Comparison of Classification Models

Binary Outcome Prediction

Logistic Regression (Penalized) SVM & Random Forest

Data Cleaning

1. **Removed blank or missing values**
   By replacing them with appropriate values (mean imputation for numerical columns), ensuring no empty cells interfere with analysis.

2. **Ensured consistent data types**
   Such as, converting the target variable (LABEL) into a categorical format suitable for classification.

3. **Dropped Constant Columns**
   Features with only one unique value, as they do not provide any meaningful information for modeling.

4. **Performed Class-Balanced Splitting**
   Features with only one unique value, as they do not provide any meaningful information for modeling.

5. **Retained all necessary columns**
   Particularly the target variable, after filtering to prevent any accidental data loss.

6. **Removed duplicate or redundant features**
   Especially before applying dimensionality reduction techniques like PCA.

7. **Validated dataset structure and integrity**
   This has been performed post-cleaning to ensure it was ready for downstream processing such as feature engineering and modeling.

Class Imbalance Handling

Our dataset involves a binary classification task where the target variable is LABEL.
During stratified train-test splitting, we observed that the training set had class imbalance, i.e., unequal representation of class 0 and class 1.
This imbalance can bias the model toward the majority class, reducing the predictive performance for the minority class.

A classifier trained on imbalanced data tends to ignore the minority class, which may be the more important one (e.g., fraud detection, disease diagnosis).
Metrics like accuracy become misleading, as a model might achieve high accuracy by simply predicting the majority class.
We needed a technique to improve class balance without losing data integrity.

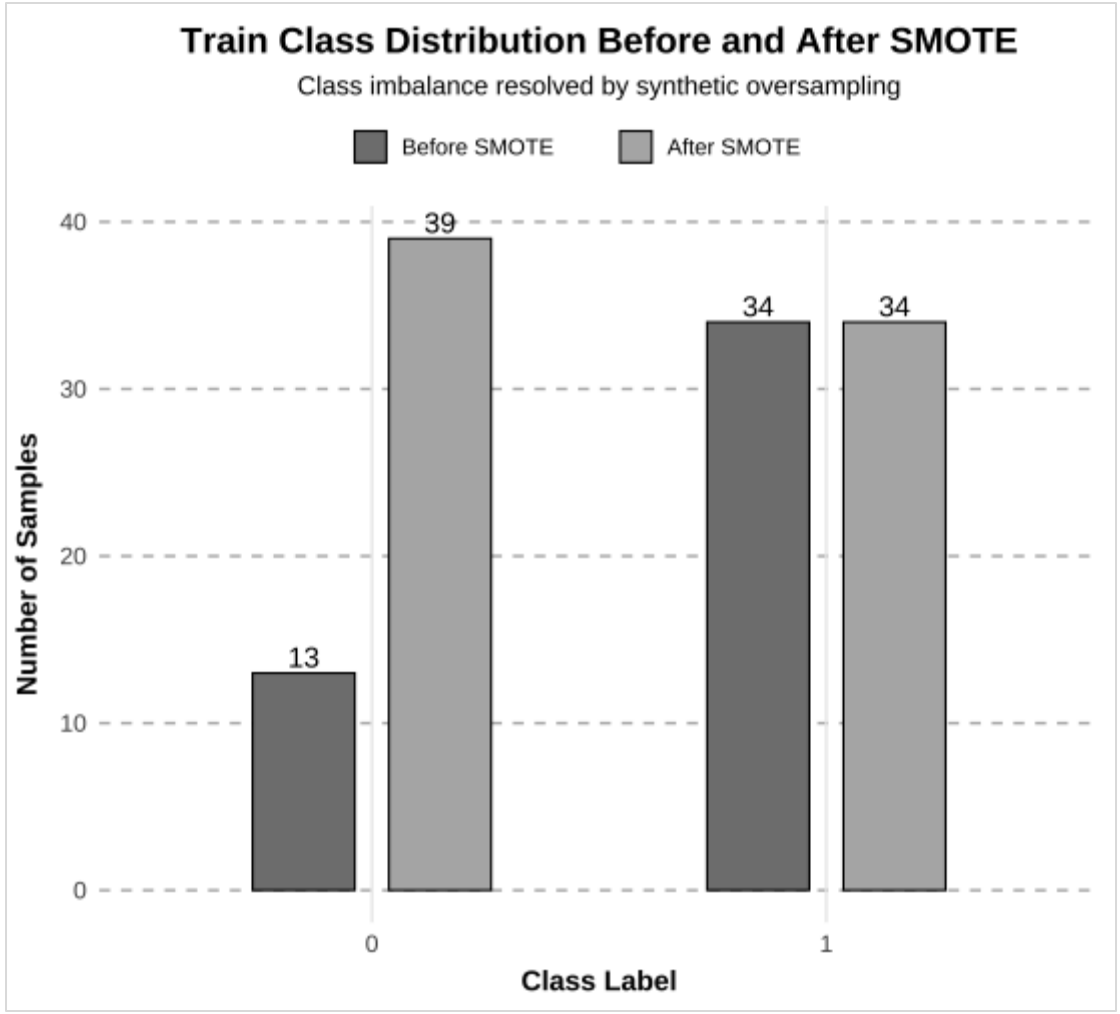
Solution Applied: SMOTE

We applied SMOTE (Synthetic Minority Oversampling Technique) to the training dataset.

Parameters Used in SMOTE

K = 5: Each synthetic sample was generated using 5 nearest neighbors.
dup_size = 2: Approximately 2x synthetic samples were created for each minority instance.
Applied Only to Training Set: Ensured test set remains untouched for fair model evaluation.

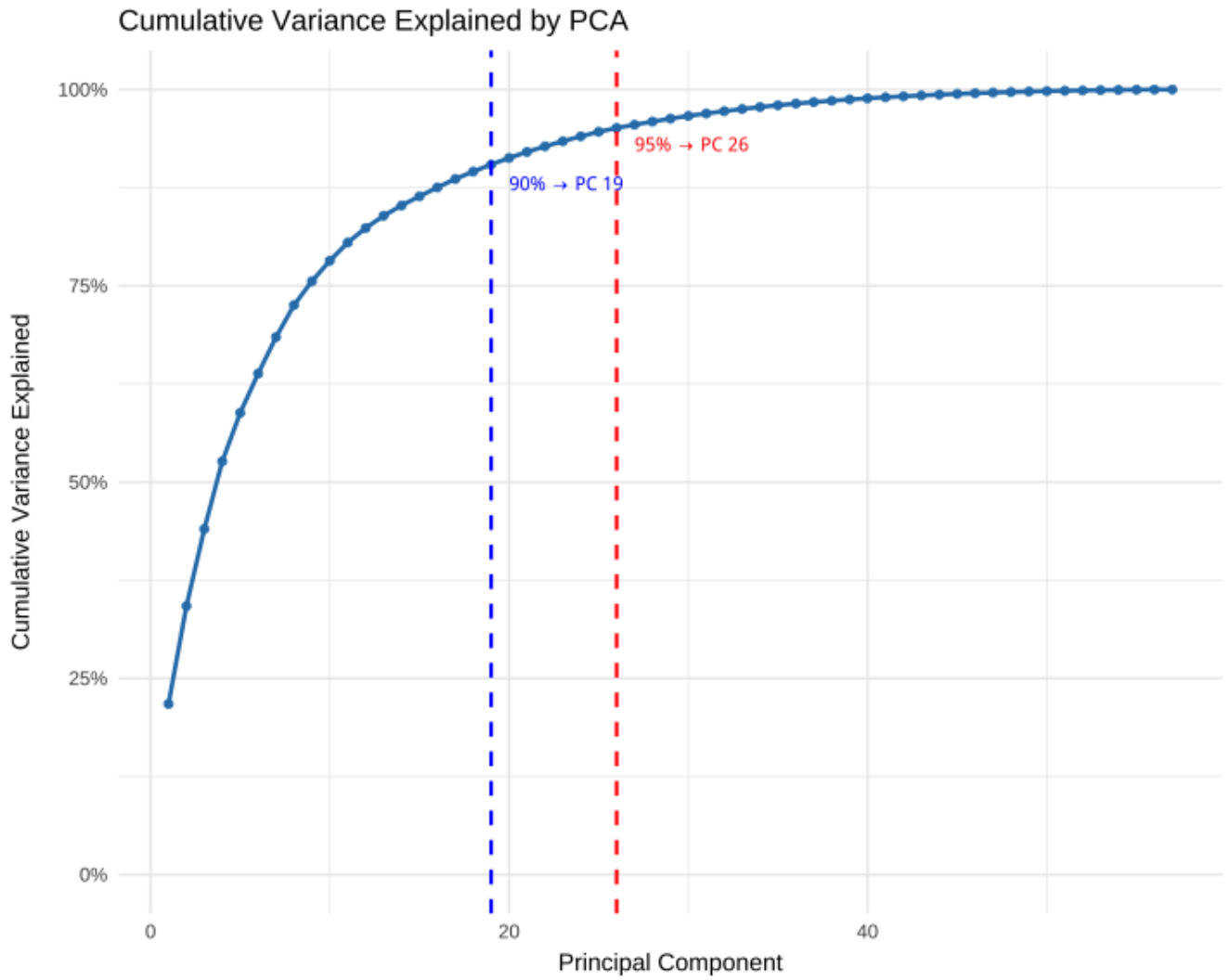
```
from imblearn.over_sampling import SMOTE

# === Separate features and label ===
X = training_data_full.iloc[:, :-1]  # All columns except the last one (assumed to be LABEL)
y = training_data_full.iloc[:, -1]   # The LABEL column

# === Apply SMOTE ===
smote = SMOTE(k_neighbors=5, sampling_strategy='auto')
X_smote, y_smote = smote.fit_resample(X, y)
```



Principal Component Analysis

Principal Component Analysis (PCA) was used to reduce dimensionality by transforming correlated features into uncorrelated components ranked by explained variance. Based on cumulative variance and feature contribution analysis, the top 26 impactful features were selected for modeling.

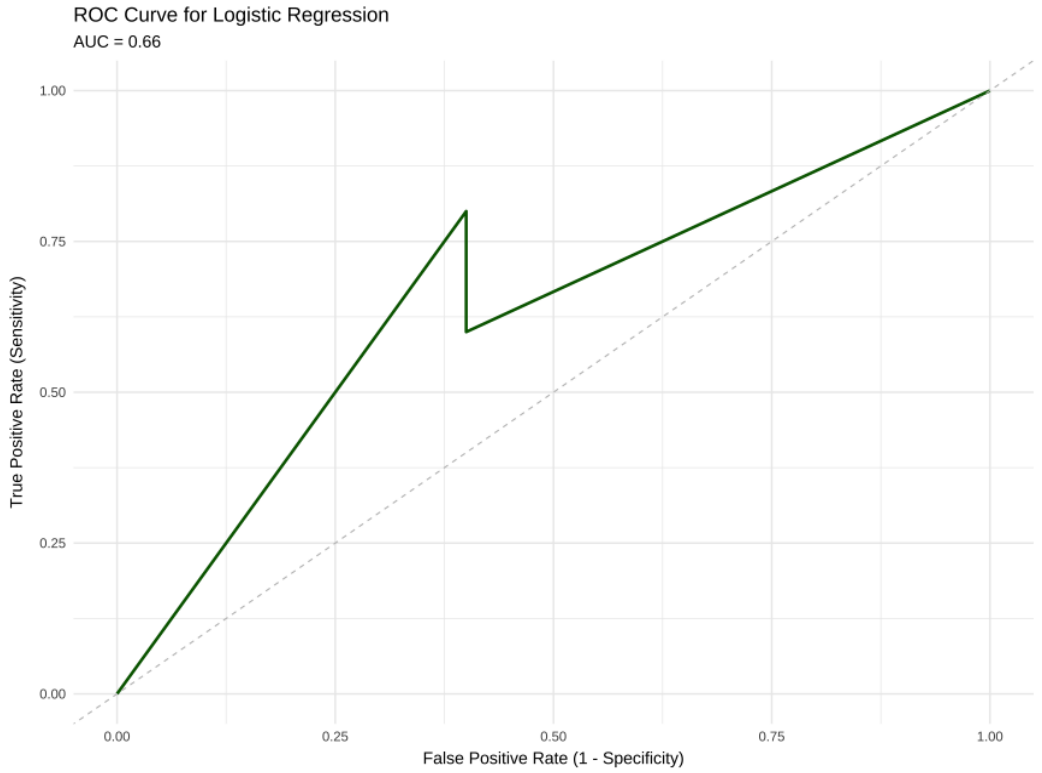
```
> head(top_features,26)
[1] "vp_original_firstorder_InterquartileRange_2"
[2] "vp_original_firstorder_RobustMeanAbsoluteDeviation_2"
[3] "tau_original_firstorder_Energy_1"
[4] "fp_original_shape_Sphericity_2"
[5] "ktrans_original_shape_Sphericity_2"
[6] "ve_original_shape_Sphericity_2"
[7] "vp_original_shape_Sphericity_2"
[8] "vp_original_firstorder_90Percentile_1"
[9] "vp_diagnostics_Image.original_Mean_1"
[10] "tau_original_shape_Sphericity_2"
[11] "vp_original_firstorder_90Percentile_2"
[12] "ktrans_original_firstorder_RobustMeanAbsoluteDeviation_1"
[13] "fp_original_firstorder_Variance_1"
[14] "ktrans_original_firstorder_90Percentile_1"
[15] "vp_original_gldm_GrayLevelNonUniformity_1"
[16] "vp_original_gldm_LargeDependenceLowGrayLevelEmphasis_1"
[17] "vp_original_glcm_SumAverage_2"
[18] "vp_original_glcm_JointAverage_2"
[19] "vp_original_glcm_Autocorrelation_2"
[20] "tau_original_gldm_GrayLevelNonUniformity_1"
[21] "tau_diagnostics_Mask.original_VoxelNum_1"
[22] "ktrans_diagnostics_Image.original_Mean_1"
[23] "ktrans_original_firstorder_InterquartileRange_2"
[24] "ktrans_original_firstorder_InterquartileRange_1"
[25] "vp_original_firstorder_InterquartileRange_1"
[26] "vp_original_firstorder_RootMeanSquared_2"
```



Logistic Regression

Logistic Regression Model Evaluation Summary:
The logistic regression model showed a moderate ability to distinguish between classes, achieving an **AUC of 0.66**. However, with an accuracy of 6**0%** and low sensitivity and specificity, the model's predictive power remains limited and may require further optimization or more informative features.

```
          Reference
Prediction 0 1
         0 3 2
         1 2 3

               Accuracy : 0.6
                 95% CI : (0.2624, 0.8784)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.377
```

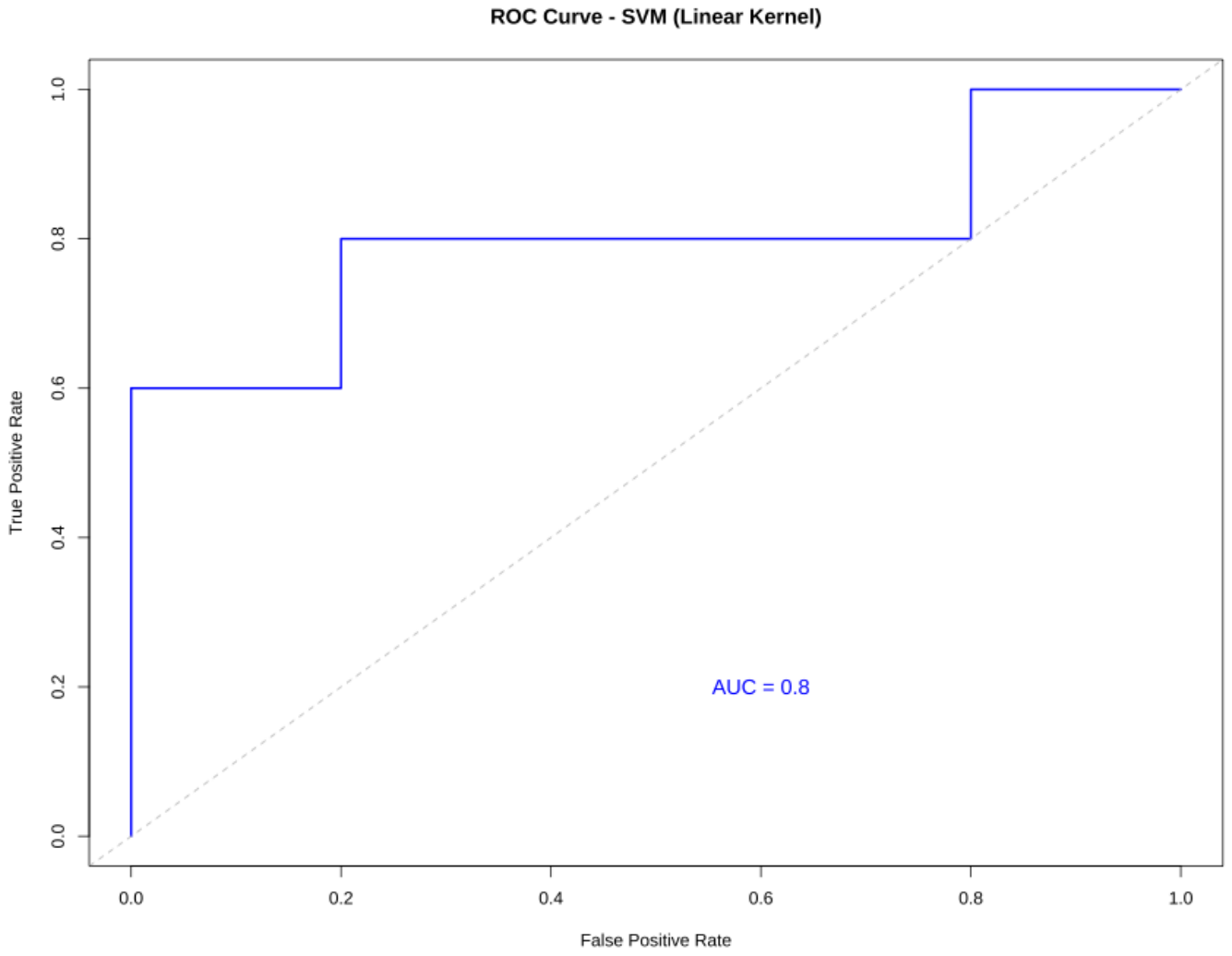


Support Vector Machine

SVM Summary:

- Accuracy: The model achieved an 80% accuracy, indicating it correctly classified 8 out of 10 instances.
- Sensitivity & Specificity: Both were 0.8, showing balanced performance in detecting both positive and negative classes.

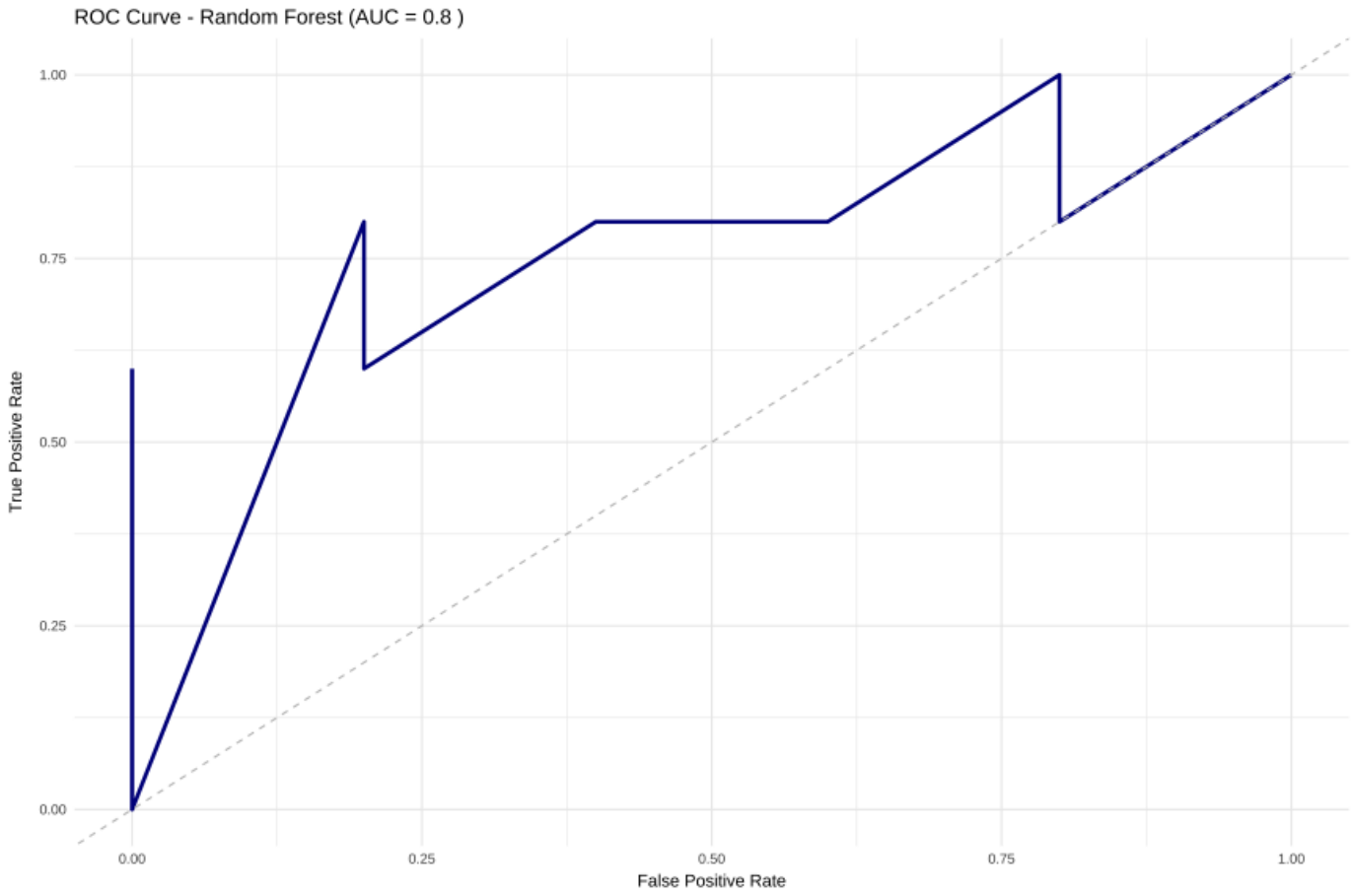
```
> print(conf_matrix)
Confusion Matrix and Statistics

          Reference
Prediction 0 1
         0 4 1
         1 1 4

               Accuracy : 0.8
                 95% CI : (0.4439, 0.9748)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.05469
```



Random Forest

The Random Forest model achieved an accuracy of 60% with a balanced accuracy of 60%, indicating moderate performance. It showed strong specificity (80%) but relatively low sensitivity (40%), suggesting it was better at identifying class 1 than class 0. The precision for class 0 was 66.7%, and the Kappa score of 0.2 indicates slight agreement beyond chance. Overall, the model performs reasonably but may benefit from further tuning to improve class 0 detection.

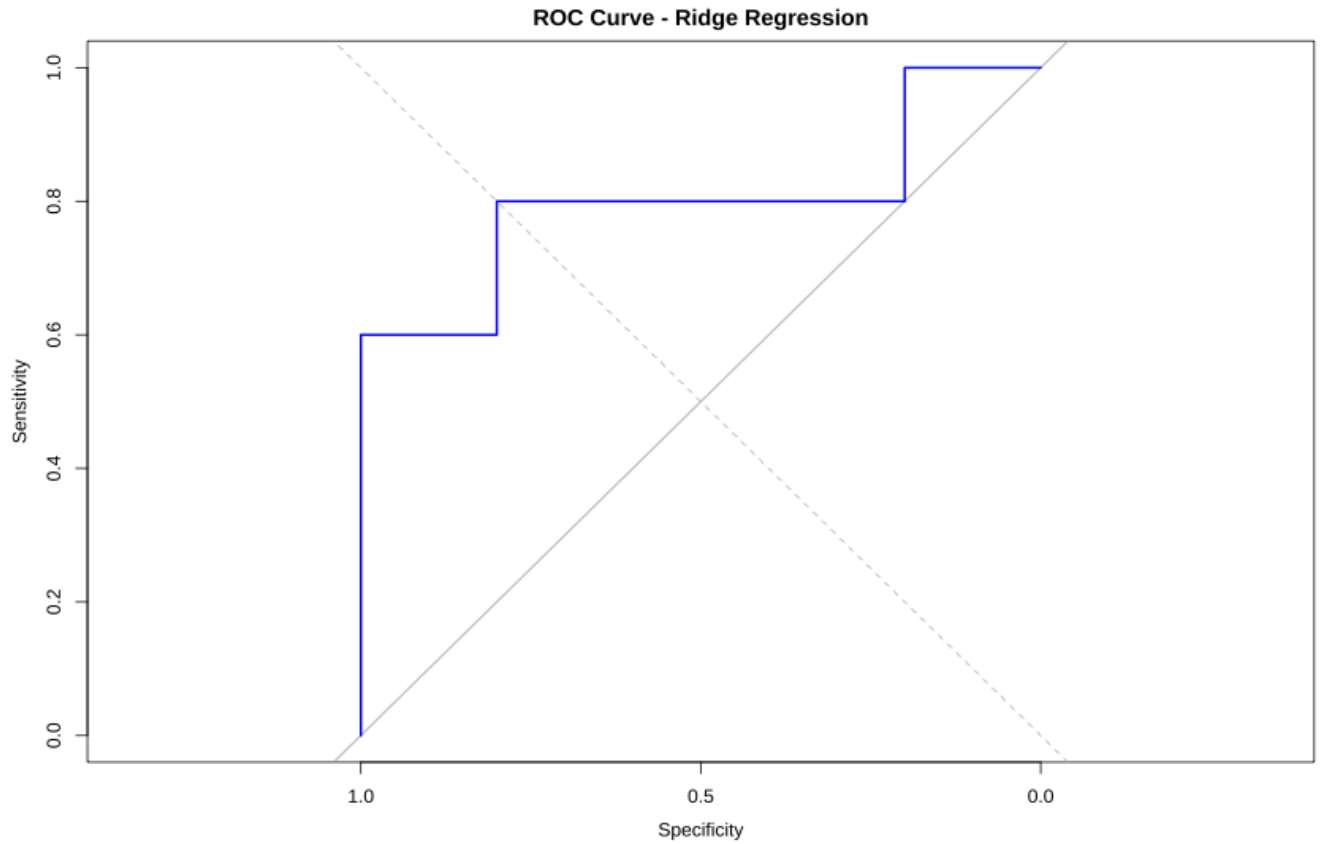
```
> print(conf_matrix)
Confusion Matrix and Statistics

          Reference
Prediction 0 1
         0 2 1
         1 3 4

               Accuracy : 0.6
                 95% CI : (0.2624, 0.8784)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.3770
```



Ridge Regression

Ridge Regression was used as a regularized logistic model to address multicollinearity and prevent overfitting. The optimal regularization parameter (lambda) was selected as **0.0395** using cross-validation. The model achieved a **70% accuracy** with a **balanced accuracy of 70%**, **sensitivity of 60%**, and **specificity of 80%**. The AUC score of **0.80** indicated good classification performance.

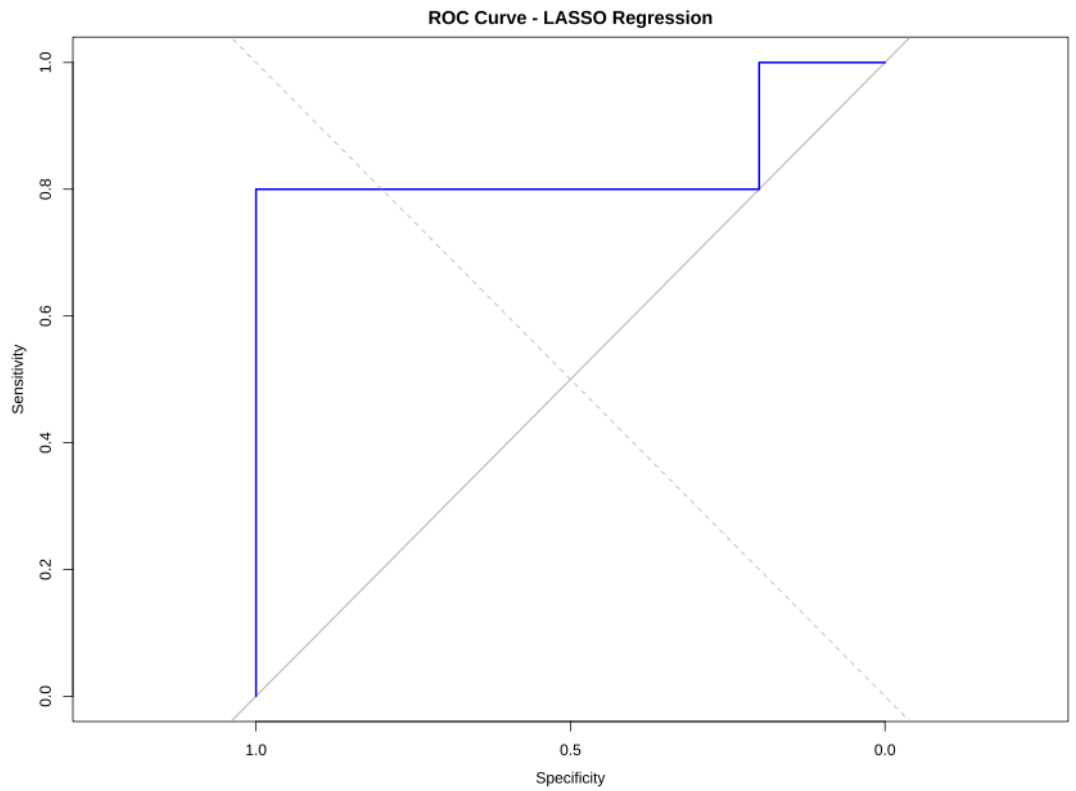
```
==== Ridge Regression ====
Best lambda: 0.03954
Confusion Matrix and Statistics

          Reference
Prediction 0 1
         0 3 1
         1 2 4

               Accuracy : 0.7
                 95% CI : (0.3475, 0.9333)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.1719
```



LASSO Regression

LASSO Regression was used to perform variable selection and regularization, effectively shrinking less important coefficients to zero. This helped improve model generalization and interpretability. The best lambda value was found to be 0.00872. The model achieved 80% accuracy, balanced accuracy of 80%, and both sensitivity and specificity were 0.80, indicating strong performance. The AUC of 0.84 further confirms excellent model discrimination capability.

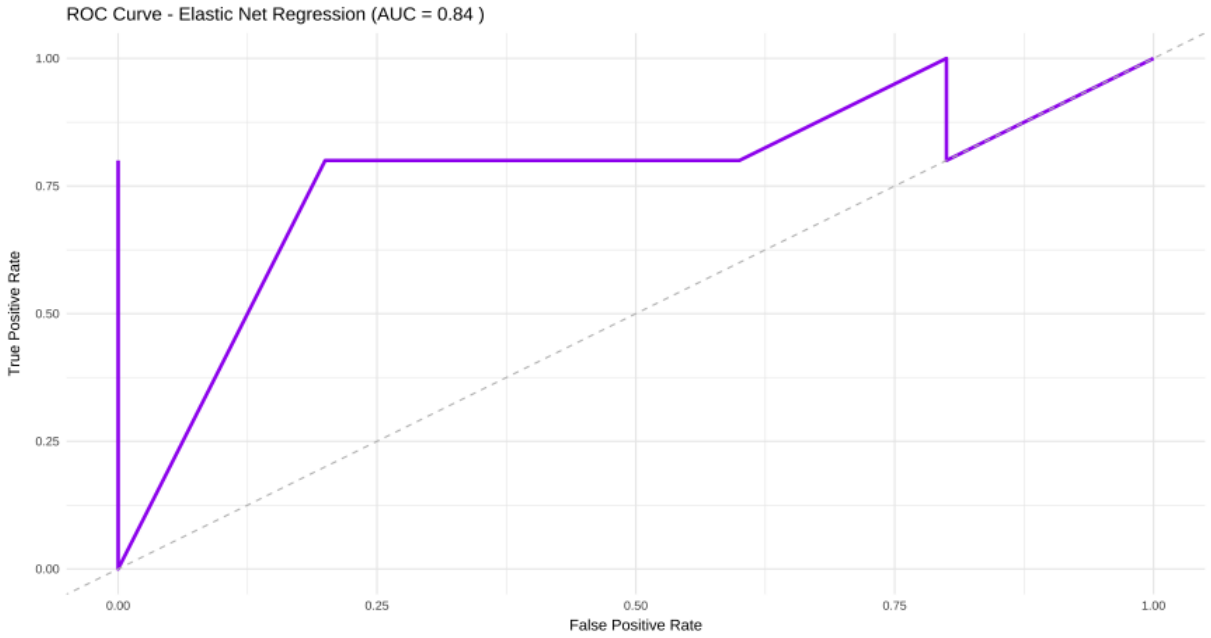
```
==== LASSO Regression ====
Best lambda: 0.00872
Confusion Matrix and Statistics

          Reference
Prediction 0 1
         0 4 1
         1 1 4

               Accuracy : 0.8
                 95% CI : (0.4439, 0.9748)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.05469
```



ELASTICNET Regression

Elastic Net Regression, which combines both L1 (LASSO) and L2 (Ridge) penalties, was used to balance feature selection and regularization. The optimal lambda was **0.01095**, yielding **80% accuracy** and **balanced accuracy** of **0.80**. Both sensitivity and specificity were also **0.80**, indicating consistent classification performance across both classes. The **AUC of 0.84** reflects excellent model discrimination capability, affirming the effectiveness of Elastic Net for this binary classification task.

```
==== Elastic Net Regression ====
Best lambda: 0.01095
Confusion Matrix and Statistics

          Reference
Prediction 0 1
         0 4 1
         1 1 4

               Accuracy : 0.8
                 95% CI : (0.4439, 0.9748)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.05469
```



fwd Selection

```
data[col].fillna(data[col].mean(), inplace=True)
```

Top Selected Features (Forward Selection):
1. fp_diagnostics_Image.original_Mean_1
2. ktrans_diagnostics_Image.original_Mean_1
3. tau_diagnostics_Image.original_Mean_1
4. ve_diagnostics_Image.original_Mean_1
5. vp_diagnostics_Image.original_Mean_1
6. fp_diagnostics_Image.original_Mean_2
7. ktrans_diagnostics_Image.original_Mean_2
8. ve_diagnostics_Image.original_Mean_2
9. vp_diagnostics_Image.original_Mean_2
10. fp_diagnostics_Image.original_Maximum_1
11. ktrans_diagnostics_Image.original_Maximum_1
12. tau_diagnostics_Image.original_Maximum_1
13. ve_diagnostics_Image.original_Maximum_1
14. vp_diagnostics_Image.original_Maximum_1
15. fp_diagnostics_Image.original_Maximum_2
16. ktrans_diagnostics_Image.original_Maximum_2
17. tau_diagnostics_Image.original_Maximum_2
18. ve_diagnostics_Image.original_Maximum_2
19. vp_diagnostics_Image.original_Maximum_2
20. fp_diagnostics_Mask.original_VoxelNum_2
21. tau_diagnostics_Mask.original_VoxelNum_2
22. fp_diagnostics_Mask.original_VolumeNum_1
23. ktrans_diagnostics_Mask.original_VolumeNum_1
24. tau_diagnostics_Mask.original_VolumeNum_1
25. ve_diagnostics_Mask.original_VolumeNum_1
26. fp_diagnostics_Mask.original_VolumeNum_2

Elastic NET on Complete Data

```
> run_penalized_logreg(alpha_val = 0.5, label = "Elastic Net Regression")

==== Elastic Net Regression ====
Best lambda: 0.19459
Confusion Matrix and Statistics

          Reference
Prediction 0 1
         0 0 0
         1 2 6

               Accuracy : 0.75
                 95% CI : (0.3491, 0.9681)
    No Information Rate : 0.75
    P-Value [Acc > NIR] : 0.6785

                  Kappa : 0

 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 0.00
            Specificity : 1.00
         Pos Pred Value :  NaN
         Neg Pred Value : 0.75
             Prevalence : 0.25
         Detection Rate : 0.00
   Detection Prevalence : 0.00
      Balanced Accuracy : 0.50

       'Positive' Class : 0
```

```
> head(important_vars, 26)$feature
 [1] "tau_original_firstorder_10Percentile_1"          "ve_original_firstorder_10Percentile_1"
 [3] "ve_diagnostics_Image.original_Maximum_2"         "ve_diagnostics_Image.original_Maximum_1"
 [5] "vp_original_firstorder_10Percentile_1"           "ktrans_original_shape_Flatness_1"
 [7] "fp_original_shape_Flatness_1"                    "tau_original_shape_Flatness_1"
 [9] "ve_original_shape_Flatness_1"                    "vp_original_shape_Flatness_1"
[11] "vp_original_shape_Sphericity_1"                  "ve_original_shape_Sphericity_1"
[13] "fp_original_shape_Sphericity_1"                  "tau_original_shape_Sphericity_1"
[15] "ktrans_original_shape_Sphericity_1"              "ktrans_original_firstorder_MeanAbsoluteDeviation_1"
[17] "ktrans_original_firstorder_RootMeanSquared_1"    "ktrans_original_firstorder_Variance_2"
[19] "ktrans_original_firstorder_Range_1"              "ktrans_original_firstorder_Maximum_1"
[21] "vp_original_shape_Sphericity_2"                  "ve_original_shape_Sphericity_2"
[23] "tau_original_shape_Sphericity_2"                 "ktrans_original_shape_Sphericity_2"
[25] "fp_original_shape_Sphericity_2"                  "vp_original_glrlm_LongRunLowGrayLevelEmphasis_1"
```