Echocardiogram Dataset

**Survival Analysis Problem**

# Objective –

The problem addressed was to predict from the other variables whether the patient will survive at least one year. The most difficult part of this problem is correctly predicting that the patient will NOT survive.

# Data Summary –

Number of rows: 132

|  |  |
| --- | --- |
| **Variable Name** | **Description** |
| Survival | The number of months patient survived (has survived, if patient is still alive) |
| Still-alive | A binary variable. 0=dead at end of survival period, 1 means still alive |
| Age | Age in years when heart attack occurred |
| Pericardial-effusion | Binary. Pericardial effusion is fluid around the heart. 0=no fluid, 1=fluid |
| Fractional-shortening | A measure of contractility around the heart lower numbers are increasingly abnormal |
| EPSS | E-point septal separation, another measure of contractility. Larger numbers are increasingly abnormal. |
| LVDD | Left ventricular end-diastolic dimension. This is a measure of the size of the heart at end-diastole. Large hearts tend to be sick hearts. |
| Wall-motion-score | A measure of how the segments of the left ventricle are moving |
| Wall-motion-index | Equals wall-motion-score divided by number of segments seen. Usually 12-13 segments are seen in an echocardiogram. To be used INSTEAD of the wall motion score. |

## Steps:

1. Missing Value treatment – replaced NAs with Mean value
2. Create dependent Variable, if Survival times > 12, then Target = 1, else 0
3. Split data into 70:30, train-test ratio
4. Fit a logistic Regression and analyze output
5. Apply SMOTE Technique for creating Balanced Dataset
6. Apply Decision tree & Random Forest
7. Compare all outputs

# **PART A: Creating Dependent Variable and Missing Value Treatment**

# Dependent Variable:

Patients with Survival (in months) < 12 and Still\_Alive = 1 were removed as they couldn’t be considered for Predictions.

For rest, patients with Survival > 12 Months, Target = 1, else 0

# Missing Value Treatment:

Replaced few the NAs with Mean and deleted rows with multiple columns of missing values. (16 rows were dropped)

After Treatment of missing values, Number of rows = 116

Ratio of Target Variable:

Alive,1 = 8/116 (7%)

Dead, 0 = 108/116 (93%)

Final independent variables used for the model:

1. Age
2. Pericardial Effusion
3. Fractional Shortening
4. EPSS
5. LVDD
6. Wallmotion-index

# **PART B: Logistic Regression**

# Code:

## **Logistic Regression:**

Split data into 70:30 for train & test.

> sample = sample.split(data\_1$Target,SplitRatio = 0.70)

> train.data = data\_1[sample == TRUE,]

> test.data = data\_1[sample == FALSE,]

> #Logistic Regression - 1

> model1 = glm(train.data$Target~.,family="binomial",train.data)

> summary(model1)

Call:

glm(formula = train.data$Target ~ ., family = "binomial", data = train.data)

Deviance Residuals:

Min 1Q Median 3Q Max

-1.2469 -0.3164 -0.2049 -0.1054 2.7078

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -8.28419 5.50926 -1.504 0.133

age 0.11056 0.06851 1.614 0.107

pericardialeffusion 0.94616 1.18488 0.799 0.425

fractionalshortening -8.88681 7.40297 -1.200 0.230

epss 0.17044 0.11740 1.452 0.147

lvdd -0.19548 0.82764 -0.236 0.813

wallmotion.index -0.94333 1.74255 -0.541 0.588

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 40.020 on 64 degrees of freedom

Residual deviance: 28.013 on 58 degrees of freedom

AIC: 42.013

Number of Fisher Scoring iterations: 7

> vif(model1)

|  |  |
| --- | --- |
| Variable | VIF Value |
| age  pericardialeffusion  fractionalshortening  epss  lvdd  wallmotion.index | 1.376 1.055 1.2719 2.6392 2.063 1.419 |

Optimal Cut-off –

> optCutOff

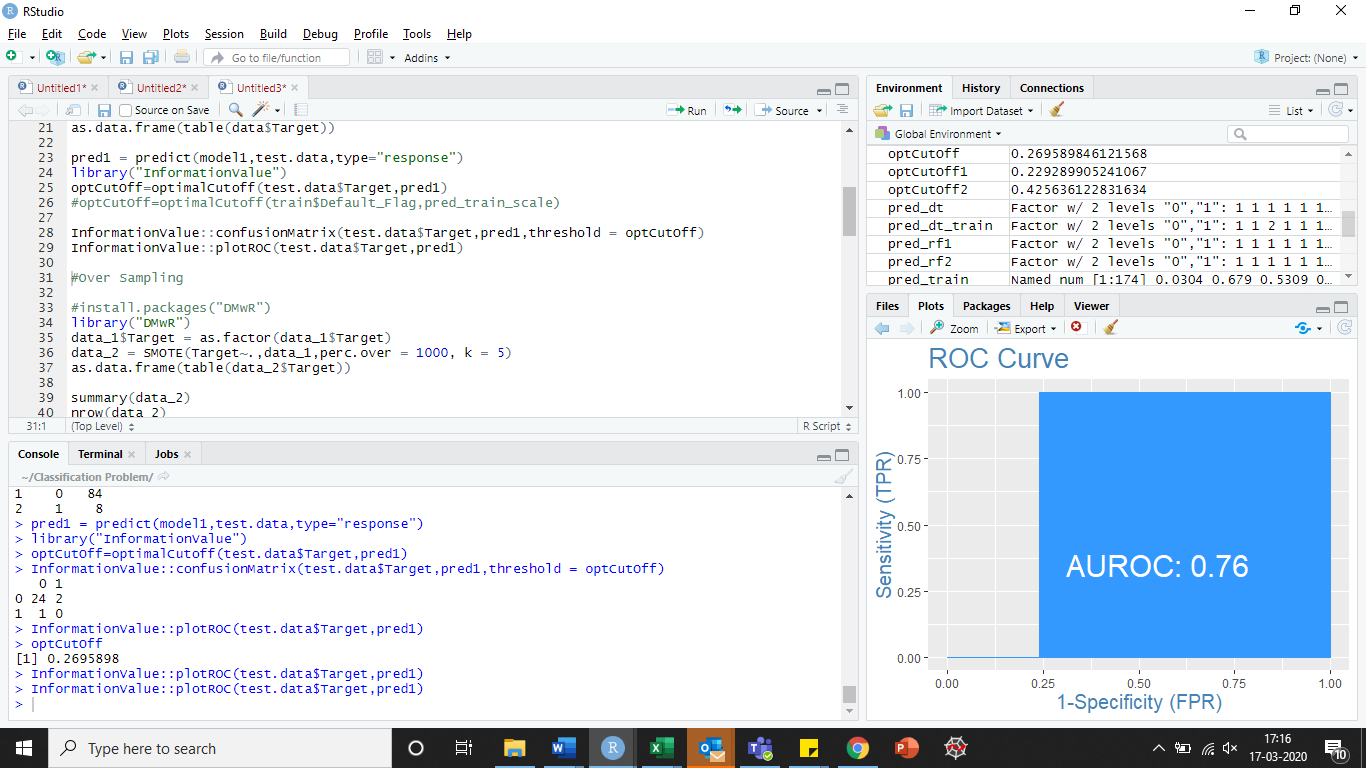
[1] 0.2695898

Confusion Matrix (Test Data) –

> confusionMatrix(test.data$Target,pred1,threshold = optCutOff)

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 24 | 2 |
| 1 | 1 | 0 |

ROC –



Since no variable came out to be significant, applying **SMOTE Technique** to get balanced data set.

# **PART C: SMOTE Technique**

**SMOTE** stands for Synthetic Minority Oversampling Technique. This is a statistical technique for increasing the number of cases in your dataset in a balanced way. The module works by generating new instances from existing minority cases that you supply as input. This implementation of SMOTE does **not** change the number of majority cases.

The new instances are not just copies of existing minority cases; instead, the algorithm takes samples of the feature space for each target class and its nearest neighbors, and generates new examples that combine features of the target case with features of its neighbors. This approach increases the features available to each class and makes the samples more general.

Before SMOTE –

> as.data.frame(table(data\_1$Target))

Var1 Freq

1 0 84

2 1 8

After SMOTE –

> data\_2 = SMOTE(Target~.,data\_1,perc.over = 1000, k = 5)

> as.data.frame(table(data\_2$Target))

Var1 Freq

1 0 160

2 1 24

# **PART D: Classification on Balanced Dataset**

## **Logistic Regression on balanced dataset:**

> model2 = glm(train.data1$Target ~ .,train.data1, family="binomial")

> summary(model2)

Call:

glm(formula = train.data1$Target ~ ., family = "binomial", data = train.data1)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.36159 -0.33824 -0.14593 -0.06147 2.67305

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -15.83285 4.80969 -3.292 0.000995 \*\*\*

age 0.13146 0.04954 2.654 0.007957 \*\*

pericardialeffusion 0.81504 0.85304 0.955 0.339343

fractionalshortening -18.67648 5.94816 -3.140 0.001690 \*\*

epss 0.02437 0.06060 0.402 0.687508

lvdd 1.18929 0.64702 1.838 0.066048 .

wallmotion.index 2.04572 1.16973 1.749 0.080311 .

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 100.559 on 128 degrees of freedom

Residual deviance: 55.699 on 122 degrees of freedom

AIC: 69.699

Number of Fisher Scoring iterations: 7

* At 99% Confidence Interval, only age and fractional shortening are significant.
* At 90% Confidence Interval, lvdd and wallmotion index are also significant.

Equation for Logistic Regression:

**TRAIN DATA PERFORMANCE METRICS (Logistic Regression):**

Optimal Cut-off:

> optCutOff2

[1] 0.4084902

Confusion Matrix:

> InformationValue::confusionMatrix(train.data1$Target,pred\_train,threshold = optCutOff2)

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 109 | 4 |
| 1 | 3 | 13 |

Accuracy:

> Accuracy(pred\_logit,train.data1$Target)

[1] 0.9457364

Sensitivity:

> Sensitivity(pred\_logit,train.data1$Target)

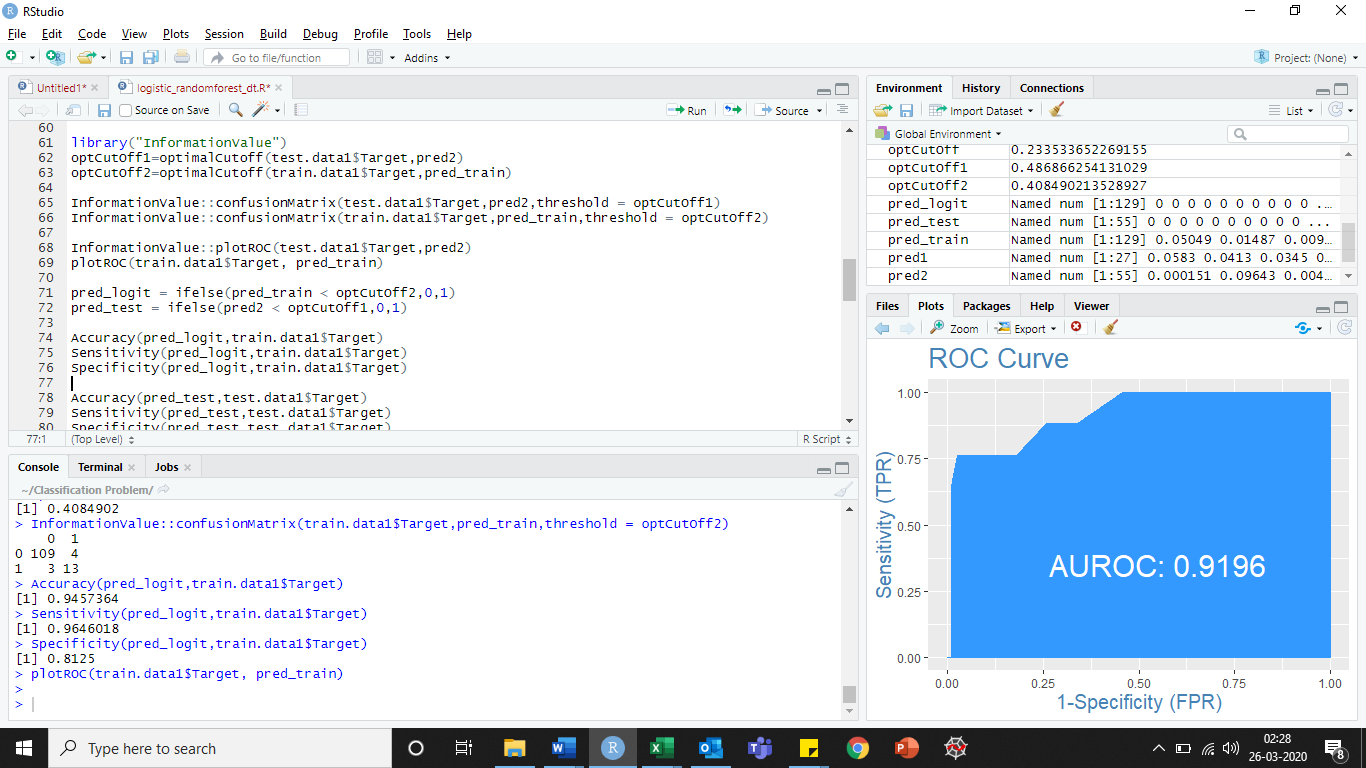
[1] 0.9646018

Specificity:

> Specificity(pred\_logit,train.data1$Target)

[1] 0.8125

ROC:



**TEST DATA PERFORMANCE METRICS (Logistic Regression):**

Optimal Cutoff:

> optCutOff1

[1] 0.4868663

Confusion Matrix:

> InformationValue::confusionMatrix(test.data1$Target,pred2,threshold = optCutOff1)

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 47 | 5 |
| 1 | 1 | 2 |

Accuracy:

> Accuracy(pred\_test,test.data1$Target)

[1] 0.8909091

Sensitivity:

> Sensitivity(pred\_test,test.data1$Target)

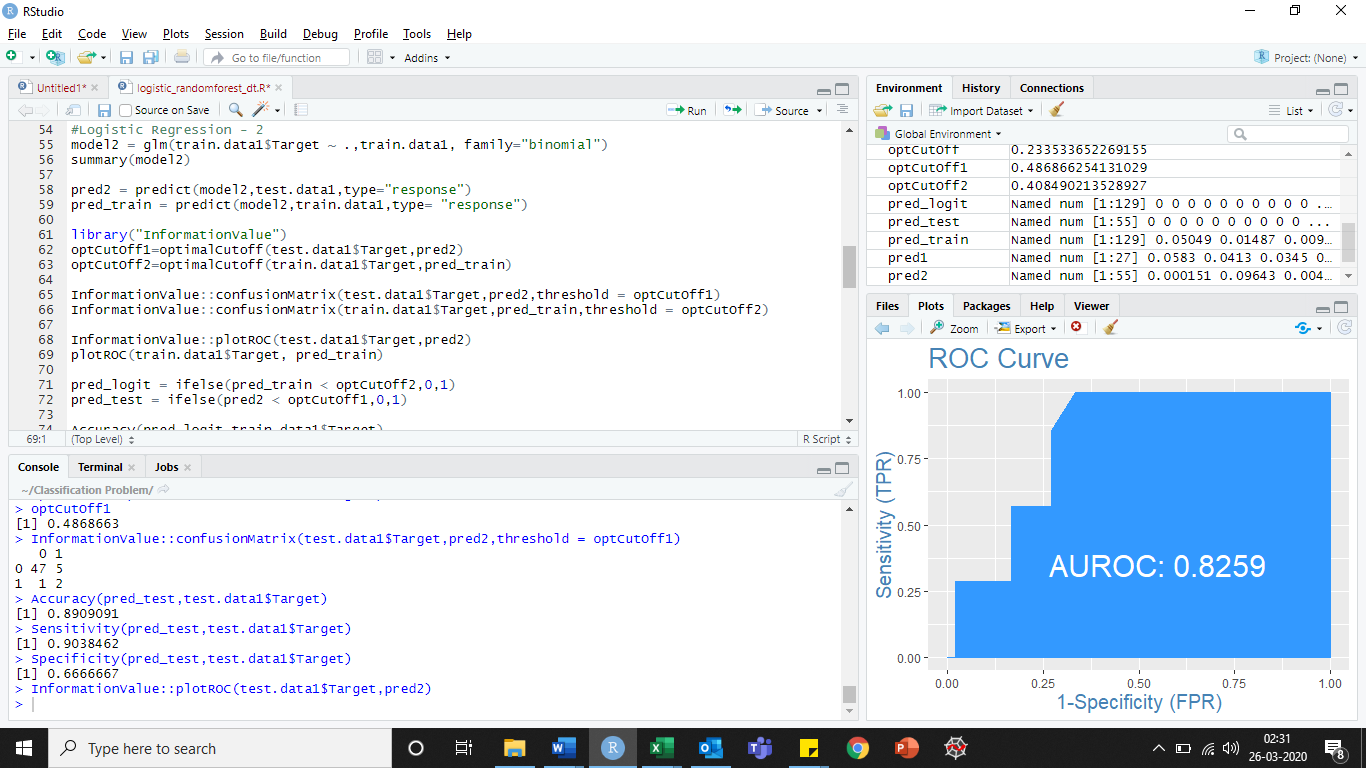
[1] 0.9038462

Specificity:

> Specificity(pred\_test,test.data1$Target)

[1] 0.6666667

ROC:



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | | **Sensitivity** | | **Specificity** | |
| **TRAIN** | **TEST** | **TRAIN** | **TEST** | **TRAIN** | **TEST** |
| Logistic Regression (After SMOTE) | 0.945 | 0.890 | 0.964 | 0.903 | 0.8125 | 0.667 |

## **Decision Tree (On Balanced Dataset):**

> model3 = rpart(train.data1$Target ~., data = train.data1, method = 'class')

> model3

n= 129

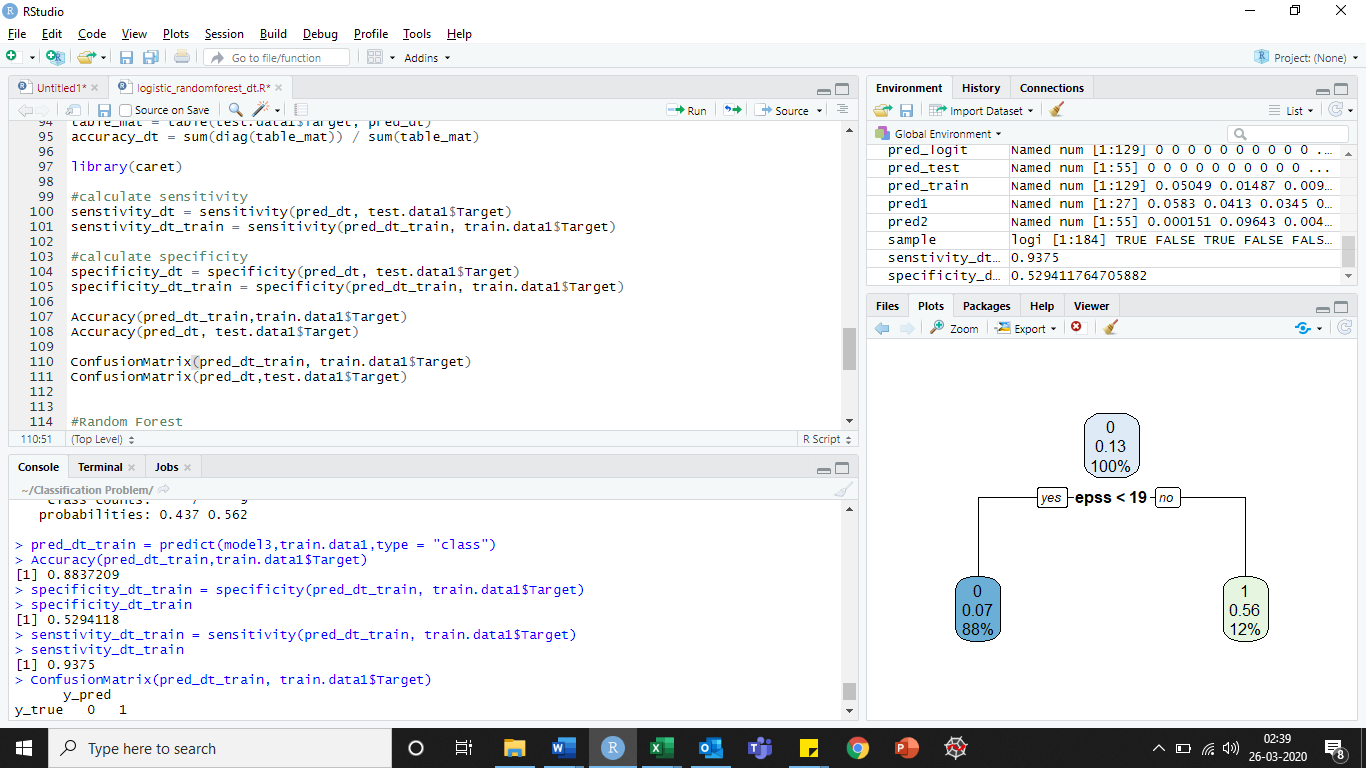
node), split, n, loss, yval, (yprob)

\* denotes terminal node

1) root 129 17 0 (0.86821705 0.13178295)

2) epss< 18.6581 113 8 0 (0.92920354 0.07079646) \*

3) epss>=18.6581 16 7 1 (0.43750000 0.56250000) \*



**TRAIN DATA PERFORMANCE METRICS (Decision Trees):**

Accuracy:

> Accuracy(pred\_dt\_train,train.data1$Target)

[1] 0.8837209

Sensitivity:

> senstivity\_dt\_train

[1] 0.9375

Specificity:

> specificity\_dt\_train

[1] 0.5294118

Confusion Matrix:

> ConfusionMatrix(pred\_dt\_train, train.data1$Target)

|  |  |  |
| --- | --- | --- |
| Y\_pred | 0 | 1 |
| Y\_true |  |  |
| 0 | 105 | 7 |
| 1 | 8 | 9 |

**TEST DATA PERFORMANCE METRICS (Decision Trees):**

Accuracy:

> Accuracy(pred\_dt, test.data1$Target)

[1] 0.8545455

Sensitivity:

> senstivity\_dt

[1] 0.9583333

Specificity:

> specificity\_dt

[1] 0.1428571

Confusion Matrix:

> ConfusionMatrix(pred\_dt,test.data1$Target)

|  |  |  |
| --- | --- | --- |
| Y\_pred | 0 | 1 |
| Y\_true |  |  |
| 0 | 46 | 2 |
| 1 | 6 | 1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | | **Sensitivity** | | **Specificity** | |
| **TRAIN** | **TEST** | **TRAIN** | **TEST** | **TRAIN** | **TEST** |
| Decision Tree | 0.883 | 0.854 | 0.937 | 0.958 | 0.529 | 0.142 |

## **Random Forest (On Balanced Dataset):**

> model4 = randomForest(train.data1$Target~., data = train.data1)

> model4

Call:

randomForest(formula = train.data1$Target ~ ., data = train.data1)

Type of random forest: classification

Number of trees: 500

No. of variables tried at each split: 2

OOB estimate of error rate: 6.98%

Confusion matrix:

0 1 class.error

0 110 2 0.01785714

1 7 10 0.41176471

**TRAIN DATA PERFORMANCE METRICS (Random Forest):**

Accuracy:

> Accuracy(pred\_rf2,train.data1$Target)

[1] 1

Sensitivity:

> Sensitivity(pred\_rf2,train.data1$Target)

[1] 1

Specificity:

> Specificity(pred\_rf2,train.data1$Target)

[1] 1

Confusion Matrix:

> ConfusionMatrix(pred\_rf2,train.data1$Target)

|  |  |  |
| --- | --- | --- |
| Y\_pred | 0 | 1 |
| Y\_true |  |  |
| 0 | 112 | 0 |
| 1 | 0 | 17 |

**TEST DATA PERFORMANCE METRICS (Random Forest):**

Accuracy:

> Accuracy(pred\_rf1,test.data1$Target)

[1] 0.8727273

Sensitivity:

> Sensitivity(pred\_rf1,test.data1$Target)

[1] 0.8867925

Specificity:

> Specificity(pred\_rf1,test.data1$Target)

[1] 0.5

Confusion Matrix:

> ConfusionMatrix(pred\_rf1,test.data1$Target)

|  |  |  |
| --- | --- | --- |
| Y\_pred | 0 | 1 |
| Y\_true |  |  |
| 0 | 47 | 1 |
| 1 | 6 | 1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | | **Sensitivity** | | **Specificity** | |
| **TRAIN** | **TEST** | **TRAIN** | **TEST** | **TRAIN** | **TEST** |
| Random Forest | 1 | 0.872 | 1 | 0.886 | 1 | 0.5 |

# Conclusion:

* After using SMOTE technique, Logistic Regression performs well
* Random Forest overfits the data as Accuracy, Sensitivity & Specificity are 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | | **Sensitivity** | | **Specificity** | |
| **TRAIN** | **TEST** | **TRAIN** | **TEST** | **TRAIN** | **TEST** |
| Logistic Regression (After SMOTE) | 0.945 | 0.890 | 0.964 | 0.903 | 0.8125 | 0.667 |
| Decision Tree | 0.883 | 0.854 | 0.937 | 0.958 | 0.529 | 0.142 |
| Random Forest | 1 | 0.872 | 1 | 0.886 | 1 | 0.5 |

**Logistic Regression** is the classification technique chosen for the following problem.