College Admission

```
#Loading dataset

CA <- read.csv("C:/Users/Harshada/Data/College_admission.csv")
View(CA)

#Check if any missing value
result.mean<-mean(CA)
print(result.mean)

#Second way
is.na(CA)

#Finding structure of dataset

str(CA)

#Factoring
f<-factor(c(CA$gpa))
as.numeric(f)
View(CA)

#Ploting
```

```
barplot(table(CA$gre))

barplot(table(CA$gpa))


#Outliers detection using histogram


hist(CA$gre,xlab = "gre",main = "Histogram of gre",breaks = sqrt(nrow(CA)))


#or using ggplot

library(ggplot2)

ggplot(CA) + aes(x=gre) + geom_histogram(bins=30L,fill="red")+ theme_minimal()


#Boxplots also useful to detect potential outliers

boxplot(CA$ses,ylab="ses")

boxplot(CA$admit,ylab="admit")


#To extract exact values of outliers


boxplot.stats(CA$gre)$out


#To extract row number corresponding to outliers


out <- boxplot.stats(CA$gre)$out

out_ind <- which(CA$gre %in% c(out))

out_ind
```

```
#Variables for this outliers

CA[out_ind,]


library(outliers)

SD<-CA[1:20,]


#For lowest outlier


test<-dixon.test(SD$gre)

test


#for Highest outlier


test<-dixon.test(SD$gre,opposite = TRUE)

test


#Visualization of outliers using boxplot


out <-boxplot.stats(SD$gre)$out

boxplot(SD$gre,ylab="gre")

mtext(paste("Outliers: ",paste(out,collapse = ",")))


#Normality distribution test


shapiro.test(CA$gre)
```

```r
#here p value<0.05 hence data is not normaly distributed


#normalization using scale function

library(caret)

da<-as.data.frame(scale(CA[,2]))

summary(CA$gre)


#Reducing variables

library(olsrr)


model <-lm(admit~ gre + gpa + ses + Gender_Male + Race + rank,data = CA)

ols_step_all_possible(model)


#plot method shows fit criteria for all possible regression methods


model<-lm(admit~ gre + gpa + ses + Gender_Male + Race + rank,data = CA)

k <-ols_step_all_possible(model)

plot(k)


#Best subset regression


#select best subset of predictors such as having largest R2 value or smallest MSE
```

```
model <-lm(admit~ gre + gpa + ses + Gender_Male + Race + rank,data=CA)

ols_step_best_subset(model)



# plot for best subset regression


model<-lm(admit~ gre+ gpa + ses + Gender_Male + Race+ rank,data=CA)

k<-ols_step_best_subset(model)




#Variable selection


#stepwise forward regression


model<- lm(admit~.,data = CA)

ols_step_forward_p(model)


k<-ols_step_forward_p(model)

plot(k)


#Detailed output


ols_step_forward_p(model,details = TRUE)
```

```
#Logistic model

head(CA)
summary(CA)
sapply(CA, sd)

xtabs(~admit +rank,data=CA)

CA$rank<-factor(CA$rank)
CA$rank

CA_logit <-glm(admit~gre + gpa+rank ,data=CA,family = "binomial")
summary(CA_logit)

#Obtain confidence interval
#CIS using profiled log-likelihood

confint(CA_logit)
```

#CIS using standard errors

```
confint.default(CA_logit)
```

#Overall effect of rank using WALD test

```
library(aod)
```

#wald.test(b=coef(CA_logit),sigma=vcov(CA_logit),Terms=4:6)

#Odds ratio only

```
exp(coef(CA_logit))
```

#Odds ratio and 95% CI

```
exp(cbind(OR = coef(CA_logit), confint(CA_logit)))
```

#We will start by calculating the predicted probability of admission at each value of rank, holding gre and gpa at their means. First we create and view the data frame.

```
newdata1 <- with(CA, data.frame(gre = mean(gre), gpa = mean(gpa), rank = factor(1:4)))
newdata1
```

#In the below output we see that the predicted probability of being accepted into a graduate program is 0.52 for students from the highest prestige undergraduate institutions (rank=1), and 0.18 for students from the lowest ranked institutions (rank=4), holding gre and gpa at their means.

```
newdata1$rankP <- predict(CA_logit, newdata = newdata1, type = "response")

newdata1
```

#We can do something very similar to create a table of predicted probabilities varying the value of gre and rank. We are going to plot these, so we will create 100 values of gre between 200 and 800, at each value of rank (i.e., 1, 2, 3, and 4).

```
newdata2 <- with(CA, data.frame(gre = rep(seq(from = 200, to = 800, length.out = 100),
                                4), gpa = mean(gpa), rank = factor(rep(1:4, each = 100))))

newdata2
```

#The code to generate the predicted probabilities (the first line below) is the same as before, except we are also going to ask for standard errors so we can plot a confidence interval. We get the estimates on the link scale and back transform both the predicted values and confidence limits into probabilities.

```
newdata3 <- cbind(newdata2, predict(CA_logit, newdata = newdata2, type = "link",
                   se = TRUE))

newdata3 <- within(newdata3, {
```

```
  PredictedProb <- plogis(fit)

  LL <- plogis(fit - (1.96 * se.fit))

  UL <- plogis(fit + (1.96 * se.fit))

})
```

```
## view first few rows of final dataset

head(newdata3)
```

#It can also be helpful to use graphs of predicted probabilities to understand and/or present the model. We will use the ggplot2 package for graphing. Below we make a plot with the predicted probabilities, and 95% confidence intervals.

```
library(ggplot2)

ggplot(newdata3, aes(x = gre, y = PredictedProb)) + geom_ribbon(aes(ymin = LL,
    ymax = UL, fill = rank), alpha = 0.2) + geom_line(aes(colour = rank),
    size = 1)
```

#To find the difference in deviance for the two models (i.e., the test statistic) we can use the command:

```
with(CA_logit, null.deviance - deviance)
```

#The degrees of freedom for the difference between the two models is equal to the number of predictor variables in the mode, and can be obtained using:

```
with(CA_logit, df.null - df.residual)
```

P value can be obtained using

with(CA_logit, pchisq(null.deviance - deviance, df.null - df.residual, lower.tail = FALSE))

#The chi-square of 41.46 with 5 degrees of freedom and an associated p-value of less than 0.001 tells us that our model as a whole fits significantly better than an empty model. This is sometimes called a likelihood ratio test (the deviance residual is -2*log likelihood). To see the model's log likelihood,

#7.58e-08=7.58*10^-8=0.0000000758

#Models log liklihood

logLik(CA_logit)

checking accuracy of model

#Plot ROC

library(ROCR)

library(Metrics)

library(caret)

split<-createDataPartition(y=CA$admit,p=0.6,list = FALSE)

new_train <- CA[split]

new_test <- CA[split]

log_predict<-predict(CA_logit,newdata=CA,type="response")

```
log_predict<-ifelse(log_predict>0.5,1,0)

pr<-prediction(log_predict,CA$admit)

perf<-performance(pr,measure = "tpr",x.measure = "fpr")

plot(perf)

auc(CA$admit,log_predict)


# Our AUC score is 0.5833. In roc plot we always try to move up and top left

# corner .from this plot we can say the model is predicting more negative values incorrectly

# to move up increase our threshold value to 0.6 and check the performance.



#Confusion matrix

confusionMatrix(table(predict(CA_logit,type="response")>=0.5,CA$admit==1))



#Plot confusion matrix

ctable<-as.table(matrix(c(254,97,19,30),nrow = 2,byrow = TRUE))

fourfoldplot(ctable,color = c("#CC6666","#99CC99"),conf.level = 0,margin = 1,main="Confusion Matrix")



#Decision Tree model


library(party)
```

```
# Create the input data frame.
input.dat <- CA[c(2:200),]

# Give the chart file a name.
png(file = "decision_tree.png")

# Create the tree.
output.tree <- ctree(
  admit ~ gre+ ses +Race+rank,
  data = input.dat)

# Plot the tree.
plot(output.tree)

# Save the file.
dev.off()


#Decision tree plot

library(rpart)
library(rpart.plot)
fit<-rpart(admit~.,data=CA,method='class')
rpart.plot(fit,extra=106)
```

```
#Confusion matrix

pu<-predict(fit,CA,type='class')

tm<-table(CA$admit,pu)

tm

#Accuracy of decision tree model


AC<-sum(diag(tm))/sum(tm)

paste('Accuracy for test',AC)


library(party)

library(randomForest)



#RandomForest model


# Create the forest.

output.forest <- randomForest(admit~ gre+ ses + Race + rank,

                   data = input.dat)


# View the forest results.

print(output.forest)


# Importance of each predictor.

importance(output.forest,type=2)

print(importance(output.forest,type = 2))
```

```
plot(output.forest)



#Categorize gre attribute into Low,Medium and High class



library(dplyr)


CA<-CA%>%
  mutate(gre_class=case_when(
    gre<400~"Low",
    gre>440&gre<580~"Medium",
    gre>580~"High"
  ))
CA


CA$gre_class=factor(CA$gre_class,levels = c("Low","Medium","High"))
XT=xtabs(~gre_class+gre,data = CA)
XT


#Count levels in gre
nlevels(CA$gre_class)


#Count no of elements in each levels in gre
count(CA,CA$gre_class)
```

#Random forest gives better result out of all these models.

#Also Rank and grad are most influencing factor which affects on admission process.

…………………………………………………......OUTPUT………………………………………….............

APC: Amemiya Prediction Criteria

```
> model<-lm(admit~ gre+ gpa + ses + Gender_Male + Race+ rank,data=CA)

> k<-ols_step_best_subset(model)

> plot(k)

> model<- lm(admit~.,data = CA)

> ols_step_forward_p(model)


                  Selection Summary
------------------------------------------------------------------------
      Variable                Adj.
Step    Entered    R-Square    R-Square     C(p)       AIC        RMSE
------------------------------------------------------------------------
   1    rank        0.0588      0.0564    15.9995    505.1975    0.4527
   2    gpa         0.0859      0.0813     6.1469    495.5214    0.4467
   3    gre         0.0960      0.0892     3.7151    493.0665    0.4448
```

```
  4    Race         0.0990     0.0899     4.4111    493.7462    0.4447
----------------------------------------------------------------------
> k<-ols_step_forward_p(model)
> plot(k)
> ols_step_forward_p(model,details = TRUE)
Forward Selection Method
---------------------------

Candidate Terms:

1. gre
2. gpa
3. ses
4. Gender_Male
5. Race
6. rank

We are selecting variables based on p value...


Forward Selection: Step 1
```

```
+ rank

                    Model Summary
---------------------------------------------------------------
R                     0.243      RMSE              0.453
R-Squared             0.059      Coef. Var       142.596
Adj. R-Squared        0.056      MSE               0.205
Pred R-Squared        0.049      MAE               0.408
---------------------------------------------------------------
 RMSE: Root Mean Square Error
 MSE: Mean Square Error
 MAE: Mean Absolute Error

                         ANOVA
--------------------------------------------------------------------
            Sum of
           Squares       DF    Mean Square      F        Sig.
--------------------------------------------------------------------
Regression      5.098       1        5.098     24.87    0.0000
Residual       81.580     398        0.205
Total          86.678     399
--------------------------------------------------------------------
```

Parameter Estimates

model	Beta	Std. Error	Std. Beta	t	Sig	lower	upper
(Intercept)	0.615	0.064		9.640	0.000	0.490	0.740
rank	-0.120	0.024	-0.243	-4.987	0.000	-0.167	-0.072

Forward Selection: Step 2

+ gpa

Model Summary

R	0.293	RMSE	0.447
R-Squared	0.086	Coef. Var	140.706
Adj. R-Squared	0.081	MSE	0.200
Pred R-Squared	0.072	MAE	0.397

RMSE: Root Mean Square Error

MSE: Mean Square Error

MAE: Mean Absolute Error

ANOVA

	Sum of Squares	DF	Mean Square	F	Sig.
Regression	7.445	2	3.722	18.651	0.0000
Residual	79.233	397	0.200		
Total	86.677	399			

Parameter Estimates

model	Beta	Std. Error	Std. Beta	t	Sig	lower	upper
(Intercept)	-0.081	0.212		-0.381	0.703	-0.499	0.337
rank	-0.115	0.024	-0.233	-4.849	0.000	-0.162	-0.068
gpa	0.202	0.059	0.165	3.429	0.001	0.086	0.318

Forward Selection: Step 3

+ gre

Model Summary

R	0.310	RMSE	0.445
R-Squared	0.096	Coef. Var	140.102
Adj. R-Squared	0.089	MSE	0.198
Pred R-Squared	0.078	MAE	0.393

RMSE: Root Mean Square Error

MSE: Mean Square Error

MAE: Mean Absolute Error

ANOVA

	Sum of Squares	DF	Mean Square	F	Sig.

Regression	8.322	3	2.774	14.02	0.0000
Residual	78.355	396	0.198		
Total	86.678	399			

Parameter Estimates

model	Beta	Std. Error	Std. Beta	t	Sig	lower	upper
(Intercept)	-0.182	0.217		-0.841	0.401	-0.609	0.244
rank	-0.110	0.024	-0.222	-4.608	0.000	-0.156	-0.063
gpa	0.151	0.063	0.123	2.383	0.018	0.026	0.276
gre	0.000	0.000	0.110	2.106	0.036	0.000	0.001

Forward Selection: Step 4

+ Race

Model Summary

R	0.315	RMSE	0.445
R-Squared	0.099	Coef. Var	140.047
Adj. R-Squared	0.090	MSE	0.198
Pred R-Squared	0.076	MAE	0.392

RMSE: Root Mean Square Error

MSE: Mean Square Error

MAE: Mean Absolute Error

ANOVA

	Sum of Squares	DF	Mean Square	F	Sig.
Regression	8.580	4	2.145	10.85	0.0000
Residual	78.097	395	0.198		
Total	86.678	399			

Parameter Estimates

model	Beta	Std. Error	Std. Beta	t	Sig	lower	upper
(Intercept)	-0.134	0.221		-0.606	0.545	-0.568	0.300
rank	-0.109	0.024	-0.220	-4.571	0.000	-0.155	-0.062
gpa	0.157	0.064	0.128	2.471	0.014	0.032	0.282
gre	0.000	0.000	0.105	2.020	0.044	0.000	0.001
Race	-0.031	0.027	-0.055	-1.143	0.254	-0.084	0.022

No more variables to be added.

Variables Entered:

+ rank

+ gpa

+ gre

+ Race

Final Model Output

Model Summary

R	0.315	RMSE	0.445
R-Squared	0.099	Coef. Var	140.047
Adj. R-Squared	0.090	MSE	0.198
Pred R-Squared	0.076	MAE	0.392

RMSE: Root Mean Square Error

MSE: Mean Square Error

MAE: Mean Absolute Error

ANOVA

	Sum of Squares	DF	Mean Square	F	Sig.
Regression	8.580	4	2.145	10.85	0.0000
Residual	78.097	395	0.198		
Total	86.678	399			

Parameter Estimates

model	Beta	Std. Error	Std. Beta	t	Sig	lower	upper
(Intercept)	-0.134	0.221		-0.606	0.545	-0.568	0.300
rank	-0.109	0.024	-0.220	-4.571	0.000	-0.155	-0.062
gpa	0.157	0.064	0.128	2.471	0.014	0.032	0.282
gre	0.000	0.000	0.105	2.020	0.044	0.000	0.001
Race	-0.031	0.027	-0.055	-1.143	0.254	-0.084	0.022

Selection Summary

Step	Variable Entered	R-Square	Adj. R-Square	C(p)	AIC	RMSE
1	rank	0.0588	0.0564	15.9995	505.1975	0.4527
2	gpa	0.0859	0.0813	6.1469	495.5214	0.4467
3	gre	0.0960	0.0892	3.7151	493.0665	0.4448
4	Race	0.0990	0.0899	4.4111	493.7462	0.4447

```
----------------------------------------------------------------------

> head(CA)
  admit gre  gpa ses Gender_Male Race rank
1     0 380 3.61   1           0    3    3
2     1 660 3.67   2           0    2    3
3     1 800 4.00   2           0    2    1
4     1 640 3.19   1           1    2    4
5     0 520 2.93   3           1    2    4
6     1 760 3.00   2           1    1    2
> summary(CA)
     admit             gre             gpa             ses         Gender_Male
 Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000   Min.   :0.000
 1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:1.000   1st Qu.:0.000
 Median :0.0000   Median :580.0   Median :3.395   Median :2.000   Median :0.000
 Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :1.992   Mean   :0.475
 3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000   3rd Qu.:1.000
 Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :3.000   Max.   :1.000
      Race            rank
 Min.   :1.000   Min.   :1.000
 1st Qu.:1.000   1st Qu.:2.000
 Median :2.000   Median :2.000
 Mean   :1.962   Mean   :2.485
```

```
 3rd Qu.:3.000   3rd Qu.:3.000

 Max.   :3.000   Max.   :4.000

> sapply(CA, sd)

     admit        gre        gpa        ses Gender_Male       Race       rank

  0.4660867 115.5165364   0.3805668   0.8087515   0.5000000   0.8232789
0.9444602

> xtabs(~admit +rank,data=CA)

     rank

admit  1  2  3  4

    0 28 97 93 55

    1 33 54 28 12

> CA$rank<-factor(CA$rank)

> CA$rank

  [1] 3 3 1 4 4 2 1 2 3 2 4 1 1 2 1 3 4 3 2 1 3 2 4 4 2 1 1 4 2 1 4 3 3 3 1 2 1 3 2

 [40] 3 2 2 2 3 2 3 2 4 4 3 3 4 4 2 3 3 3 3 2 4 2 4 3 3 3 2 4 1 1 1 3 4 4 2 4 3 3 3

 [79] 1 1 4 2 2 4 3 2 2 2 1 2 2 1 2 2 2 2 4 2 2 3 3 3 4 3 2 2 1 2 3 2 4 4 3 1 3 3 2

[118] 2 1 3 2 2 3 3 3 4 1 4 2 4 2 2 2 3 2 3 4 3 2 1 2 4 4 3 4 3 2 3 1 1 1 2 2 3 3 4

[157] 2 1 2 3 2 2 2 2 2 1 4 3 3 3 3 3 3 2 4 2 2 3 3 3 3 4 2 2 4 2 3 2 2 2 2 3 3 4 2

[196] 2 3 4 3 4 3 2 1 4 1 3 1 1 3 2 4 2 2 3 2 3 1 1 1 2 3 3 1 3 2 3 2 4 2 2 4 3 2 3

[235] 1 2 2 2 4 3 2 1 3 2 1 3 2 2 3 3 4 4 2 4 4 3 2 3 2 2 2 2 3 3 3 3 4 3 2 3 2 3 2

[274] 1 2 2 3 1 4 2 2 3 4 4 2 4 1 4 4 4 2 2 2 1 1 3 1 2 2 3 2 3 2 2 3 4 1 2 2 3 3 2

[313] 3 4 4 2 2 4 4 1 3 2 4 2 3 1 2 2 2 4 3 3 1 3 3 1 3 4 1 3 4 3 4 2 3 3 2 2 2 2 2
```

```
[352] 3 3 2 2 1 2 1 3 3 1 1 2 2 1 3 3 3 1 2 2 3 1 1 2 4 2 2 3 2 2 2 2 1 2 1 2 2 2 2

[391] 2 2 3 2 3 2 3 2 2 3

Levels: 1 2 3 4

> CA_logit <-glm(admit~gre + gpa+rank ,data=CA,family = "binomial")

> summary(CA_logit)


Call:

glm(formula = admit ~ gre + gpa + rank, family = "binomial",

   data = CA)


Deviance Residuals:

   Min      1Q   Median      3Q     Max

-1.6268  -0.8662  -0.6388   1.1490   2.0790


Coefficients:

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

(Intercept) -3.989979   1.139951  -3.500 0.000465 ***

gre          0.002264   0.001094   2.070 0.038465 *

gpa          0.804038   0.331819   2.423 0.015388 *

rank2       -0.675443   0.316490  -2.134 0.032829 *

rank3       -1.340204   0.345306  -3.881 0.000104 ***

rank4       -1.551464   0.417832  -3.713 0.000205 ***
```

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

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 499.98  on 399  degrees of freedom
Residual deviance: 458.52  on 394  degrees of freedom
AIC: 470.52

Number of Fisher Scoring iterations: 4

> confint(CA_logit)
Waiting for profiling to be done...
                    2.5 %       97.5 %
(Intercept) -6.2716202334 -1.792547080
gre          0.0001375921  0.004435874
gpa          0.1602959439  1.464142727
rank2       -1.3008888002 -0.056745722
rank3       -2.0276713127 -0.670372346
rank4       -2.4000265384 -0.753542605
> confint.default(CA_logit)
                    2.5 %       97.5 %
```

```
(Intercept) -6.2242418514 -1.755716295
gre          0.0001202298  0.004408622
gpa          0.1536836760  1.454391423
rank2       -1.2957512650 -0.055134591
rank3       -2.0169920597 -0.663415773
rank4       -2.3703986294 -0.732528724
> #Overall effect of rank using WALD test
> library(aod)
> exp(coef(CA_logit))
(Intercept)        gre        gpa      rank2      rank3      rank4
  0.0185001   1.0022670   2.2345448   0.5089310   0.2617923   0.2119375
> #Odds ratio and 95% CI
> exp(cbind(OR = coef(CA_logit), confint(CA_logit)))
Waiting for profiling to be done...
                   OR       2.5 %    97.5 %
(Intercept) 0.0185001 0.001889165 0.1665354
gre         1.0022670 1.000137602 1.0044457
gpa         2.2345448 1.173858216 4.3238349
rank2       0.5089310 0.272289674 0.9448343
rank3       0.2617923 0.131641717 0.5115181
rank4       0.2119375 0.090715546 0.4706961
```

```
> newdata1 <- with(CA, data.frame(gre = mean(gre), gpa = mean(gpa), rank =
factor(1:4)))

> newdata1

    gre    gpa rank

1 587.7 3.3899    1

2 587.7 3.3899    2

3 587.7 3.3899    3

4 587.7 3.3899    4

> newdata1$rankP <- predict(CA_logit, newdata = newdata1, type = "response")

> newdata1

    gre    gpa rank     rankP

1 587.7 3.3899    1 0.5166016

2 587.7 3.3899    2 0.3522846

3 587.7 3.3899    3 0.2186120

4 587.7 3.3899    4 0.1846684

> newdata2 <- with(CA, data.frame(gre = rep(seq(from = 200, to = 800, length.out
= 100),

+                                          4), gpa = mean(gpa), rank = factor(rep(1:4, each =
100))))

> newdata2

         gre    gpa rank

1   200.0000 3.3899    1

2   206.0606 3.3899    1
```

3	212.1212	3.3899	1
4	218.1818	3.3899	1
5	224.2424	3.3899	1
6	230.3030	3.3899	1
7	236.3636	3.3899	1
8	242.4242	3.3899	1
9	248.4848	3.3899	1
10	254.5455	3.3899	1
11	260.6061	3.3899	1
12	266.6667	3.3899	1
13	272.7273	3.3899	1
14	278.7879	3.3899	1
15	284.8485	3.3899	1
16	290.9091	3.3899	1
17	296.9697	3.3899	1
18	303.0303	3.3899	1
19	309.0909	3.3899	1
20	315.1515	3.3899	1
21	321.2121	3.3899	1
22	327.2727	3.3899	1
23	333.3333	3.3899	1
24	339.3939	3.3899	1

25	345.4545	3.3899	1
26	351.5152	3.3899	1
27	357.5758	3.3899	1
28	363.6364	3.3899	1
29	369.6970	3.3899	1
30	375.7576	3.3899	1
31	381.8182	3.3899	1
32	387.8788	3.3899	1
33	393.9394	3.3899	1
34	400.0000	3.3899	1
35	406.0606	3.3899	1
36	412.1212	3.3899	1
37	418.1818	3.3899	1
38	424.2424	3.3899	1
39	430.3030	3.3899	1
40	436.3636	3.3899	1
41	442.4242	3.3899	1
42	448.4848	3.3899	1
43	454.5455	3.3899	1
44	460.6061	3.3899	1
45	466.6667	3.3899	1
46	472.7273	3.3899	1

47	478.7879	3.3899	1
48	484.8485	3.3899	1
49	490.9091	3.3899	1
50	496.9697	3.3899	1
51	503.0303	3.3899	1
52	509.0909	3.3899	1
53	515.1515	3.3899	1
54	521.2121	3.3899	1
55	527.2727	3.3899	1
56	533.3333	3.3899	1
57	539.3939	3.3899	1
58	545.4545	3.3899	1
59	551.5152	3.3899	1
60	557.5758	3.3899	1
61	563.6364	3.3899	1
62	569.6970	3.3899	1
63	575.7576	3.3899	1
64	581.8182	3.3899	1
65	587.8788	3.3899	1
66	593.9394	3.3899	1
67	600.0000	3.3899	1
68	606.0606	3.3899	1

69	612.1212	3.3899	1
70	618.1818	3.3899	1
71	624.2424	3.3899	1
72	630.3030	3.3899	1
73	636.3636	3.3899	1
74	642.4242	3.3899	1
75	648.4848	3.3899	1
76	654.5455	3.3899	1
77	660.6061	3.3899	1
78	666.6667	3.3899	1
79	672.7273	3.3899	1
80	678.7879	3.3899	1
81	684.8485	3.3899	1
82	690.9091	3.3899	1
83	696.9697	3.3899	1
84	703.0303	3.3899	1
85	709.0909	3.3899	1
86	715.1515	3.3899	1
87	721.2121	3.3899	1
88	727.2727	3.3899	1
89	733.3333	3.3899	1
90	739.3939	3.3899	1

91	745.4545	3.3899	1
92	751.5152	3.3899	1
93	757.5758	3.3899	1
94	763.6364	3.3899	1
95	769.6970	3.3899	1
96	775.7576	3.3899	1
97	781.8182	3.3899	1
98	787.8788	3.3899	1
99	793.9394	3.3899	1
100	800.0000	3.3899	1
101	200.0000	3.3899	2
102	206.0606	3.3899	2
103	212.1212	3.3899	2
104	218.1818	3.3899	2
105	224.2424	3.3899	2
106	230.3030	3.3899	2
107	236.3636	3.3899	2
108	242.4242	3.3899	2
109	248.4848	3.3899	2
110	254.5455	3.3899	2
111	260.6061	3.3899	2
112	266.6667	3.3899	2

113	272.7273	3.3899	2
114	278.7879	3.3899	2
115	284.8485	3.3899	2
116	290.9091	3.3899	2
117	296.9697	3.3899	2
118	303.0303	3.3899	2
119	309.0909	3.3899	2
120	315.1515	3.3899	2
121	321.2121	3.3899	2
122	327.2727	3.3899	2
123	333.3333	3.3899	2
124	339.3939	3.3899	2
125	345.4545	3.3899	2
126	351.5152	3.3899	2
127	357.5758	3.3899	2
128	363.6364	3.3899	2
129	369.6970	3.3899	2
130	375.7576	3.3899	2
131	381.8182	3.3899	2
132	387.8788	3.3899	2
133	393.9394	3.3899	2
134	400.0000	3.3899	2

135	406.0606	3.3899	2
136	412.1212	3.3899	2
137	418.1818	3.3899	2
138	424.2424	3.3899	2
139	430.3030	3.3899	2
140	436.3636	3.3899	2
141	442.4242	3.3899	2
142	448.4848	3.3899	2
143	454.5455	3.3899	2
144	460.6061	3.3899	2
145	466.6667	3.3899	2
146	472.7273	3.3899	2
147	478.7879	3.3899	2
148	484.8485	3.3899	2
149	490.9091	3.3899	2
150	496.9697	3.3899	2
151	503.0303	3.3899	2
152	509.0909	3.3899	2
153	515.1515	3.3899	2
154	521.2121	3.3899	2
155	527.2727	3.3899	2
156	533.3333	3.3899	2

157	539.3939	3.3899	2
158	545.4545	3.3899	2
159	551.5152	3.3899	2
160	557.5758	3.3899	2
161	563.6364	3.3899	2
162	569.6970	3.3899	2
163	575.7576	3.3899	2
164	581.8182	3.3899	2
165	587.8788	3.3899	2
166	593.9394	3.3899	2
167	600.0000	3.3899	2
168	606.0606	3.3899	2
169	612.1212	3.3899	2
170	618.1818	3.3899	2
171	624.2424	3.3899	2
172	630.3030	3.3899	2
173	636.3636	3.3899	2
174	642.4242	3.3899	2
175	648.4848	3.3899	2
176	654.5455	3.3899	2
177	660.6061	3.3899	2
178	666.6667	3.3899	2

179	672.7273	3.3899	2
180	678.7879	3.3899	2
181	684.8485	3.3899	2
182	690.9091	3.3899	2
183	696.9697	3.3899	2
184	703.0303	3.3899	2
185	709.0909	3.3899	2
186	715.1515	3.3899	2
187	721.2121	3.3899	2
188	727.2727	3.3899	2
189	733.3333	3.3899	2
190	739.3939	3.3899	2
191	745.4545	3.3899	2
192	751.5152	3.3899	2
193	757.5758	3.3899	2
194	763.6364	3.3899	2
195	769.6970	3.3899	2
196	775.7576	3.3899	2
197	781.8182	3.3899	2
198	787.8788	3.3899	2
199	793.9394	3.3899	2
200	800.0000	3.3899	2

201	200.0000	3.3899	3
202	206.0606	3.3899	3
203	212.1212	3.3899	3
204	218.1818	3.3899	3
205	224.2424	3.3899	3
206	230.3030	3.3899	3
207	236.3636	3.3899	3
208	242.4242	3.3899	3
209	248.4848	3.3899	3
210	254.5455	3.3899	3
211	260.6061	3.3899	3
212	266.6667	3.3899	3
213	272.7273	3.3899	3
214	278.7879	3.3899	3
215	284.8485	3.3899	3
216	290.9091	3.3899	3
217	296.9697	3.3899	3
218	303.0303	3.3899	3
219	309.0909	3.3899	3
220	315.1515	3.3899	3
221	321.2121	3.3899	3
222	327.2727	3.3899	3

223	333.3333	3.3899	3
224	339.3939	3.3899	3
225	345.4545	3.3899	3
226	351.5152	3.3899	3
227	357.5758	3.3899	3
228	363.6364	3.3899	3
229	369.6970	3.3899	3
230	375.7576	3.3899	3
231	381.8182	3.3899	3
232	387.8788	3.3899	3
233	393.9394	3.3899	3
234	400.0000	3.3899	3
235	406.0606	3.3899	3
236	412.1212	3.3899	3
237	418.1818	3.3899	3
238	424.2424	3.3899	3
239	430.3030	3.3899	3
240	436.3636	3.3899	3
241	442.4242	3.3899	3
242	448.4848	3.3899	3
243	454.5455	3.3899	3
244	460.6061	3.3899	3

245	466.6667	3.3899	3
246	472.7273	3.3899	3
247	478.7879	3.3899	3
248	484.8485	3.3899	3
249	490.9091	3.3899	3
250	496.9697	3.3899	3
251	503.0303	3.3899	3
252	509.0909	3.3899	3
253	515.1515	3.3899	3
254	521.2121	3.3899	3
255	527.2727	3.3899	3
256	533.3333	3.3899	3
257	539.3939	3.3899	3
258	545.4545	3.3899	3
259	551.5152	3.3899	3
260	557.5758	3.3899	3
261	563.6364	3.3899	3
262	569.6970	3.3899	3
263	575.7576	3.3899	3
264	581.8182	3.3899	3
265	587.8788	3.3899	3
266	593.9394	3.3899	3

267	600.0000	3.3899	3
268	606.0606	3.3899	3
269	612.1212	3.3899	3
270	618.1818	3.3899	3
271	624.2424	3.3899	3
272	630.3030	3.3899	3
273	636.3636	3.3899	3
274	642.4242	3.3899	3
275	648.4848	3.3899	3
276	654.5455	3.3899	3
277	660.6061	3.3899	3
278	666.6667	3.3899	3
279	672.7273	3.3899	3
280	678.7879	3.3899	3
281	684.8485	3.3899	3
282	690.9091	3.3899	3
283	696.9697	3.3899	3
284	703.0303	3.3899	3
285	709.0909	3.3899	3
286	715.1515	3.3899	3
287	721.2121	3.3899	3
288	727.2727	3.3899	3

289	733.3333	3.3899	3
290	739.3939	3.3899	3
291	745.4545	3.3899	3
292	751.5152	3.3899	3
293	757.5758	3.3899	3
294	763.6364	3.3899	3
295	769.6970	3.3899	3
296	775.7576	3.3899	3
297	781.8182	3.3899	3
298	787.8788	3.3899	3
299	793.9394	3.3899	3
300	800.0000	3.3899	3
301	200.0000	3.3899	4
302	206.0606	3.3899	4
303	212.1212	3.3899	4
304	218.1818	3.3899	4
305	224.2424	3.3899	4
306	230.3030	3.3899	4
307	236.3636	3.3899	4
308	242.4242	3.3899	4
309	248.4848	3.3899	4
310	254.5455	3.3899	4

311	260.6061	3.3899	4
312	266.6667	3.3899	4
313	272.7273	3.3899	4
314	278.7879	3.3899	4
315	284.8485	3.3899	4
316	290.9091	3.3899	4
317	296.9697	3.3899	4
318	303.0303	3.3899	4
319	309.0909	3.3899	4
320	315.1515	3.3899	4
321	321.2121	3.3899	4
322	327.2727	3.3899	4
323	333.3333	3.3899	4
324	339.3939	3.3899	4
325	345.4545	3.3899	4
326	351.5152	3.3899	4
327	357.5758	3.3899	4
328	363.6364	3.3899	4
329	369.6970	3.3899	4
330	375.7576	3.3899	4
331	381.8182	3.3899	4
332	387.8788	3.3899	4

```
333 393.9394 3.3899    4
 [ reached 'max' / getOption("max.print") -- omitted 67 rows ]
> newdata3 <- cbind(newdata2, predict(CA_logit, newdata = newdata2, type = "link",
+                                     se = TRUE))
> newdata3 <- within(newdata3, {
+   PredictedProb <- plogis(fit)
+   LL <- plogis(fit - (1.96 * se.fit))
+   UL <- plogis(fit + (1.96 * se.fit))
+ })
> ## view first few rows of final dataset
> head(newdata3)
       gre    gpa rank        fit    se.fit residual.scale        UL        LL
1 200.0000 3.3899    1 -0.8114870 0.5147714              1 0.5492064 0.1393812
2 206.0606 3.3899    1 -0.7977632 0.5090986              1 0.5498513 0.1423880
3 212.1212 3.3899    1 -0.7840394 0.5034491              1 0.5505074 0.1454429
4 218.1818 3.3899    1 -0.7703156 0.4978239              1 0.5511750 0.1485460
5 224.2424 3.3899    1 -0.7565919 0.4922237              1 0.5518545 0.1516973
6 230.3030 3.3899    1 -0.7428681 0.4866494              1 0.5525464 0.1548966
  PredictedProb
1     0.3075737
2     0.3105042
```

```
3    0.3134499

4    0.3164108

5    0.3193867

6    0.3223773
```

> #It can also be helpful to use graphs of predicted probabilities to understand and/or present the model. We will use the ggplot2 package for graphing. Below we make a plot with the predicted probabilities, and 95% confidence intervals.

```
> library(ggplot2)

> ggplot(newdata3, aes(x = gre, y = PredictedProb)) + geom_ribbon(aes(ymin = LL,

+                                         ymax = UL, fill = rank), alpha = 0.2) + geom_line(aes(colour = rank),

+                                                                          size = 1)

> with(CA_logit, null.deviance - deviance)

[1] 41.45903

> with(CA_logit, df.null - df.residual)

[1] 5

> with(CA_logit, pchisq(null.deviance - deviance, df.null - df.residual, lower.tail = FALSE))

[1] 7.578194e-08

> logLik(CA_logit)

'log Lik.' -229.2587 (df=6)

> library(ROCR)

> library(Metrics)
```

```
> library(caret)

> split<-createDataPartition(y=CA$admit,p=0.6,list = FALSE)

> new_train <- CA[split]

> new_test <- CA[split]

> log_predict<-predict(CA_logit,newdata=CA,type="response")

> log_predict<-ifelse(log_predict>0.5,1,0)

> pr<-prediction(log_predict,CA$admit)

> perf<-performance(pr,measure = "tpr",x.measure = "fpr")

> plot(perf)

> auc(CA$admit,log_predict)

[1] 0.5833117

> #Confusion matrix

> confusionMatrix(table(predict(CA_logit,type="response")>=0.5,CA$admit==1))

Confusion Matrix and Statistics



       FALSE TRUE

  FALSE   254   97

  TRUE     19   30



          Accuracy : 0.71

            95% CI : (0.6628, 0.754)
```

No Information Rate : 0.6825

P-Value [Acc > NIR] : 0.1293

Kappa : 0.1994

Mcnemar's Test P-Value : 8.724e-13

Sensitivity : 0.9304

Specificity : 0.2362

Pos Pred Value : 0.7236

Neg Pred Value : 0.6122

Prevalence : 0.6825

Detection Rate : 0.6350

Detection Prevalence : 0.8775

Balanced Accuracy : 0.5833

'Positive' Class : FALSE

```
> #Plot confusion matrix
> ctable<-as.table(matrix(c(254,97,19,30),nrow = 2,byrow = TRUE))
> fourfoldplot(ctable,color = c("#CC6666","#99CC99"),conf.level = 0,margin = 1,main="Confusion Matrix")
```

```
> library(party)

> # Create the input data frame.

> input.dat <- CA[c(2:200),]

> # Give the chart file a name.

> png(file = "decision_tree.png")

> # Create the tree.

> output.tree <- ctree(

+   admit ~ gre+ ses +Race+rank,

+   data = input.dat)

> # Plot the tree.

> plot(output.tree)

> # Save the file.

> dev.off()

RStudioGD

      2

> library(rpart)

> library(rpart.plot)

> fit<-rpart(admit~.,data=CA,method='class')

> rpart.plot(fit,extra=106)

> #Confusion matrix

> pu<-predict(fit,CA,type='class')

> tm<-table(CA$admit,pu)
```

```
> tm
   pu
     0   1
  0 254  19
  1  78  49
> AC<-sum(diag(tm))/sum(tm)
> paste('Accuracy for test',AC)
[1] "Accuracy for test 0.7575"
> library(party)
> library(randomForest)
> # Create the forest.
> output.forest <- randomForest(admit~ gre+ ses + Race + rank,
+                            data = input.dat)
Warning message:
In randomForest.default(m, y, ...) :
  The response has five or fewer unique values.  Are you sure you want to do regression?
> # View the forest results.
> print(output.forest)


Call:
 randomForest(formula = admit ~ gre + ses + Race + rank, data = input.dat)
```

```
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 1

          Mean of squared residuals: 0.195543
                    % Var explained: 3.3
> # Importance of each predictor.
> importance(output.forest,type=2)
     IncNodePurity
gre       6.043341
ses       1.972306
Race      1.952104
rank      4.524398
> print(importance(output.forest,type = 2))
     IncNodePurity
gre       6.043341
ses       1.972306
Race      1.952104
rank      4.524398
> plot(output.forest)
> library(dplyr)
> CA<-CA%>%
```

```
+   mutate(gre_class=case_when(
+     gre<400~"Low",
+     gre>440&gre<580~"Medium",
+     gre>580~"High"
+   ))
> CA
   admit gre  gpa ses Gender_Male Race rank gre_class
1      0 380 3.61   1           0    3    3       Low
2      1 660 3.67   2           0    2    3      High
3      1 800 4.00   2           0    2    1      High
4      1 640 3.19   1           1    2    4      High
5      0 520 2.93   3           1    2    4    Medium
6      1 760 3.00   2           1    1    2      High
7      1 560 2.98   2           1    2    1    Medium
8      0 400 3.08   2           0    2    2      <NA>
9      1 540 3.39   1           1    1    3    Medium
10     0 700 3.92   1           0    2    2      High
11     0 800 4.00   1           1    1    4      High
12     0 440 3.22   3           0    2    1      <NA>
13     1 760 4.00   3           1    2    1      High
14     0 700 3.08   2           0    2    2      High
15     1 700 4.00   2           1    1    1      High
```

16	0	480	3.44	3	0	1	3	Medium
17	0	780	3.87	2	0	3	4	High
18	0	360	2.56	3	1	3	3	Low
19	0	800	3.75	1	1	3	2	High
20	1	540	3.81	1	0	3	1	Medium
21	0	500	3.17	3	0	2	3	Medium
22	1	660	3.63	1	0	1	2	High
23	0	600	2.82	1	0	3	4	High
24	0	680	3.19	1	0	1	4	High
25	1	760	3.35	2	0	2	2	High
26	1	800	3.66	2	1	1	1	High
27	1	620	3.61	2	0	1	1	High
28	1	520	3.74	2	0	3	4	Medium
29	1	780	3.22	1	0	1	2	High
30	0	520	3.29	1	0	1	1	Medium
31	0	540	3.78	1	1	1	4	Medium
32	0	760	3.35	2	1	1	3	High
33	0	600	3.40	3	0	1	3	High
34	1	800	4.00	3	0	1	3	High
35	0	360	3.14	1	1	2	1	Low
36	0	400	3.05	3	0	2	2	<NA>
37	0	580	3.25	1	0	2	1	<NA>

38	0	520	2.90	2	0	2	3	Medium
39	1	500	3.13	2	0	2	2	Medium
40	1	520	2.68	2	0	1	3	Medium
41	0	560	2.42	1	1	3	2	Medium
42	1	580	3.32	1	0	1	2	<NA>
43	1	600	3.15	2	1	1	2	High
44	0	500	3.31	2	0	2	3	Medium
45	0	700	2.94	1	0	3	2	High
46	1	460	3.45	2	1	3	3	Medium
47	1	580	3.46	3	1	1	2	<NA>
48	0	500	2.97	3	0	2	4	Medium
49	0	440	2.48	3	0	3	4	<NA>
50	0	400	3.35	3	0	1	3	<NA>
51	0	640	3.86	2	1	3	3	High
52	0	440	3.13	2	0	2	4	<NA>
53	0	740	3.37	2	1	3	4	High
54	1	680	3.27	2	0	2	2	High
55	0	660	3.34	1	0	1	3	High
56	1	740	4.00	1	1	2	3	High
57	0	560	3.19	3	1	1	3	Medium
58	0	380	2.94	3	0	2	3	Low
59	0	400	3.65	3	1	2	2	<NA>

60	0	600	2.82	3	1	1	4	High
61	1	620	3.18	2	1	1	2	High
62	0	560	3.32	1	0	3	4	Medium
63	0	640	3.67	1	1	2	3	High
64	1	680	3.85	1	1	3	3	High
65	0	580	4.00	2	1	3	3	<NA>
66	0	600	3.59	1	0	1	2	High
67	0	740	3.62	3	1	2	4	High
68	0	620	3.30	2	1	3	1	High
69	0	580	3.69	3	0	3	1	<NA>
70	0	800	3.73	1	1	1	1	High
71	0	640	4.00	1	1	1	3	High
72	0	300	2.92	1	1	1	4	Low
73	0	480	3.39	2	0	2	4	Medium
74	0	580	4.00	3	0	3	2	<NA>
75	0	720	3.45	2	1	2	4	High
76	0	720	4.00	2	0	3	3	High
77	0	560	3.36	1	1	2	3	Medium
78	1	800	4.00	3	0	3	3	High
79	0	540	3.12	3	1	2	1	Medium
80	1	620	4.00	2	0	2	1	High
81	0	700	2.90	2	0	2	4	High

82	0	620	3.07	3	1	2	2	High
83	0	500	2.71	2	0	3	2	Medium
84	0	380	2.91	3	1	2	4	Low
85	1	500	3.60	1	1	1	3	Medium
86	0	520	2.98	2	0	2	2	Medium
87	0	600	3.32	1	0	3	2	High
88	0	600	3.48	1	0	1	2	High
89	0	700	3.28	3	0	3	1	High
90	1	660	4.00	1	1	1	2	High
91	0	700	3.83	2	0	2	2	High
92	1	720	3.64	2	0	2	1	High
93	0	800	3.90	3	1	1	2	High
94	0	580	2.93	3	1	1	2	<NA>
95	1	660	3.44	2	0	3	2	High
96	0	660	3.33	2	1	3	2	High
97	0	640	3.52	2	1	3	4	High
98	0	480	3.57	3	1	2	2	Medium
99	0	700	2.88	2	1	3	2	High
100	0	400	3.31	3	1	2	3	<NA>
101	0	340	3.15	2	0	1	3	Low
102	0	580	3.57	1	1	2	3	<NA>
103	0	380	3.33	3	0	3	4	Low

104	0	540	3.94	3	0	1	3	Medium
105	1	660	3.95	2	1	1	2	High
106	1	740	2.97	1	1	1	2	High
107	1	700	3.56	1	1	2	1	High
108	0	480	3.13	2	0	1	2	Medium
109	0	400	2.93	1	1	3	3	<NA>
110	0	480	3.45	3	0	1	2	Medium
111	0	680	3.08	3	0	3	4	High
112	0	420	3.41	2	1	3	4	<NA>
113	0	360	3.00	1	0	1	3	Low
114	0	600	3.22	3	1	2	1	High
115	0	720	3.84	1	1	2	3	High
116	0	620	3.99	2	1	2	3	High
117	1	440	3.45	1	1	3	2	<NA>
118	0	700	3.72	2	1	2	2	High
119	1	800	3.70	1	0	2	1	High
120	0	340	2.92	3	1	2	3	Low
121	1	520	3.74	2	0	2	2	Medium
122	1	480	2.67	1	0	1	2	Medium
123	0	520	2.85	3	0	1	3	Medium
124	0	500	2.98	3	0	2	3	Medium
125	0	720	3.88	2	0	3	3	High

```
 [ reached 'max' / getOption("max.print") -- omitted 275 rows ]

> CA$gre_class=factor(CA$gre_class,levels = c("Low","Medium","High"))

> XT=xtabs(~gre_class+gre,data = CA)

> XT

        gre

gre_class 220 300 340 360 380 460 480 500 520 540 560 600 620 640 660 680 700
720

   Low      1   3   4   4   8   0   0   0   0   0   0   0   0   0   0   0   0   0

   Medium   0   0   0   0   0  14  16  21  24  27  24   0   0   0   0   0   0   0

   High     0   0   0   0   0   0   0   0   0   0   0  23  30  21  24  20  22  11

        gre

gre_class 740 760 780 800

   Low      0   0   0   0

   Medium   0   0   0   0

   High    11   5   5  25

> #Count levels in gre

> nlevels(CA$gre_class)

[1] 3

> #Count no of elements in each levels in gre

> count(CA,CA$gre_class)

  CA$gre_class   n

1         Low  20
```

2 Medium 126

3 High 197

4 <NA> 57

>

> #Random forest gives better result out of all these models.

>