3. CLASSIFICATION _ Thi Tinh Lo (22236226)

2023-08-19

CLASSIFICATION:

1. LOGISTIC REGRESSION

2. DECISION TREE

3. RANDOM FOREST

4. NEUTRAL NETWORK

Read data

```
setwd('C:/Users/tinhl/OneDrive/Documents')
data <- read.csv(file ='data_processed.csv')
```

```
summary(data)
```

```
##      State          Response          Coverage       Education    
##  Min.   :1.000   Min.   :0.0000   Min.   :1.000   Min.   :1.000  
##  1st Qu.:2.000   1st Qu.:0.0000   1st Qu.:2.000   1st Qu.:3.000  
##  Median :2.000   Median :0.0000   Median :3.000   Median :4.000  
##  Mean   :2.742   Mean   :0.1432   Mean   :2.519   Mean   :3.712  
##  3rd Qu.:4.000   3rd Qu.:0.0000   3rd Qu.:3.000   3rd Qu.:5.000  
##  Max.   :5.000   Max.   :1.0000   Max.   :3.000   Max.   :5.000  
##  EmploymentStatus      Gender     Location_Code   Marital_Status  Policy_Type   
##  Min.   :1.000    Min.   :1.00   Min.   :1.000   Min.   :1.00   Min.   :1.000  
##  1st Qu.:2.000    1st Qu.:1.00   1st Qu.:2.000   1st Qu.:2.00   1st Qu.:2.000  
##  Median :2.000    Median :1.00   Median :2.000   Median :2.00   Median :3.000  
##  Mean   :2.826    Mean   :1.49   Mean   :2.021   Mean   :2.12   Mean   :2.702  
##  3rd Qu.:5.000    3rd Qu.:2.00   3rd Qu.:2.000   3rd Qu.:3.00   3rd Qu.:3.000  
##  Max.   :5.000    Max.   :2.00   Max.   :3.000   Max.   :3.00   Max.   :3.000  
##      Policy      Renew_Offer_Type Sales_Channel    Vehicle_Class  
##  Min.   :1.000   Min.   :1.00     Min.   :1.000   Min.   :1.000  
##  1st Qu.:6.000   1st Qu.:1.00     1st Qu.:1.000   1st Qu.:1.000  
##  Median :8.000   Median :2.00     Median :2.000   Median :1.000  
##  Mean   :7.425   Mean   :1.97     Mean   :2.103   Mean   :3.036  
##  3rd Qu.:9.000   3rd Qu.:3.00     3rd Qu.:3.000   3rd Qu.:5.000  
##  Max.   :9.000   Max.   :4.00     Max.   :4.000   Max.   :6.000  
##   Vehicle_Size  Customer_Lifetime_Value     Income        Monthly_Premium_Auto
##  Min.   :1.00   Min.   :-0.8888         Min.   :-1.2395   Min.   :-0.9364     
##  1st Qu.:2.00   1st Qu.:-0.5837         1st Qu.:-1.2395   1st Qu.:-0.7329     
```

```
##  Median :2.00   Median :-0.3238         Median :-0.1240   Median :-0.2970
##  Mean   :1.91   Mean   : 0.0000         Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.:2.00   3rd Qu.: 0.1393         3rd Qu.: 0.8118   3rd Qu.: 0.4586
##  Max.   :3.00   Max.   :10.9621         Max.   : 2.0515   Max.   : 5.9515
##  Months_Since_Last_Claim Months_Since_Policy_Inception
##  Min.   :-1.4987         Min.   :-1.722376
##  1st Qu.:-0.9031         1st Qu.:-0.862345
##  Median :-0.1089         Median :-0.002315
##  Mean   : 0.0000         Mean   : 0.000000
##  3rd Qu.: 0.7846         3rd Qu.: 0.821881
##  Max.   : 1.9758         Max.   : 1.825250
##  Number_of_Open_Complaints Number_of_Policies Total_Claim_Amount
##  Min.   :-0.4222           Min.   :-0.8226    Min.   :-1.4939
##  1st Qu.:-0.4222           1st Qu.:-0.8226    1st Qu.:-0.5571
##  Median :-0.4222           Median :-0.4042    Median :-0.1726
##  Mean   : 0.0000           Mean   : 0.0000    Mean   : 0.0000
##  3rd Qu.:-0.4222           3rd Qu.: 0.4325    3rd Qu.: 0.3905
##  Max.   : 5.0700           Max.   : 2.5244    Max.   : 8.4652
```

Create the training and test data

```r
set.seed(42)
n= nrow(data)
trainIndex = sample(1:n, size= round(0.7*n), replace=FALSE)
train = data[trainIndex,]
test = data[-trainIndex,]
```

Rows of training data and test data

```r
nrow(train)
```

```
## [1] 6394
```

```r
nrow(test)
```

```
## [1] 2740
```

1. LOGISTIC REGRESSION

Build model

Estimates a logistic regression model using the glm

```r
set.seed(42)
model <- glm(Response ~ ., data = train, family = "binomial")
summary(model)
```

```
##
## Call:
## glm(formula = Response ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2581  -0.6210  -0.4769  -0.3207   2.4466
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    0.536327   0.391540   1.370  0.17075
## State                         -0.014333   0.028537  -0.502  0.61549
## Coverage                       0.035270   0.063974   0.551  0.58141
## Education                     -0.095212   0.033858  -2.812  0.00492 **
## EmploymentStatus              -0.016092   0.040050  -0.402  0.68784
## Gender                         0.003229   0.073719   0.044  0.96506
## Location_Code                  0.041637   0.064683   0.644  0.51976
## Marital_Status                -0.452459   0.058533  -7.730 1.08e-14 ***
## Policy_Type                   -0.124376   0.157612  -0.789  0.43004
## Policy                         0.006627   0.047463   0.140  0.88895
## Renew_Offer_Type              -0.575076   0.045859 -12.540  < 2e-16 ***
## Sales_Channel                 -0.195313   0.036487  -5.353 8.65e-08 ***
## Vehicle_Class                  0.013449   0.017328   0.776  0.43766
## Vehicle_Size                   0.279868   0.067903   4.122 3.76e-05 ***
## Customer_Lifetime_Value       -0.068328   0.041290  -1.655  0.09796 .
## Income                         0.116835   0.054228   2.155  0.03120 *
## Monthly_Premium_Auto          -0.152651   0.060331  -2.530  0.01140 *
## Months_Since_Last_Claim       -0.068249   0.037157  -1.837  0.06624 .
## Months_Since_Policy_Inception -0.015504   0.036538  -0.424  0.67133
## Number_of_Open_Complaints     -0.055188   0.037563  -1.469  0.14177
## Number_of_Policies            -0.079527   0.037483  -2.122  0.03386 *
## Total_Claim_Amount             0.255875   0.057674   4.437 9.14e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5310.7  on 6393  degrees of freedom
## Residual deviance: 4963.2  on 6372  degrees of freedom
## AIC: 5007.2
##
## Number of Fisher Scoring iterations: 5
```

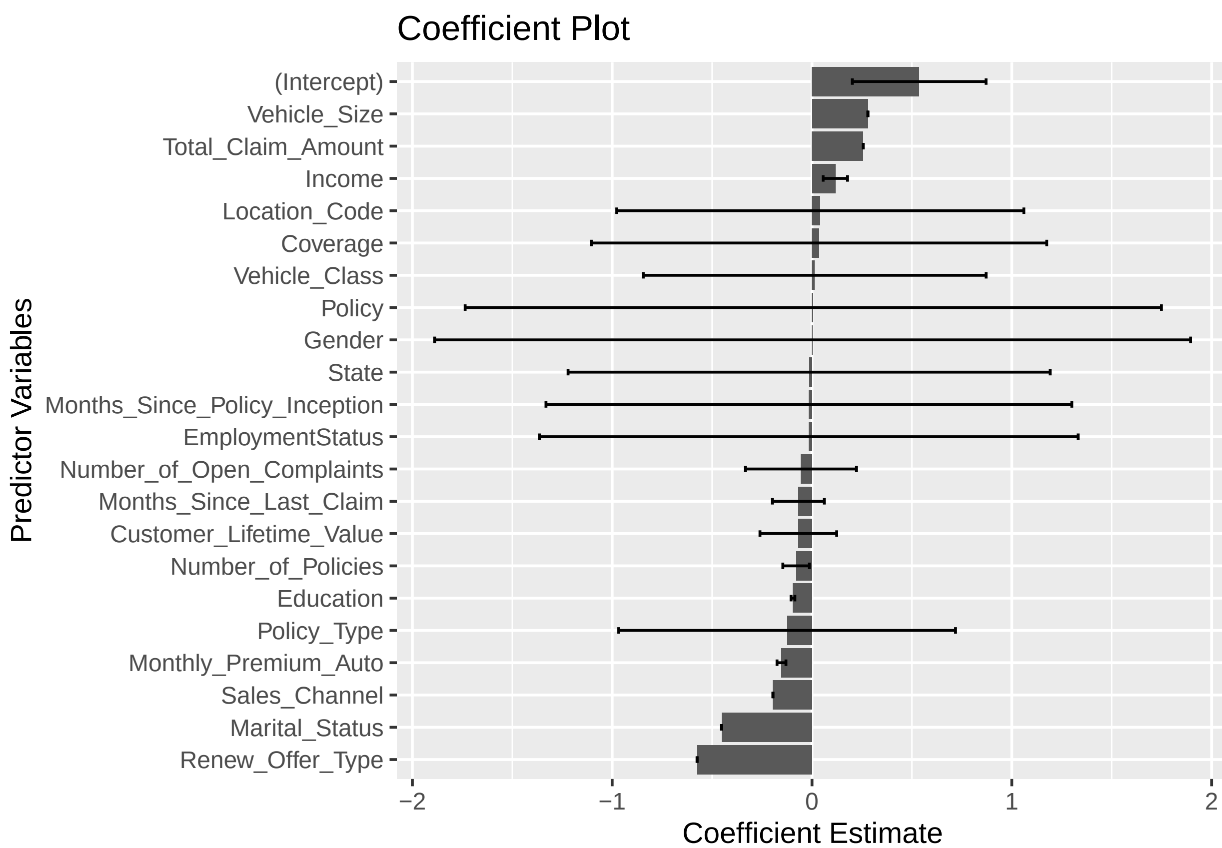
Coefficient

```
coef_df <- as.data.frame(summary(model)$coefficients)
coef_df$Variable <- rownames(coef_df)
coef_df
```

```
##                                   Estimate Std. Error      z value     Pr(>|z|)
## (Intercept)                    0.536326749 0.39154018   1.36978725 1.707533e-01
## State                         -0.014332795 0.02853706  -0.50225189 6.154903e-01
## Coverage                       0.035270029 0.06397387   0.55131926 5.814148e-01
```

```
## Education                    -0.095211822 0.03385833  -2.81206458 4.922462e-03
## EmploymentStatus             -0.016091667 0.04004989  -0.40179057 6.878382e-01
## Gender                        0.003229210 0.07371925   0.04380415 9.650605e-01
## Location_Code                 0.041637265 0.06468329   0.64370979 5.197636e-01
## Marital_Status               -0.452459425 0.05853304  -7.72998321 1.075608e-14
## Policy_Type                  -0.124376480 0.15761237  -0.78912894 4.300367e-01
## Policy                        0.006627466 0.04746276   0.13963508 8.889483e-01
## Renew_Offer_Type             -0.575076293 0.04585896 -12.54010630 4.503910e-36
## Sales_Channel                -0.195313445 0.03648695  -5.35296773 8.652328e-08
## Vehicle_Class                 0.013449376 0.01732817   0.77615685 4.376564e-01
## Vehicle_Size                  0.279868113 0.06790348   4.12155763 3.763193e-05
## Customer_Lifetime_Value      -0.068328444 0.04129007  -1.65483971 9.795702e-02
## Income                        0.116835378 0.05422773   2.15453194 3.119848e-02
## Monthly_Premium_Auto         -0.152650512 0.06033083  -2.53022408 1.139897e-02
## Months_Since_Last_Claim      -0.068248866 0.03715682  -1.83677907 6.624252e-02
## Months_Since_Policy_Inception -0.015504278 0.03653847  -0.42432745 6.713270e-01
## Number_of_Open_Complaints    -0.055187914 0.03756252  -1.46922815 1.417709e-01
## Number_of_Policies           -0.079527422 0.03748276  -2.12170687 3.386236e-02
## Total_Claim_Amount            0.255874573 0.05767430   4.43654408 9.141462e-06
##                                                    Variable
## (Intercept)                                     (Intercept)
## State                                                 State
## Coverage                                           Coverage
## Education                                         Education
## EmploymentStatus                           EmploymentStatus
## Gender                                               Gender
## Location_Code                                 Location_Code
## Marital_Status                               Marital_Status
## Policy_Type                                     Policy_Type
## Policy                                               Policy
## Renew_Offer_Type                           Renew_Offer_Type
## Sales_Channel                                 Sales_Channel
## Vehicle_Class                                 Vehicle_Class
## Vehicle_Size                                   Vehicle_Size
## Customer_Lifetime_Value             Customer_Lifetime_Value
## Income                                               Income
## Monthly_Premium_Auto                   Monthly_Premium_Auto
## Months_Since_Last_Claim             Months_Since_Last_Claim
## Months_Since_Policy_Inception Months_Since_Policy_Inception
## Number_of_Open_Complaints         Number_of_Open_Complaints
## Number_of_Policies                       Number_of_Policies
## Total_Claim_Amount                       Total_Claim_Amount
```

```
ggplot(coef_df, aes(x = reorder(Variable, Estimate), y = Estimate)) +
  geom_bar(stat = "identity") +
  geom_errorbar(aes(ymin = Estimate - 1.96 * `Pr(>|z|)`, ymax = Estimate + 1.96 * `Pr(>|z|)`), width = 
  coord_flip() +
  labs(title = "Coefficient Plot", x = "Predictor Variables", y = "Coefficient Estimate")
```



TRAIN DATA

Predict train data

```
pred <- predict(model, newdata = train, type = "response")
pred_value <- ifelse(pred > 0.5, 1, 0)
result <- table(pred_value, train$Response)
result
```

```
##
## pred_value    0    1
##          0 5460  932
##          1    2    0
```

Calculate accuracy, precision, recall, F1-score

```
TN <- result[1, 1]
TP <- result[2, 2]
FP <- result[1, 2]
FN <- result[2, 1]

accuracy <- (TN + TP) / (TN + TP + FP + FN)
accuracy
```

```
## [1] 0.8539256
```

```r
precision <- TP / (TP + FP)
precision
```

```
## [1] 0
```

```r
recall <- TP / (TP + FN)
recall
```

```
## [1] 0
```

```r
f1_score <- 2 * (precision * recall) / (precision + recall)
f1_score
```

```
## [1] NaN
```

-> (TP) value is 0, it means that the model did not correctly predict any positive instances. Both precision and recall will also be 0, and the F1 score cannot be calculated.

TEST DATA

Predict test data

```r
pred2 <- predict(model, newdata = test, type = "response")
pred_value2 <- ifelse(pred2 > 0.5, 1, 0)
result <- table(pred_value2, test$Response)
result
```

```
##
## pred_value2    0    1
##           0 2363  376
##           1    1    0
```

Calculate accuracy, precision, recall, F1-score

```r
TN <- result[1, 1]
TP <- result[2, 2]
FP <- result[1, 2]
FN <- result[2, 1]

accuracy <- (TN + TP) / (TN + TP + FP + FN)
accuracy
```

```
## [1] 0.8624088
```

```r
precision <- TP / (TP + FP)
precision
```

```
## [1] 0
```

```r
recall <- TP / (TP + FN)
recall
```

```
## [1] 0
```

```r
f1_score <- 2 * (precision * recall) / (precision + recall)
f1_score
```

```
## [1] NaN
```

ROC Curve

```r
# Create ROC curves for train and test data
roc_data <- roc(train$Response, pred)
```

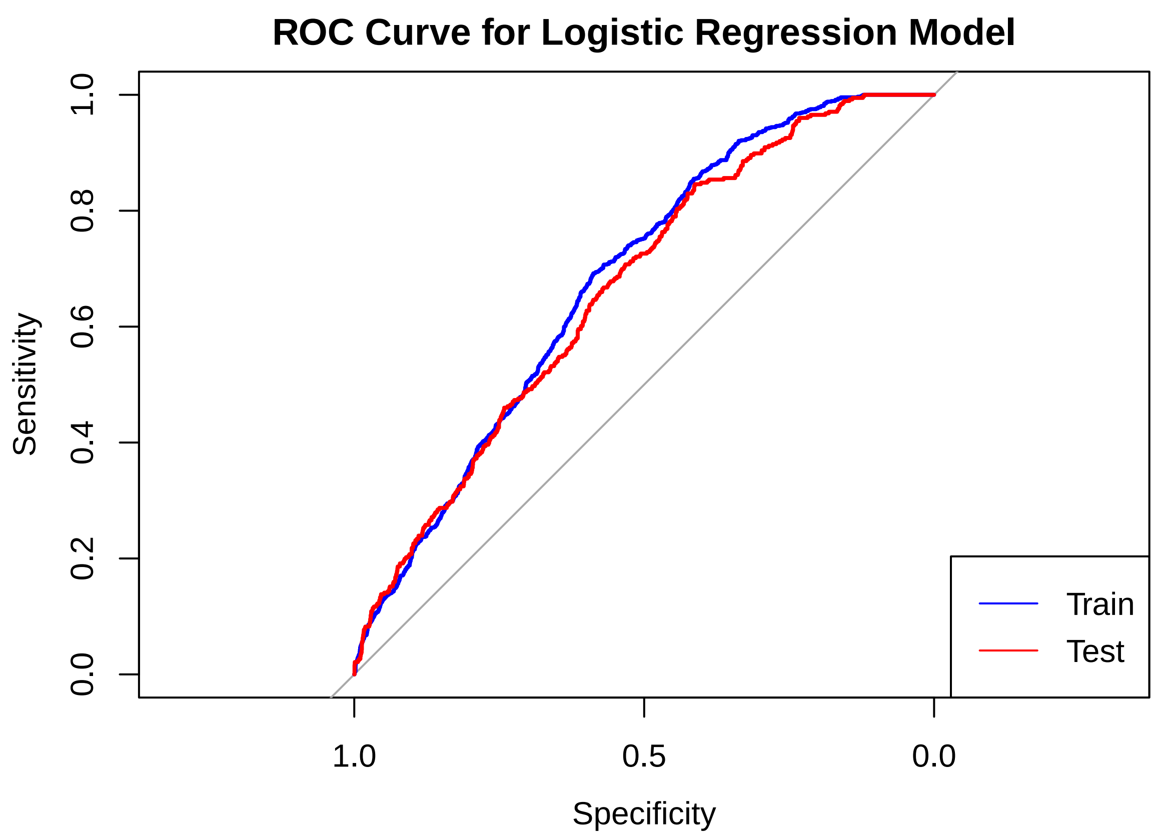
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
roc_data2 <- roc(test$Response, pred2)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
# Combine ROC curves into a single plot
plot(roc_data, col = "blue", main = "ROC Curve for Logistic Regression Model")
lines(roc_data2, col = "red")
legend("bottomright", legend = c("Train", "Test"), col = c("blue", "red"), lty = 1)
```



2. DECISION TREE

Build model

```
set.seed(42)
data.tree = rpart(Response ~ .,data = train, method="class")
data.tree
```

```
## n= 6394
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 6394 932 0 (0.85423835 0.14576165)
##     2) Renew_Offer_Type>=2.5 1736  19 0 (0.98905530 0.01094470) *
##     3) Renew_Offer_Type< 2.5 4658 913 0 (0.80399313 0.19600687)
##       6) Marital_Status>=1.5 3908 681 0 (0.82574207 0.17425793)
##        12) Income< -0.900723 1117 107 0 (0.90420770 0.09579230) *
##        13) Income>=-0.900723 2791 574 0 (0.79433895 0.20566105)
##          26) EmploymentStatus< 3.5 2684 485 0 (0.81929955 0.18070045) *
##          27) EmploymentStatus>=3.5 107  18 1 (0.16822430 0.83177570) *
##       7) Marital_Status< 1.5 750 232 0 (0.69066667 0.30933333)
##        14) Income>=-0.3808399 438  89 0 (0.79680365 0.20319635) *
```

```
##        15) Income< -0.3808399 312 143 0 (0.54166667 0.45833333)
##          30) Months_Since_Last_Claim>=1.032735 39   2 0 (0.94871795 0.05128205) *
##          31) Months_Since_Last_Claim< 1.032735 273 132 1 (0.48351648 0.51648352)
##            62) Total_Claim_Amount< -0.4598129 22   0 0 (1.00000000 0.00000000) *
##            63) Total_Claim_Amount>=-0.4598129 251 110 1 (0.43824701 0.56175299)
##             126) Income< -0.8519737 96  37 0 (0.61458333 0.38541667) *
##             127) Income>=-0.8519737 155  51 1 (0.32903226 0.67096774)
##               254) EmploymentStatus< 3.5 98  47 1 (0.47959184 0.52040816)
##                 508) Education>=3.5 32   5 0 (0.84375000 0.15625000) *
##                 509) Education< 3.5 66  20 1 (0.30303030 0.69696970) *
##               255) EmploymentStatus>=3.5 57   4 1 (0.07017544 0.92982456) *
```

```r
summary(data.tree)
```

```
## Call:
## rpart(formula = Response ~ ., data = train, method = "class")
##   n= 6394
##
##           CP nsplit rel error    xerror       xstd
## 1 0.01904506      0 1.0000000 1.0000000 0.03027482
## 2 0.01421674      4 0.9238197 0.9012876 0.02898280
## 3 0.01180258      8 0.8669528 0.8980687 0.02893881
## 4 0.01000000     10 0.8433476 0.8937768 0.02887997
##
## Variable importance
##          EmploymentStatus          Renew_Offer_Type                    Income
##                        37                        22                        15
##            Marital_Status        Total_Claim_Amount   Months_Since_Last_Claim
##                         6                         5                         4
##                 Education   Customer_Lifetime_Value             Location_Code
##                         4                         2                         2
## Number_of_Open_Complaints       Monthly_Premium_Auto
##                         1                         1
##
## Node number 1: 6394 observations,    complexity param=0.01904506
##   predicted class=0  expected loss=0.1457617  P(node) =1
##     class counts:  5462   932
##    probabilities: 0.854 0.146
##   left son=2 (1736 obs) right son=3 (4658 obs)
##   Primary splits:
##       Renew_Offer_Type   < 2.5        to the right, improve=86.62472, (0 missing)
##       Marital_Status     < 1.5        to the right, improve=20.03729, (0 missing)
##       Sales_Channel      < 1.5        to the right, improve=18.23999, (0 missing)
##       Total_Claim_Amount < -0.5269586 to the left,  improve=15.55030, (0 missing)
##       Income             < -0.900723  to the left,  improve=14.07473, (0 missing)
##   Surrogate splits:
##       Income < 2.047492   to the right, agree=0.729, adj=0.001, (0 split)
##
## Node number 2: 1736 observations
##   predicted class=0  expected loss=0.0109447  P(node) =0.2715045
##     class counts:  1717    19
##    probabilities: 0.989 0.011
##
## Node number 3: 4658 observations,    complexity param=0.01904506
```

```
##   predicted class=0  expected loss=0.1960069  P(node) =0.7284955
##     class counts:  3745   913
##    probabilities: 0.804 0.196
##   left son=6 (3908 obs) right son=7 (750 obs)
##   Primary splits:
##       Marital_Status     < 1.5        to the right, improve=22.96143, (0 missing)
##       Income             < -0.900723  to the left,  improve=21.71285, (0 missing)
##       EmploymentStatus   < 4.5        to the right, improve=21.19408, (0 missing)
##       Sales_Channel      < 1.5        to the right, improve=16.47464, (0 missing)
##       Total_Claim_Amount < -0.4618456 to the left,  improve=16.03380, (0 missing)
##   Surrogate splits:
##       Customer_Lifetime_Value < -0.8729307 to the right, agree=0.839, adj=0.001, (0 split)
##
## Node number 6: 3908 observations,    complexity param=0.01904506
##   predicted class=0  expected loss=0.1742579  P(node) =0.611198
##     class counts:  3227   681
##    probabilities: 0.826 0.174
##   left son=12 (1117 obs) right son=13 (2791 obs)
##   Primary splits:
##       Income             < -0.900723  to the left,  improve=19.25914, (0 missing)
##       EmploymentStatus   < 4.5        to the right, improve=18.79774, (0 missing)
##       Renew_Offer_Type   < 1.5        to the left,  improve=16.86070, (0 missing)
##       Sales_Channel      < 1.5        to the right, improve=12.97736, (0 missing)
##       Total_Claim_Amount < -0.5269586 to the left,  improve=10.55279, (0 missing)
##   Surrogate splits:
##       EmploymentStatus        < 4.5        to the right, agree=0.998, adj=0.994, (0 split)
##       Total_Claim_Amount      < 0.9343285  to the right, agree=0.751, adj=0.129, (0 split)
##       Marital_Status          < 2.5        to the right, agree=0.740, adj=0.090, (0 split)
##       Customer_Lifetime_Value < -0.8162598 to the left,  agree=0.728, adj=0.047, (0 split)
##
## Node number 7: 750 observations,    complexity param=0.01421674
##   predicted class=0  expected loss=0.3093333  P(node) =0.1172975
##     class counts:   518   232
##    probabilities: 0.691 0.309
##   left son=14 (438 obs) right son=15 (312 obs)
##   Primary splits:
##       Income                  < -0.3808399 to the right, improve=23.721620, (0 missing)
##       EmploymentStatus        < 3.5        to the left,  improve=14.556200, (0 missing)
##       Total_Claim_Amount      < -0.4255145 to the left,  improve=12.664510, (0 missing)
##       Customer_Lifetime_Value < -0.8157533 to the right, improve=10.373730, (0 missing)
##       Monthly_Premium_Auto    < -0.7765437 to the right, improve= 7.850406, (0 missing)
##   Surrogate splits:
##       EmploymentStatus              < 2.5        to the left,  agree=0.847, adj=0.631, (0 split)
##       Total_Claim_Amount            < 0.1364937  to the left,  agree=0.656, adj=0.173, (0 split)
##       Customer_Lifetime_Value       < -0.810636  to the right, agree=0.625, adj=0.099, (0 split)
##       Monthly_Premium_Auto          < -0.8056067 to the right, agree=0.591, adj=0.016, (0 split)
##       Months_Since_Policy_Inception < 1.55649    to the left,  agree=0.588, adj=0.010, (0 split)
##
## Node number 12: 1117 observations
##   predicted class=0  expected loss=0.0957923  P(node) =0.174695
##     class counts:  1010   107
##    probabilities: 0.904 0.096
##
## Node number 13: 2791 observations,    complexity param=0.01904506
```

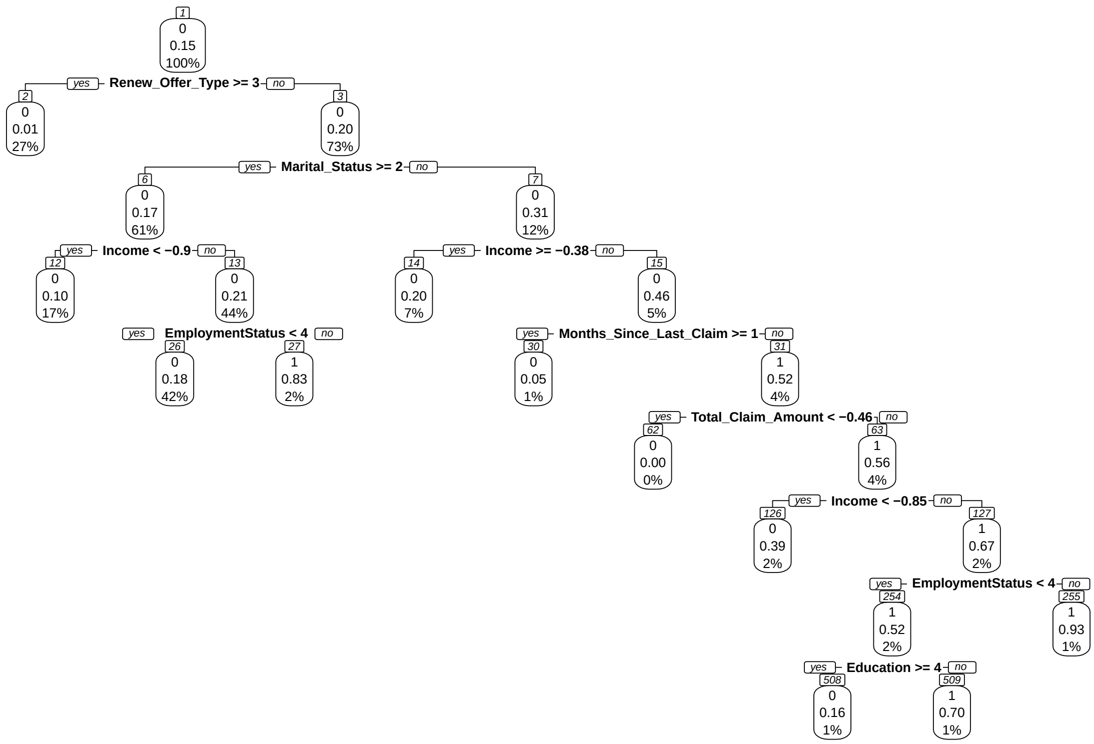
```
##   predicted class=0  expected loss=0.2056611  P(node) =0.436503
##     class counts:  2217   574
##    probabilities: 0.794 0.206
##   left son=26 (2684 obs) right son=27 (107 obs)
##   Primary splits:
##       EmploymentStatus   < 3.5        to the left,  improve=87.23662, (0 missing)
##       Total_Claim_Amount < -0.5269586 to the left,  improve=20.47424, (0 missing)
##       Location_Code      < 2.5        to the right, improve=12.51777, (0 missing)
##       Renew_Offer_Type   < 1.5        to the left,  improve=12.42603, (0 missing)
##       Sales_Channel      < 1.5        to the right, improve=12.24559, (0 missing)
##   Surrogate splits:
##       Income                  < -0.8884781 to the right, agree=0.965, adj=0.075, (0 split)
##       Customer_Lifetime_Value < -0.8349271 to the right, agree=0.963, adj=0.028, (0 split)
##
## Node number 14: 438 observations
##   predicted class=0  expected loss=0.2031963  P(node) =0.06850172
##     class counts:   349    89
##    probabilities: 0.797 0.203
##
## Node number 15: 312 observations,    complexity param=0.01421674
##   predicted class=0  expected loss=0.4583333  P(node) =0.04879575
##     class counts:   169   143
##    probabilities: 0.542 0.458
##   left son=30 (39 obs) right son=31 (273 obs)
##   Primary splits:
##       Months_Since_Last_Claim < 1.032735   to the right, improve=14.770150, (0 missing)
##       Income                  < -0.8519737 to the left,  improve=13.450730, (0 missing)
##       Total_Claim_Amount      < -0.4598129 to the left,  improve=12.923710, (0 missing)
##       EmploymentStatus        < 4.5        to the right, improve=10.991790, (0 missing)
##       Customer_Lifetime_Value < 0.2840898  to the left,  improve= 9.392127, (0 missing)
##
## Node number 26: 2684 observations
##   predicted class=0  expected loss=0.1807004  P(node) =0.4197685
##     class counts:  2199   485
##    probabilities: 0.819 0.181
##
## Node number 27: 107 observations
##   predicted class=1  expected loss=0.1682243  P(node) =0.01673444
##     class counts:    18    89
##    probabilities: 0.168 0.832
##
## Node number 30: 39 observations
##   predicted class=0  expected loss=0.05128205  P(node) =0.006099468
##     class counts:    37     2
##    probabilities: 0.949 0.051
##
## Node number 31: 273 observations,    complexity param=0.01421674
##   predicted class=1  expected loss=0.4835165  P(node) =0.04269628
##     class counts:   132   141
##    probabilities: 0.484 0.516
##   left son=62 (22 obs) right son=63 (251 obs)
##   Primary splits:
##       Total_Claim_Amount      < -0.4598129 to the left,  improve=12.765990, (0 missing)
##       Income                  < -0.8519737 to the left,  improve=12.549180, (0 missing)
```

```
##       EmploymentStatus        < 4.5       to the right, improve=10.253690, (0 missing)
##       Customer_Lifetime_Value < 0.2840898  to the left,  improve= 8.864469, (0 missing)
##       Location_Code           < 2.5       to the right, improve= 6.696476, (0 missing)
##   Surrogate splits:
##       Location_Code           < 2.5       to the right, agree=0.963, adj=0.545, (0 split)
##       Monthly_Premium_Auto    < -0.8927959 to the left,  agree=0.938, adj=0.227, (0 split)
##       Customer_Lifetime_Value < -0.833638  to the left,  agree=0.923, adj=0.045, (0 split)
##
## Node number 62: 22 observations
##   predicted class=0  expected loss=0  P(node) =0.003440726
##     class counts:    22     0
##    probabilities: 1.000 0.000
##
## Node number 63: 251 observations,    complexity param=0.01421674
##   predicted class=1  expected loss=0.438247  P(node) =0.03925555
##     class counts:   110   141
##    probabilities: 0.438 0.562
##   left son=126 (96 obs) right son=127 (155 obs)
##   Primary splits:
##       Income                  < -0.8519737 to the left,  improve=9.667781, (0 missing)
##       Customer_Lifetime_Value < -0.7443817 to the right, improve=8.763435, (0 missing)
##       EmploymentStatus        < 4.5       to the right, improve=7.937910, (0 missing)
##       Number_of_Policies      < 2.315234   to the right, improve=4.544674, (0 missing)
##       Monthly_Premium_Auto    < 0.2406626  to the right, improve=4.376346, (0 missing)
##   Surrogate splits:
##       EmploymentStatus          < 4.5       to the right, agree=0.988, adj=0.969, (0 split)
##       Vehicle_Size              < 2.5       to the right, agree=0.673, adj=0.146, (0 split)
##       Customer_Lifetime_Value   < 0.9578242  to the right, agree=0.661, adj=0.115, (0 split)
##       Number_of_Open_Complaints < 1.225431   to the right, agree=0.653, adj=0.094, (0 split)
##       Education                 < 4.5       to the right, agree=0.637, adj=0.052, (0 split)
##
## Node number 126: 96 observations
##   predicted class=0  expected loss=0.3854167  P(node) =0.01501408
##     class counts:    59    37
##    probabilities: 0.615 0.385
##
## Node number 127: 155 observations,    complexity param=0.01180258
##   predicted class=1  expected loss=0.3290323  P(node) =0.02424148
##     class counts:    51   104
##    probabilities: 0.329 0.671
##   left son=254 (98 obs) right son=255 (57 obs)
##   Primary splits:
##       EmploymentStatus              < 3.5       to the left,  improve=12.081750, (0 missing)
##       Number_of_Policies            < 2.315234   to the right, improve= 6.600872, (0 missing)
##       Customer_Lifetime_Value       < -0.7877115 to the right, improve= 5.275323, (0 missing)
##       Months_Since_Policy_Inception < -1.417781  to the left,  improve= 4.610038, (0 missing)
##       Vehicle_Class                 < 5.5       to the right, improve= 4.585292, (0 missing)
##   Surrogate splits:
##       Number_of_Open_Complaints < 0.1269926  to the left,  agree=0.671, adj=0.105, (0 split)
##       Total_Claim_Amount        < -0.3658708 to the right, agree=0.671, adj=0.105, (0 split)
##       Months_Since_Last_Claim   < 0.8838254  to the left,  agree=0.665, adj=0.088, (0 split)
##       Customer_Lifetime_Value   < -0.8278004 to the right, agree=0.658, adj=0.070, (0 split)
##       Education                 < 3.5       to the left,  agree=0.652, adj=0.053, (0 split)
##
```

```
## Node number 254: 98 observations,    complexity param=0.01180258
##   predicted class=1  expected loss=0.4795918  P(node) =0.01532687
##     class counts:    47    51
##    probabilities: 0.480 0.520
##   left son=508 (32 obs) right son=509 (66 obs)
##   Primary splits:
##       Education                  < 3.5        to the right, improve=12.602080, (0 missing)
##       Number_of_Open_Complaints < 0.1269926  to the right, improve= 5.367806, (0 missing)
##       Customer_Lifetime_Value   < -0.7911586 to the right, improve= 4.860449, (0 missing)
##       State                      < 4.5        to the right, improve= 4.083203, (0 missing)
##       Income                     < -0.3898919 to the left,  improve= 3.743864, (0 missing)
##   Surrogate splits:
##       Months_Since_Last_Claim   < 0.7349162  to the right, agree=0.704, adj=0.094, (0 split)
##       Number_of_Open_Complaints < 0.1269926  to the right, agree=0.704, adj=0.094, (0 split)
##       Total_Claim_Amount        < 3.150641   to the right, agree=0.694, adj=0.063, (0 split)
##       Vehicle_Size               < 2.5        to the right, agree=0.684, adj=0.031, (0 split)
##       Customer_Lifetime_Value   < 2.49875    to the right, agree=0.684, adj=0.031, (0 split)
##
## Node number 255: 57 observations
##   predicted class=1  expected loss=0.07017544  P(node) =0.008914607
##     class counts:     4    53
##    probabilities: 0.070 0.930
##
## Node number 508: 32 observations
##   predicted class=0  expected loss=0.15625  P(node) =0.005004692
##     class counts:    27     5
##    probabilities: 0.844 0.156
##
## Node number 509: 66 observations
##   predicted class=1  expected loss=0.3030303  P(node) =0.01032218
##     class counts:    20    46
##    probabilities: 0.303 0.697
```

plot tree

```r
prp(data.tree, type = 2, extra = "auto", nn = TRUE, branch = 1, varlen = 0, yesno = 2)
```



Get the number of nodes

```r
num_nodes <- data.tree$n
num_nodes
```

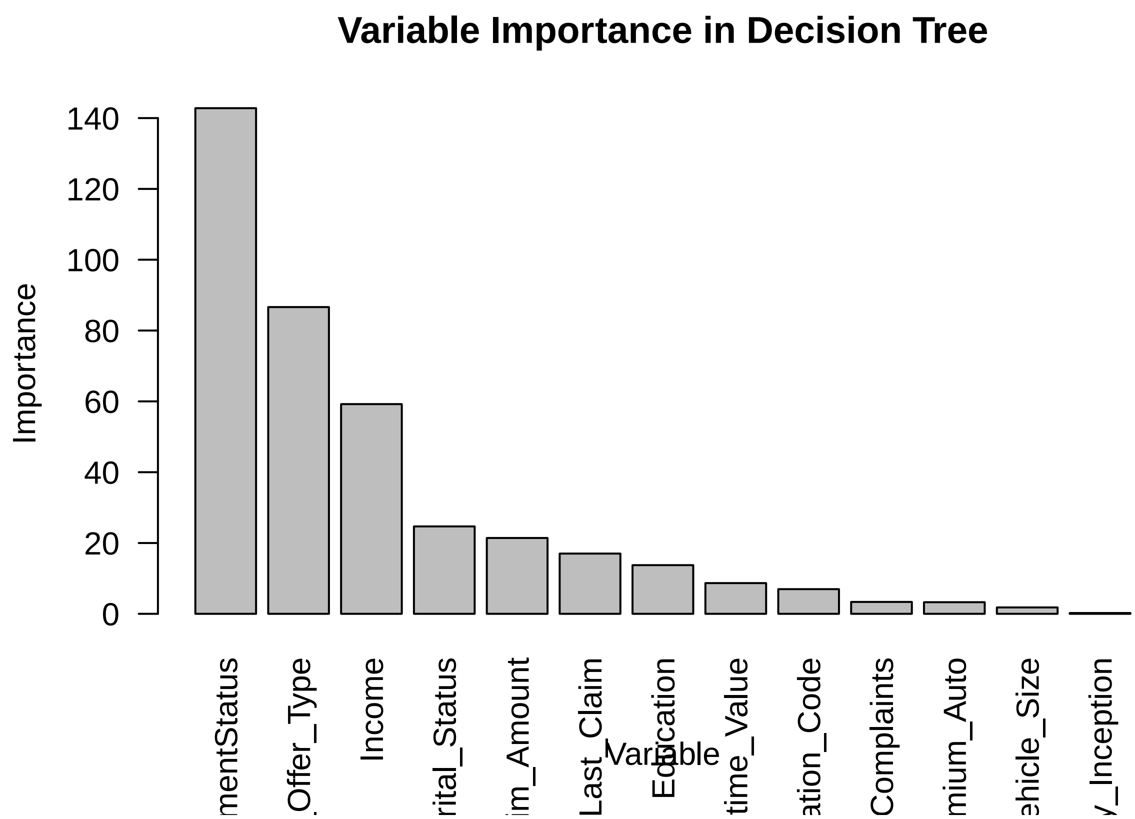
```
## [1] 4
```

Get variable importance

```r
var_importance <- data.tree$variable.importance
var_importance
```

```
##            EmploymentStatus             Renew_Offer_Type
##                 142.8005446                   86.6247246
##                      Income               Marital_Status
##                  59.2207982                   24.6856104
##          Total_Claim_Amount      Months_Since_Last_Claim
##                  21.4138733                   17.0113937
##                   Education      Customer_Lifetime_Value
##                  13.7414911                    8.6769706
##               Location_Code     Number_of_Open_Complaints
##                   6.9632678                    3.3595621
##        Monthly_Premium_Auto                 Vehicle_Size
##                   3.2815157                    1.8036997
## Months_Since_Policy_Inception
##                   0.2280925
```

```r
# Create a bar plot of variable importance with rotated x-axis labels
barplot(var_importance, main = "Variable Importance in Decision Tree",
        xlab = "Variable", ylab = "Importance",
        names.arg = names(var_importance), las = 2)
```



TRAIN DATA Predict train data

```r
pred = predict(data.tree, train, type = 'prob')
pred_value <- ifelse(pred[, "1"] > 0.5, 1, 0)
# accuracy train data
result = table(pred_value, train$Response)
result
```

```
##
## pred_value    0    1
##          0 5420  744
##          1   42  188
```

Calculate accuracy, precision, recall, F1-score

```r
TN <- result[1, 1]
TP <- result[2, 2]
FP <- result[1, 2]
FN <- result[2, 1]
```

```r
accuracy <- (TN + TP) / (TN + TP + FP + FN)
accuracy
```

```
## [1] 0.8770723
```

```r
precision <- TP / (TP + FP)
precision
```

```
## [1] 0.2017167
```

```r
recall <- TP / (TP + FN)
recall
```

```
## [1] 0.8173913
```

```r
f1_score <- 2 * (precision * recall) / (precision + recall)
f1_score
```

```
## [1] 0.32358
```

TEST DATA

Predict test data

```r
pred2 = predict(data.tree, test, type = 'prob')
pred_value2 <- ifelse(pred2[, "1"] > 0.5, 1, 0)
# accuracy test data
result = table(pred_value2, test$Response)
result
```

```
##
## pred_value2    0    1
##           0 2340  312
##           1   24   64
```

Calculate accuracy, precision, recall, F1-score

```r
TN <- result[1, 1]
TP <- result[2, 2]
FP <- result[1, 2]
FN <- result[2, 1]

accuracy <- (TN + TP) / (TN + TP + FP + FN)
accuracy
```

```
## [1] 0.8773723
```

```r
precision <- TP / (TP + FP)
precision
```

```
## [1] 0.1702128
```

```r
recall <- TP / (TP + FN)
recall
```

```
## [1] 0.7272727
```

```r
f1_score <- 2 * (precision * recall) / (precision + recall)
f1_score
```

```
## [1] 0.2758621
```

ROC Curve

```r
# Create ROC data
roc_data <- roc(train$Response, pred_value)
```

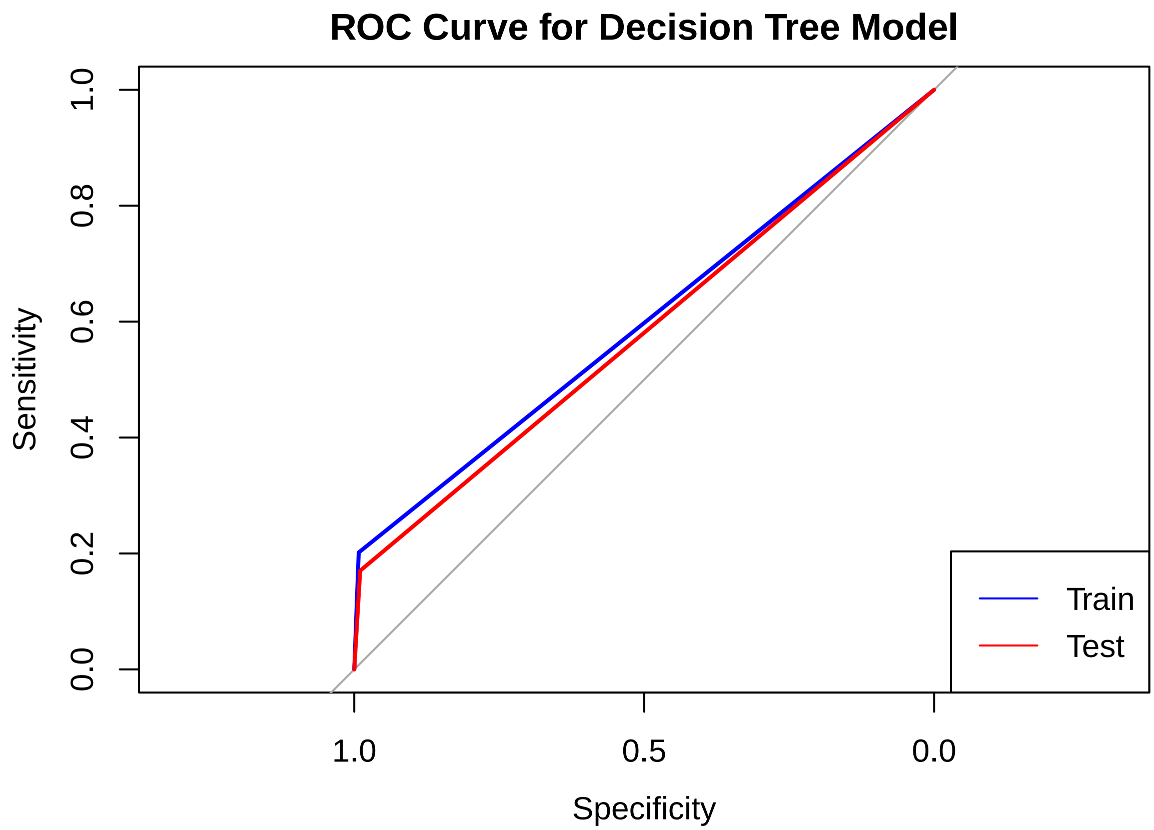
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
roc_data2 <- roc(test$Response, pred_value2)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
# Combine ROC curves into a single plot
plot(roc_data, col = "blue", main = "ROC Curve for Decision Tree Model")
lines(roc_data2, col = "red")
legend("bottomright", legend = c("Train", "Test"), col = c("blue", "red"), lty = 1)
```



3. RANDOM FOREST

Build model

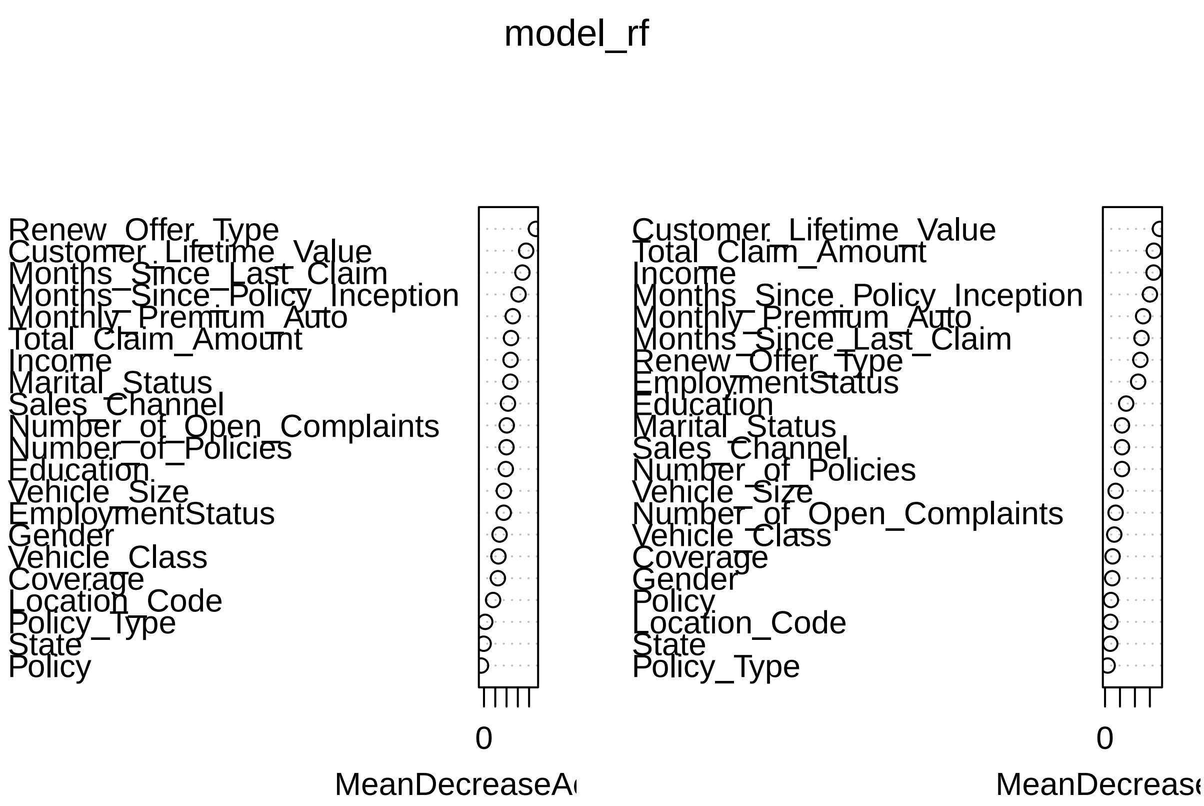
```
set.seed(42)
train$Response <- factor(train$Response)
model_rf <- randomForest(Response ~ ., data = train, ntree = 15, mtry = 7, importance = TRUE)
model_rf
```

```
##
## Call:
##  randomForest(formula = Response ~ ., data = train, ntree = 15,      mtry = 7, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 15
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 1.77%
## Confusion matrix:
##      0   1 class.error
## 0 5374  82  0.01502933
## 1   31 901  0.03326180
```

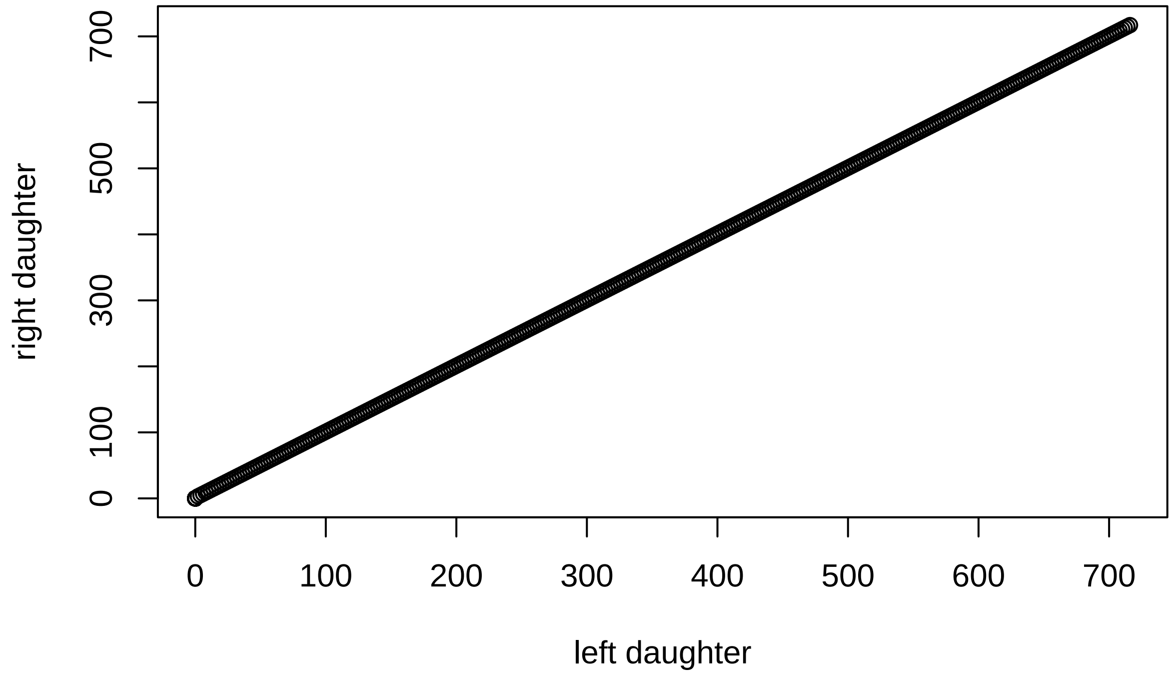
```
summary(model_rf)
```

```
##                 Length Class  Mode
## call                6  -none- call
## type                1  -none- character
## predicted        6394  factor numeric
## err.rate           45  -none- numeric
## confusion           6  -none- numeric
## votes           12788  matrix numeric
## oob.times        6394  -none- numeric
## classes             2  -none- character
## importance         84  -none- numeric
## importanceSD       63  -none- numeric
## localImportance     0  -none- NULL
## proximity           0  -none- NULL
## ntree               1  -none- numeric
## mtry                1  -none- numeric
## forest             14  -none- list
## y                6394  factor numeric
## test                0  -none- NULL
## inbag               0  -none- NULL
## terms               3  terms  call
```

```r
varImpPlot(model_rf)
```




```r
library(randomForest)
tree_plot <- getTree(model_rf)
plot(tree_plot)
```



TRAIN DATA

Predicting on train set

```
pred <- predict(model_rf, train, type = "response")
# Checking classification accuracy
result = table(pred, train$Response)
result
```

```
##
## pred    0    1
##    0 5462    1
##    1    0  931
```

Calculate accuracy, precision, recall, F1-score

```
TN <- result[1, 1]
TP <- result[2, 2]
FP <- result[1, 2]
FN <- result[2, 1]

accuracy <- (TN + TP) / (TN + TP + FP + FN)
accuracy
```

```
## [1] 0.9998436
```

```r
precision <- TP / (TP + FP)
precision
```

```
## [1] 0.998927
```

```r
recall <- TP / (TP + FN)
recall
```

```
## [1] 1
```

```r
f1_score <- 2 * (precision * recall) / (precision + recall)
f1_score
```

```
## [1] 0.9994632
```

TEST DATA

Predicting on test set

```r
pred2 <- predict(model_rf, test, type = "class")
# Checking classification accuracy
result = table(pred2, test$Response)
result
```

```
##
## pred2    0    1
##     0 2348   12
##     1   16  364
```

Calculate accuracy, precision, recall, F1-score

```r
TN <- result[1, 1]
TP <- result[2, 2]
FP <- result[1, 2]
FN <- result[2, 1]

accuracy <- (TN + TP) / (TN + TP + FP + FN)
accuracy
```

```
## [1] 0.989781
```

```r
precision <- TP / (TP + FP)
precision
```

```
## [1] 0.9680851
```

```r
recall <- TP / (TP + FN)
recall
```

```
## [1] 0.9578947
```

```r
f1_score <- 2 * (precision * recall) / (precision + recall)
f1_score
```

```
## [1] 0.962963
```

ROC Curve

```r
# Create a binary vector indicating if the predicted class is 1 or 0
pred_class <- as.numeric(pred == "1")
pred_class2 <- as.numeric(pred2 == "1")

# Create ROC data
roc_data <- roc(train$Response, pred_class)
```

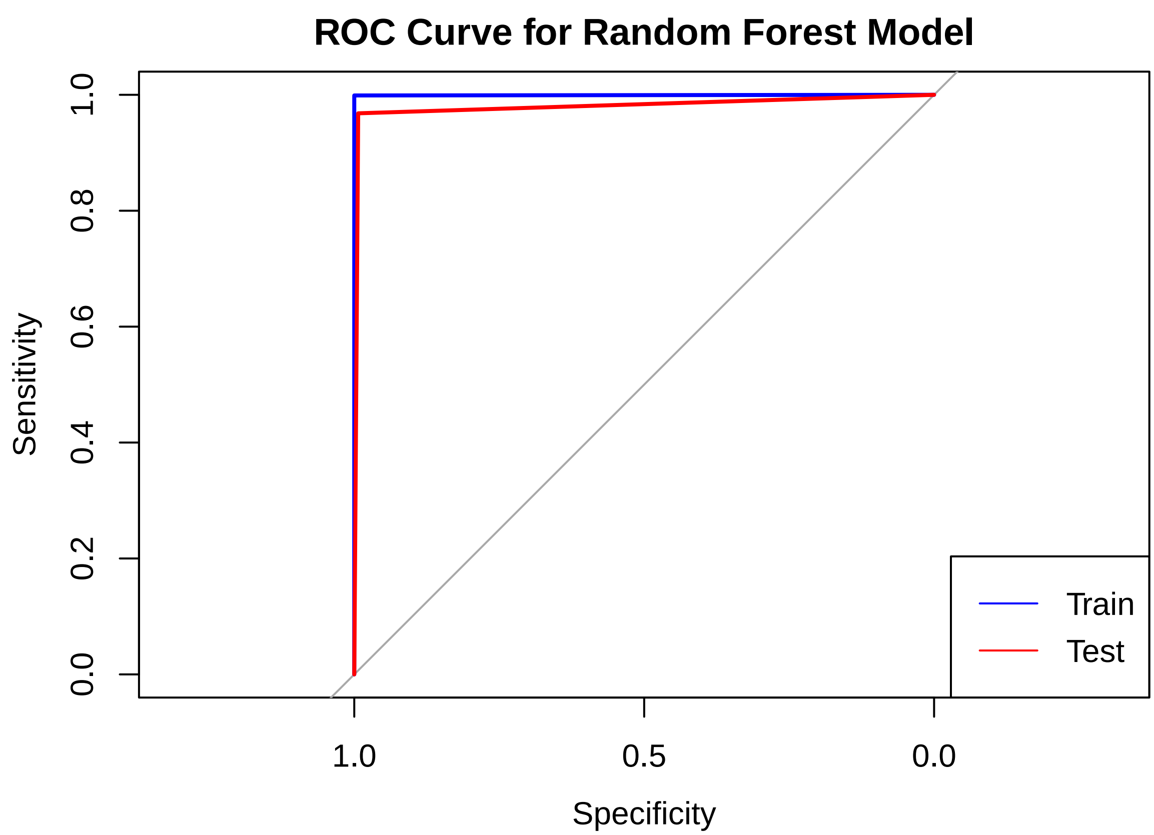
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
roc_data2 <- roc(test$Response, pred_class2)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
# Combine ROC curves into a single plot
plot(roc_data, col = "blue", main = "ROC Curve for Random Forest Model")
lines(roc_data2, col = "red")
legend("bottomright", legend = c("Train", "Test"), col = c("blue", "red"), lty = 1)
```



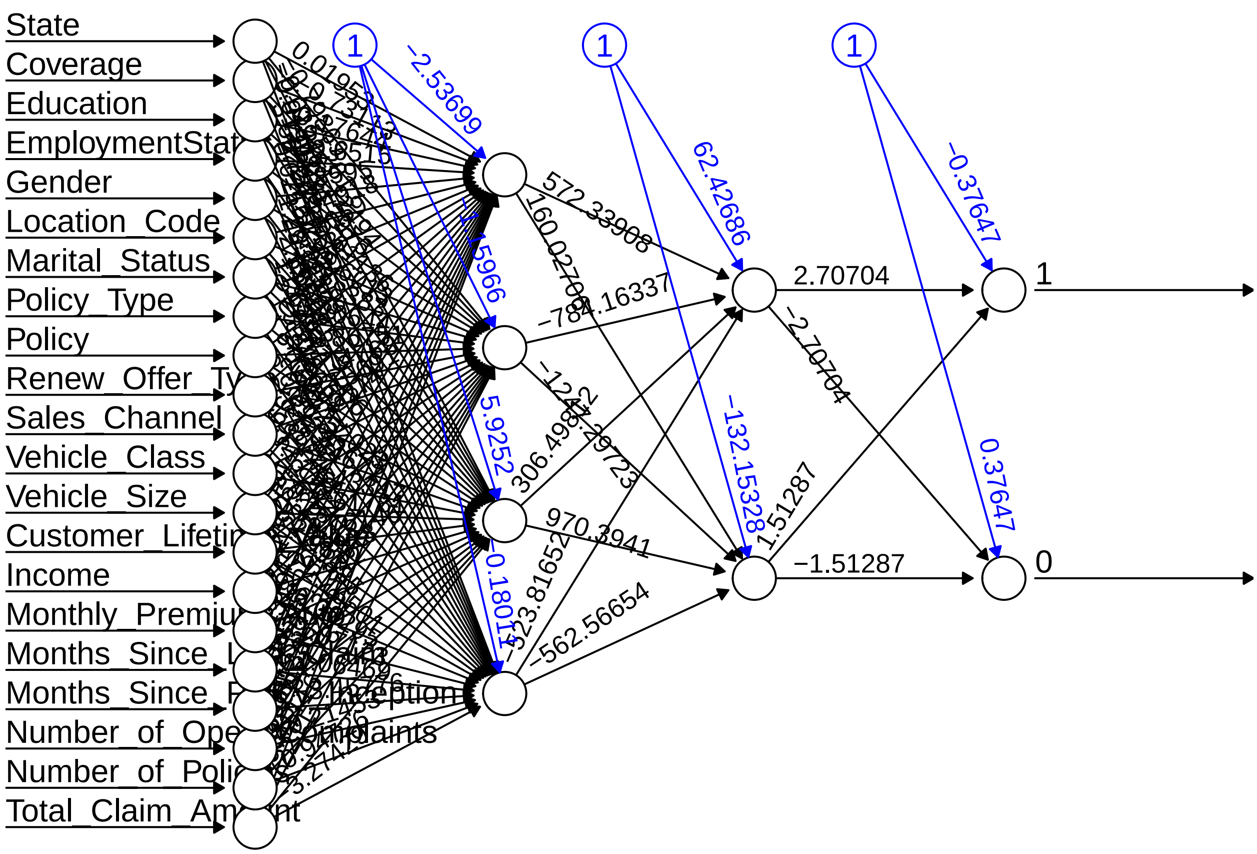
4. NEUTRAL NETWORK

Build model

```
set.seed(123)
model = neuralnet(Response ~ ., data=train, hidden=c(4,2),linear.output = FALSE, act.fct = "logistic")
summary(model)
```

```
##                     Length Class      Mode
## call                     6 -none-     call
## response             12788 -none-     logical
## covariate           134274 -none-     numeric
## model.list               2 -none-     list
## err.fct                  1 -none-     function
## act.fct                  1 -none-     function
## linear.output            1 -none-     logical
## data                    22 data.frame list
## exclude                  0 -none-     NULL
## net.result               1 -none-     list
## weights                  1 -none-     list
## generalized.weights      1 -none-     list
## startweights             1 -none-     list
## result.matrix          107 -none-     numeric
```

```r
plot(model,rep = "best")
```



TRAIN DATA

Predicting on train set

```r
pred <- predict(model, newdata = train)
pred_class <- ifelse(pmax(pred) > 0.5, 1, 0)[,2]
result <- table(train$Response, pred_class)
result
```

```
##    pred_class
##        0    1
##   0 5265  197
##   1  645  287
```

Calculate accuracy, precision, recall, F1-score

```r
TN <- result[1, 1]
TP <- result[2, 2]
FP <- result[1, 2]
FN <- result[2, 1]
```

```r
accuracy <- (TN + TP) / (TN + TP + FP + FN)
accuracy
```

```
## [1] 0.868314
```

```r
precision <- TP / (TP + FP)
precision
```

```
## [1] 0.5929752
```

```r
recall <- TP / (TP + FN)
recall
```

```
## [1] 0.3079399
```

```r
f1_score <- 2 * (precision * recall) / (precision + recall)
f1_score
```

```
## [1] 0.4053672
```

TEST DATA

Predicting on test set

```r
pred2 <- predict(model, newdata = test)
pred_class <- ifelse(pmax(pred2) > 0.5, 1, 0)[,2]
result <- table(test$Response, pred_class)
result
```

```
##    pred_class
##        0    1
##   0 2237  127
##   1  265  111
```

Calculate accuracy, precision, recall, F1-score

```r
TN <- result[1, 1]
TP <- result[2, 2]
FP <- result[1, 2]
FN <- result[2, 1]

accuracy <- (TN + TP) / (TN + TP + FP + FN)
accuracy
```

```
## [1] 0.8569343
```

```r
precision <- TP / (TP + FP)
precision
```

```
## [1] 0.4663866
```

```r
recall <- TP / (TP + FN)
recall
```

```
## [1] 0.2952128
```

```r
f1_score <- 2 * (precision * recall) / (precision + recall)
f1_score
```

```
## [1] 0.3615635
```

ROC Curve

```r
# Create ROC curve
roc_data <- roc(train$Response, pred[,2])
```

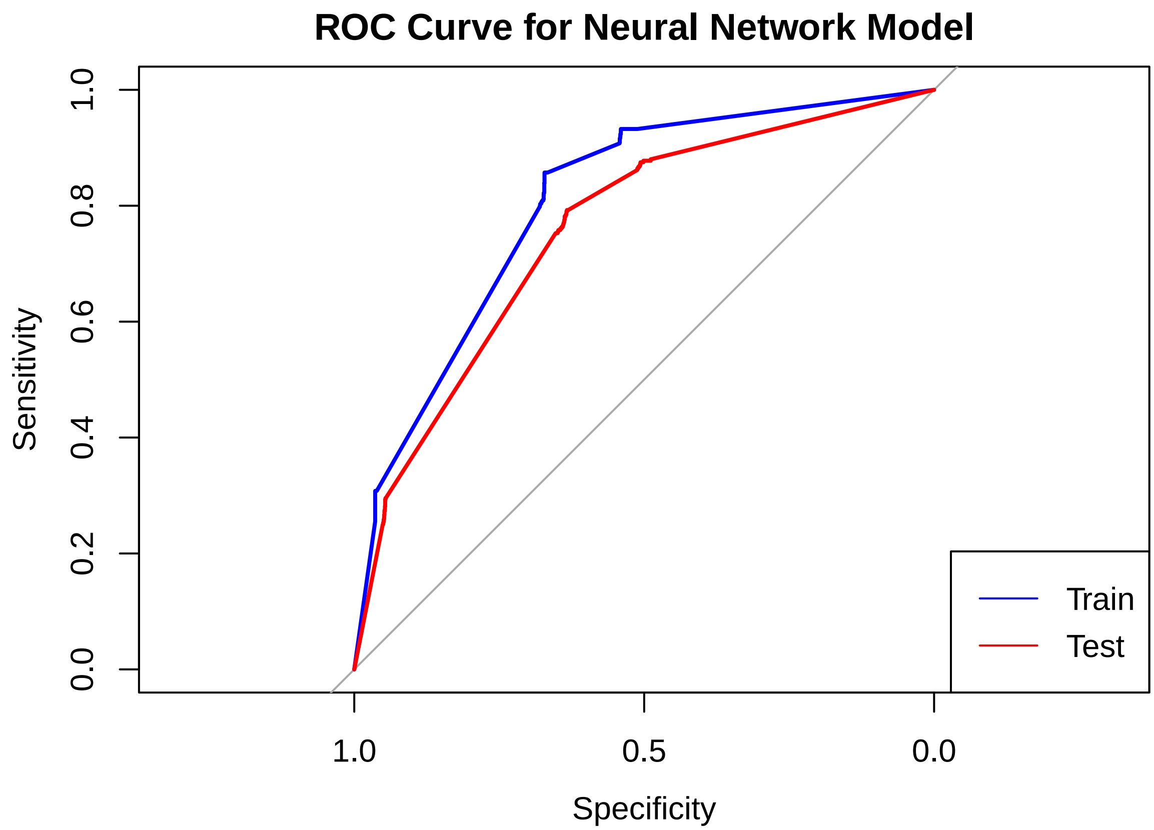
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
roc_data2 <- roc(test$Response, pred2[,2])
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
# Combine ROC curves into a single plot
plot(roc_data, col = "blue", main = "ROC Curve for Neural Network Model")
lines(roc_data2, col = "red")
legend("bottomright", legend = c("Train", "Test"), col = c("blue", "red"), lty = 1)
```


ROC Curve for Neural Network Model
Sensitivity
Specificity
1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.5
0.0
Train
Test