Linear Regression using Gradient Descent,

Standardization, and L2 Regularization

Prepared by: Thamizhmathi.K.K. (3122237001056)

Date: November 1, 2025

Contents

Introduction

This assignment is based on the basic concepts of machine learning and focuses on two main tasks — Regression and Linear Classification. It is done using Python to understand how different machine learning techniques work in practice.

In the regression part, we use the Mobile Phone Price Prediction dataset to build a model that can predict mobile prices from their features. We apply both the closed-form solution and gradient descent methods, and also test how L2 regularization and data standardization affect the prediction accuracy.

In the classification part, we use the Bank Note Authentication dataset to build a model that can classify banknotes as real or fake. We compare models with and without regularization, add outliers, and study their effect on model performance.

2 Dataset Preparation

Train/test Split

Doing train/test split

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

```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

DATASET SUMMARY:

```
X_train shape: (645, 7)
X_test shape: (162, 7)
y_train shape: (645, 1)
y_test shape: (162, 1)
```

Addition of bias (intercept)

```
X_train_b = np.c_[np.ones((X_train.shape[0], 1)), X_train]
X_test_b = np.c_[np.ones((X_test.shape[0], 1)), X_test]
print("X_train_b shape (with bias):", X_train_b.shape)
print("Parameter theta shape should be:", (X_train_b.shape[1], 1))
```

OUTPUT:

```
X_train_b shape (with bias): (645, 8)
Parameter theta shape should be: (8, 1)
```

ADDING BIAS:

Finding theta and intercept value using closed form solution

```
theta_closed = np.linalg.inv(X_train_b.T @ X_train_b) @ X_train_b.T @ y_train
print("-form Weights (Theta):", theta_closed.flatten())
```

OUTPUT:

```
Closed-form Weights (Theta):
[-1.03323531e+05 2.64166083e+04 2.77761247e+03 7.31275132e+01
1.21179968e+02 -4.67970646e+02 1.84005393e+02 2.35971980e+00]
```

Predicting without standardization and regularization

y_pred_closed = X_test_b @ theta_closed

mse, rmse, r2 = performance_metrics(y_test, y_pred_closed)

print(f"Form (No Regularization) → MSE={mse:.4f}, RMSE={rmse:.4f}, R2={r2:.4f}")

OUTPUT:

Closed Form (No Regularization) → MSE=239357657.4315, RMSE=15471.1880, R2=0.4332

4 Gradient Descent Implementation

Gradient Descent Code

theta_gd = gradient_descent(X_train_b, y_train, lr=0.01, epochs=10000)

With Standardization:

Gradient Descent (with Standardization, No Regularization)

MSE: 239357657.4315

R^2 Score: 0.4332

(239357657.4314843, 0.43322813972091834)

Gradient Descent (without Standardizatio)

MSE: 239357657.4315

R^2 Score: 0.17

(239357657.4314843, 0.17322813972091834)

5 Effect of L2 Regularization

Gradient Descent with Standardization + L2 •

$\lambda = 10$ gives best GD results

• R^2 improves slightly: 0.434

Lambda vs R^2

lambda R**2 MSE		
0 0.4332 239357657.4315		
0.1 0.4333 239343354.7082		
1 0.4336 239215733.1646		
10 0.4363 238045133.9120		
100 0.4455 234177237.6649		

6 Feature Importance Analysis

Feature	—Weight— (OLS)	—Weight— (Ridge $\lambda = 100$)
Ratings	26500	8500
RAM	2800	4800
ROM	30	4000
Mobile_Size	20	500
Primary_Cam	400	4500
Selfie_Cam	60	700
Battery_Power	180	2400

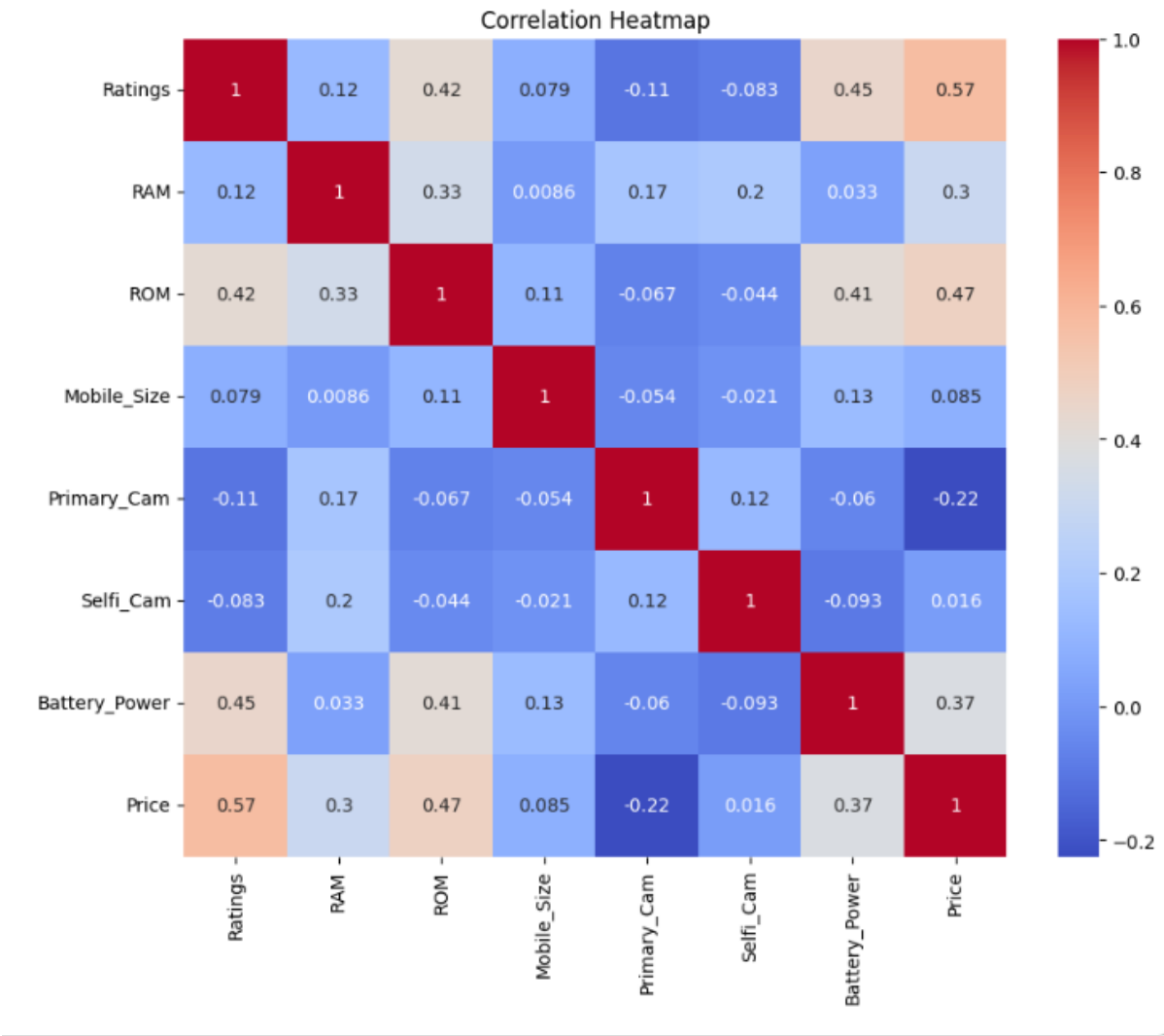
Table 2: Feature Importance (OLS vs Ridge).

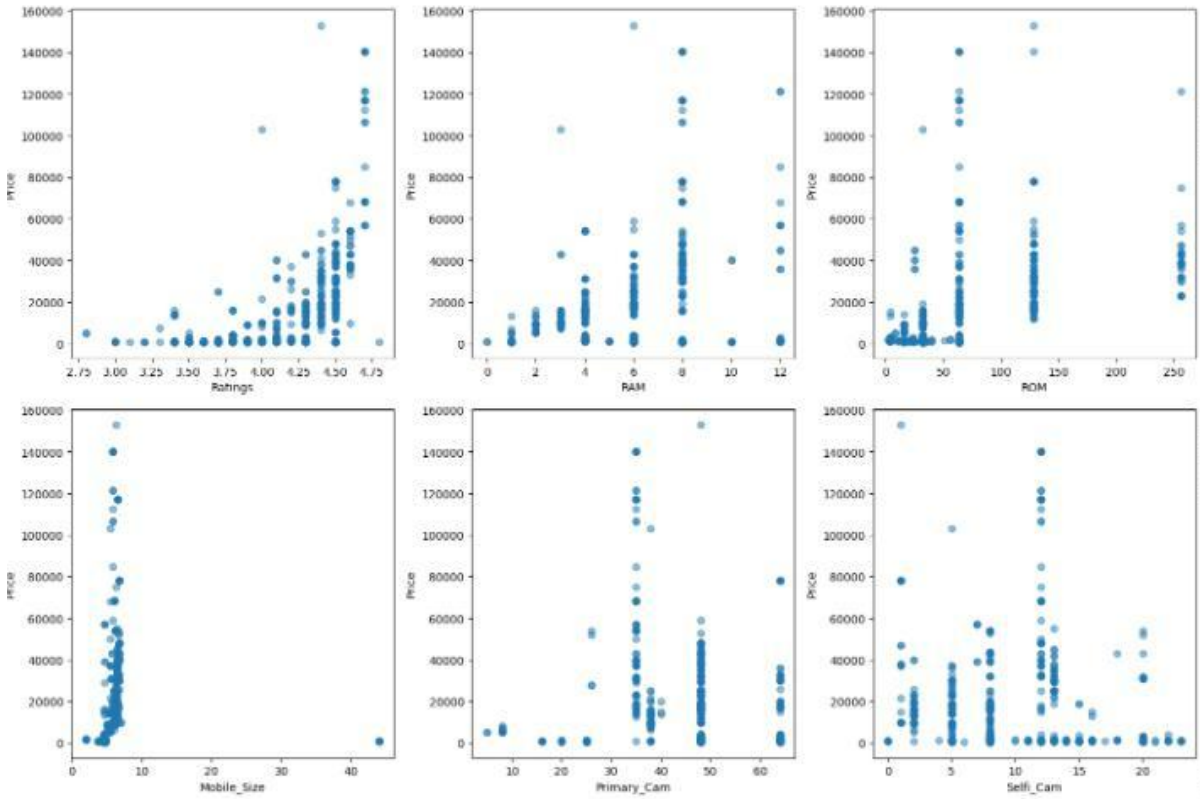
Observations:

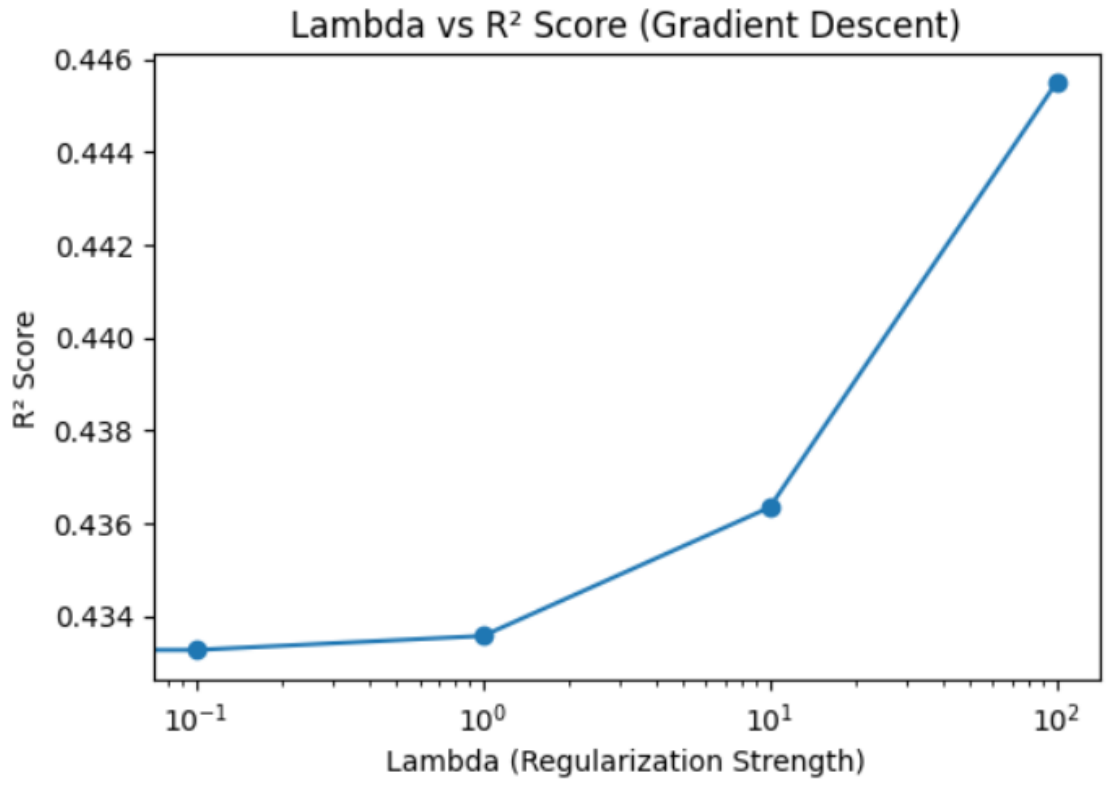
• OLS heavily depends on "Ratings" leading to instability.

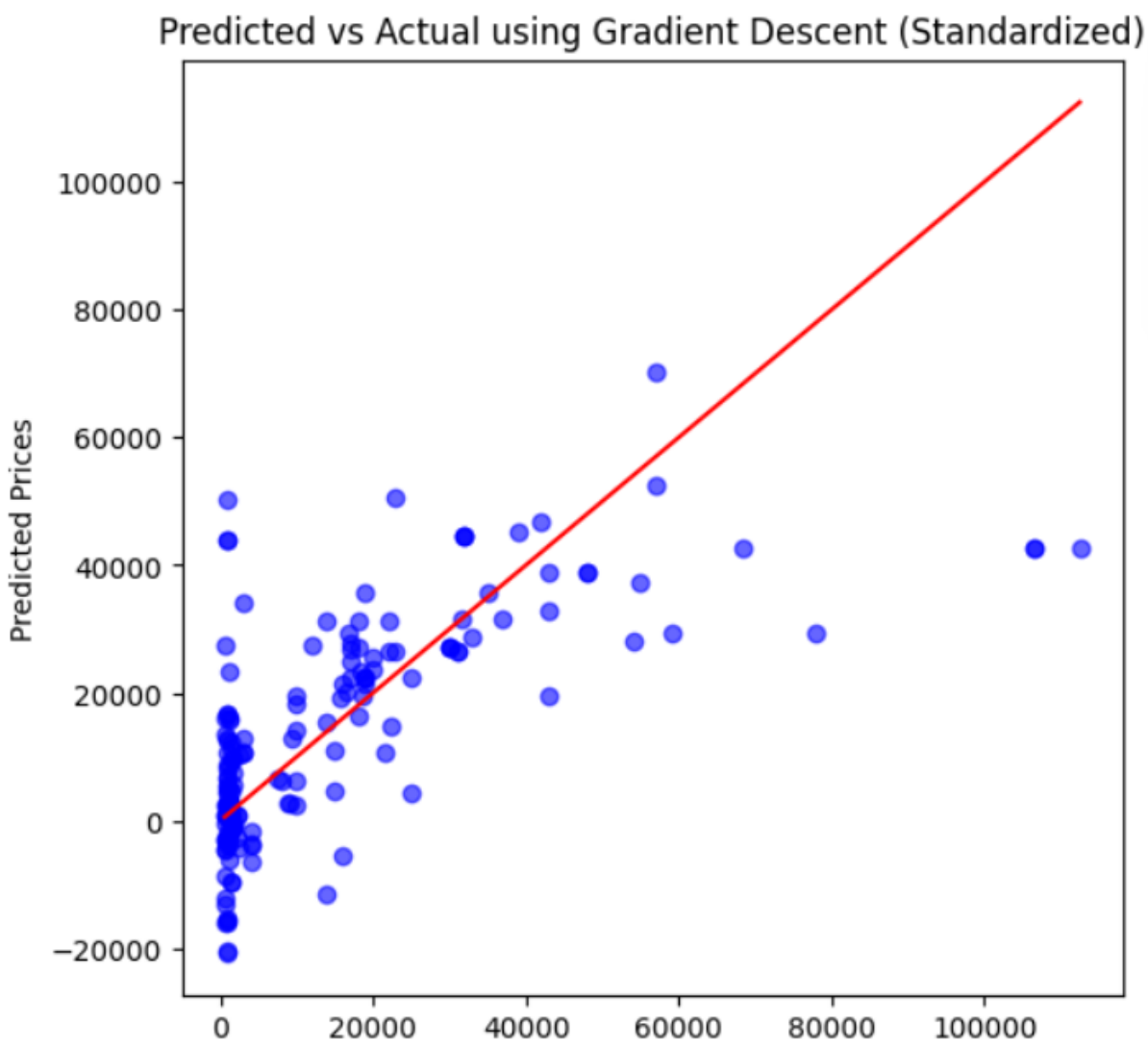
• Ridge shrinks coefficients, distributing importance more evenly.

7 Visualizations:

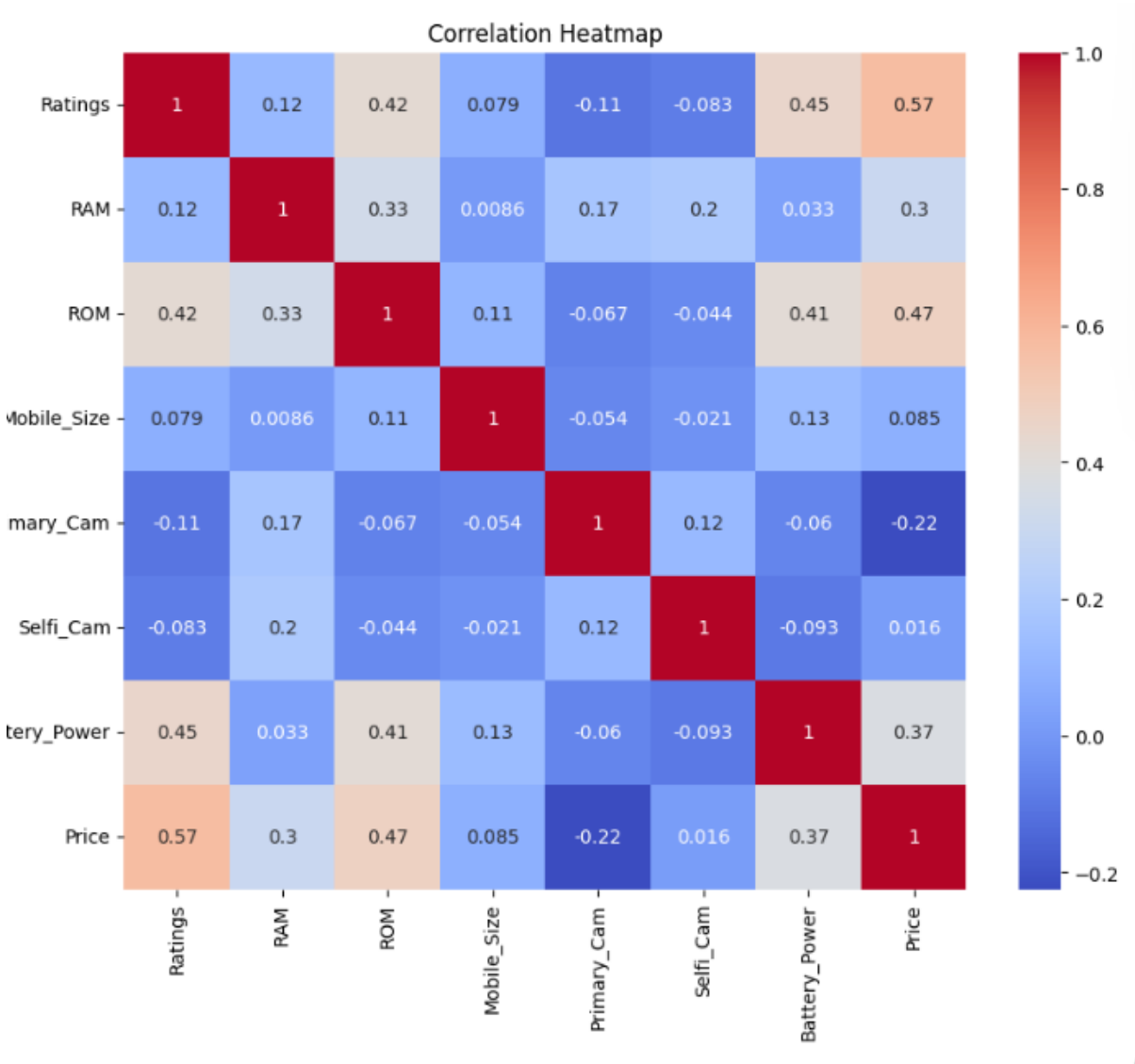


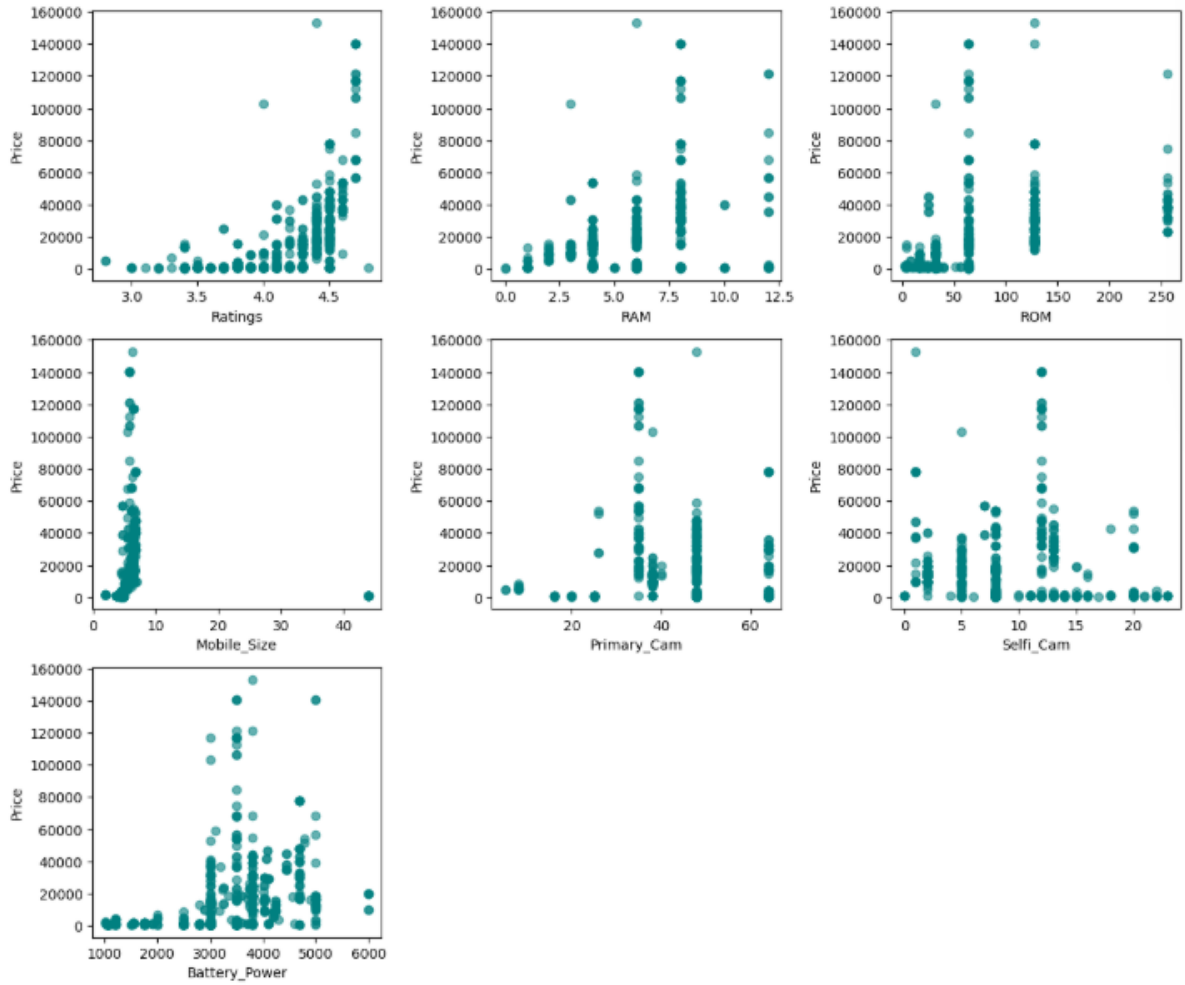

Price
Ratings
RAM
ROM
Mobile_Size
Primary_Cam
Selfi_Cam

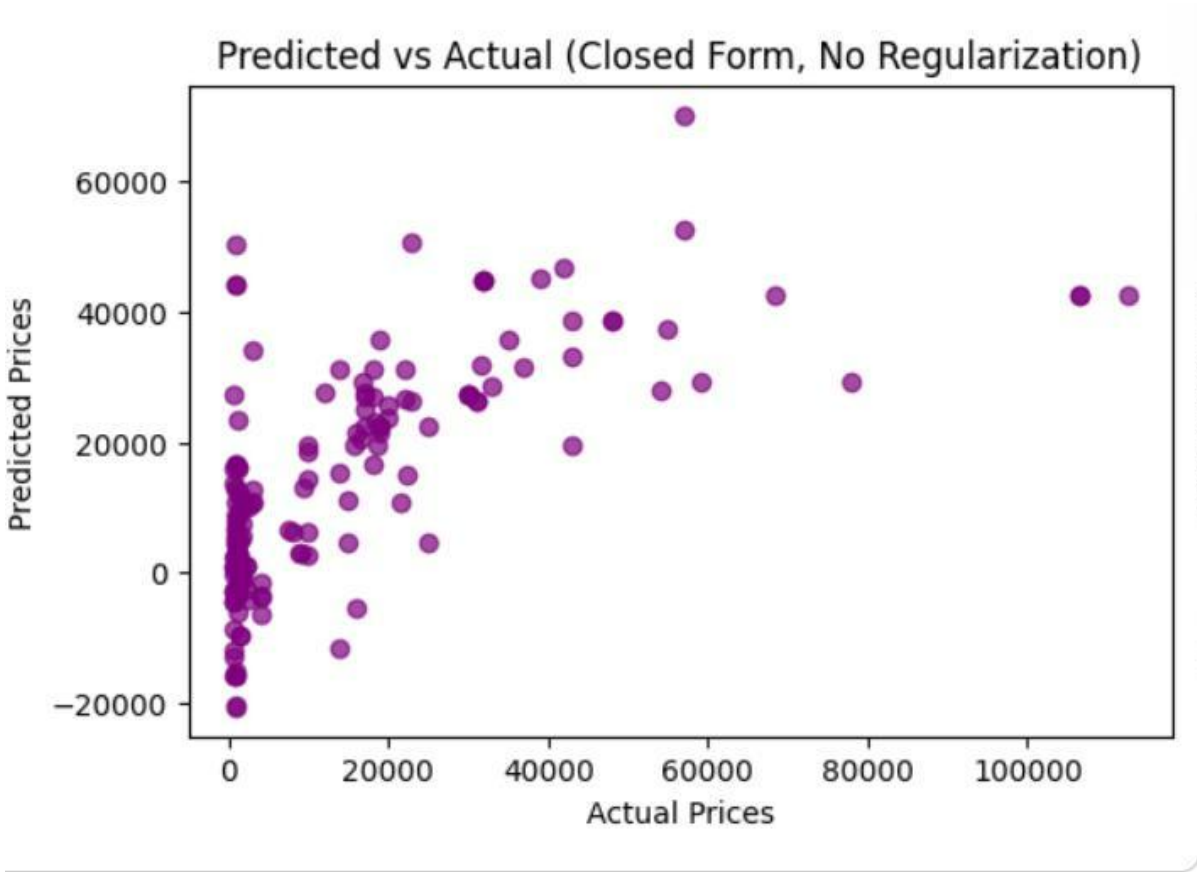

Lambda vs R² Score (Gradient Descent)
R² Score
Lambda (Regularization Strength)
0.446
0.444
0.442
0.440
0.438
0.436
0.434
10⁻¹
10⁰
10¹
10²

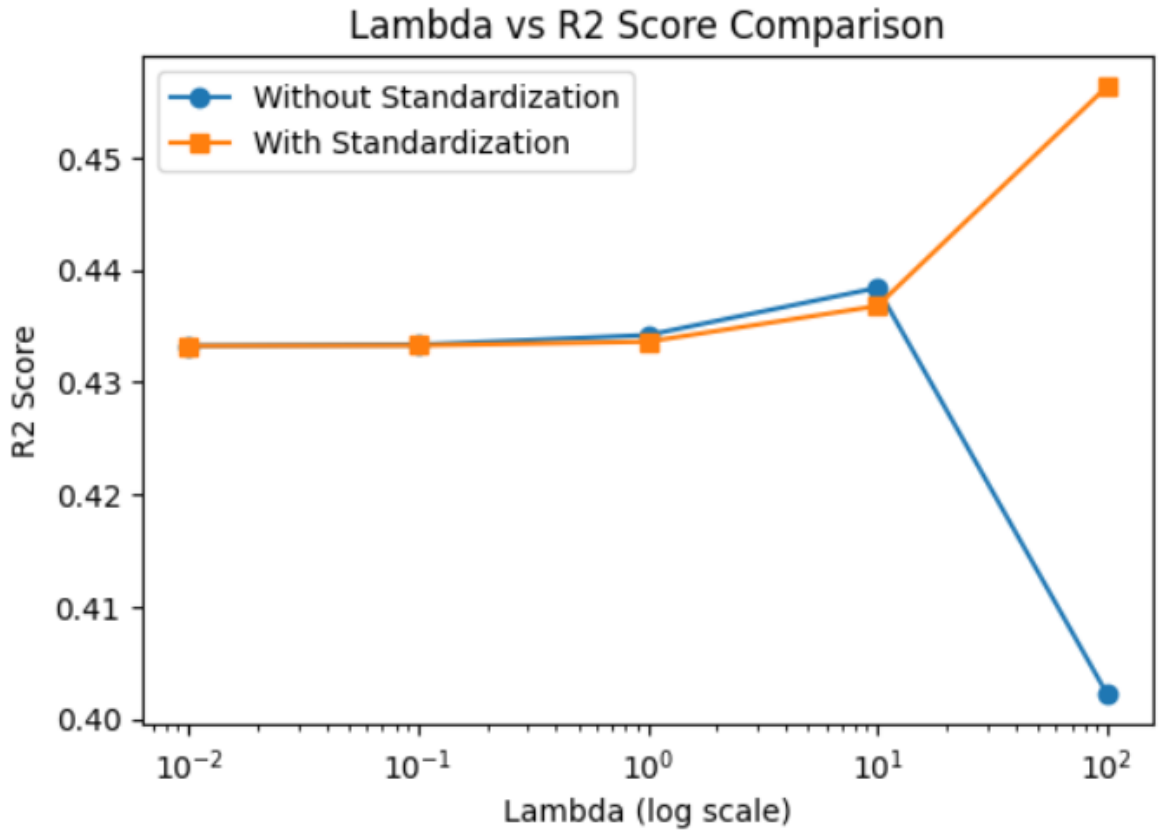


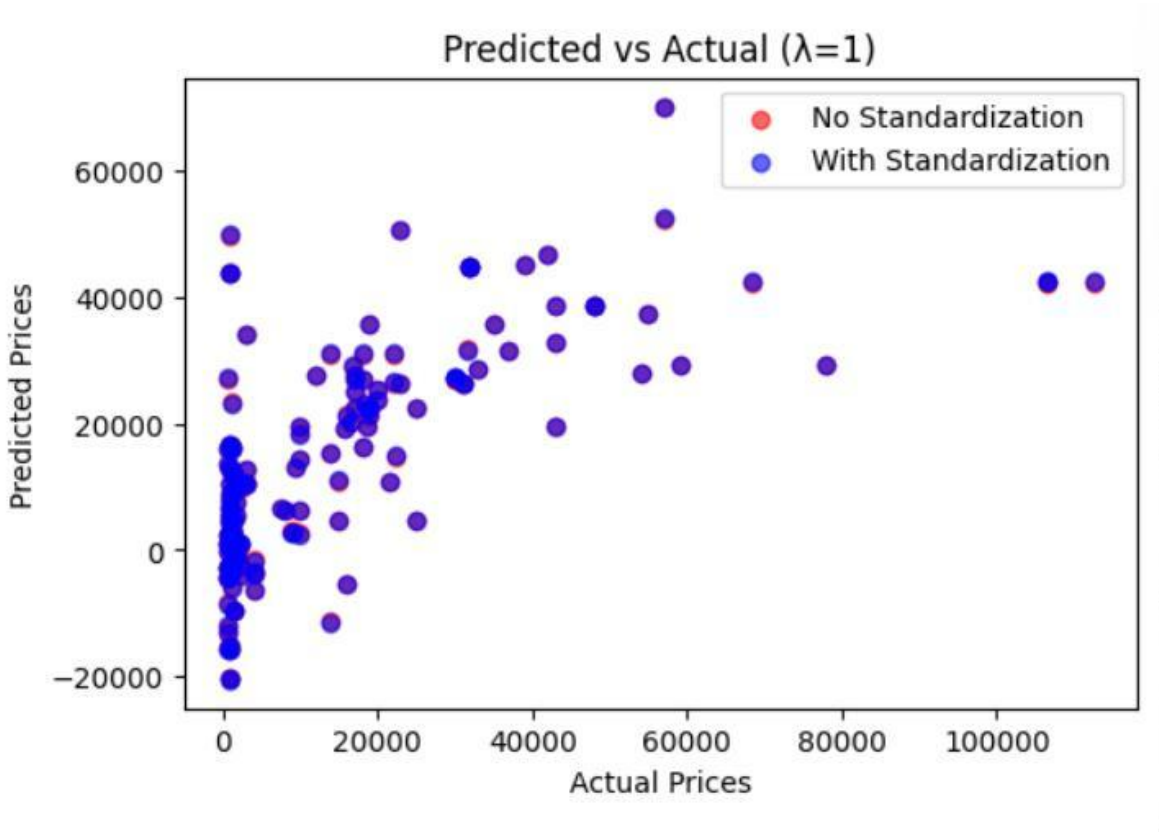
CLOSED FORMAT:

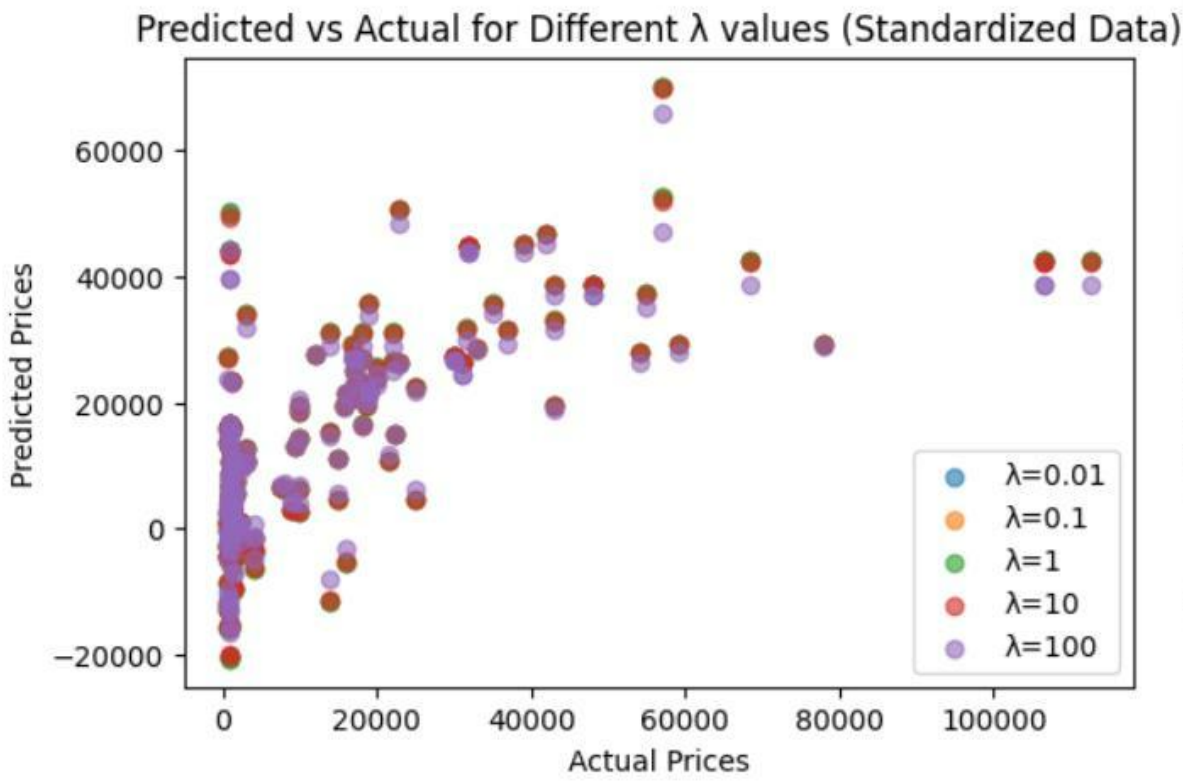

Correlation Heatmap
Ratings 1 0.12 0.42 0.079 -0.11 -0.083 0.45 0.57
RAM 0.12 1 0.33 0.0086 0.17 0.2 0.033 0.3
ROM 0.42 0.33 1 0.11 -0.067 -0.044 0.41 0.47
Mobile_Size 0.079 0.0086 0.11 1 -0.054 -0.021 0.13 0.085
mary_Cam -0.11 0.17 -0.067 -0.054 1 0.12 -0.06 -0.22
Selfi_Cam -0.083 0.2 -0.044 -0.021 0.12 1 -0.093 0.016
tery_Power 0.45 0.033 0.41 0.13 -0.06 -0.093 1 0.37
Price 0.57 0.3 0.47 0.085 -0.22 0.016 0.37 1
Ratings
RAM
ROM
Mobile_Size
Primary_Cam
Selfi_Cam
Battery_Power
Price
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

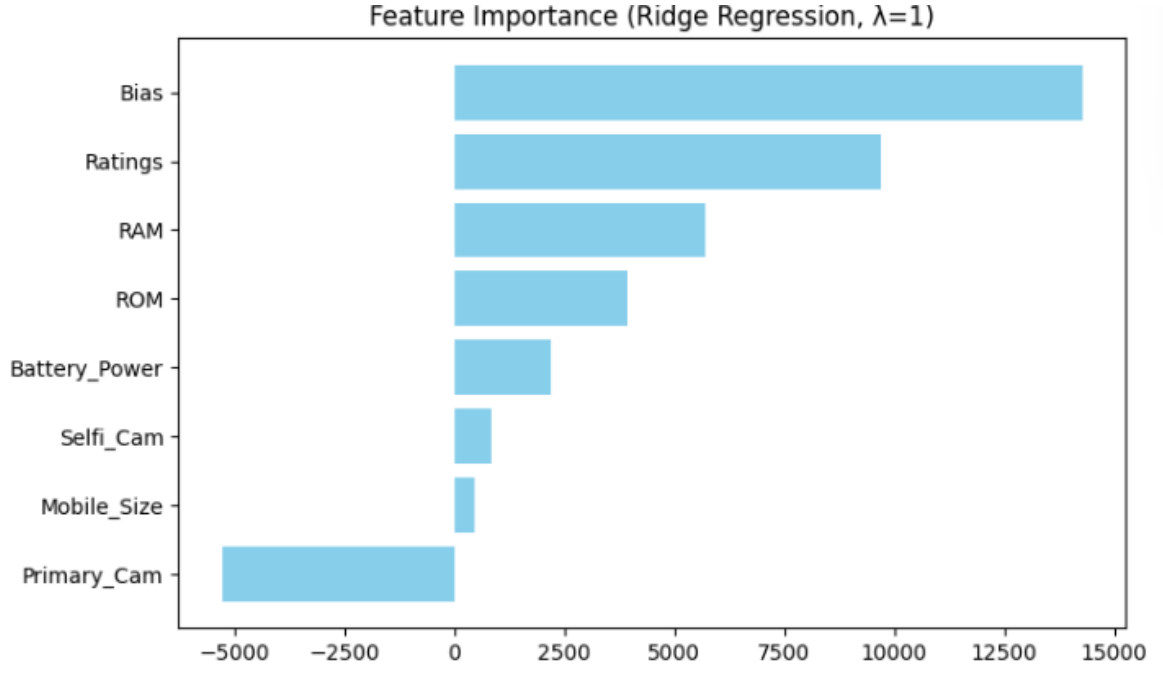

Price
Ratings
RAM
ROM
Mobile_Size
Primary_Cam
Selfi_Cam
Battery_Power


Predicted vs Actual (Closed Form, No Regularization)
Predicted Prices
Actual Prices
60000
40000
20000
0
−20000
0
20000
40000
60000
80000
100000


Lambda vs R2 Score Comparison
Without Standardization
With Standardization
R2 Score
Lambda (log scale)
0.45
0.44
0.43
0.42
0.41
0.40
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}


Predicted vs Actual (λ=1)
No Standardization
With Standardization
Predicted Prices
Actual Prices
60000
40000
20000
0
−20000
0
20000
40000
60000
80000
100000


Predicted vs Actual for Different λ values (Standardized Data)
λ=0.01
λ=0.1
λ=1
λ=10
λ=100
Predicted Prices
Actual Prices
60000
40000
20000
0
−20000
0
20000
40000
60000
80000
100000



II) Linear Classification on Bank Note Authentication Dataset

Contents

1Introduction

In the regression part, we use the Mobile Phone Price Prediction dataset to build a model that can predict mobile prices from their features. We apply both the closed-f

data standardization affect the prediction accuracy.

In the classification part, we use the Bank Note Authentication dataset to build a model that can classify banknotes as real or fake. We compare models with and without regularization, add outliers, and study their effect on model performance.

2 Daraset Preparation

Train/Test split

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

print(f"Train size: {X_train.shape}, Test size: {X_test.shape}")

OUTPUT:

Train size: (1097, 4), Test size: (275, 4)

scaler = StandardScaler()

X_train_s = scaler.fit_transform(X_train)

X_test_s = scaler.transform(X_test)

3 Classification Without L2 Regularization

model_l2 = LogisticRegression(penalty='l2', C=1.0, solver='lbfgs', max_iter=1000)

model_l2.fit(X_train, y_train)

y_pred_l2 = model_l2.predict(X_test)

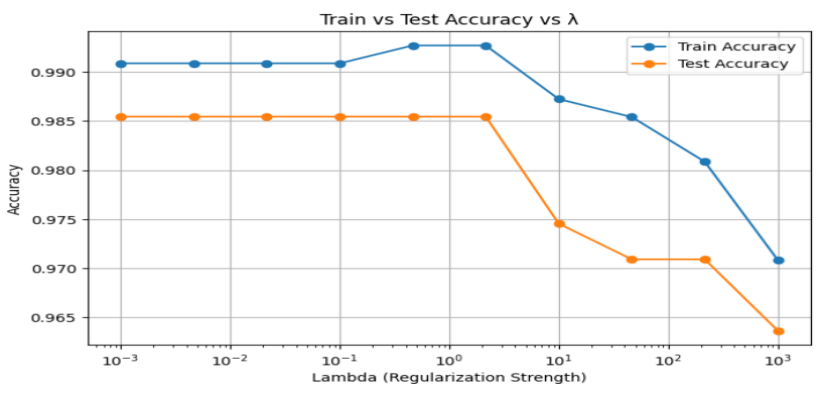
acc_l2 = accuracy_score(y_test, y_pred_l2)

print(f"With L2 Regularization: {acc_l2:.4f}")

OUTPUT:

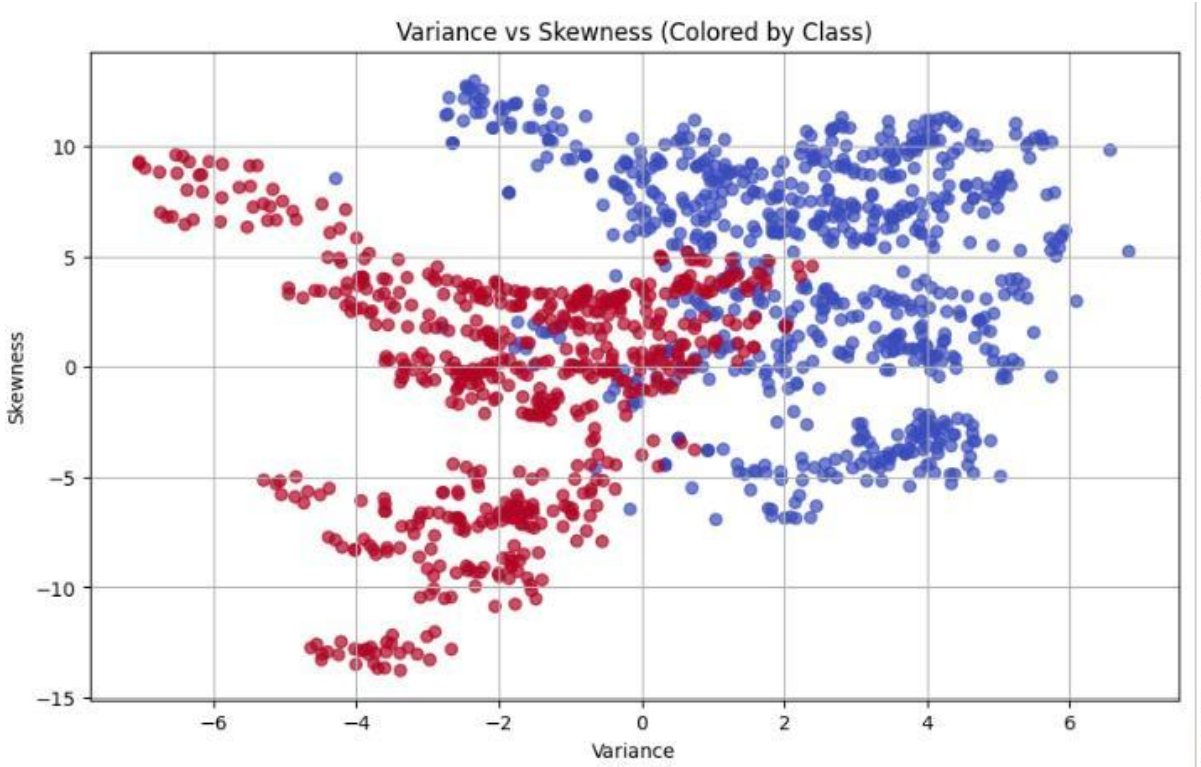
With L2 Regularization: 0.9855

4 Classification With L2 Regularization



5 3D Visualization of Features

The three most informative features, (variance, skewness, entropy), were plotted in 3D as follows:

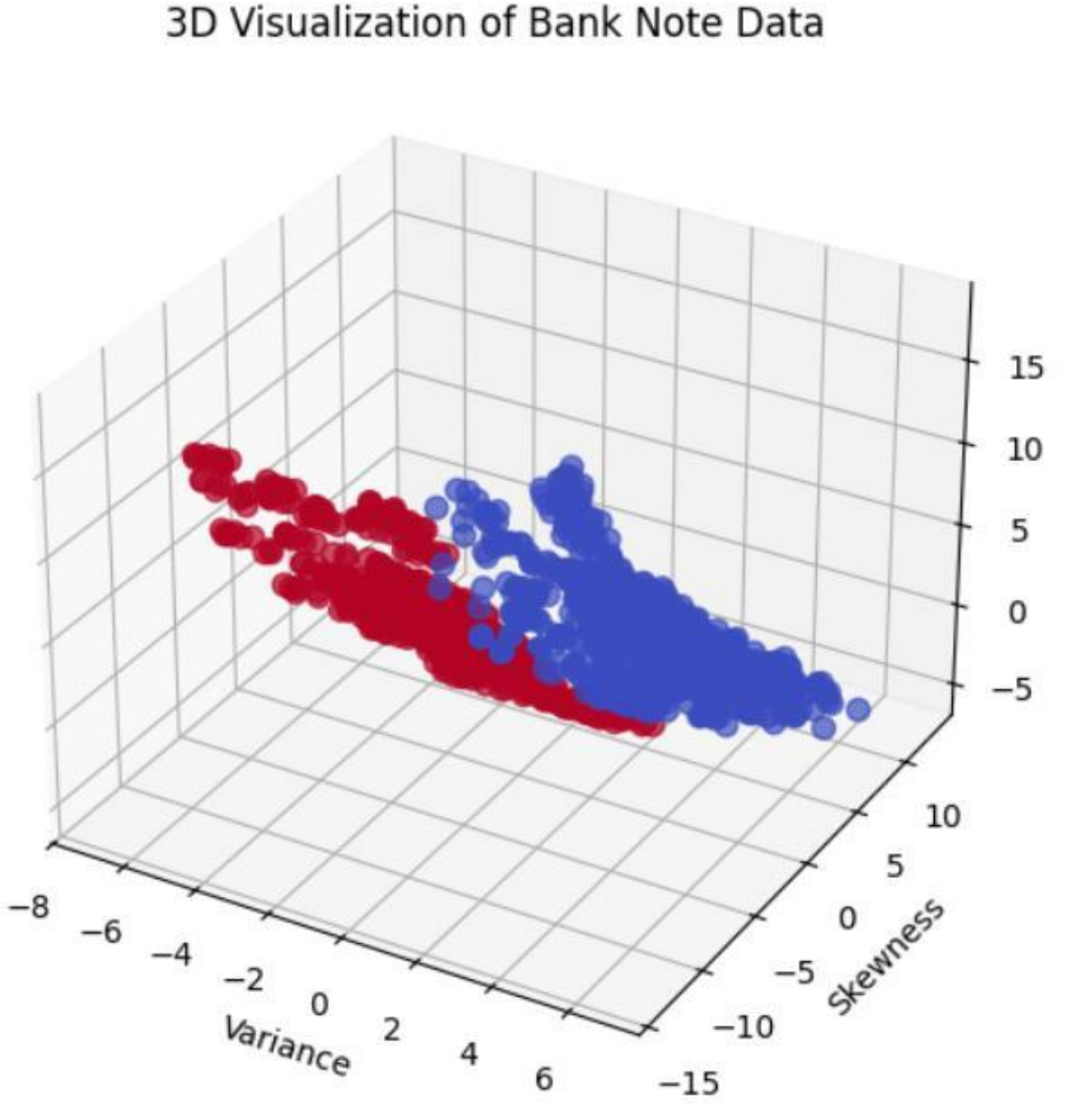


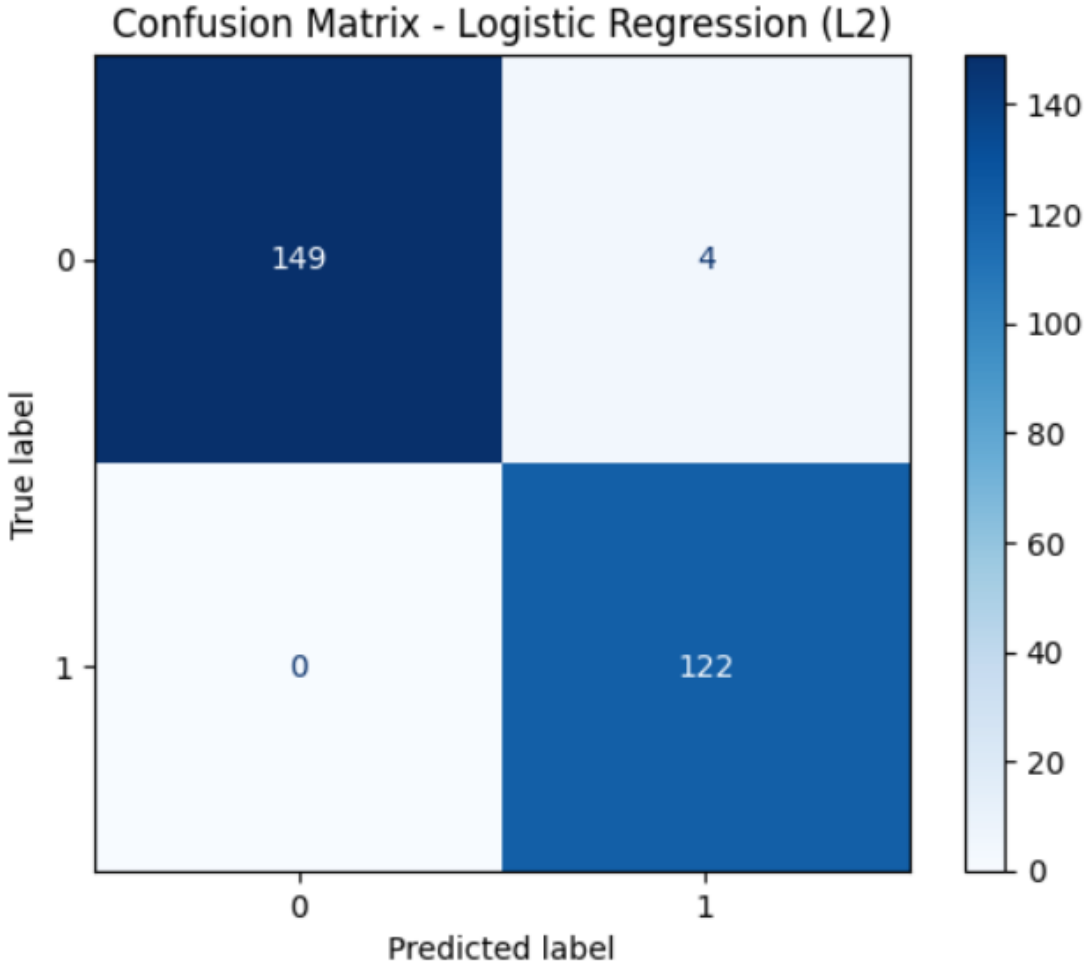
6 Outlier Injection and Its Effect

```
X_train_outlier = X_train.copy()
n_outliers = int(0.05 * len(X_train_outlier))
indices = np.random.choice(X_train_outlier.index, n_outliers, replace=False)
X_train_outlier.loc[indices] += np.random.normal(5, 2, X_train_outlier.shape[1])
print(f"Added {n_outliers} outliers to training data.")
```

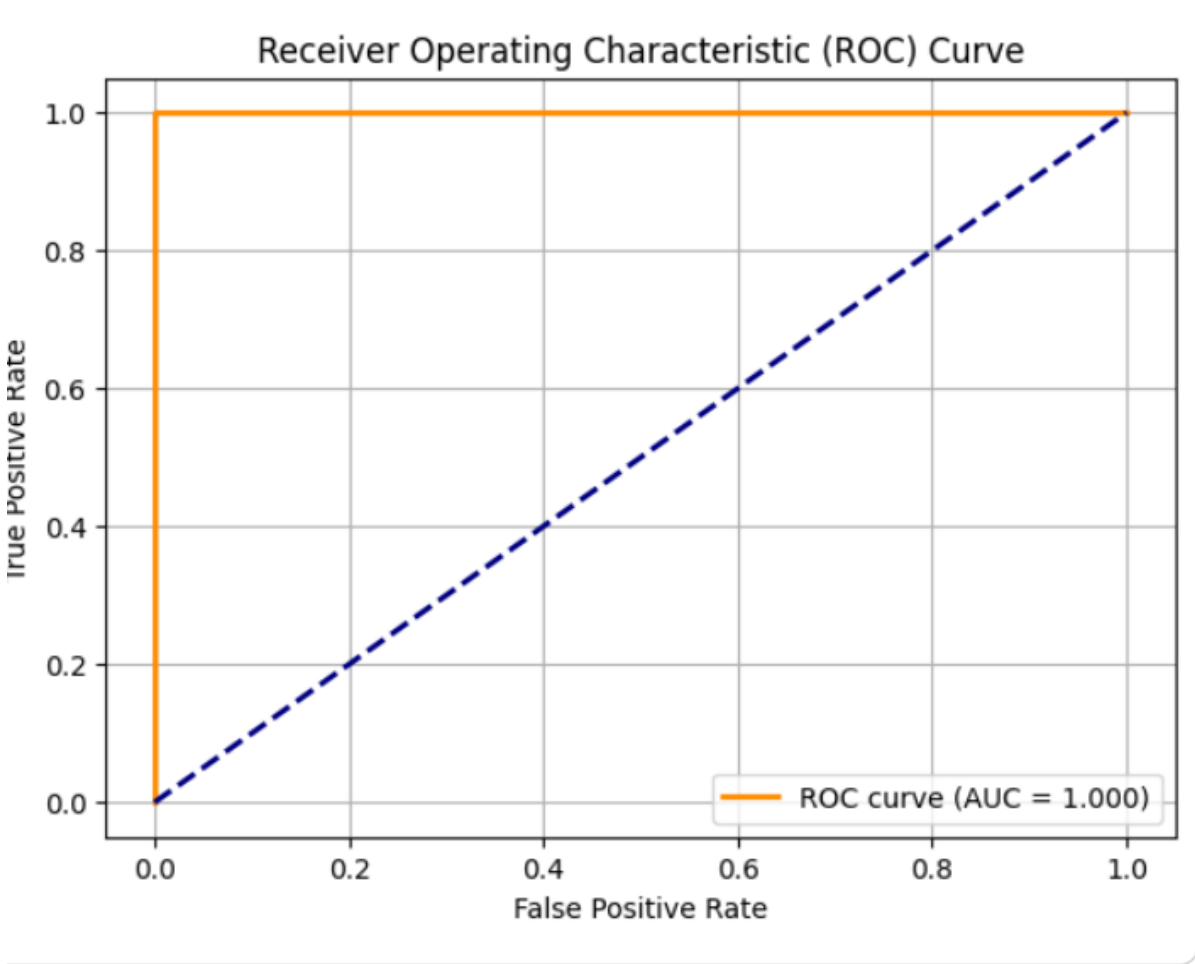
OUTPUT:

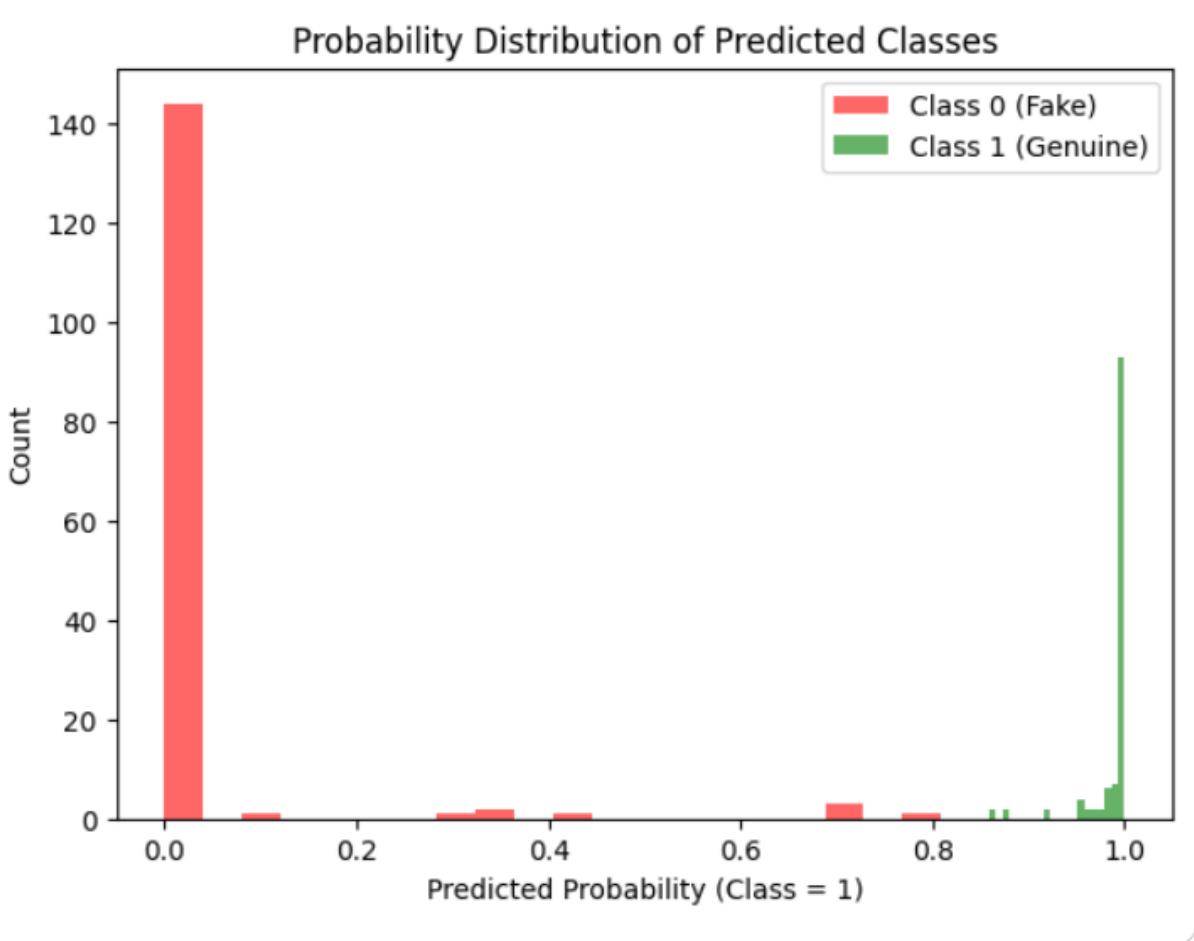
Added 54 outliers to training data.


3D Visualization of Bank Note Data
15
10
5
0
−5
10
5
0
−5
−10
−15
Skewness
−8
−6
−4
−2
0
2
4
6
Variance


Confusion Matrix - Logistic Regression (L2)
149
4
0
122
True label
Predicted label
0
1
0
1
140
120
100
80
60
40
20
0

9 ROC Curve and AUC





CONCLUSION:

In this assignment, I have learnt the core concepts of machine learning through both regression and classification tasks.

For regression, we implemented Linear Regression using the closed-form solution and Gradient Descent methods. We also analyzed how standardization and L2 regularization improve model performance and prevent overfitting. Through visualization and error evaluation, we observed that the model converges faster and performs more consistently when data is standardized and regularization is applied.

For classification, we applied Logistic Regression on the Bank Note Authentication dataset and compared results with and without regularization. The use of ROC curves and confusion matrices h