> # loading libraries

> library(caret)

> library(mlbench)

> library(magrittr)

> library(dplyr)

> library(ggplot2)

> library(ggpubr)

> # setting ggplot Themes

> theme\_set(theme\_pubr())

>

> # load the CSV file from the local directory

> dataset <- read.csv("C:/Users/HOME/Downloads/College\_admission.csv", header=TRUE)

> # Check first few rows

> head(dataset)

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 the data

> summary(dataset)

admit gre gpa ses

Min. :0.0000 Min. :220.0 Min. :2.260 Min. :1.000

1st Qu.:0.0000 1st Qu.:520.0 1st Qu.:3.130 1st Qu.:1.000

Median :0.0000 Median :580.0 Median :3.395 Median :2.000

Mean :0.3175 Mean :587.7 Mean :3.390 Mean :1.992

3rd Qu.:1.0000 3rd Qu.:660.0 3rd Qu.:3.670 3rd Qu.:3.000

Max. :1.0000 Max. :800.0 Max. :4.000 Max. :3.000

Gender\_Male Race rank

Min. :0.000 Min. :1.000 Min. :1.000

1st Qu.:0.000 1st Qu.:1.000 1st Qu.:2.000

Median :0.000 Median :2.000 Median :2.000

Mean :0.475 Mean :1.962 Mean :2.485

3rd Qu.:1.000 3rd Qu.:3.000 3rd Qu.:3.000

Max. :1.000 Max. :3.000 Max. :4.000

>

>

>

> # summarize the admission distribution

> percentage <- prop.table(table(dataset$admit)) \* 100

> cbind(freq=table(dataset$admit), percentage=percentage)

freq percentage

0 273 68.25

1 127 31.75

>

> #Find the missing values. (if any, perform missing value treatment)

> is.na(dataset)

admit gre gpa ses Gender\_Male Race rank

[1,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[2,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[3,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[4,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[5,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[6,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[7,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[8,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[9,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[10,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[11,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[12,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[13,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[14,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[15,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[16,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[17,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[18,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[19,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[20,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[21,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[22,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[23,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[24,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[25,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[26,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[27,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[28,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[29,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[30,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[31,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[32,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[33,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[34,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[35,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[36,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[37,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[38,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[39,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[40,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[41,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[42,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[43,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[44,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[45,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[46,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[47,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[48,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[49,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[50,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[51,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[52,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[53,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[54,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[55,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[56,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[57,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[58,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[59,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[60,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[61,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[62,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[63,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[64,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[65,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[66,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[67,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[68,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[69,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[70,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[71,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[72,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[73,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[74,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[75,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[76,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[77,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[78,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[79,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[80,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[81,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[82,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[83,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[84,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[85,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[86,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[87,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[88,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[89,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[90,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[91,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[92,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[93,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[94,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[95,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[96,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[97,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[98,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[99,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[100,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[101,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[102,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[103,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[104,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[105,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[106,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[107,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[108,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[109,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[110,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[111,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[112,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[113,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[114,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[115,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[116,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[117,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[118,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[119,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[120,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[121,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[122,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[123,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[124,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[125,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[126,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[127,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[128,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[129,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[130,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[131,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[132,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[133,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[134,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[135,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[136,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[137,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[138,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[139,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[140,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[141,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[142,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[ reached getOption("max.print") -- omitted 258 rows ]

> sum(is.na(dt))

[1] 0

Warning message:

In is.na(dt) : is.na() applied to non-(list or vector) of type 'closure'

> # No missing values in dataset

>

> #Find outliers (if any, then perform outlier treatment)

> outliers <- boxplot(dataset$disp, plot=FALSE)$out

> dataset[which(dataset$disp %in% outliers),]

[1] admit gre gpa ses Gender\_Male

[6] Race rank

<0 rows> (or 0-length row.names)

> # No outliers in dataset

>

> #Find the structure of the data set

> str(dataset)

'data.frame': 400 obs. of 7 variables:

$ admit : int 0 1 1 1 0 1 1 0 1 0 ...

$ gre : int 380 660 800 640 520 760 560 400 540 700 ...

$ gpa : num 3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...

$ ses : int 1 2 2 1 3 2 2 2 1 1 ...

$ Gender\_Male: int 0 0 0 1 1 1 1 0 1 0 ...

$ Race : int 3 2 2 2 2 1 2 2 1 2 ...

$ rank : int 3 3 1 4 4 2 1 2 3 2 ...

>

> # list types for each attribute

> sapply(dataset, class)

admit gre gpa ses Gender\_Male

"integer" "integer" "numeric" "integer" "integer"

Race rank

"integer" "integer"

>

> #transform the numeric data type to factor and vice-versa.

> dataset$admit=as.factor(dataset$admit)

>

>

> # list types after tansfer to factor

> sapply(dataset, class)

admit gre gpa ses Gender\_Male

"factor" "integer" "numeric" "integer" "integer"

Race rank

"integer" "integer"

>

> #Find whether the data is normally distributed or not. Use the plot to determine the same.

> x <- dataset[,2:7]

> y <- dataset[,1]

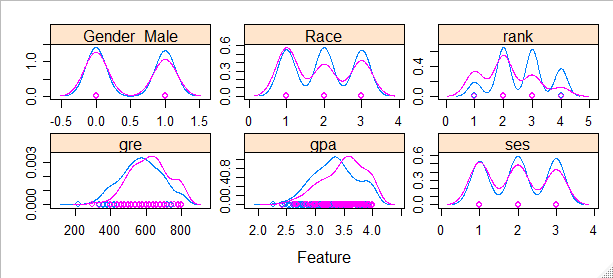
>

> # density plots for each attribute by class value

> scales <- list(x=list(relation="free"), y=list(relation="free"))

> featurePlot(x=x, y=y, plot="density", scales=scales)

>

> 

>

>

> # create a list of 80% of the rows in the original dataset we can use for training

> validation\_index <- createDataPartition(dataset$admit, p=0.80, list=FALSE)

> # select 20% of the data for validation

> validation <- dataset[-validation\_index,]

> # use the remaining 80% of data to training and testing the models

> dataset <- dataset[validation\_index,]

>

>

>

> # Use variable reduction techniques to identify significant variables.

> set.seed(7)

> # calculate correlation matrix

> control <- trainControl(method="repeatedcv", number=10, repeats=3)

> # train the model

> model <- train(admit~., data=dataset, method="lvq", preProcess="scale", trControl=control)

> # estimate variable importance

> importance <- varImp(model, scale=FALSE)

> # summarize importance

> print(importance)

ROC curve variable importance

Importance

rank 0.6327

gpa 0.6187

gre 0.6052

Gender\_Male 0.5244

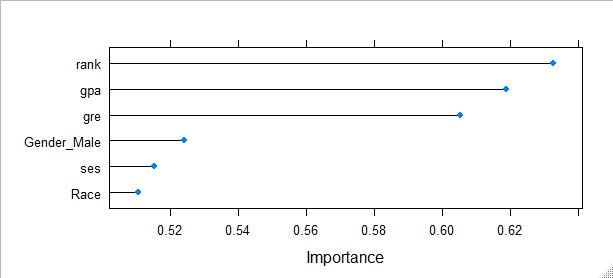
ses 0.5153

Race 0.5107

> # plot importance

> plot(importance)

>

> 

>

> #Run logistic model to determine the factors that influence the admission process of a student (Drop insignificant variables)

> #Calculate the accuracy of the model and run validation techniques.

> #Try other modelling techniques like decision tree and SVM and select a champion model

> #Determine the accuracy rates for each kind of model

> #Select the most accurate model

> #Identify other Machine learning or statistical techniques

> set.seed(7)

> fit.glm <- train(admit~ rank + gre + gpa , data=dataset, method="glm", metric=metric, trControl=control)

> set.seed(7)

> fit.svm <- train(admit~ rank + gre + gpa , data=dataset, method="svmRadial", metric=metric, trControl=control)

>

>

> # summarize accuracy of models

> results <- resamples(list(lrg=fit.glm, svm=fit.svm))

> summary(results)

Call:

summary.resamples(object = results)

Models: lrg, svm

Number of resamples: 30

Accuracy

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

lrg 0.59375 0.6799395 0.6969697 0.6978260 0.7361391 0.81250 0

svm 0.59375 0.6588542 0.6875000 0.6926177 0.7187500 0.78125 0

Kappa

Min. 1st Qu. Median Mean 3rd Qu. Max.

lrg -0.09473684 0.069767442 0.1495984 0.1558887 0.2502854 0.5102041

svm -0.09473684 0.002808989 0.1306380 0.1154531 0.1891768 0.4105263

NA's

lrg 0

svm 0

>

>

> dotplot(results)

>

> # summarize Best Model

> print(fit.glm)

Generalized Linear Model

321 samples

3 predictor

2 classes: '0', '1'

No pre-processing

Resampling: Cross-Validated (10 fold, repeated 3 times)

Summary of sample sizes: 289, 288, 289, 289, 288, 289, ...

Resampling results:

Accuracy Kappa

0.697826 0.1558887

>

>

> predictions <- predict(fit.glm, validation)

> confusionMatrix(predictions, validation$admit)

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 50 19

1 4 6

Accuracy : 0.7089

95% CI : (0.5958, 0.8057)

No Information Rate : 0.6835

P-Value [Acc > NIR] : 0.363355

Kappa : 0.1978

Mcnemar's Test P-Value : 0.003509

Sensitivity : 0.9259

Specificity : 0.2400

Pos Pred Value : 0.7246

Neg Pred Value : 0.6000

Prevalence : 0.6835

Detection Rate : 0.6329

Detection Prevalence : 0.8734

Balanced Accuracy : 0.5830

'Positive' Class : 0

>

>

>

> #Descriptive:

> # Categorize the average of grade point into High, Medium, and Low (with admission probability percentages) and plot it on a point chart.

> #Cross grid for admission variables with GRE Categorization is shown below:

> gre\_Table <- dataset %>%

+ mutate(grep\_prob =

+ case\_when(gre <440 ~ "Low",

+ gre <580 ~ "Medium",

+ gre >=580 ~ "High"))

>

> # Print table

> df <- gre\_Table %>%

+ group\_by(gre\_prob) %>%

+ summarise(counts = n())

Error: Column `gre\_prob` is unknown

> df

# A tibble: 3 x 2

grep\_prob counts

*<chr>* *<int>*

1 High 55

2 Low 11

3 Medium 31

>

>

> # Printpoint chart

> ggplot(df, aes(x=grep\_prob, y=counts, color=as.factor(grep\_prob), shape=as.factor(grep\_prob))) +

+ geom\_linerange(

+ aes(x = grep\_prob, ymin = 0, ymax = counts),

+ color = "lightgray", size = 1.5

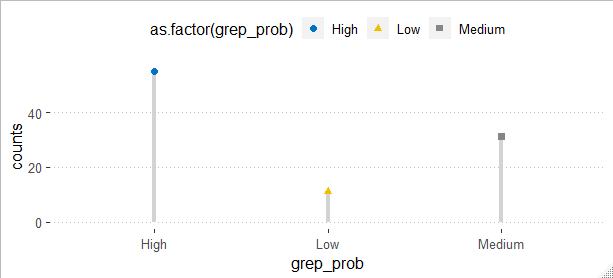
+ )+

+ geom\_point(aes(color = grep\_prob), size = 2)+

+ ggpubr::color\_palette("jco")+

+ theme\_pubclean()

>



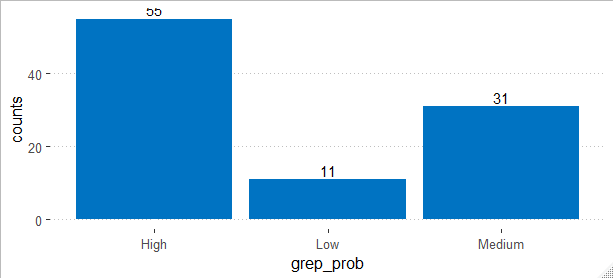
> # Print bar chart

> ggplot(df, aes(x = grep\_prob, y = counts)) +

+ geom\_bar(fill = "#0073C2FF", stat = "identity") +

+ geom\_text(aes(label = counts), vjust = -0.3) +

+ theme\_pubclean()

> 

>

> # Analysis Tasks for admission

> # Summary: Most accuarate modle logistic regression with 61% accuracy with feautere

> # importances GRE,Rank and GPA

>

> # Descriptive for admission

> # GRE scpore plays a vital role in admission . we can see same by data and graphs