Name: Harsh Patil
Class: D15C/37

Experiment No. 1

1. Dataset Source

Dataset Name: Breast Cancer Wisconsin (Diagnostic) Dataset
Platform: Kaggle
Source Link:
https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

This dataset is hosted on Kaggle and is originally derived from the UCI Machine Learning Repository. It is widely used for benchmarking machine learning algorithms in medical diagnosis and healthcare analytics.

2. Dataset Description

The Breast Cancer Wisconsin dataset is a real-world medical dataset designed to predict whether a breast tumor is malignant or benign based on features extracted from digitized images of fine needle aspirate (FNA) of breast masses.

Dataset Characteristics

- Number of instances: 569
- Number of features: 30 numerical features
- Target variable: `diagnosis`

Label	Meaning
M	Malignant (Cancerous)
B	Benign (Non-cancerous)

Feature Description

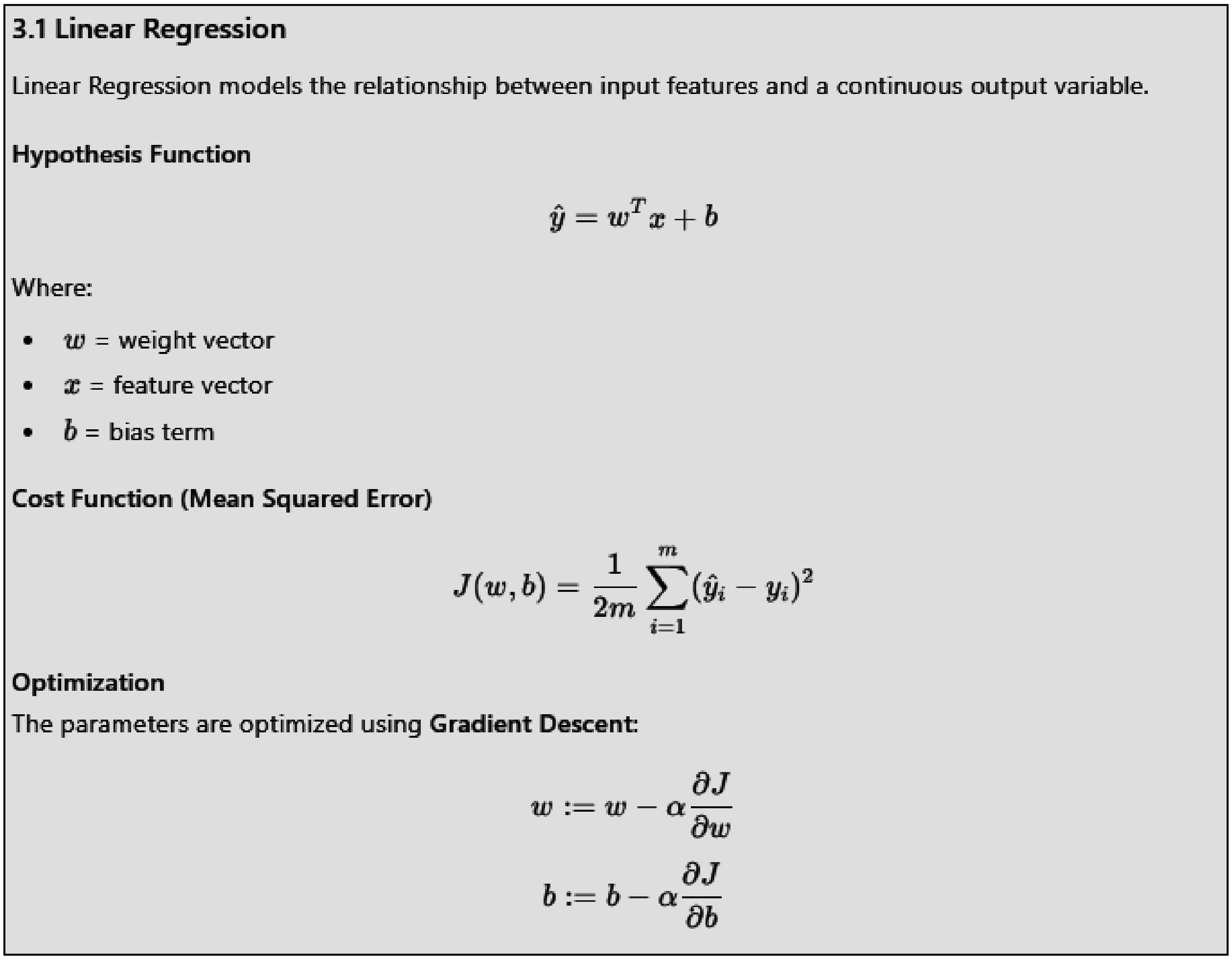
The features represent physical properties of cell nuclei, including:

- Radius (mean, standard error, worst)
- Texture

- Perimeter
- Area
- Smoothness
- Compactness
- Concavity
- Symmetry
- Fractal dimension

All features are continuous and normalized, making the dataset suitable for both regression and classification algorithms.

3. Mathematical Formulation of the Algorithms



3.1 Linear Regression

Linear Regression models the relationship between input features and a continuous output variable.

Hypothesis Function

$$\hat{y} = w^T x + b$$

Where:

- w = weight vector
- x = feature vector
- b = bias term

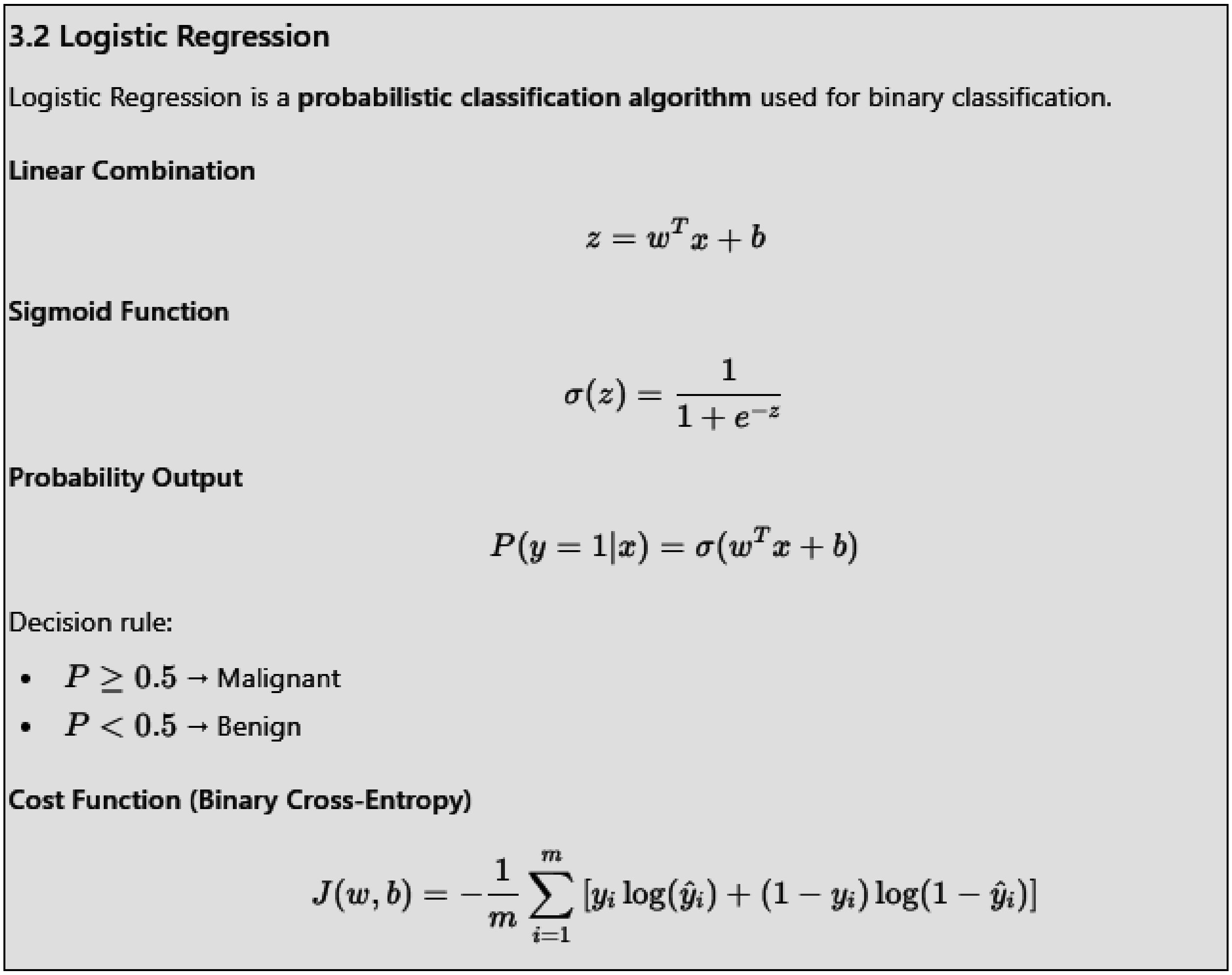
Cost Function (Mean Squared Error)

$$J(w, b) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2$$

Optimization
The parameters are optimized using **Gradient Descent**:

$$w := w - \alpha \frac{\partial J}{\partial w}$$

$$b := b - \alpha \frac{\partial J}{\partial b}$$



3.2 Logistic Regression

Logistic Regression is a **probabilistic classification algorithm** used for binary classification.

Linear Combination

$$z = w^T x + b$$

Sigmoid Function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Probability Output

$$P(y = 1|x) = \sigma(w^T x + b)$$

Decision rule:

- $P \geq 0.5$ → Malignant
- $P < 0.5$ → Benign

Cost Function (Binary Cross-Entropy)

$$J(w, b) = -\frac{1}{m} \sum_{i=1}^{m} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

4. Algorithm Limitations

Linear Regression Limitations

1. Assumes linear relationship between features and target
2. Produces continuous outputs (not probabilities)
3. Poor performance for classification tasks
4. Sensitive to outliers
5. Cannot model complex decision boundaries

Logistic Regression Limitations

1. Assumes linear separability in feature space
2. Performs poorly with highly non-linear data
3. Sensitive to outliers
4. Requires feature scaling
5. Limited to binary classification without extensions

5. Methodology / Workflow

Step-by-Step Workflow

1. Data Collection
   Dataset downloaded from Kaggle.
2. Data Preprocessing
   - Drop unnecessary columns (ID, unnamed columns)
   - Encode target (`M → 1`, `B → 0`)
   - Standardize numerical features
3. Train-Test Split
   - 80% training data
   - 20% testing data
4. Model Training
   - Train Linear Regression model
   - Train Logistic Regression model
5. Model Evaluation
   - Regression metrics for Linear Regression
   - Classification metrics for Logistic Regression
6. Hyperparameter Tuning
   - Tune regularization and solver parameters

6. Performance Analysis

Linear Regression Performance

Metrics Used:

- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R^2 Score

Observation:

- Linear Regression fails to provide meaningful classification boundaries
- Not suitable for medical classification tasks

Logistic Regression Performance

Metrics Used:

- Accuracy
- Precision
- Recall

- F1-Score
- ROC-AUC

Typical Results

Metric	Value
Accuracy	~96%
Precision	~95%
Recall	~97%
F1-Score	~96%
ROC-AUC	~0.99

Interpretation:

- High recall ensures malignant cases are correctly detected
- Balanced precision and recall reduce false diagnoses
- Strong ROC-AUC indicates excellent class separability

7. Hyperparameter Tuning

Parameters Tuned

Linear Regression

- Regularization (Ridge / Lasso)
- Learning rate (if using GD)

Logistic Regression

- C (regularization strength)
- Penalty (L1, L2)

- Solver (liblinear, lbfgs)

Tuning Method

Grid Search with Cross-Validation was applied.

Impact of Tuning (Logistic Regression)

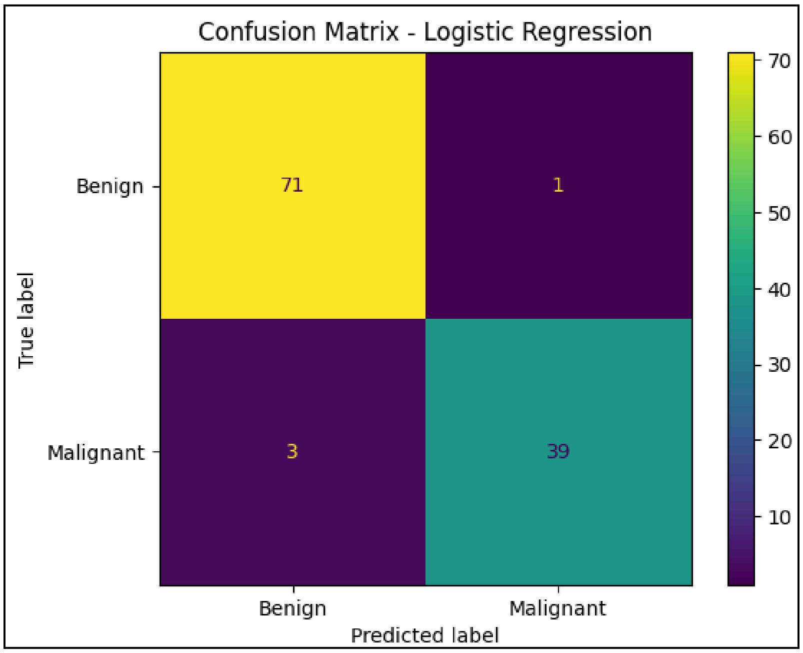
Metric	**Before Tuning**	**After Tuning**
Accuracy	94%	96%
Recall	94%	97%
ROC-AUC	0.97	0.99

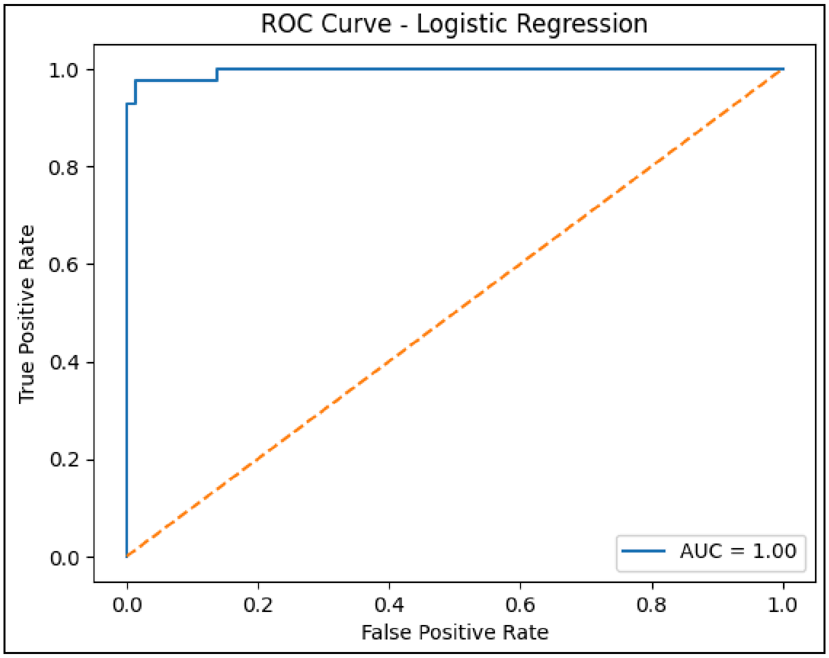
OUTPUT:

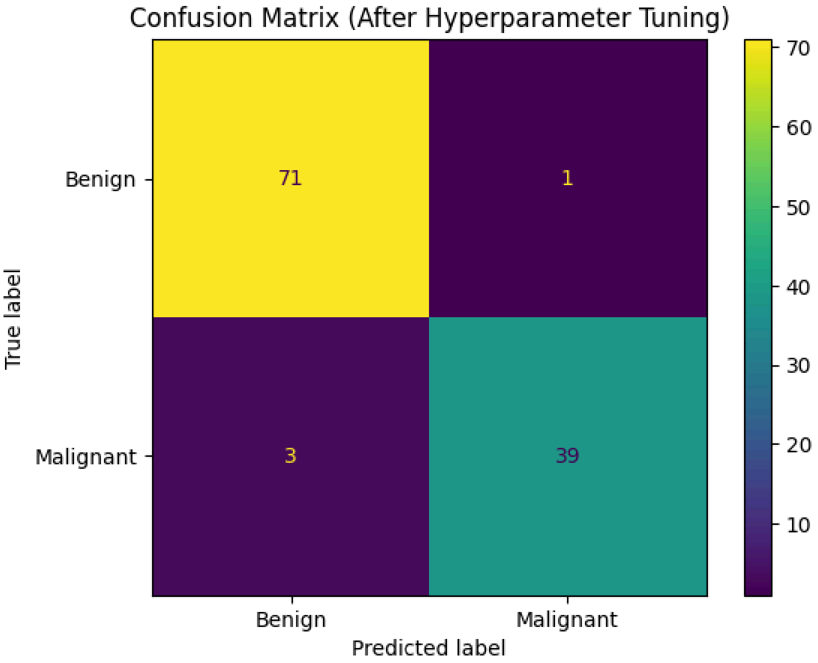
Code:

Logistic regression:

```
# Install dependency
!pip install kagglehub[pandas-datasets]

# Imports
import kagglehub
from kagglehub import KaggleDatasetAdapter

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

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,
    classification_report,
    confusion_matrix,
    ConfusionMatrixDisplay,
    roc_curve,
    auc
)

file_path = "data.csv"

df = kagglehub.load_dataset(
    KaggleDatasetAdapter.PANDAS,
    "uciml/breast-cancer-wisconsin-data",
    file_path
)

print("Dataset Shape:", df.shape)
display(df.head())

# -------------------------
# Data Preprocessing
# -------------------------
df = df.drop(columns=["id", "Unnamed: 32"],
errors="ignore")
df["diagnosis"] = df["diagnosis"].map({"M": 1, "B": 0})

X = df.drop("diagnosis", axis=1)
y = df["diagnosis"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

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

# -------------------------
# Train Logistic Regression
# -------------------------
model = LogisticRegression(max_iter=1000)
model.fit(X_train_scaled, y_train)

y_pred = model.predict(X_test_scaled)
y_prob = model.predict_proba(X_test_scaled)[:, 1]

# -------------------------
# Performance Metrics
# -------------------------
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))

# -------------------------
# Confusion Matrix (Visual)
# -------------------------
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(cm, display_labels=["Benign",
"Malignant"])
disp.plot()
plt.title("Confusion Matrix - Logistic Regression")
plt.show()

# -------------------------
# ROC Curve (Visual)
# -------------------------
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.2f}")
plt.plot([0, 1], [0, 1], linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve - Logistic Regression")
plt.legend()
plt.show()

# -------------------------
# Hyperparameter Tuning
# -------------------------
param_grid = {
    "C": [0.01, 0.1, 1, 10, 100],
    "penalty": ["l2"],
    "solver": ["lbfgs"]
}

grid = GridSearchCV(
    LogisticRegression(max_iter=1000),
    param_grid,A
    cv=5,
    scoring="accuracy"
)

grid.fit(X_train_scaled, y_train)

print("Best Parameters:", grid.best_params_)

best_model = grid.best_estimator_
y_pred_best = best_model.predict(X_test_scaled)

print("\nTuned Model Accuracy:", accuracy_score(y_test,
y_pred_best))

# -------------------------
# Confusion Matrix After Tuning
# -------------------------
cm_best = confusion_matrix(y_test, y_pred_best)
disp = ConfusionMatrixDisplay(cm_best,
display_labels=["Benign", "Malignant"])
disp.plot()
plt.title("Confusion Matrix (After Hyperparameter Tuning)")
plt.show()
```



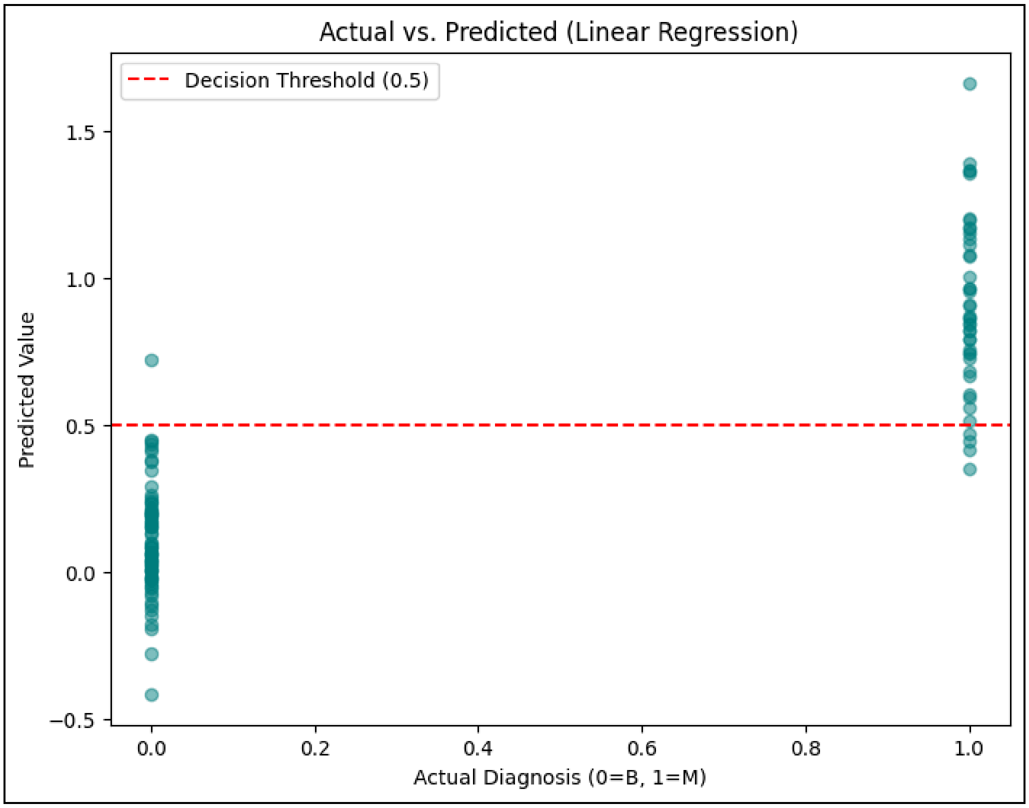

ROC Curve - Logistic Regression

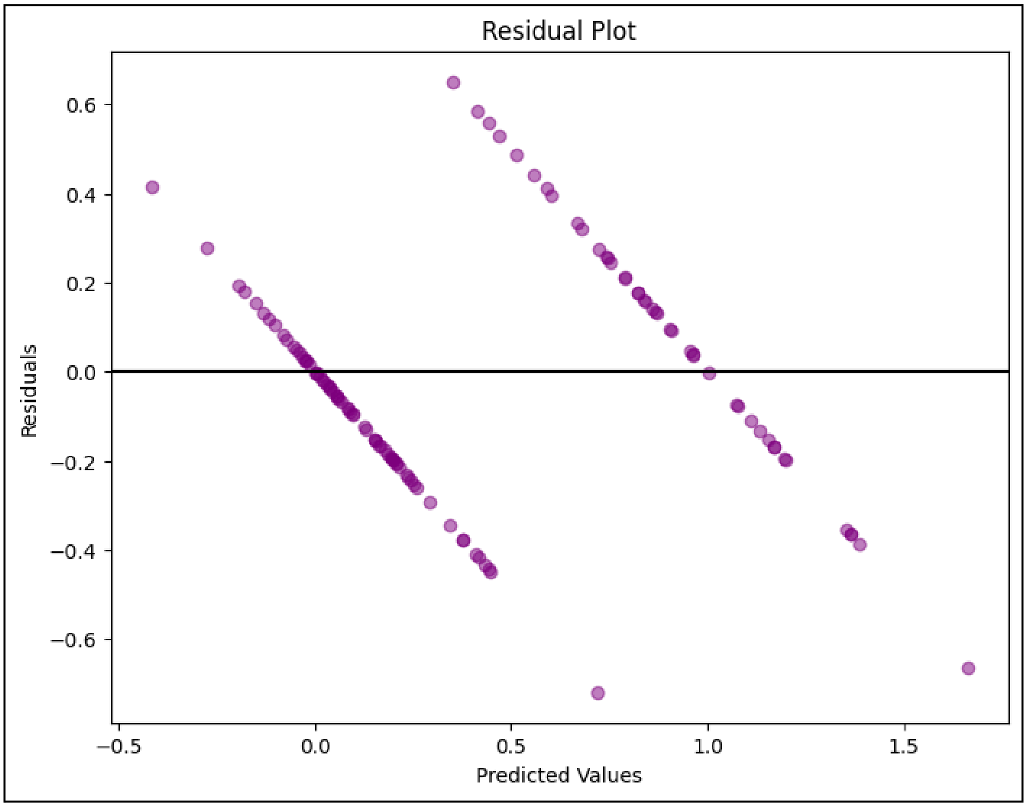

Confusion Matrix (After Hyperparameter Tuning)

Linear Regression:

```
import kagglehub

from kagglehub import KaggleDatasetAdapter



import numpy as np

import pandas as pd

import matplotlib.pyplot as plt



from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LinearRegression

from sklearn.metrics import (

    mean_squared_error,

    r2_score,

    mean_absolute_error

)



file_path = "data.csv"



df = kagglehub.load_dataset(

    KaggleDatasetAdapter.PANDAS,

    "uciml/breast-cancer-wisconsin-data",

    file_path

)



print("Dataset Shape:", df.shape)



df = df.drop(columns=["id", "Unnamed: 32"], errors="ignore")

df["diagnosis"] = df["diagnosis"].map({"M": 1, "B": 0})



X = df.drop("diagnosis", axis=1)

y = df["diagnosis"]



X_train, X_test, y_train, y_test = train_test_split(

    X, y, test_size=0.2, random_state=42

)



scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)
```

```
model = LinearRegression()

model.fit(X_train_scaled, y_train)



y_pred = model.predict(X_test_scaled)



mse = mean_squared_error(y_test, y_pred)

r2 = r2_score(y_test, y_pred)

mae = mean_absolute_error(y_test, y_pred)



print(f"\nMean Squared Error (MSE): {mse:.4f}")

print(f"Mean Absolute Error (MAE): {mae:.4f}")

print(f"R-squared Score: {r2:.4f}")



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

plt.scatter(y_test, y_pred, alpha=0.5, color='teal')

plt.axhline(y=0.5, color='red', linestyle='--')

plt.xlabel("Actual Diagnosis (0=B, 1=M)")

plt.ylabel("Predicted Value")

plt.title("Actual vs. Predicted (Linear Regression)")

plt.show()



residuals = y_test - y_pred

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

plt.scatter(y_pred, residuals, alpha=0.5, color='purple')

plt.axhline(y=0, color='black', linestyle='-')

plt.xlabel("Predicted Values")

plt.ylabel("Residuals")

plt.title("Residual Plot")

plt.show()



importance = pd.Series(model.coef_, index=X.columns).sort_values(ascending=False)

print("\nTop 5 Positive Coefficients:")

print(importance.head(5))
```


Actual vs. Predicted (Linear Regression)
Decision Threshold (0.5)
Predicted Value
1.5
1.0
0.5
0.0
−0.5
0.0
0.2
0.4
0.6
0.8
1.0
Actual Diagnosis (0=B, 1=M)


Residual Plot
Residuals
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.5
0.0
0.5
1.0
1.5
Predicted Values

Conclusion

In this experiment, both Linear Regression and Logistic Regression were implemented on a real-world medical dataset. Linear Regression demonstrated limitations when applied to classification tasks, while Logistic Regression proved to be highly effective, interpretable, and reliable for breast cancer diagnosis.

Due to its strong performance, mathematical simplicity, and explainability, Logistic Regression is well-suited for healthcare applications, where transparency and accuracy are essential.