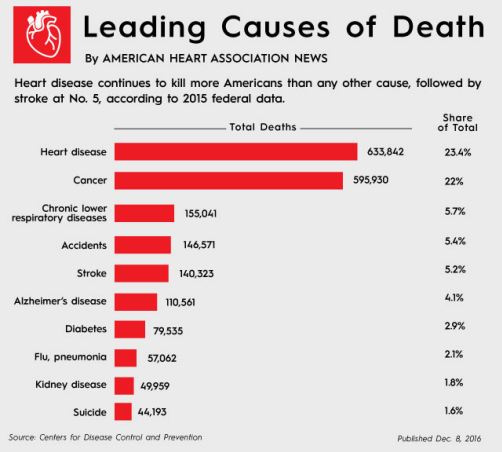
Project on Classification module:

Motivation: Now a day's heart failure diseases are increasing day by day. if we look at the statistics of US 23% of Mortality rate is due to Heart disease.



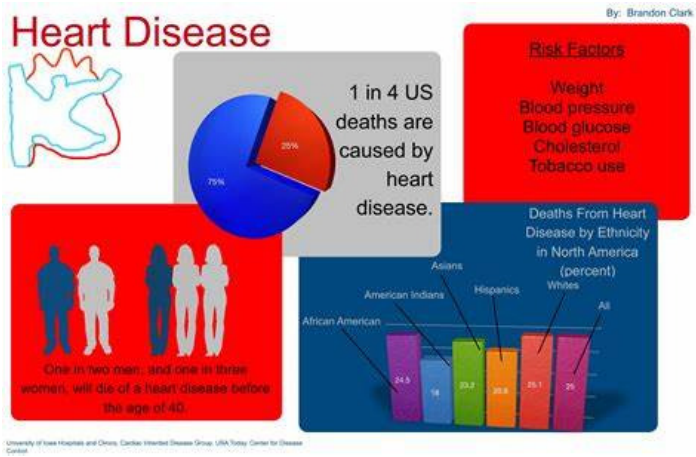


Figure 1 World statistics of Death

We are going to predict mortality by heart failure based on the 12 features included in the data set. This can be used to help hospitals in assessing the severity of patients with cardiovascular diseases (CVDs).

People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need early detection and management wherein a machine learning model can be of great help.

In this case study we will first split the data in training and test data set. Fit the logistic regression model on training data set and then test the model on test data set to check the accuracy of the model.

Data source:

The data is selected from [www.kaggle.com-Heart Failure Prediction.]

Data set contains following features:

Features	Data type	Details
age	Continuous	Age of a person
anaemia	Categorical	If a person is suffering from Anaemia if Yes- 1, If no 0.(Anaemia is a deficiency in the number or quality of red blood cells in your body)
creatinine_phosphokinase	Continuous	A creatine phosphokinase (CPK) determination is the most specific test for muscular dystrophy (MD).
diabetes	Categorical	if a person is sufferring from Diabets, if Yes- 1, IF no -0
ejection_fraction	Continuous	Image result for ejection fraction normal range Normal Heart. A normal left ventricular ejection fraction (LVEF) ranges from 55% to 70%.
high_blood_pressure	Categorical	high_blood_pressure- Yes/No
platelets	Continuous	Platelet level

serum_creatinine	Continuous	Level of serum creatinine in the blood (mg/dL)
serum_sodium	Continuous	Level of serum sodium in the blood (mEq/L)
sex	Categorical	Male = 1, Female =0
smoking	Categorical	yes -1 , No- 0
time	Continuous	Follow-up period (days)
DEATH_EVENT (Response variable)	Categorical	If the patient deceased during the follow-up period (boolean)

```
##### Importing data

data1=read.csv("D:/Upgrad/Classification/heart_failure_clinical_records_dataset.csv")

##### Checking number of observations (customers) on whom the analysis will be performed

N=nrow(data1)

N

nn=nrow(data1)

[1] 299

#### considering following featured for data visualization.

f_features = c("anaemia", "diabetes", "high_blood_pressure", "sex", "smoking", "DEATH_EVENT")

#### data visualisation with different graphs:

library(ggplot2)

plot1 = ggplot(df, aes(x = anaemia, fill = DEATH_EVENT)) +
 geom_bar(stat = "count", position = "stack", show.legend = FALSE) +
 scale_x_discrete(labels  = c("0 (False)", "1 (True)"))+
 scale_fill_manual(values = c("blue","green"),
                   name = "DEATH_EVENT",
                   labels = c("0 (False)", "1 (True)")) +
 labs(x = "Anaemia") +
 theme_minimal(base_size = 12) +
 geom_label(stat = "count", aes(label = ..count..), position = position_stack(vjust = 0.5),
            size = 5, show.legend = FALSE)

plot(plot1)
```

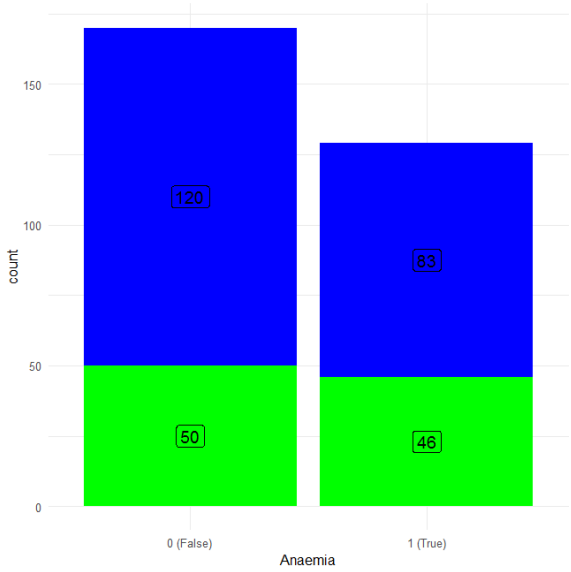


Figure 2 -Plot1- Anaemia v/s Death_Event

```
plot2 = ggplot(df, aes(x = diabetes, fill = DEATH_EVENT)) +
 geom_bar(stat = "count", position = "stack", show.legend = FALSE) +
 scale_x_discrete(labels  = c("0 (False)", "1 (True)"))+
 scale_fill_manual(values = c("blue","green"),
                   name = "DEATH_EVENT",
                   labels = c("0 (False)", "1 (True)")) +
 labs(x = "diabetes") +
 theme_minimal(base_size = 12) +
 geom_label(stat = "count", aes(label = ..count..), position =
position_stack(vjust = 0.5),
           size = 5, show.legend = FALSE)
plot(plot2)
```

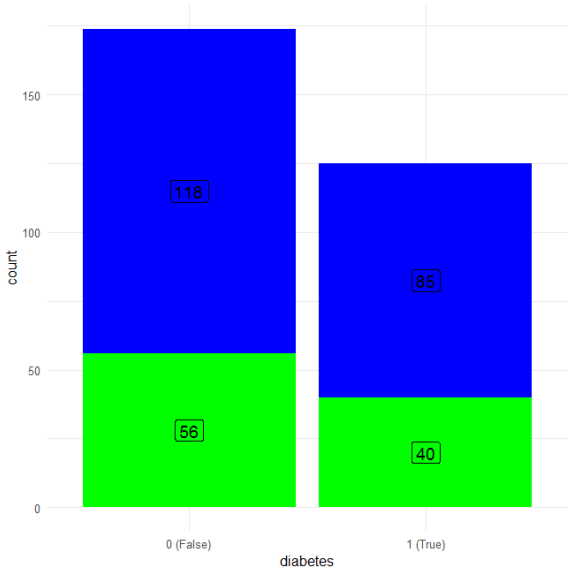


Figure 3-Plot 2 Diabetes v/s Death_Event

```
plot3= ggplot(df, aes(x = high_blood_pressure, fill =
DEATH_EVENT)) +
 geom_bar(stat = "count", position = "stack", show.legend =
FALSE) +
 scale_x_discrete(labels  = c("0 (False)", "1 (True)"))+
 scale_fill_manual(values = c("blue","green"),
                   name = "DEATH_EVENT",
                   labels = c("0 (False)", "1 (True)")) +
 labs(x = "high_blood_pressure") +
 theme_minimal(base_size = 12) +
 geom_label(stat = "count", aes(label = ..count..), position =
position_stack(vjust = 0.5),
           size = 5, show.legend = FALSE)
plot(plot3)
```

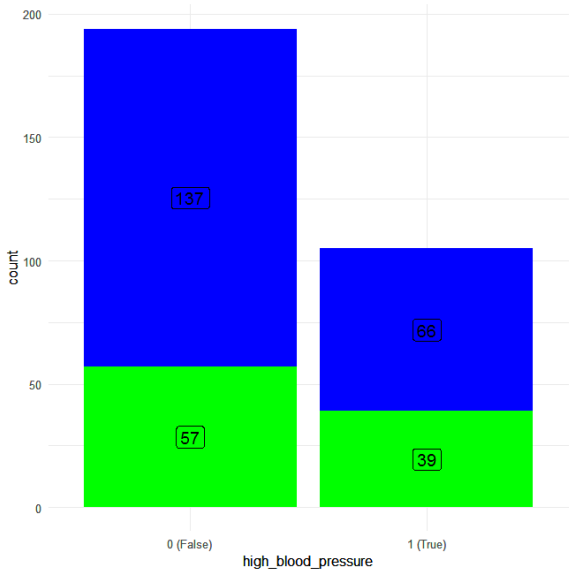


Figure 4- Plot 3 Blood pressure v/s Death_Event

```
plot4= ggplot(df, aes(x = sex, fill = DEATH_EVENT)) +
 geom_bar(stat = "count", position = "stack", show.legend = FALSE) +
 scale_x_discrete(labels  = c("0 (False)", "1 (True)"))+
 scale_fill_manual(values = c("blue","green"),
                   name = "DEATH_EVENT",
                   labels = c("0 (False)", "1 (True)")) +
 labs(x = "sex ") +
 theme_minimal(base_size = 12) +
 geom_label(stat = "count", aes(label = ..count..), position = position_stack(vjust = 0.5),
            size = 5, show.legend = FALSE)
plot(plot4)
```

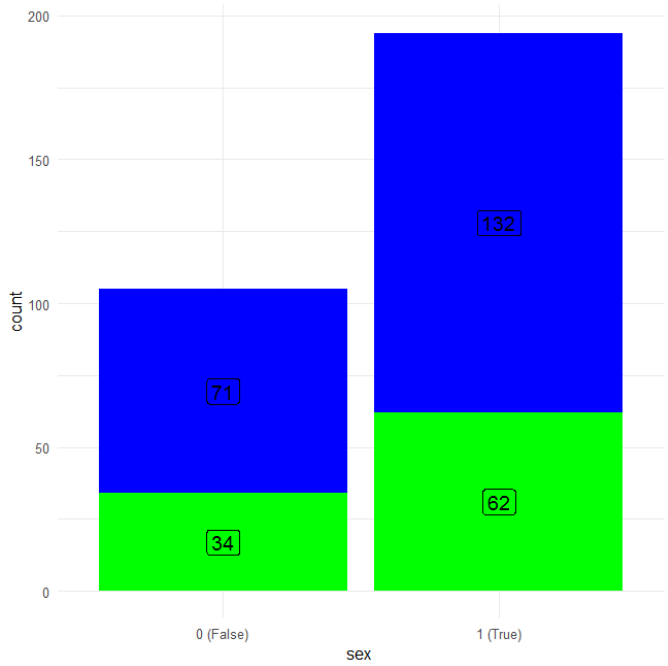


Figure 5 Plot 4- Sex v/s Death Event

```
plot5= ggplot(df, aes(x = sex, fill = DEATH_EVENT)) +
 geom_bar(stat = "count", position = "stack", show.legend = FALSE) +
 scale_x_discrete(labels  = c("0 (False)", "1 (True)"))+
 scale_fill_manual(values = c("blue","green"),
                   name = "DEATH_EVENT",
                   labels = c("0 (False)", "1 (True)")) +
 labs(x = "sex ") +
 theme_minimal(base_size = 12) +
 geom_label(stat = "count", aes(label = ..count..), position = position_stack(vjust = 0.5),
            size = 5, show.legend = FALSE)
plot(plot5)
```

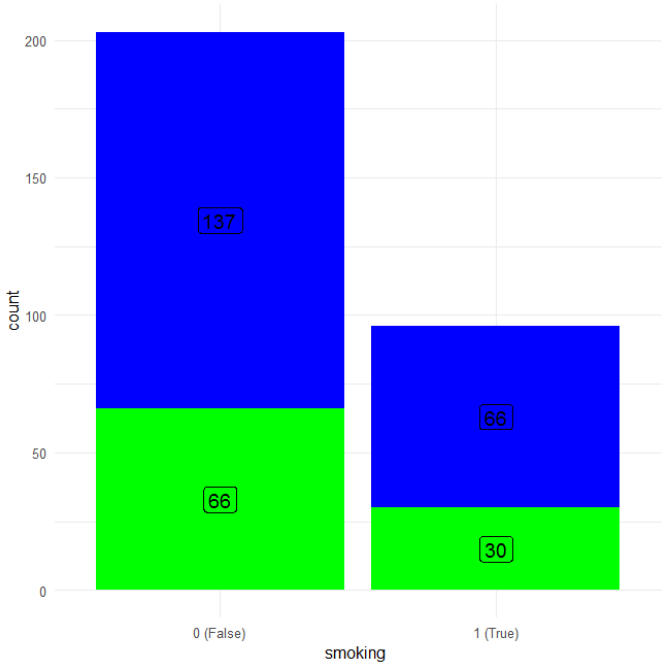


Figure 6 Plot 5- Smoking v/s Death_Event

```
plot6= ggplot(df, aes(x = DEATH_EVENT, fill =
DEATH_EVENT)) +

 geom_bar(stat = "count", position = "stack", show.legend =
FALSE) +

 scale_x_discrete(labels  = c("0 (False)", "1 (True)"))+

 scale_fill_manual(values = c("blue","green"),

                   name = "DEATH_EVENT",

                   labels = c("0 (False)", "1 (True)")) +

 labs(x = "DEATH_EVENT ") +

 theme_minimal(base_size = 12) +

 geom_label(stat = "count", aes(label = ..count..), position =
position_stack(vjust = 0.5),

           size = 5, show.legend = FALSE)

plot(plot6)
```

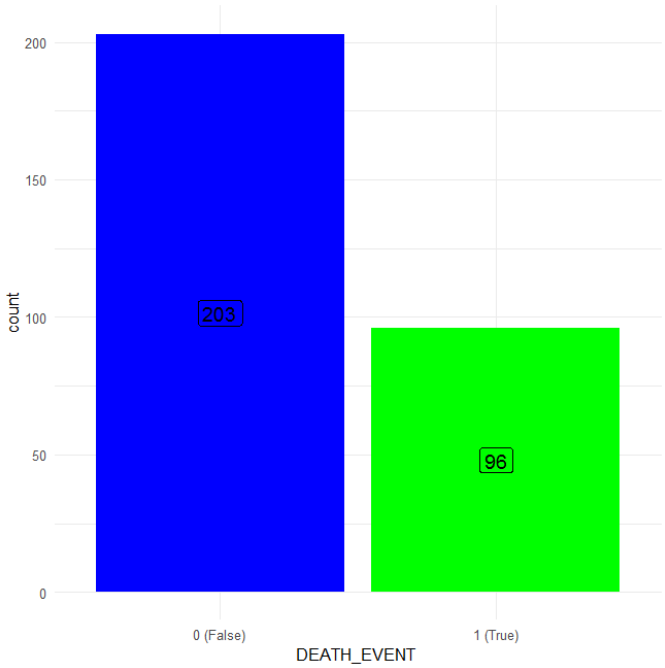


Figure 7- Plot 6- Death event

```
#### perform Shapiro-Wilk's test for checking the normality of dataset:

shapiro.test(data1$age)

shapiro.test(data1$anaemia)

shapiro.test(data1$creatinine_phosphokinase)

shapiro.test(data1$diabetes)

shapiro.test(data1$ejection_fraction)

shapiro.test(data1$high_blood_pressure)

shapiro.test(data1$platelets)

shapiro.test(data1$serum_creatinine)

shapiro.test(data1$serum_sodium)

shapiro.test(data1$sex)

shapiro.test(data1$smoking)

shapiro.test(data1$time)
```

```
> shapiro.test(data1$age)
Error in shapiro.test(data1$age) : is.numeric(x) is not TRUE
data:  data1$anaemia
W = 0.62961, p-value < 2.2e-16

data:  data1$creatinine_phosphokinase
W = 0.51426, p-value < 2.2e-16

data:  data1$diabetes
W = 0.62665, p-value < 2.2e-16

> shapiro.test(data1$ejection_fraction)
Error in shapiro.test(data1$ejection_fraction) :   is.numeric(x) is not TRUE

data:  data1$high_blood_pressure
W = 0.60343, p-value < 2.2e-16

data:  data1$platelets
W = 0.91151, p-value = 2.883e-12

> shapiro.test(data1$serum_creatinine)
Error in shapiro.test(data1$serum_creatinine) : is.numeric(x) is not TRUE

data:  data1$serum_sodium
W = 0.93903, p-value = 9.215e-10

> shapiro.test(data1$sex)

data:  data1$sex
W = 0.60343, p-value < 2.2e-16
data:  data1$smoking
W = 0.58814, p-value < 2.2e-16

> shapiro.test(data1$time)
Error in shapiro.test(data1$time) : is.numeric(x) is not TRUE
```

```
#### Perform two-sample t-tests on predictors


t.test(age~anaemia, data=df)
```

Welch Two Sample t-test

data: age by anaemia

t = -1.5241, df = 276.75, p-value = 0.1286

alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0

95 percent confidence interval:

 -4.8354244 0.6152894

sample estimates:

mean in group 0 mean in group 1

59.92353 62.03360

```
> t.test(serum_creatinine ~anaemia, data=df)

Welch Two Sample t-test

ata:  serum_creatinine by anaemia

t = -0.84604, df = 199.48, p-value = 0.3985

alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0

95 percent confidence interval:

 -0.3623722  0.1447817

sample estimates:

mean in group 0 mean in group 1

      1.346941       1.455736

> t.test(time~anaemia, data=df)

Welch Two Sample t-test

data:  time by anaemia

t = 2.4807, df = 283, p-value = 0.01369

alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0

95 percent confidence interval:

  4.568916 39.678508

sample estimates:

mean in group 0 mean in group 1

      139.8059       117.6822

t.test(platelets~anaemia, data=df)

Welch Two Sample t-test

data:  platelets by anaemia

t = 0.76896, df = 290.75, p-value = 0.4425

alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0

95 percent confidence interval:

 -13461.83  30725.88

sample estimates:

mean in group 0 mean in group 1

      267082.2       258450.2
```

```
### Creating training and test set

set.seed(123)

indx=sample(1:nn,0.8*nn)

traindata=data1[indx,]

testdata=data1[-indx,]

#### Fitting full logistic regression (LR) model with all features

fullmod=glm(as.factor(DEATH_EVENT)~as.factor(anaemia)
+creatinine_phosphokinase+age+as.factor(diabetes)+ejection_fraction+as.factor(high_blood_pressure)+platel
ets+serum_creatinine+serum_sodium+as.factor(sex)+as.factor(smoking)+time,data=traindata,family="binom
ial")

summary(fullmod)
```

```
Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.4646  -0.4676  -0.1527   0.3352   2.2099

Coefficients:
                                  Estimate    Std. Error   z value   Pr(>|z|)
(Intercept)                      1.169e+01   7.018e+00     1.666    0.09566 .
as.factor(anaemia)1              6.993e-02   4.300e-01     0.163    0.87080
creatinine_phosphokinase         2.160e-04   2.115e-04     1.021    0.30702
age                              5.643e-02   1.978e-02     2.852    0.00434 **
as.factor(diabetes)1             3.590e-01   4.274e-01     0.840    0.40100
ejection_fraction               -8.354e-02   1.999e-02    -4.180    2.91e-05 ***
as.factor(high_blood_pressure)1  6.338e-02   4.239e-01     0.150    0.88116
platelets                       -1.638e-06   2.230e-06    -0.734    0.46273
serum_creatinine                 8.573e-01   2.154e-01     3.981    6.87e-05 ***
serum_sodium                    -7.818e-02   4.851e-02    -1.612    0.10706
as.factor(sex)1                 -4.638e-01   4.953e-01    -0.936    0.34908
as.factor(smoking)1             -1.971e-01   4.933e-01    -0.400    0.68944
time                            -2.723e-02   4.231e-03    -6.437    1.22e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 301.89  on 238  degrees of freedom
Residual deviance: 152.08  on 226  degrees of freedom
AIC: 178.08

Number of Fisher Scoring iterations: 6
```

Comment: The Null and alternative hypotheses are as follows:

H0:β=0; HA:β≠0

Any predictor variable for which we obtain the p-value>0.05, signifies that we fail to reject the null hypothesis and that this variable is insignificant for the model.

Comment:

1) From the above model we see that factors Age, ejection_fraction, Serum_creatinine & time are the important factor in prediction of model.

2) When residual deviance is low, the reliability and fit of the model is high. This means that the saturated and fitted models will be as close as possible. Here residual deviance is 152.08 on 226 DOF with the AIC value of 178.08 after 6 iterations.

```
### Selecting features for fitting reduced logistic regression model

library(MASS)

step=stepAIC(fullmod)

mod2=glm(as.factor(DEATH_EVENT)~age+ejection_fraction+serum_creatinine+time, family = "binomial",

 data = traindata)

summary(mod2)
```

```
> step=stepAIC(fullmod)

Step:  AIC=167.93
as.factor(DEATH_EVENT) ~ age + ejection_fraction + serum_creatinine +  serum_sodium + time

                          Df        Deviance   AIC
<none>                    1         55.93      167.93
- serum_sodium            1         158.88     168.88
- age                     1         163.95     173.95
- serum_creatinine        1         175.25     185.25
- ejection_fraction       1         177.23     187.23
- time                    1         236.55     246.55
```

```
> mod2=glm(as.factor(DEATH_EVENT)~age+ejection_fraction+serum_creatinine+time, family = "binomial",
+        data = traindata)
> summary(mod2)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.3610  -0.5327  -0.1588   0.3819   2.4680

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       0.589849   1.207625   0.488  0.62524
age               0.052614   0.018080   2.910  0.00361 **
ejection_fraction -0.081188   0.018681  -4.346 1.39e-05 ***
serum_creatinine   0.898479   0.208153   4.316 1.59e-05 ***
time              -0.026530   0.004043  -6.562 5.33e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 301.89  on 238  degrees of freedom
Residual deviance: 158.88  on 234  degrees of freedom
AIC: 168.88
```

Number of Fisher Scoring iterations: 6

predicting success probabilities using the LR model

```
testdata_new=testdata[,c(1,5,8,12)]

pred_prob=predict(mod2,testdata_new,type="response")

hist(pred_prob)
```

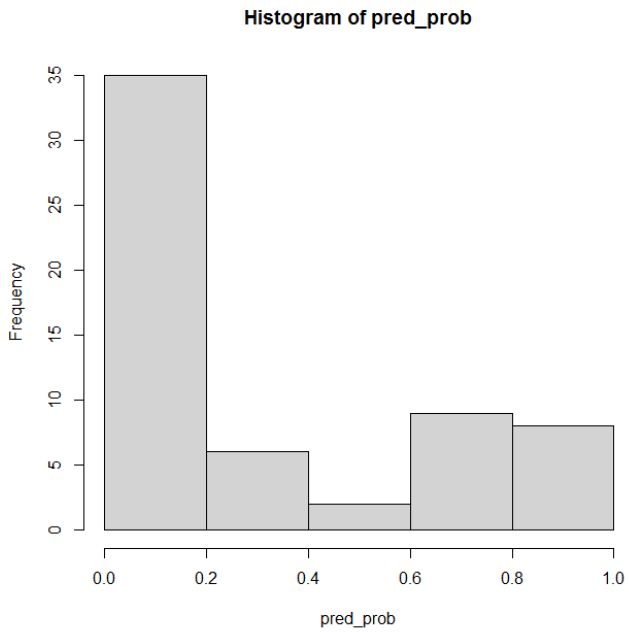


Figure 8 Histogram for Probability prediction

predicting success probability for an individual

```
library(stats)

library(datasets)

sampletest=data.frame(t(c(50,50,2.0,60)))

colnames(sampletest)=c("age","ejection_fraction","serum_
creatinine","time")

predict(mod2,sampletest,type="response")

      1
0.3466664
```

Plotting ROC

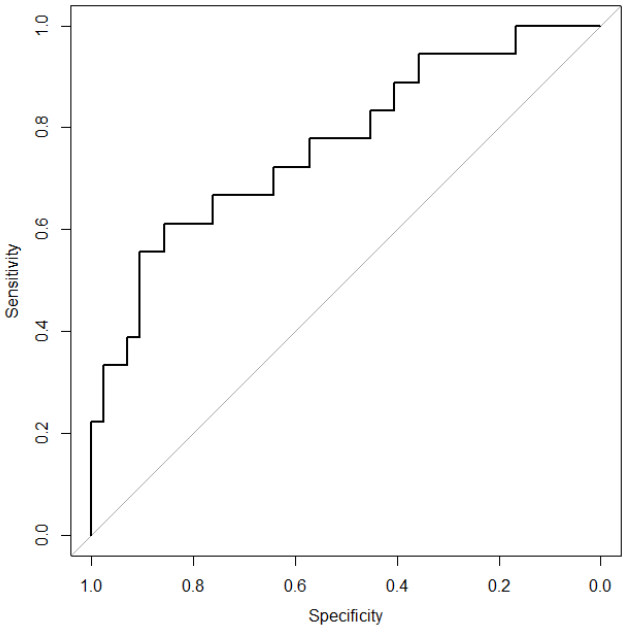
```
library(pROC)

roc1=roc(testdata[,13],pred_prob,plot=TRUE,legacy.axes=TRUE)

plot(roc1)

roc1$auc
```

Area under the curve: 0.7672



Comment: Based on the Area under curve we will restrict our sensitivity value to 0.76

Using ROC in deciding threshold
```
library(lattice)
library(ggplot2)
thres=data.frame(sen=roc1$sensitivities, spec=roc1$specificities,thresholds=roc1$thresholds)
thres[thres$sen>0.70&thres$spec>0.4,]
pred_Y=ifelse(pred_prob > 0.093,1,0)
```

	sen	spec	thresholds
20	0.8888889	0.4047619	0.04705110
21	0.8333333	0.4047619	0.04759412
22	0.8333333	0.4285714	0.04972988
23	0.8333333	0.4523810	0.05211152
24	0.7777778	0.4523810	0.05914300
25	0.7777778	0.4761905	0.06913301
26	0.7777778	0.5000000	0.07309830
27	0.7777778	0.5238095	0.07930653
28	**0.7777778**	**0.5476190**	**0.09351073**
29	0.7777778	0.5714286	0.10219457
30	0.7222222	0.5714286	0.10747605
31	0.7222222	0.5952381	0.11310991
32	0.7222222	0.6190476	0.11495553
33	0.7222222	0.6428571	0.12646671

Comment: So the threshold value that we can consider is 0.093 from the above prediction range with a sensitivity of 0.77.

```
confusionMatrix(as.factor(testdata[,13]), as.factor(pred_Y))
```

Confusion Matrix and Statistics

	Reference	
Prediction	0	1
0	23	19
1	4	14

Accuracy	**:**	**0.6167**
95% CI	:	(0.4821, 0.7393)
No Information Rate	:	0.55
P-Value [Acc > NIR]	:	0.182141
Kappa	:	0.2628
Mcnemar's Test P-Value	:	0.003509
Sensitivity	:	0.8519
Specificity	:	0.4242
Pos Pred Value	:	0.5476
Neg Pred Value	:	0.7778
Prevalence	:	0.4500
Detection Rate	:	0.3833
Detection Prevalence	:	0.7000
Balanced Accuracy	:	0.6380
'Positive' Class	:	0

Comment:

- **So by using a thresh of 0.093 we are getting the accuracy o f model to be 61.67% (62%)**
- **From the confusion matrix it can be observed that there are 19 patients who are wrongly classified by the model which will predict the death event of the patients.**
- **The sensitivity of the model is fairly high i.e 85% which means model will not allow wrong patient to be classed as predicting death.**
- **The specivity of the model is 42% which mean model has low chance, it will classify the patient correctly.**
- **Positive predictive value of the model is ~55% which of ok for the model.**

```
#### Random Forest

set.seed(0)

library(ranger)

df_new2=df

library(randomForest)

df_new2$age=(df_new2$age)

df_new2$ejection_fraction=(df_new2$ejection_fraction)

df_new2$serum_creatinine=(df_new2$serum_creatinine)

df_new2$time=(df_new2$time)

df_new2$DEATH_EVENT=as.factor(df_new2$DEATH_EVENT)

output.forest=ranger(DEATH_EVENT ~ .,data=df_new2, ntree=500,mtry=8)

output.forest
```

output.forest

```
Ranger result

Call:
 ranger(DEATH_EVENT ~ ., data = df_new2, ntree = 500, mtry = 8)

Type:                         Classification
Number of trees:              500
Sample size:                  299
Number of independent variables:  12
Mtry:                         8
Target node size:             1
Variable importance mode:     none
Splitrule:                    gini
OOB prediction error:         17.73 %
```

```
predict(output.forest,testdata[7,-13],type="response")
```

```
Ranger prediction

Type:                         Classification
Sample size:                  1
Number of independent variables:  12
```

```
####opimizing the random forest model to get least error.

output.forest2=ranger(DEATH_EVENT ~ .,data=df_new2, ntree=250,mtry=8)

output.forest2
```

```
Ranger result

Call:
 ranger(DEATH_EVENT ~ ., data = df_new2, ntree = 250, mtry = 8)

Type:                         Classification
Number of trees:              500
Sample size:                  299
Number of independent variables:  12
Mtry:                         8
Target node size:             1
Variable importance mode:     none
Splitrule:                    gini
OOB prediction error:         18.73 %
```

```
output.forest3
Ranger result
```

```
Call:
 ranger(DEATH_EVENT ~ ., data = df_new2, ntree = 650, mtry = 8)

Type:                         Classification
Number of trees:              500
Sample size:                  299
Number of independent variables:  12
Mtry:                         8
Target node size:             1
Variable importance mode:     none
Splitrule:                    gini
OOB prediction error:         17.06 %
```

```
output.forest4=ranger(DEATH_EVENT ~ .,data=df_new2, ntree=650,mtry=5)
```

output.forest4
Ranger result

```
Call:
 ranger(DEATH_EVENT ~ ., data = df_new2, ntree = 650, mtry = 5)

Type:                         Classification
Number of trees:              500
Sample size:                  299
Number of independent variables:  12
Mtry:                         5
Target node size:             1
Variable importance mode:     none
Splitrule:                    gini
OOB prediction error:         16.72 %
```

output.forest5=ranger(DEATH_EVENT ~ .,data=df_new2, ntree=650,mtry=3)
output.forest5
Ranger result

```
Call:
 ranger(DEATH_EVENT ~ ., data = df_new2, ntree = 650, mtry = 3)

Type:                         Classification
Number of trees:              500
Sample size:                  299
Number of independent variables:  12
Mtry:                         3
Target node size:             1
Variable importance mode:     none
Splitrule:                    gini
OOB prediction error:         15.72 %
```

Conclusion:

- **After fitting both logistic regression and random forest model we find that random forest model performs better .**
- **The accuracy obtained by Logistic regression model was ~ 62 with a sensitivity and specivity level of 85 &42 % respectively. The decision tree model is able to predict the with a 85%.**
- **The out of bag (OOB) error is observed to be around 15.72%. The OOB error rate is low; therefore, using a random forest is far more accurate than using logistic regression to classify observations into one of the two categories of the response variable.**
- **The hyperparameter value- number of trees are optimized from 500 to 650 and Ntry from 8 to 3 to get the least OBB values from 17.73 to 15.73%.**