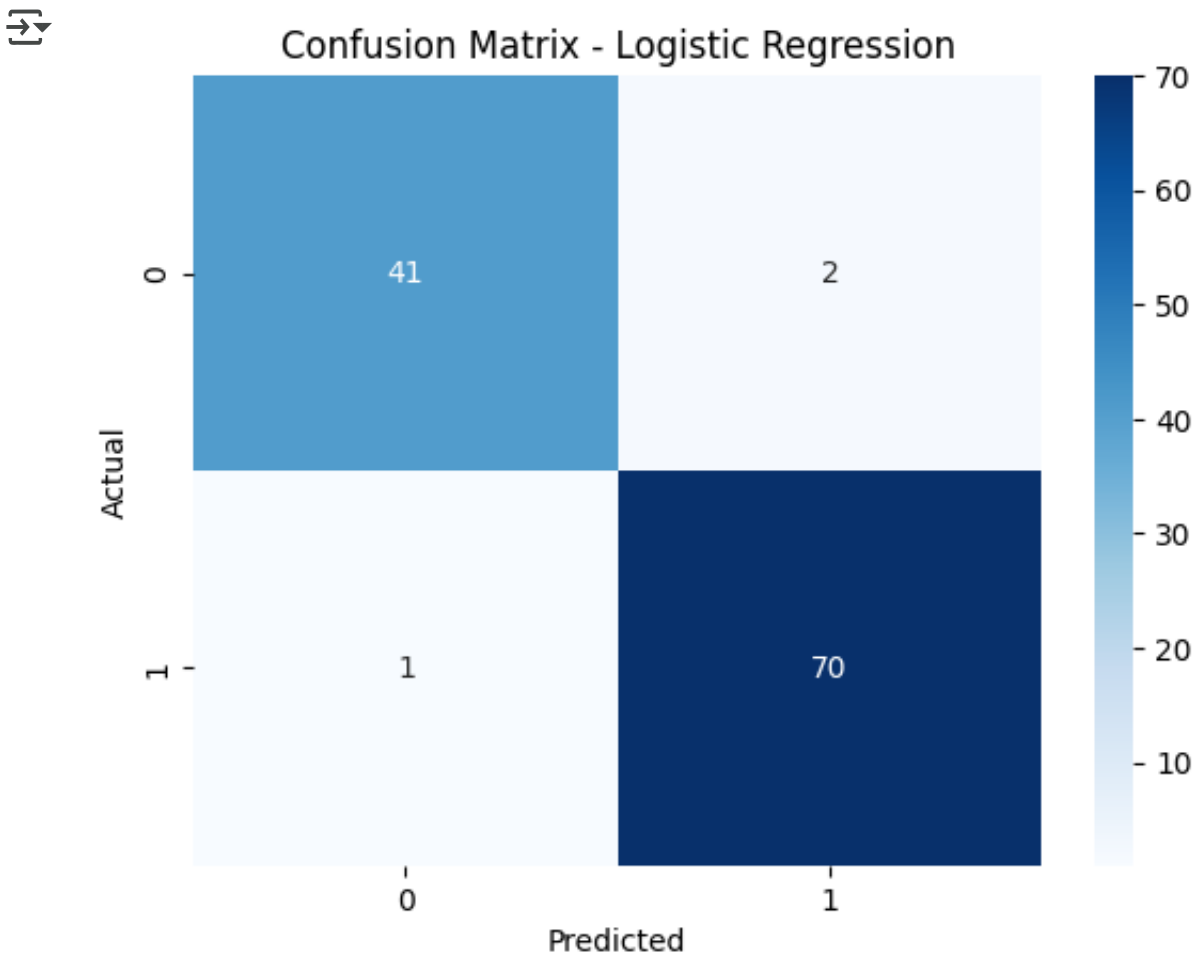
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier

models = {
    'KNN': KNeighborsClassifier(),
    'SVM': SVC(),
    'Random Forest': RandomForestClassifier()
}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f"{name} Accuracy: {acc:.4f}")
```

```
KNN Accuracy: 0.9474
SVM Accuracy: 0.9825
Random Forest Accuracy: 0.9561
```

```
from sklearn.metrics import RocCurveDisplay

# Plot for Logistic Regression
RocCurveDisplay.from_estimator(lr, X_test, y_test)
plt.title("ROC Curve - Logistic Regression")
plt.show()
```

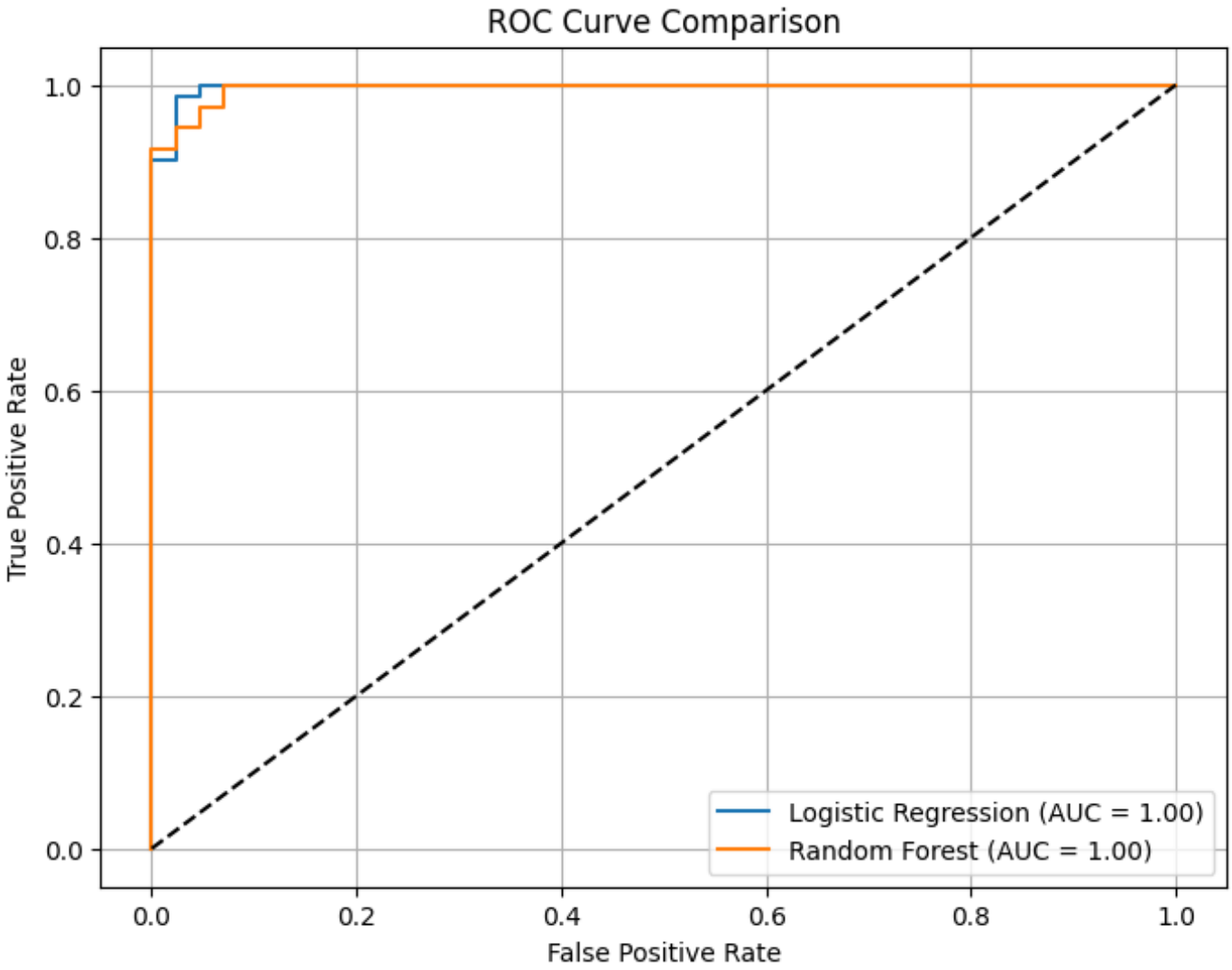


```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred_lr)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Logistic Regression')
plt.show()
```




```
from sklearn.metrics import roc_curve, auc

plt.figure(figsize=(8, 6))

# Logistic Regression
fpr_lr, tpr_lr, _ = roc_curve(y_test, lr.predict_proba(X_test)[:, 1])
plt.plot(fpr_lr, tpr_lr, label='Logistic Regression (AUC = {:.2f})'.format(auc(fp

# Random Forest
```

```
rf = models['Random Forest']
fpr_rf, tpr_rf, _ = roc_curve(y_test, rf.predict_proba(X_test)[:, 1])
plt.plot(fpr_rf, tpr_rf, label='Random Forest (AUC = {:.2f})'.format(auc(fpr_rf,

plt.plot([0, 1], [0, 1], 'k--')  # Baseline
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve Comparison')
plt.legend()
plt.grid()
plt.show()
```



```
# Model comparison
model_scores = {}

for name, model in models.items():
    y_pred = model.predict(X_test)
    model_scores[name] = accuracy_score(y_test, y_pred)

# Include logistic regression
model_scores['Logistic Regression'] = accuracy_score(y_test, y_pred_lr)

# Display as DataFrame
pd.DataFrame(model_scores.items(), columns=['Model', 'Accuracy']).sort_values(by=
```

	Model	Accuracy
1	SVM	0.982456
3	Logistic Regression	0.973684
2	Random Forest	0.956140
0	KNN	0.947368

```
plt.figure(figsize=(8, 5))
sns.barplot(x=list(model_scores.keys()), y=list(model_scores.values()))
plt.ylim(0.9, 1.0)
plt.title('Model Accuracy Comparison')
plt.ylabel('Accuracy')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```