APR Assignment 1

Overview

I have implemented Logistic Regression on the Breast Cancer dataset (binary classification) using Python and scikit-learn. The dataset contains **569 samples** with **30 features numerical** extracted from digitized images of fine needle aspirates (FNAs) of breast masses. The goal is to classify tumors into **benign (0)** or **malignant (1)** categories.

Dataset Description

Breast Cancer Wisconsin (Diagnostic) dataset
Number of samples (rows): 569
Number of features (columns): 30
Target classes:

- 0: Benign (212 samples)
- 1: Malignant (357 samples)

Code:

Cell 1:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    confusion_matrix, classification_report, roc_curve, auc, RocCurveDisplay
)
import joblib
```

Cell 2:

```
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target, name='target')  # 0 = malignant(cancerous), 1 = 
benign(non-cancerous)

print("Features shape:", X.shape)
print("Target distribution:\n", y.value_counts())
X.head()
```

Output of cell 2:

```
Features shape: (569, 30)
Target distribution:
 target
1    357
0    212
Name: count, dtype: int64
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2019.0	0.1622	0.6656
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.80	1956.0	0.1238	0.1866
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1709.0	0.1444	0.4245
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.87	567.7	0.2098	0.8663
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1575.0	0.1374	0.2050

5 rows × 30 columns

Cell 3:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
print("Train:", X_train.shape, "Test:", X_test.shape)
```

Output of Cell 3:

```
Train: (455, 30) Test: (114, 30)
```

Cell 4:

```
scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s = scaler.transform(X_test)
```

Cell 5:

```
clf = LogisticRegression(max_iter=10000, solver='liblinear', random_state=42)
clf.fit(X_train_s, y_train)
```

Output of Cell 5:

```
LogisticRegression
LogisticRegression(max_iter=10000, random_state=42, solver='liblinear')
```

Cell 6:

```
y_pred = clf.predict(X_test_s)
y_prob = clf.predict_proba(X_test_s)[:, 1]

acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy: {:.4f}".format(acc))
print("Precision: {:.4f}".format(prec))
print("Recall: {:.4f}".format(rec))
print("F1-score: {:.4f}".format(f1))
print("\nClassification report:\n", classification_report(y_test, y_pred))
```

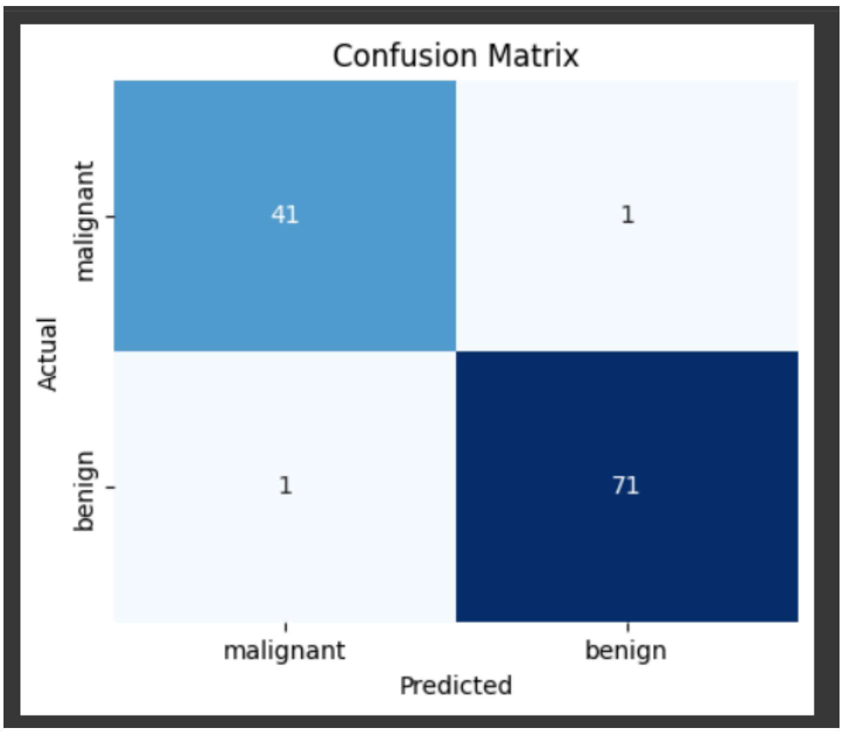
Output of Cell 6:

```
Accuracy: 0.9825
Precision: 0.9861
Recall: 0.9861
F1-score: 0.9861

Classification report:
               precision    recall  f1-score   support

           0       0.98      0.98      0.98        42
           1       0.99      0.99      0.99        72

    accuracy                           0.98       114
   macro avg       0.98      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114
```

Cell 7:

```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.xticks([0.5,1.5], data.target_names)  # 0/1 labels
plt.yticks([0.5,1.5], data.target_names)
plt.show()
```

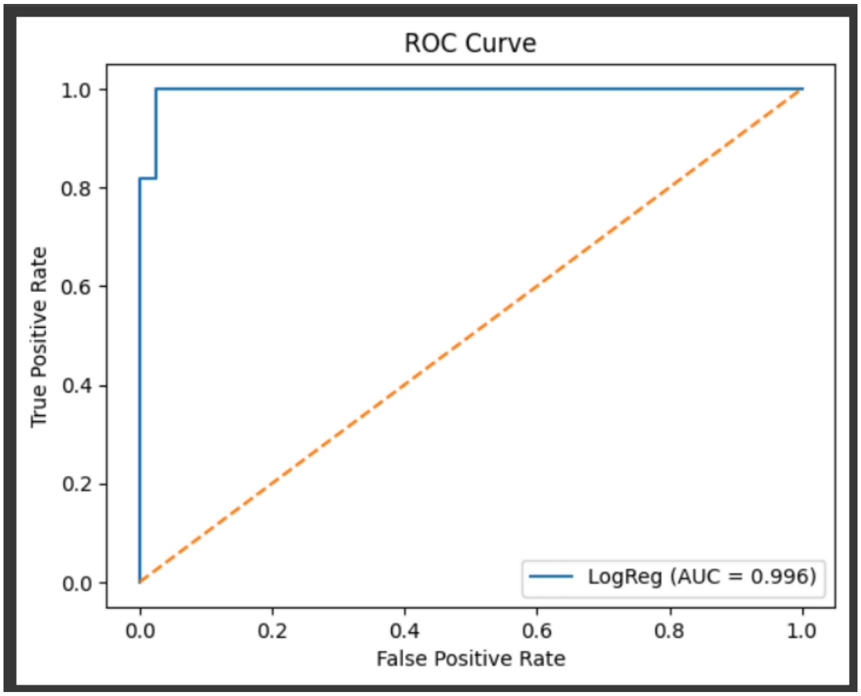
Output of Cell 7:



Cell 8:

```
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, label=f'LogReg (AUC = {roc_auc:.3f})')
plt.plot([0,1], [0,1], linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

Output of Cell 8:



Cell 9:

```
coef = pd.Series(clf.coef_.ravel(), index=X.columns).sort_values(key=abs,
ascending=False)
coef.head(10)
```

Output of Cell 9:

	0
worst texture	-1.242272
radius error	-1.087929
worst area	-0.979282
area error	-0.958096
worst radius	-0.946000
worst concave points	-0.945296
worst symmetry	-0.928729
worst concavity	-0.827180
worst perimeter	-0.764807
worst smoothness	-0.759567

dtype: float64

Cell 10:

```
print("Train Accuracy:", clf.score(X_train_s, y_train))
print("Test Accuracy:", clf.score(X_test_s, y_test))
```

Output of Cell 10:

```
Train Accuracy: 0.989010989010989
Test Accuracy: 0.9824561403508771
```