CAMPUS PLACEMENT DATASET

Presented by

TOLULOPE AYODEJI ALE

22/05/2022

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

CAMPUS RECRUITMENT PLACEMENT DATASET

A major quest for any student is to get a good job placement after graduation, but the criteria which an employer uses to determines if a graduate is a good fit for employment may not be the same with what makes a good student. Education can be subdivided into lower degree and upper degree. The quality of education in the lower degree classes determines the qualify of the foundation a student is getting for future classes. Also, some professional degree programs such an MBA could make for a distinguishable factor during placement. The dataset for this study was obtained from Kaggle

1.1 Objectives

We will be doing some analysis on a CAMPUS RECRUITMENT dataset obtained from Kaggle repository. The dataset consists of 14 attributes, 1 target attribute, 215 instances and 2 classes. We will be analyzing the dataset to predict attributes that can influence if a student will be placed or not. We will do exploratory data analysis and visualization using R.

Research Questions

1. Which variable influenced a candidate in getting placed?
2. Does previous work experience matter for one to get placed?
3. Does high percentage in employment test matters for one to get placed?
4. Which Degree type and MBA specialization is in much demand?

We will attempt to answer the research questions using some statistical learning techniques by determining the significance -p value of the attributes.

Also, We will divide the dataset into a train and test set to build a model that could predict if a student will be placed based on the attributes.

1.2 Dataset Description

The dataset has 14 attributes, 1 target attribute, 215 instances and 2 classes as shown in the table below. 148 students were placed and 67 were not placed.

Input Attributes	Values
gender	Gender (1=male; 0=female)
ssc_p	Secondary education percentage
ssc_b	Board of education (central/others)
hsc_p	Higher secondary education percentage
hsc_b	Board of education (central/others)
hsc_s	Specialization in higher secondary education
degree_p	University Degree percentage
degree-t	University Degree type (Undergraduate degree field)
workex	Work experience
etest_p	Employability test percentage
specialization	Post-graduation (MBA) specialization
mba_p	MBA percentage
status	Status of placement (Placed/Not placed)
salary	Salary offered by corporate to candidates

For analysis on this dataset, We will be loading the following R libraries as seen in the code snippet.

```
library(dplyr)
library(tidyverse)
library(ggplot2)
view(data)
```

Using the summary function, we printed out the statistics of each attribute on the dataset.

- The dataset has two classes: numeric and character.
- 215 instances and 14 attributes and the serial number (sl_no).
- The salary attribute has 67 NAs
- 8 attributes has a character data type and 6 has a numeric data type

```
> summary(data)
     sl_no          gender              ssc_p           ssc_b               hsc_p           hsc_b              hsc_s              degree_p
 Min.   :  1.0   Length:215         Min.   :40.89   Length:215         Min.   :37.00   Length:215         Length:215         Min.   :50.00
 1st Qu.: 54.5   Class :character   1st Qu.:60.60   Class :character   1st Qu.:60.90   Class :character   Class :character   1st Qu.:61.00
 Median :108.0   Mode  :character   Median :67.00   Mode  :character   Median :65.00   Mode  :character   Mode  :character   Median :66.00
 Mean   :108.0                      Mean   :67.30                      Mean   :66.33                                         Mean   :66.37
 3rd Qu.:161.5                      3rd Qu.:75.70                      3rd Qu.:73.00                                         3rd Qu.:72.00
 Max.   :215.0                      Max.   :89.40                      Max.   :97.70                                         Max.   :91.00

   degree_t            workex             etest_p      specialisation         mba_p          status              salary
 Length:215         Length:215         Min.   :50.0   Length:215         Min.   :51.21   Length:215         Min.   :200000
 Class :character   Class :character   1st Qu.:60.0   Class :character   1st Qu.:57.95   Class :character   1st Qu.:240000
 Mode  :character   Mode  :character   Median :71.0   Mode  :character   Median :62.00   Mode  :character   Median :265000
                                       Mean   :72.1                      Mean   :62.28                      Mean   :288655
                                       3rd Qu.:83.5                      3rd Qu.:66.25                      3rd Qu.:300000
                                       Max.   :98.0                      Max.   :77.89                      Max.   :940000
                                                                                                            NA's   :67
```

The below code printed out the count of Nas in the dataset. The Nas are on the salary attribute, it indicate students who are not receiving salary. Hence, they are replaced with the value “0”.

The 67 students matched the number of students that are yet to be placed, because only students with placement received salaries.

```
> sum(is.na(data))
[1] 67
> data[is.na(data)] <- 0
> #Check for null values
> sum(is.na(data))
[1] 0
```

This is a multi-class dataset comprising of numbers and character datatypes with a dimension of 215 observations and 15 attributes.

```
> str(data)
spec_tbl_df [215 x 15] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ sl_no         : num [1:215] 1 2 3 4 5 6 7 8 9 10 ...
 $ gender        : chr [1:215] "M" "M" "M" "M" ...
 $ ssc_p         : num [1:215] 67 79.3 65 56 85.8 ...
 $ ssc_b         : chr [1:215] "Others" "Central" "Central" "Central" ...
 $ hsc_p         : num [1:215] 91 78.3 68 52 73.6 ...
 $ hsc_b         : chr [1:215] "Others" "Others" "Central" "Central" ...
 $ hsc_s         : chr [1:215] "Commerce" "Science" "Arts" "Science" ...
 $ degree_p      : num [1:215] 58 77.5 64 52 73.3 ...
 $ degree_t      : chr [1:215] "Sci&Tech" "Sci&Tech" "Comm&Mgmt" "Sci&Tech" ...
 $ workex        : chr [1:215] "No" "Yes" "No" "No" ...
 $ etest_p       : num [1:215] 55 86.5 75 66 96.8 ...
 $ specialisation: chr [1:215] "Mkt&HR" "Mkt&Fin" "Mkt&Fin" "Mkt&HR" ...
 $ mba_p         : num [1:215] 58.8 66.3 57.8 59.4 55.5 ...
 $ status        : chr [1:215] "Placed" "Placed" "Placed" "Not Placed" ...
 $ salary        : num [1:215] 270000 200000 250000 0 425000 0 0 252000 231000 0
```

Visualizing the top 6 rows using the Head function

```
> #Check top5 and bottom5 data
> head(data)
# A tibble: 6 x 15
  sl_no Gender SecEducationPercent SecBoardofEduca~ HigherSecEduPer~ HigherSecBoardo~ HigherSecSpecial
  <dbl> <fct>                <dbl> <fct>                       <dbl> <fct>            <fct>
1     1 M                     67   Others                       91   Others           Commerce
2     2 M                     79.3 Central                      78.3 Others           Science
3     3 M                     65   Central                      68   Central          Arts
4     4 M                     56   Central                      52   Central          Science
5     5 M                     85.8 Central                      73.6 Central          Commerce
6     6 M                     55   Others                       49.8 Others           Science
# ... with 8 more variables: DegreePercent <dbl>, DegreeType <fct>, WorkExp <fct>,
#   EmpTestPercent <dbl>, MBAspec <fct>, MBApercent <dbl>, Status <fct>, Salary <dbl>
> 
```

Visualizing the bottom 6 rows using the Tail function

```
> tail(data)
# A tibble: 6 x 15
  sl_no Gender SecEducationPercent SecBoardofEduca~ HigherSecEduPer~ HigherSecBoardo~ HigherSecSpecial
  <dbl> <fct>                <dbl> <fct>                       <dbl> <fct>            <fct>
1   210 M                     62   Central                        72 Central          Commerce
2   211 M                     80.6 Others                         82 Others           Commerce
3   212 M                     58   Others                         60 Others           Science
4   213 M                     67   Others                         67 Others           Commerce
5   214 F                     74   Others                         66 Others           Commerce
6   215 M                     62   Central                        58 Others           Science
# ... with 8 more variables: DegreePercent <dbl>, DegreeType <fct>, WorkExp <fct>,
#   EmpTestPercent <dbl>, MBAspec <fct>, MBApercent <dbl>, Status <fct>, Salary <dbl>
> 
```

1.3 Data Cleaning

Firstly, we will be renaming the column headers using the below line of code

```
> #Renaming the variables
> data <- rename(data, Gender = gender,
+                SecEducationPercent = ssc_p,
+                SecBoardofEducation = ssc_b,
+                HigherSecEduPercent = hsc_p,
+                HigherSecBoardofEdu = hsc_b,
+                HigherSecSpecial = hsc_s,
+                DegreePercent = degree_p,
+                DegreeType = degree_t,
+                WorkExp = workex,
+                EmpTestPercent = etest_p,
+                MBAspec = specialisation,
+                MBApercent = mba_p,
+                Status = status,
+                Salary = salary)
>
```

```
> attribute <- names(data)
> attribute
 [1] "sl_no"               "Gender"              "SecEducationPercent" "SecBoardofEducation"
 [5] "HigherSecEduPercent" "HigherSecBoardofEdu" "HigherSecSpecial"    "DegreePercent"
 [9] "DegreeType"          "WorkExp"             "EmpTestPercent"      "MBAspec"
[13] "MBApercent"          "Status"              "Salary"
>
```

CHAPTER 2: EXPLORATORY DATA ANALYSIS (EDA)

With EDA we hope to make some inference on the dataset and explore answering some of the research questions such as which factor influenced a candidate in getting placed, does previous work experience matter for one to get placed, does high percentage in employment test matters for one to get placed and which degree type and MBA specialization is in much demand.

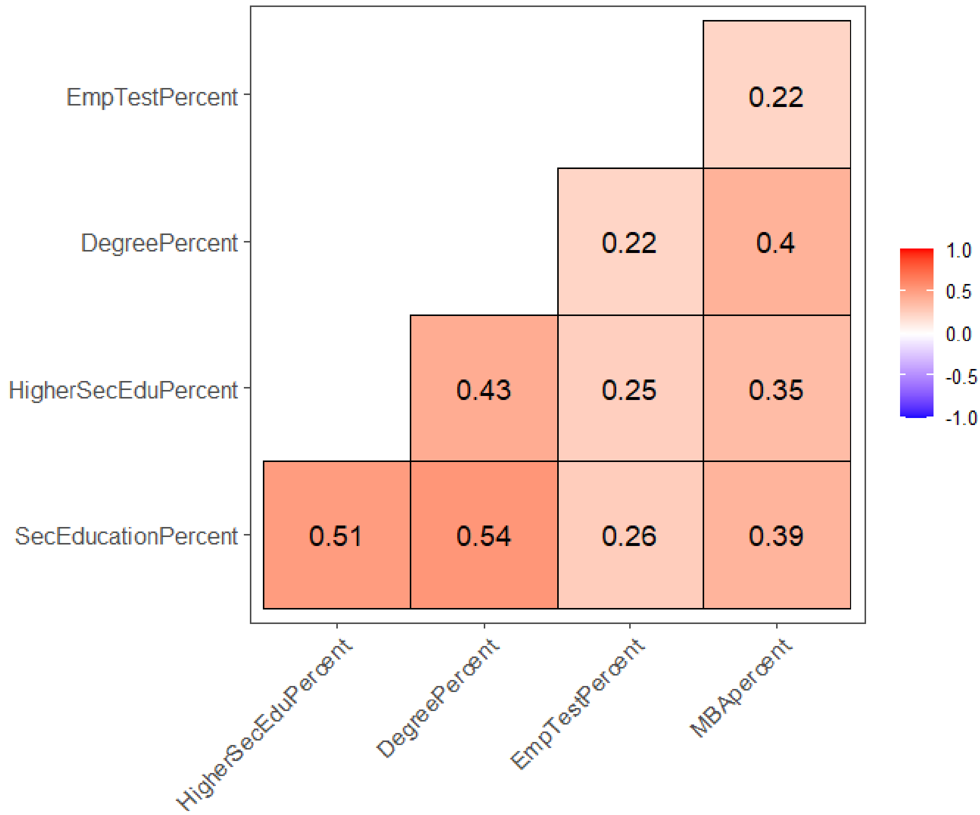
2.1 Correlation Plot

Correlation plot shows the relationship between the numeric variables

```
> #Observe the correlation between the numeric values excluding Salary
> data_corr <- select_if(data, is.numeric) %>% select(-Salary, -sl_no)
> corr <- round(cor(data_corr),2)
> corr
                     SecEducationPercent HigherSecEduPercent DegreePercent EmpTestPercent MBApercent
SecEducationPercent                 1.00                0.51          0.54           0.26       0.39
HigherSecEduPercent                 0.51                1.00          0.43           0.25       0.35
DegreePercent                       0.54                0.43          1.00           0.22       0.40
EmpTestPercent                      0.26                0.25          0.22           1.00       0.22
MBApercent                          0.39                0.35          0.40           0.22       1.00
>
```

To visualize the correlation heatmap, we installed and loaded the ggcorrplot package.

```
> #visualize the correlation matrix
> ggcorrplot(corr, method = "square",
+            ggtheme = ggthemes::theme_few,
+            outline.col = "black",
+            lab = TRUE,
+            lab_size = 5,
+            digits = 2,
+            type = "lower",
+            legend = "",
+            tl.cex = 12)
>
```



From the correlation heatmap, we can make two inferences:

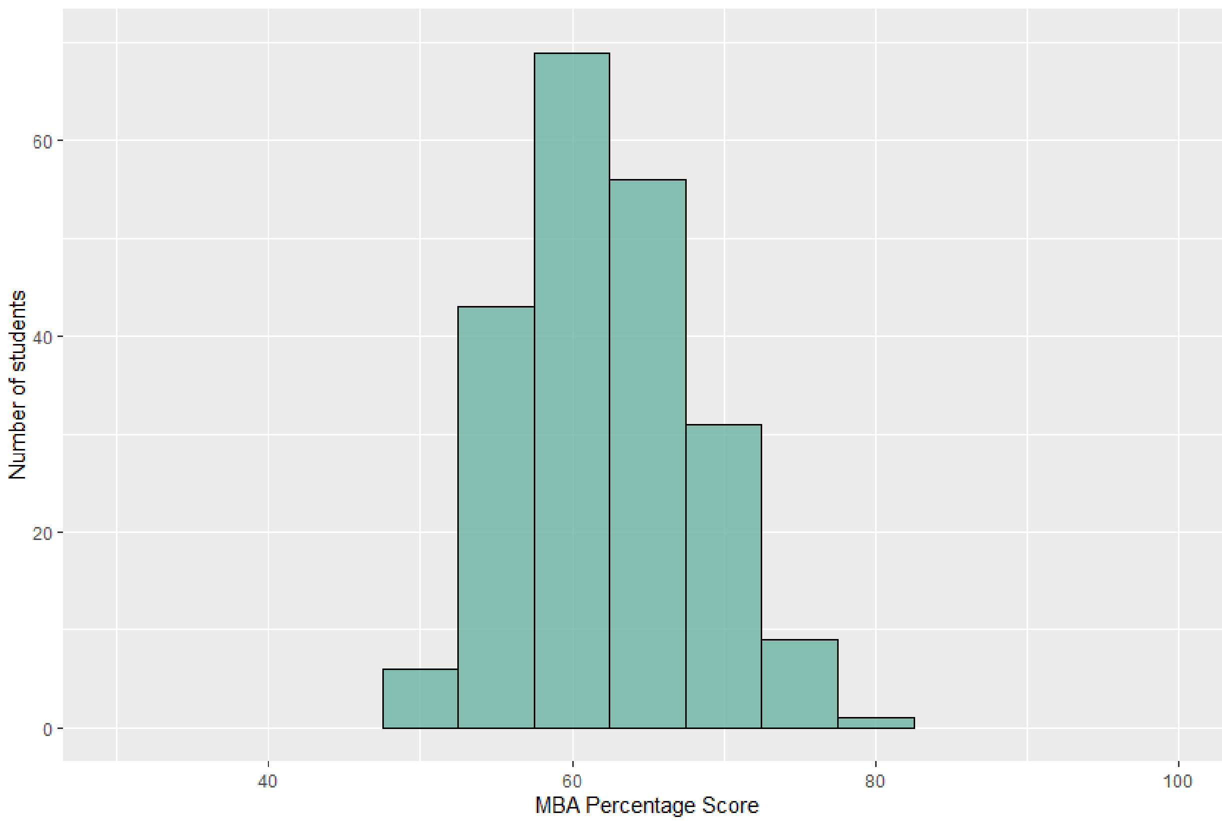
1. It can be observed that the Secondary Education Percentage, Higher Secondary Education Percentage and Degree Percentage has medium correlation. This could suggest that students who performed well in secondary education will likely perform well in further education.
2. The correlation between Employability Test Percentage and other education percentage score is low. This could suggest that the employability test was more practical, the implication of that indicates employers are looking for more than a higher academic grade.

2.2 Score Distribution

Visualizing the distribution of all the scores across the education levels should highlight if the trend observed in the correlation plot persist.

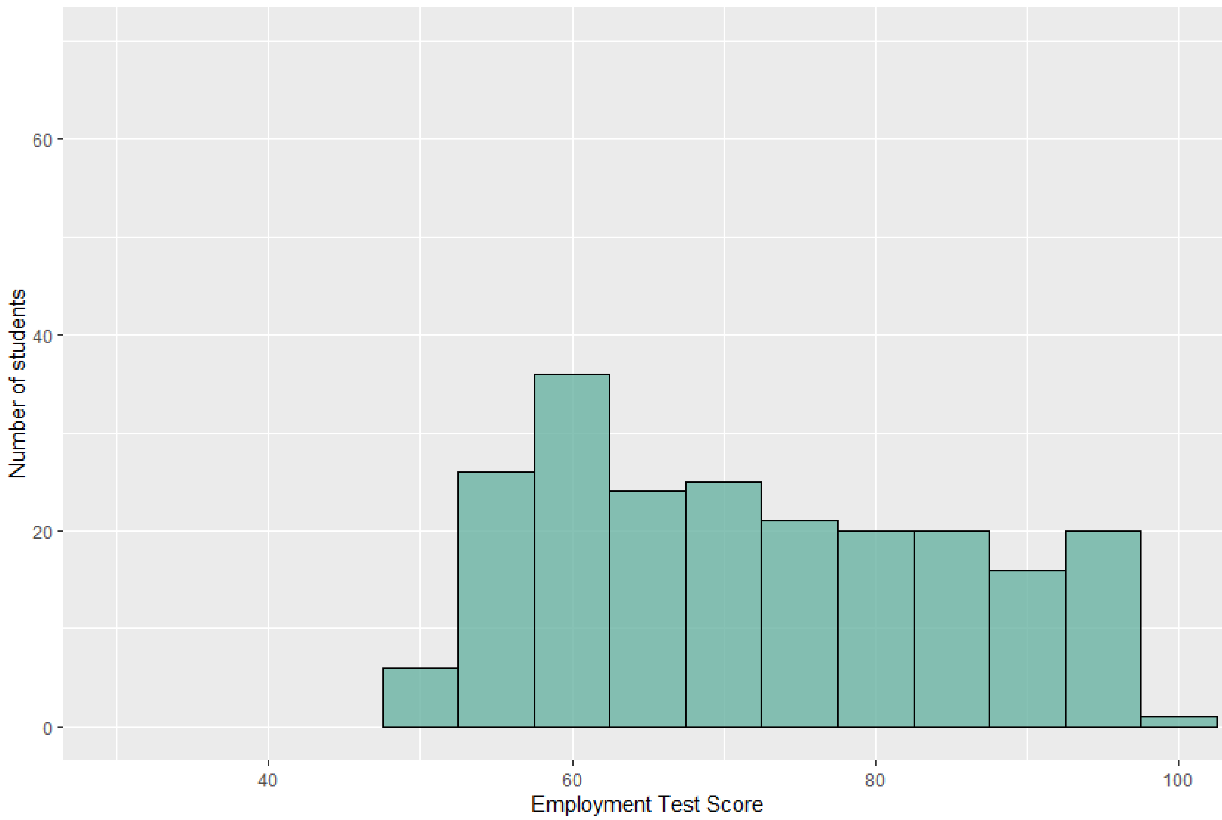
1. MBA percentage score

```
> #Histogram plot for MBApercent
> data %>%
+   ggplot( aes(x=MBApercent)) +
+   geom_histogram(fill="#69b3a2", color="black", binwidth = 5, alpha=0.8)+
+   coord_cartesian(xlim=c(30,100),
+                   ylim=c(0,70))+
+   labs(x = "MBA Percentage Score",
+        y = "Number of students")
> 
```



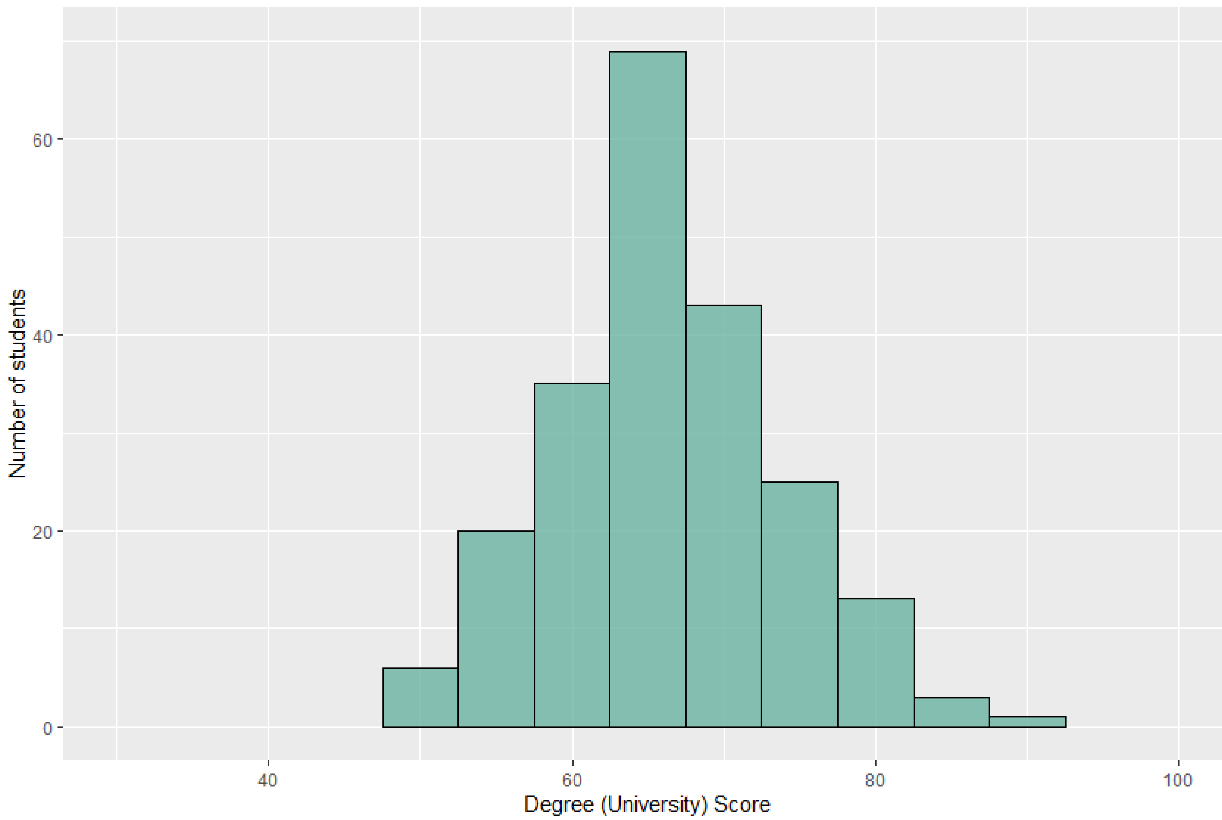
2. Employee Test Percentage Score

```
> #Histogram plot for EmpTestPercent
> data %>%
+   ggplot( aes(x=EmpTestPercent)) +
+   geom_histogram(fill="#69b3a2", color="black", binwidth = 5, alpha=0.8)+
+   coord_cartesian(xlim=c(30,100),
+                   ylim=c(0,70))+
+   labs(x = "Employment Test Score",
+        y = "Number of students")
>
```



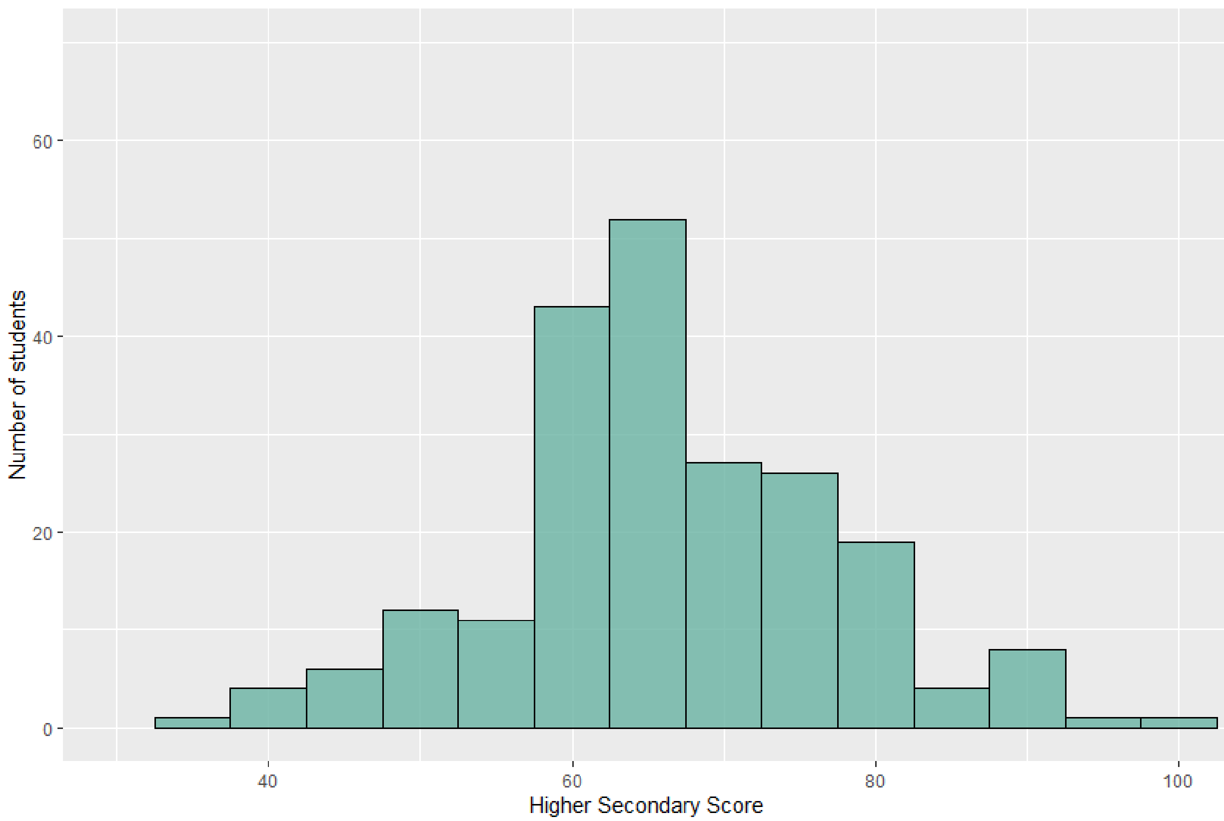
3. University Degree Percentage Score

```
> #Histogram plot for DegreePercent
> data %>%
+   ggplot( aes(x=DegreePercent)) +
+   geom_histogram(fill="#69b3a2", color="black", binwidth =5, alpha=0.8)+
+   coord_cartesian(xlim=c(30,100),
+                   ylim=c(0,70))+
+   labs(x = "Degree (University) Score",
+        y = "Number of students")
>
```



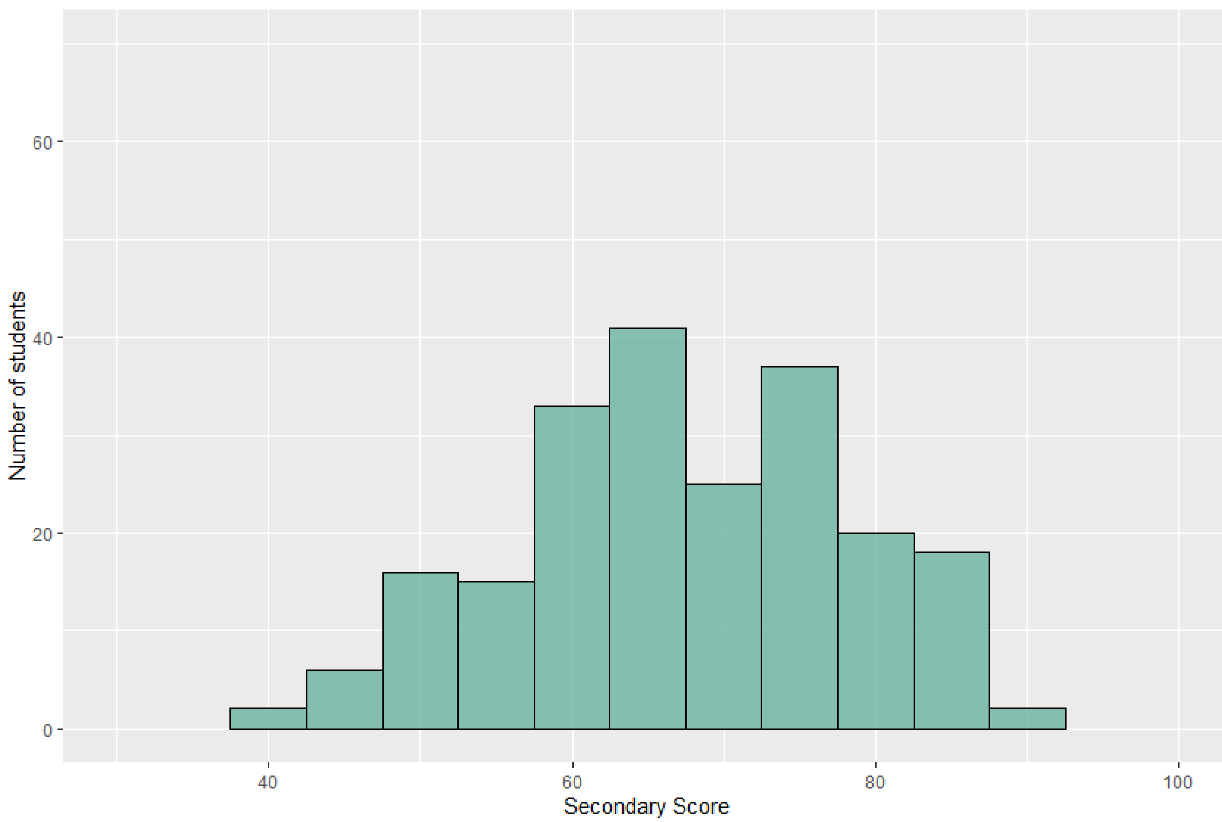
4. Higher Secondary Education Percentage Score

```
> #Higher Secondary
> data %>%
+   ggplot( aes(x=HigherSecEduPercent)) +
+   geom_histogram(fill="#69b3a2", color="black", binwidth =5, alpha=0.8)+
+   coord_cartesian(xlim=c(30,100),
+                   ylim=c(0,70))+
+   labs(x = "Higher Secondary Score",
+        y = "Number of students")
> 
```



5. Secondary Education Percentage Score

```
> #Secondary
> data %>%
+   ggplot( aes(x=SecEducationPercent)) +
+   geom_histogram(fill="#69b3a2", color="black", binwidth =5, alpha=0.8)+
+   coord_cartesian(xlim=c(30,100),
+                   ylim=c(0,70))+
+   labs(x = "Secondary Score",
+        y = "Number of students")
> 
```



- It was observed that the score distributions got narrower as the students move from Secondary Education to MBA.
- The score became more concentrated between the range 60-70 as the students' progress in education.
- We can also observe that the Employee Test Score has a different distribution, which support our earlier hypothesis on the correlation plot. The distributions were almost equal.

2.3 Gender Differences on Performance Score Using T-test

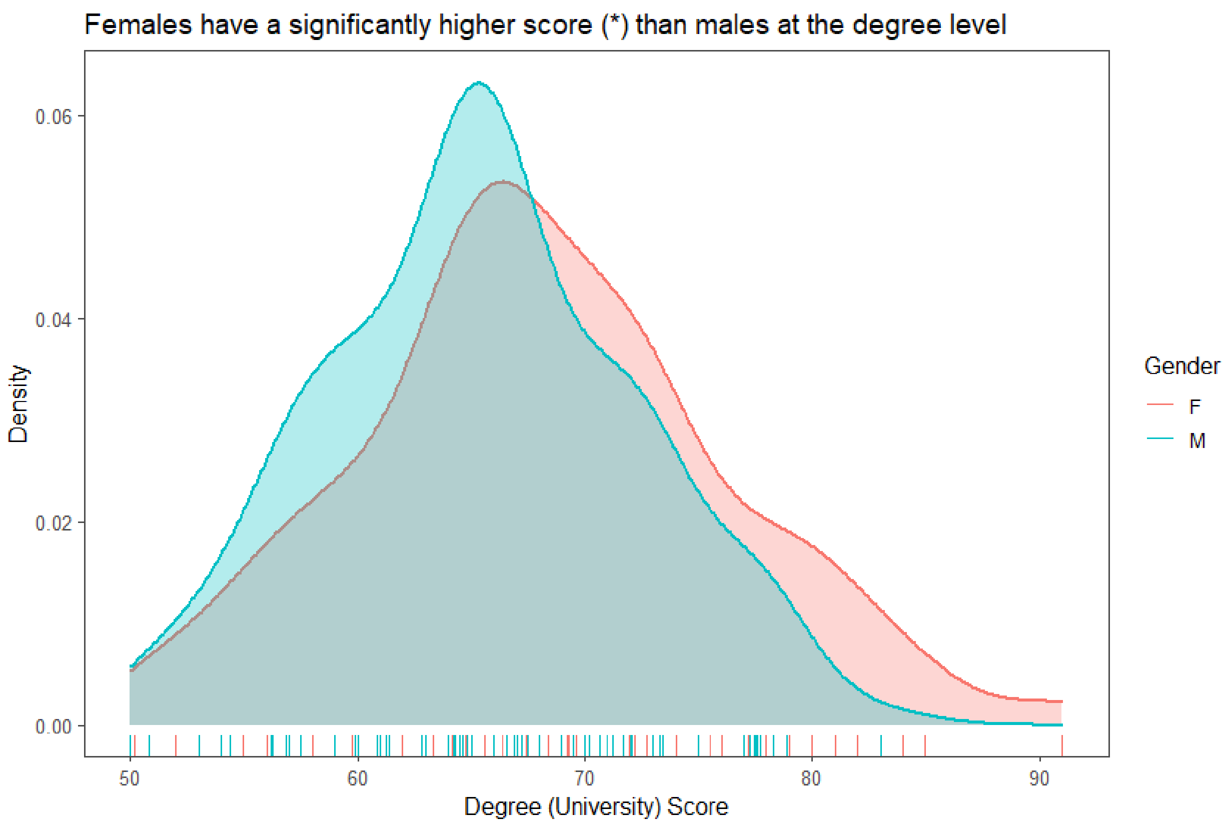
We will explore the differences between the genders on the performance score at each academic level.

T-tests in R is one of the most common tests in statistics. So, we will be using it to determine whether the means of the two genders are equal to each other. The assumption for the test is that both genders are sampled from normal distributions with equal variances. The null hypothesis is that the two means are equal, and the alternative is that they are not equal. It is known that under the null hypothesis, we can calculate a t-statistic that will follow a t-distribution with **n1 + n2 – 2 degrees of freedom**. The **p-significance** level less than <= 0.05 will help determine the significance level of a gender against the other.

Load the library 'rstatix'.

1. University Degree Percentage Score

```
> t_test_degree <- data%>%t_test(DegreePercent ~ Gender)%>%add_significance()
>
> data %>%
+   ggplot(aes(DegreePercent, fill = Gender, col = Gender))+
+   geom_density(alpha = 0.3, lwd = 1, show.legend = FALSE)+
+   geom_rug()+
+   labs(title = paste("Females have a significantly higher score
+   (", t_test_degree$p.signif, ") than males at the degree level", sep = ""),
+        col = "Gender",
+        x = "Degree (University) Score",
+        y = "Density")+
+        theme_few()+
+   theme(legend.position = "right")
> 
```



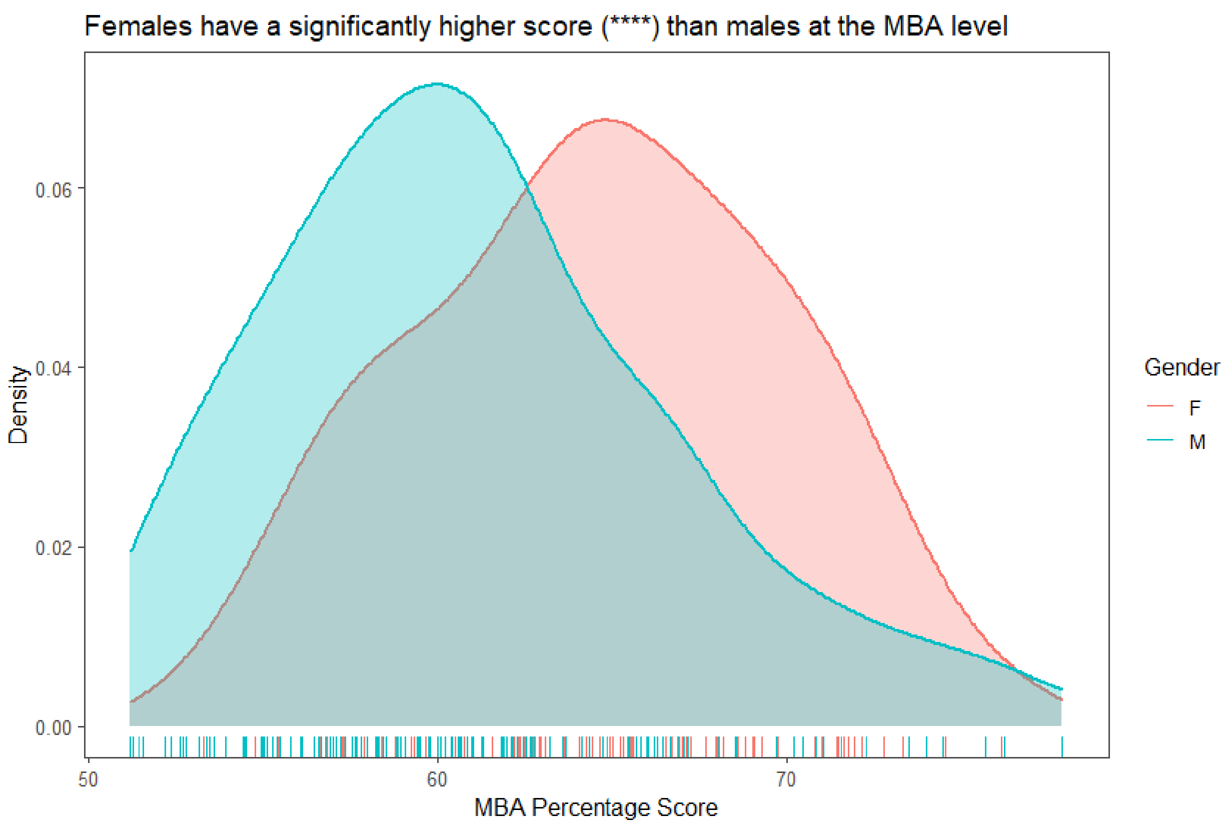

```
> t_test_degree%>%
+   kbl()%>%
+   kable_paper("hover", full_width = F)
>
```

.y.	group1	group2	n1	n2	statistic	df	p	p.signif
DegreePercent	F	M	76	139	2.431405	132.0024	0.0164	*

The **p-value score reject the null hypothesis**. Female students have a significantly higher score than their male counterpart. With the p-value score, it provide a statistical support that female students with a much higher score have a higher potential of being placed than their male counterpart.

2. MBA Percentage Score

```
> #Gender inference on MBA
>
>  t_test_mba <- data%>%t_test(MBApercent ~ Gender)%>%add_significance()
>
> data %>%
+   ggplot(aes(MBApercent, fill = Gender, col = Gender))+
+   geom_density(alpha = 0.3, lwd = 1, show.legend = FALSE)+
+   geom_rug()+
+   labs(title = paste("Females have a significantly higher score
+   (", t_test_mba$p.signif, ") than males at the MBA level", sep = ""),
+        col = "Gender",
+        x = "MBA Percentage Score",
+        y = "Density")+
+        theme_few()+
+   theme(legend.position = "right")
>
```



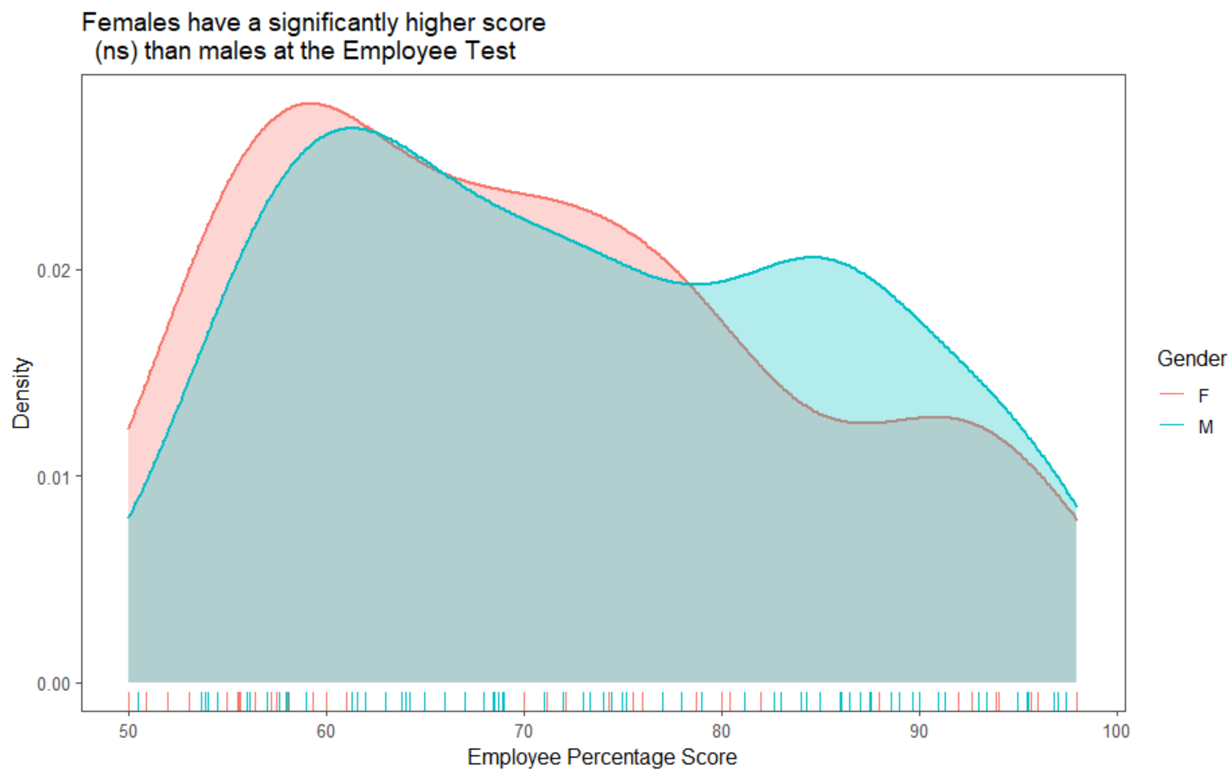

```
> t_test_mba%>%
+   kbl()%>%
+   kable_paper("hover", full_width = F)
>
```

.y.	group1	group2	n1	n2	statistic	df	p	p.signif
MBApercent	F	M	76	139	4.725214	166.8781	4.9e-06	****

Female again score significantly higher than their male counterpart. With a p-value significantly lower could mean they have a higher probability of being placed. **The null hypothesis is again rejected by this result.**

3. Employee Percentage Score

```
>  t_test_emptest <- data%>%t_test(EmpTestPercent ~ Gender)%>%add_significance()
>
> data %>%
+   ggplot(aes(EmpTestPercent, fill = Gender, col = Gender))+
+   geom_density(alpha = 0.3, lwd = 1, show.legend = FALSE)+
+   geom_rug()+
+   labs(title = paste("Females have a significantly higher score
+   (", t_test_emptest$p.signif, ") than males at the Employee Test", sep = ""),
+        col = "Gender",
+        x = "Employee Percentage Score",
+        y = "Density")+
+        theme_few()+
+   theme(legend.position = "right")
> 
```



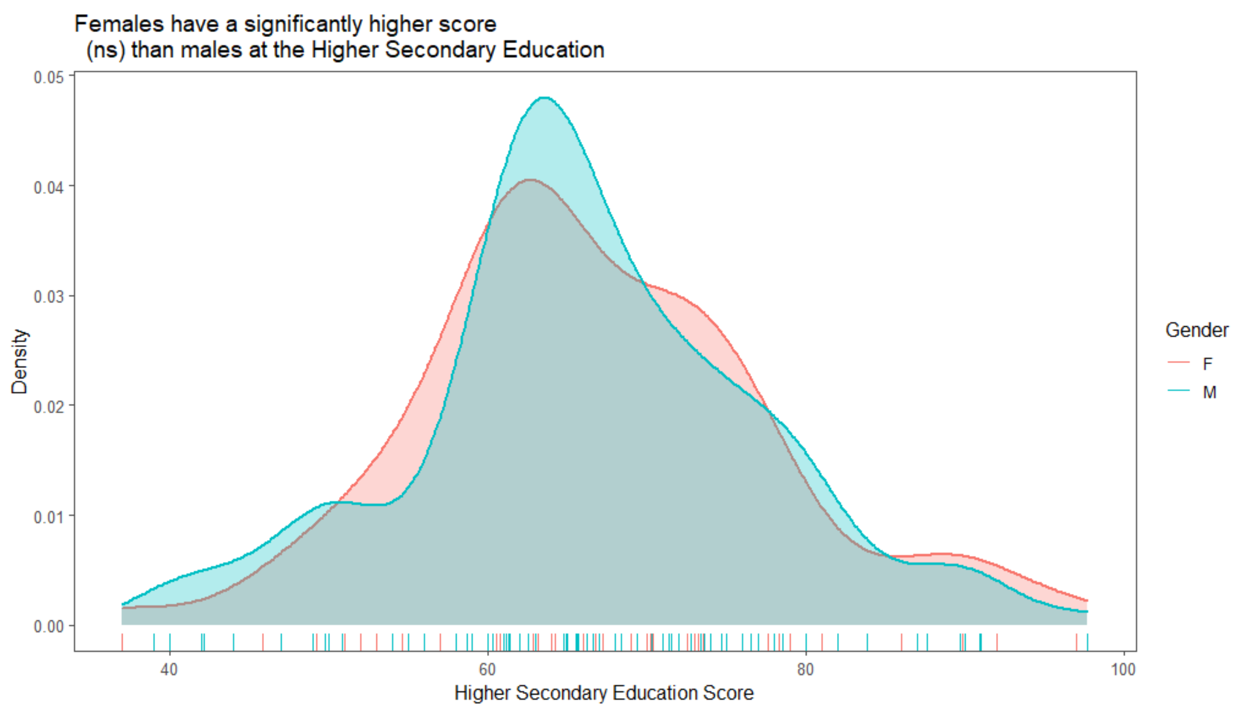

```
> t_test_emptest%>%
+   kbl()%>%
+   kable_paper("hover", full_width = F)
>
```

.y.	group1	group2	n1	n2	statistic	df	p	p.signif
EmpTestPercent	F	M	76	139	-1.231058	153.0676	0.22	ns

We don’t have enough statistical significance to reject the null hypothesis.

4. Higher Secondary Education Percentage

```
>  t_test_hse <- data%>%t_test(HigherSecEduPercent ~ Gender)%>%add_significance()
>
> data %>%
+   ggplot(aes(HigherSecEduPercent, fill = Gender, col = Gender))+
+   geom_density(alpha = 0.3, lwd = 1, show.legend = FALSE)+
+   geom_rug()+
+   labs(title = paste("Females have a significantly higher score
+   (", t_test_hse$p.signif, ") than males at the Higher Secondary Education", sep = ""),
+        col = "Gender",
+        x = "Higher Secondary Education Score",
+        y = "Density")+
+        theme_few()+
+   theme(legend.position = "right")
>
```

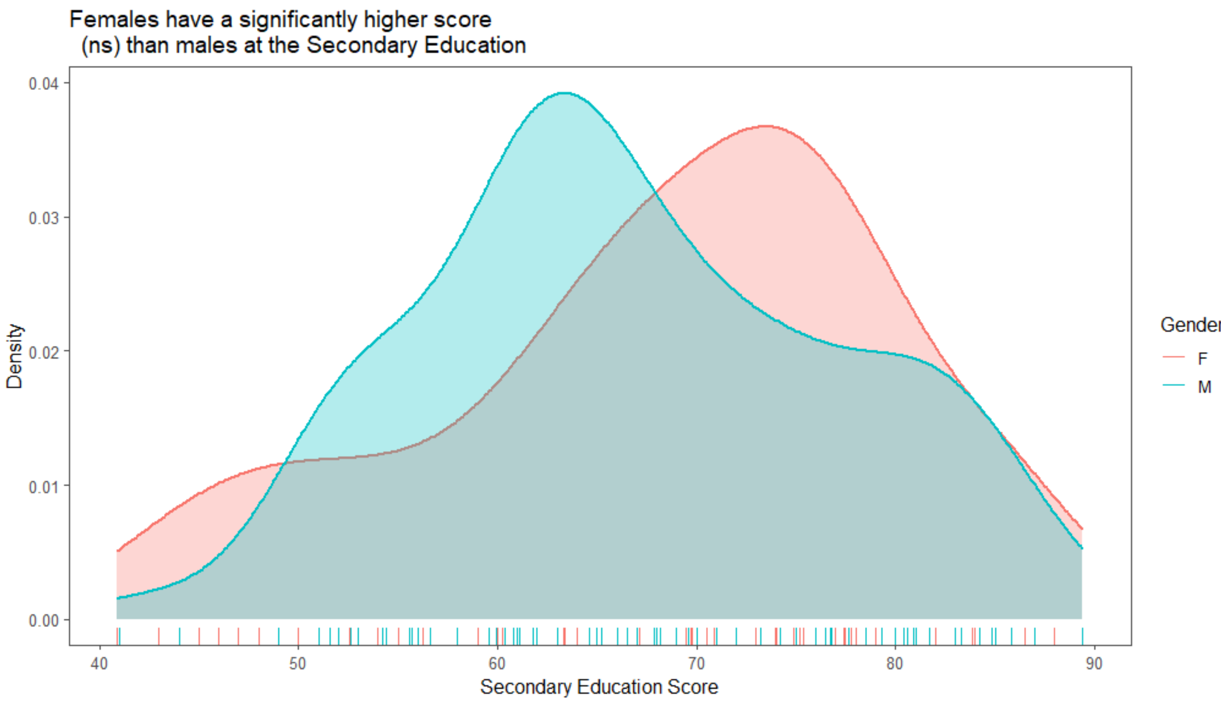



```
> t_test_hse%>%
+   kbl()%>%
+   kable_paper("hover", full_width = F)
>
```

.y.	group1	group2	n1	n2	statistic	df	p	p.signif
HigherSecEduPercent	F	M	76	139	0.3101292	152.4923	0.757	ns

5. Secondary Education Percentage

```
> t_test_se <- data%>%t_test(SecEducationPercent ~ Gender)%>%add_significance()
>
> data %>%
+   ggplot(aes(SecEducationPercent, fill = Gender, col = Gender))+
+   geom_density(alpha = 0.3, lwd = 1, show.legend = FALSE)+
+   geom_rug()+
+   labs(title = paste("Females have a significantly higher score
+   (", t_test_hse$p.signif, ") than males at the Secondary Education", sep = ""),
+        col = "Gender",
+        x = "Secondary Education Score",
+        y = "Density")+
+        theme_few()+
+   theme(legend.position = "right")
>
```




```
> t_test_se%>%
+   kbl()%>%
+   kable_paper("hover", full_width = F)
>
```

.y.	group1	group2	n1	n2	statistic	df	p	p.signif
SecEducationPercent	F	M	76	139	0.9798646	141.7762	0.329	ns

From the statistical analysis of gender on performance of each student across the academic levels, it could be observed that:

- Females have a significantly higher score than males at the University Degree and MBA levels.
- There was no significant difference in gender performance at Secondary, Higher Secondary, and Employability Test.

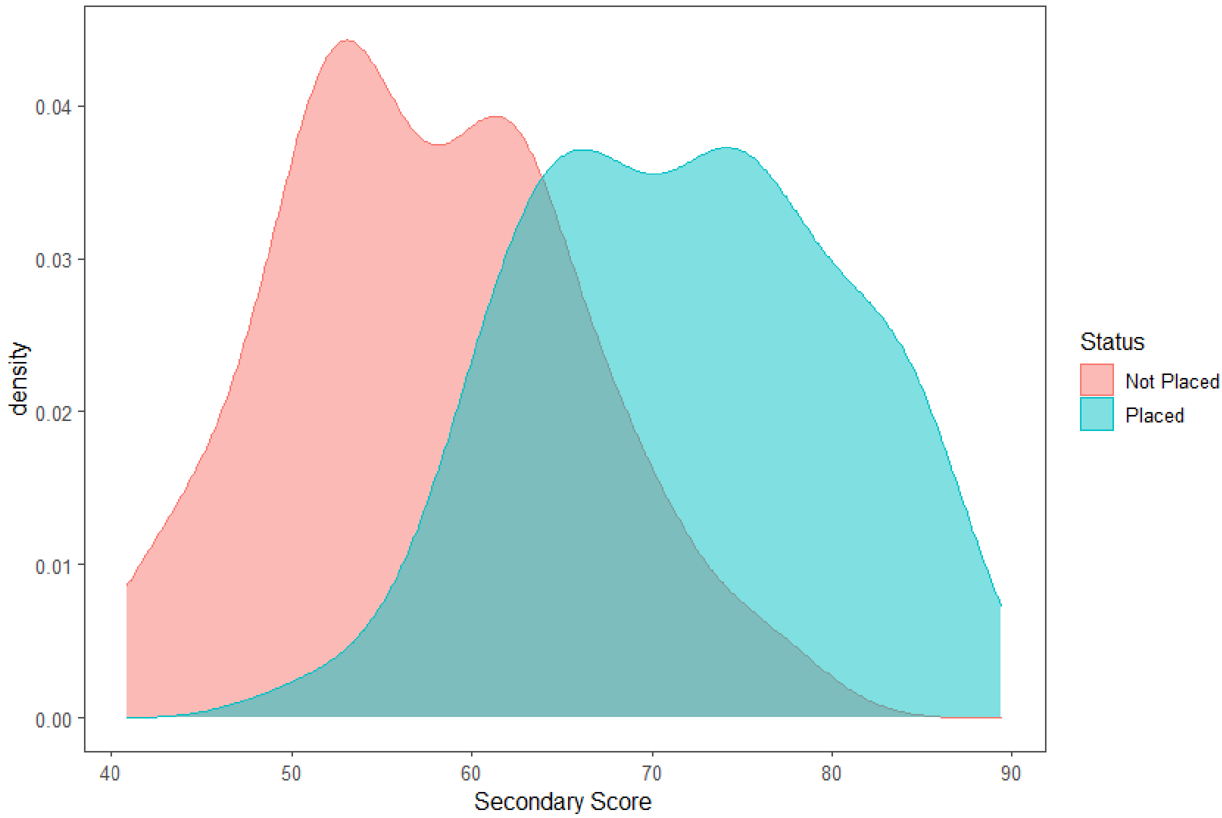
The result was supported by the p-value for each academic level.

2.4 Academic Performance Impact of Placement using T-test

It is expected that academic performance should have the most influence on a student getting placed. The earlier visualization on score distribution showed that the performance of student on the employability test was average, from this analysis we should have a clear understanding which level of education has influence on student placement.

1. Secondary Education

```
> t_test <- data%>%
+   t_test(SecEducationPercent ~ Status)%>%
+   add_significance()
>
> data %>%
+   ggplot( aes(x=SecEducationPercent, fill=Status, color=Status)) +
+   geom_density(alpha=0.5)+
+   labs(x = "Secondary Score")+
+   theme_few()
>
```

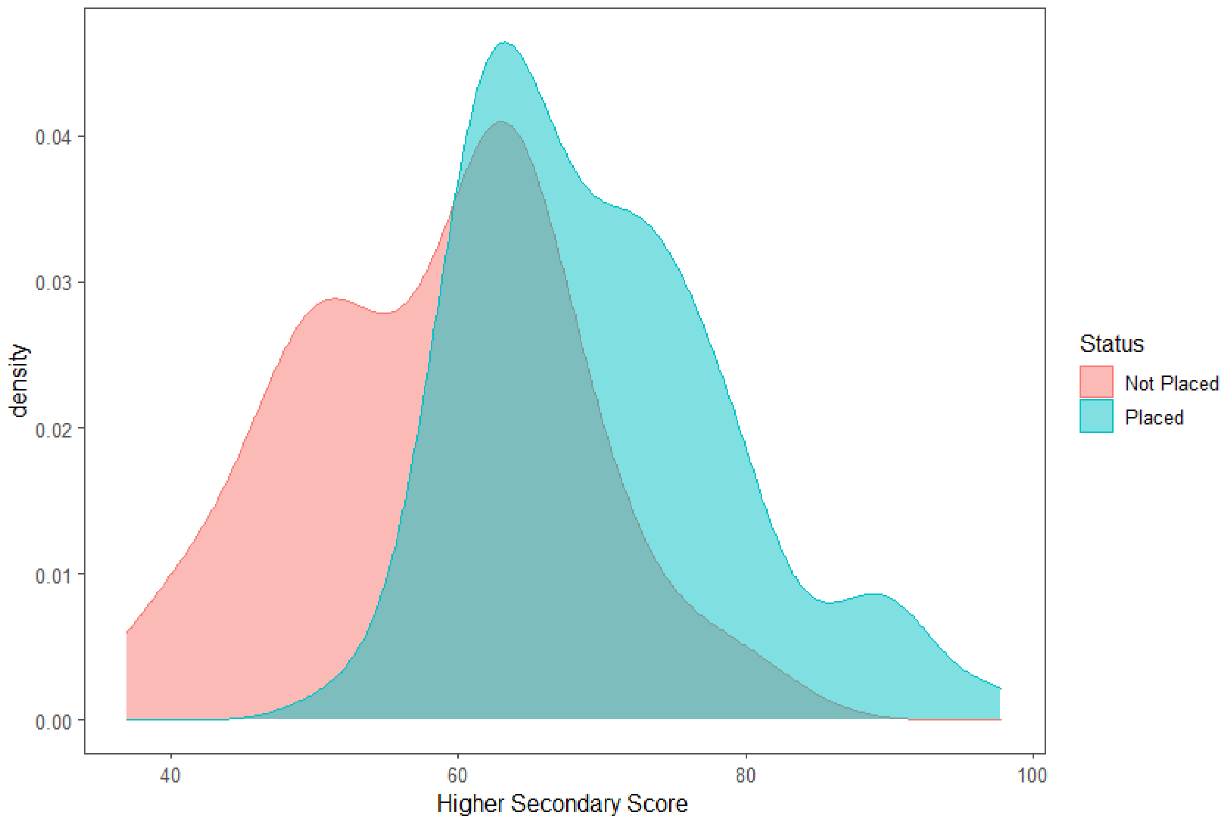



```
> t_test%>%
+   kbl()%>%
+   kable_paper("hover", full_width = F)
> 
```

.y.	group1	group2	n1	n2	statistic	df	p	p.signif
SecEducationPercent	Not Placed	Placed	67	148	-11.33316	132.0192	0	****

2. Higher Secondary Education

```
> t_test%>%
+   kbl()%>%
+   kable_paper("hover", full_width = F)
> t_test_hsc <- data%>%
+   t_test(HigherSecEduPercent ~ Status)%>%
+   add_significance()
>
> data %>%
+   ggplot( aes(x=HigherSecEduPercent, fill=Status, color=Status)) +
+   geom_density(alpha=0.5)+
+   labs(x = "Higher Secondary Score")+
+   theme_few()
> 
```

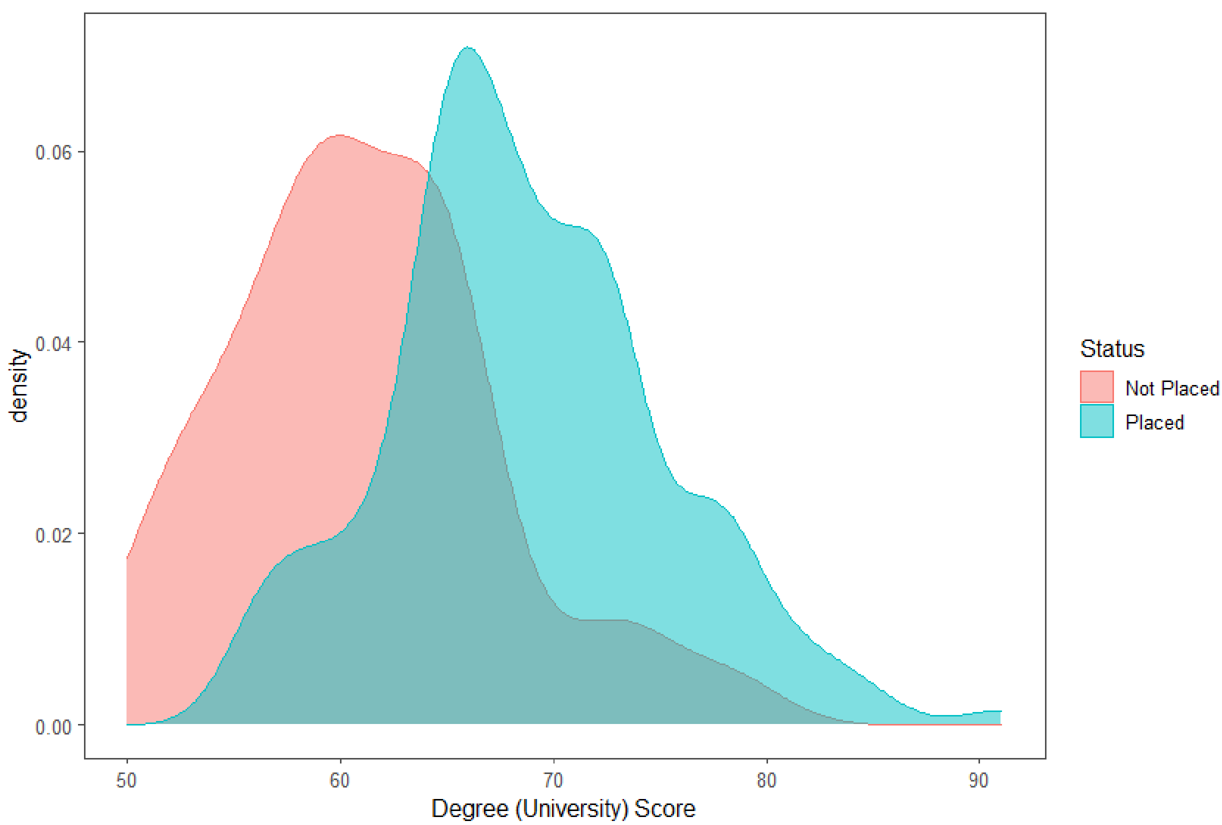



```
> t_test_hsc%>%
+   kbl()%>%
+   kable_paper("hover", full_width = F)
> 
```

.y.	group1	group2	n1	n2	statistic	df	p	p.signif
HigherSecEduPercent	Not Placed	Placed	67	148	-8.043664	120.8047	0	****

3. University Degree Score

```
> t_test_uni <- data%>%
+   t_test(DegreePercent ~ Status)%>%
+   add_significance()
>
> data %>%
+   ggplot( aes(x=DegreePercent, fill=Status, color=Status)) +
+   geom_density(alpha=0.5)+
+   labs(x = "Degree (University) Score")+
+   theme_few()
>
```

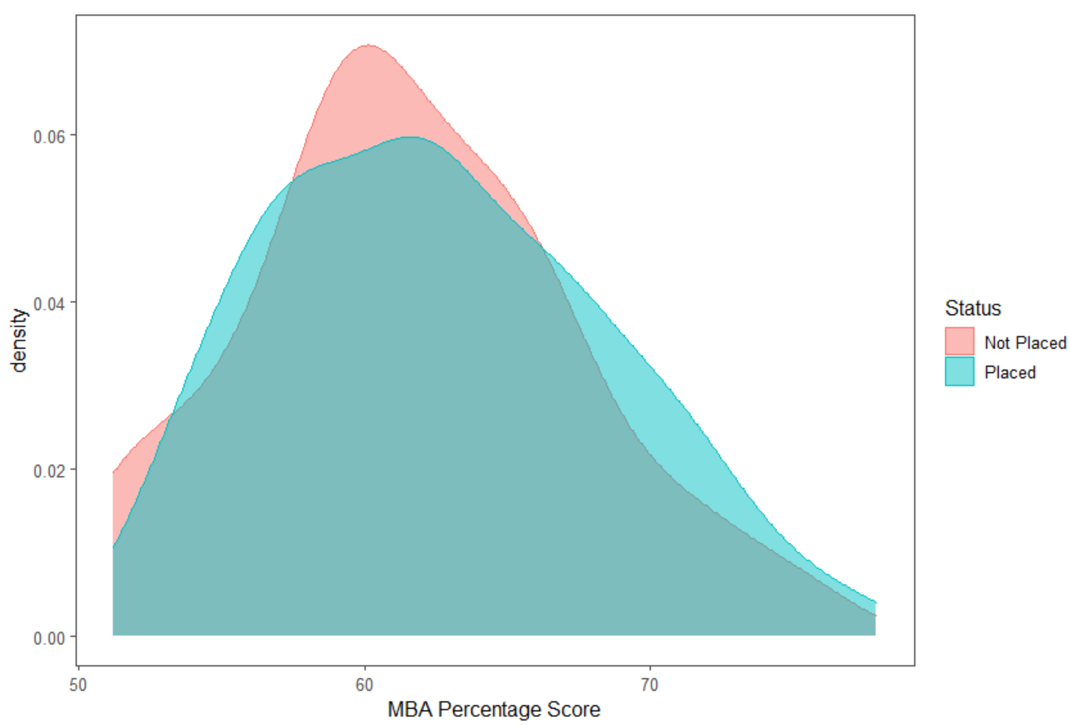


```
> t_test_uni%>%
+   kbl()%>%
+   kable_paper("hover", full_width = F)
> 
```

.y.	group1	group2	n1	n2	statistic	df	p	p.signif
DegreePercent	Not Placed	Placed	67	148	-8.054153	130.335	0	****

4. MBA Score

```
> t_test_mba <- data%>%
+   t_test(MBApercent ~ Status)%>%
+   add_significance()
> 
> data %>%
+   ggplot( aes(x=MBApercent, fill=Status, color=Status)) +
+   geom_density(alpha=0.5)+
+   labs(x = "MBA Percentage Score")+
+   theme_few()
> 
```

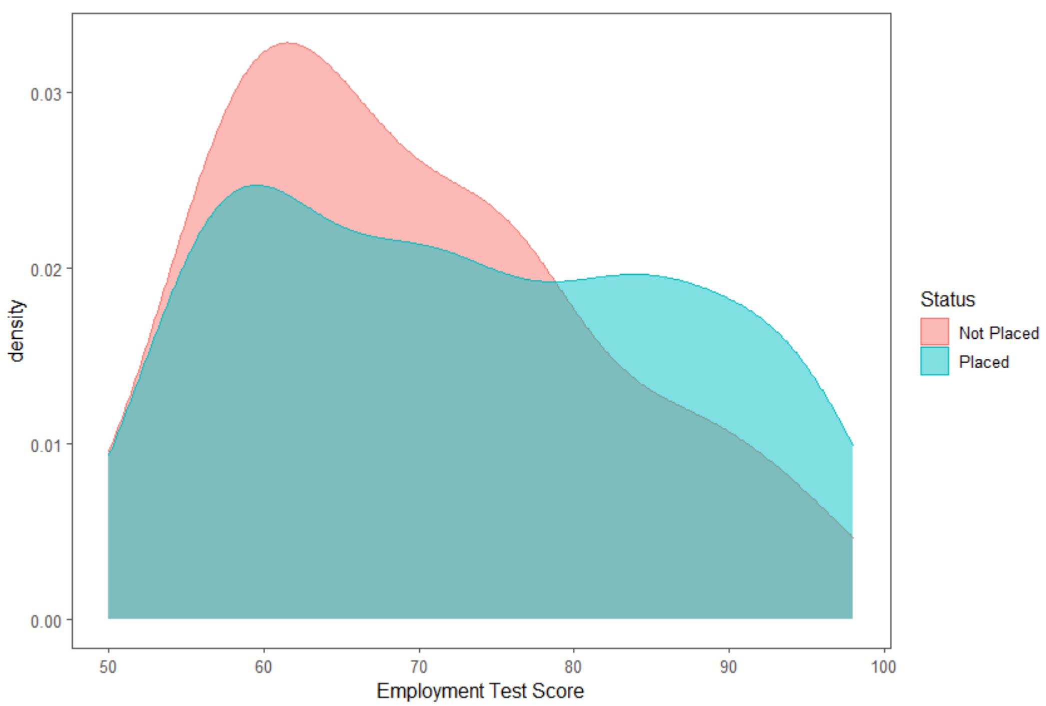


```
> t_test_mba%>%
+   kbl()%>%
+   kable_paper("hover", full_width = F)
> 
```

.y.	group1	group2	n1	n2	statistic	df	p	p.signif
MBApercent	Not Placed	Placed	67	148	-1.139201	131.2069	0.257	ns

5. Employability Test Score

```
> t_test_emp <- data%>%
+   t_test(EmpTestPercent ~ Status)%>%
+   add_significance()
> 
> data %>%
+   ggplot( aes(x=EmpTestPercent, fill=Status, color=Status)) +
+   geom_density(alpha=0.5)+
+   labs(x = "Employment Test Score")+
+   theme_few()
> 
```



```
> t_test_emp%>%
+   kbl()%>%
+   kable_paper("hover", full_width = F)
>
```

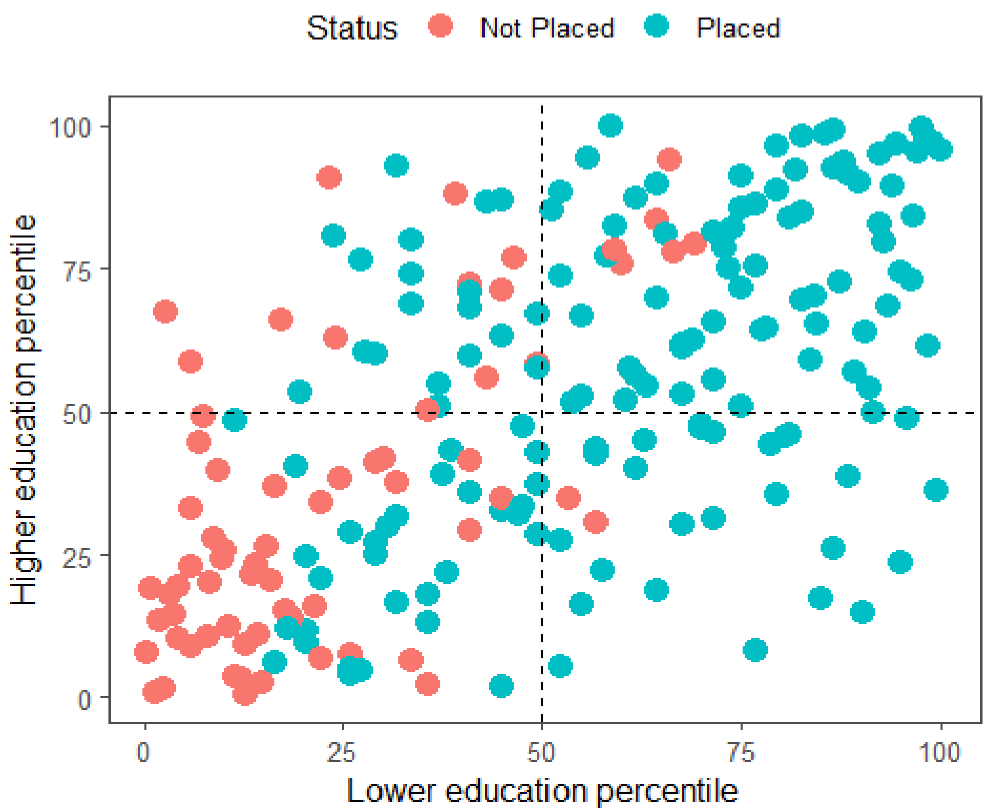
.y.	group1	group2	n1	n2	statistic	df	p	p.signif
EmpTestPercent	Not Placed	Placed	67	148	-1.980112	145.392	0.0496	*

Looking at the p-value score and analyzing the p-significance for each education level, it could be observed that:

- The score differences between those who received an offer and those that do not were highly significant at Secondary, Higher Secondary, and University Education Level.
- The Employability Test has a significant score
- But the MBA score is not significant.

From the above statistical analysis, we could assume that lower-level education has more influence than higher-level education on the placement outcome. Hence, we plot a scatter plot of lower-level education and higher-level education visualize the score distribution across the percentile.

```
> #Overall education percentile
>
> data %>%
+   mutate(lower_education = (SecEducationPercent+HigherSecEduPercent)/2,
+          higher_education = (DegreePercent+MBApercent)/2,
+          all_education = (SecEducationPercent+HigherSecEduPercent+DegreePercent+MBApercent)/4,
+          percentile_lower = round(rank(lower_education)/n()*100,2),
+          percentile_higher = round(rank(higher_education)/n()*100,2),
+          percentile_all_education = round(rank(all_education)/n()*100,2))%>%
+   ggplot(aes(percentile_lower, percentile_higher, col = Status))+
+   geom_point(aes(col = Status), size = 4)+
+   geom_vline(xintercept =50, lty = 2)+
+   geom_hline(yintercept =50, lty = 2)+
+   labs(x = "Lower education percentile",
+        y = "Higher education percentile",
+        col = "Status")+
+   theme_few()+
+   theme(legend.position = "top")
>
```



The following was observed:

- Almost all the students that did well in the lower education were placed; all the students in the top 25th percentile received placement regardless of their higher education performance.
- Majority of the student in the bottom 25th percentile across the lower education did not receive a placement.
- The hypothesis that performances in both Secondary and Higher Secondary Education having more influence on a student getting placed is supported by this analysis.

CHAPTER 3: PREPROCCESSING

3.1 Splitting the data

Now that the structure of the data is clear, we will proceed to split the data into training and testing set. We used the stratified split approach because we want even distribution of the variable Status in both the training and testing set.

The variable Salary will be dropped because it is not needed in our analysis, it can only be referenced after a placement offer has been received.

The caret library package was loaded.

```
> #Splitting the data
> #clean the model data and drop the salary feature
> data_mod <- data %>%
+   select(-Salary)%>%
+   mutate(Status = as.factor(make.names(Status)))
> set.seed(100)  # good idea to set the random seed for reproducibility
> partition <- createDataPartition(data_mod$Status, p = 0.7, list=FALSE)
> 
```

We partitioned the data in train and test set, with training set having 70% and testing set 30%.

Train_data table is the training set and Test_data table is the testing set.

```
> train_data <- data_mod[partition,]
> test_data <- data_mod[-partition,]
> train_data <- select (train_data, -c(sl_no))
> test_data <- select (test_data, -c(sl_no))
> 
```

Now we have 13 variables, the 12 predictors and the response variable. Subset selection method will be applied on the 12 variables to choose that with the best model output.

```
> colnames(train_data)
 [1] "Gender"             "SecEducationPercent" "SecBoardofEducation" "HigherSecEduPercent" "HigherSecBoardofEdu"
 [6] "HigherSecSpecial"   "DegreePercent"       "DegreeType"          "WorkExp"             "EmpTestPercent"
[11] "MBAspec"            "MBApercent"          "Status"
> 
```

3.2 Scaling the data

The data was scaled using the below code.

The three options c(“center”, “scale”, “nzv”) does scale and center. Method = “center” subtracts the mean of the predictor’s data from the predictor values while method = “scale” divides by the standard deviation. And “nzv” basically excludes variables that have near zero variance, meaning they are almost constant and most likely not useful for prediction.

```
> #scaling the data
> scaled_centered <- preProcess(train_data, method=c('center', 'scale', 'nzv'))
>
> train_data <- predict(scaled_centered, newdata = train_data)
> test_data <- predict(scaled_centered, newdata = test_data)
>
```

3.3 Subset Selection

To select the best model for further analysis, we will be implementing 3 subset selection approach to select the best variable.

1. Forward selection

```
> datafit.fwd <- regsubsets(Status ~., data = train_data, nvmax=13, method = 'forward')
> summary(datafit.fwd)
Subset selection object
Call: regsubsets.formula(Status ~ ., data = train_data, nvmax = 13,
    method = "forward")
14 Variables  (and intercept)
                           Forced in Forced out
GenderM                        FALSE      FALSE
SecEducationPercent            FALSE      FALSE
SecBoardofEducationOthers      FALSE      FALSE
HigherSecEduPercent            FALSE      FALSE
HigherSecBoardofEduOthers      FALSE      FALSE
HigherSecSpecialCommerce       FALSE      FALSE
HigherSecSpecialScience        FALSE      FALSE
DegreePercent                  FALSE      FALSE
DegreeTypeOthers               FALSE      FALSE
DegreeTypeSci&Tech             FALSE      FALSE
WorkExpYes                     FALSE      FALSE
EmpTestPercent                 FALSE      FALSE
MBAspecMkt&HR                  FALSE      FALSE
MBApercent                     FALSE      FALSE
1 subsets of each size up to 13
Selection Algorithm: forward
```

```
> coef(datafit.fwd, 7)
       (Intercept)            GenderM SecEducationPercent HigherSecEduPercent       DegreePercent  DegreeTypeSci&Tech
        1.60964401         0.10826521          0.24925404          0.08230178          0.12656824         -0.18375425
         WorkExpYes          MBApercent
        0.18847339        -0.14316827
```

2. Backward selection

```
> #Backward
> datafit.bkd <- regsubsets(Status ~., data = train_data, nvmax=13, method = 'backward')
> summary(datafit.bkd)
Subset selection object
Call: regsubsets.formula(Status ~ ., data = train_data, nvmax = 13,
    method = "backward")
14 Variables  (and intercept)
                          Forced in Forced out
GenderM                       FALSE      FALSE
SecEducationPercent           FALSE      FALSE
SecBoardofEducationOthers     FALSE      FALSE
HigherSecEduPercent           FALSE      FALSE
HigherSecBoardofEduOthers     FALSE      FALSE
HigherSecSpecialCommerce      FALSE      FALSE
HigherSecSpecialScience       FALSE      FALSE
DegreePercent                 FALSE      FALSE
DegreeTypeOthers              FALSE      FALSE
DegreeTypeSci&Tech            FALSE      FALSE
WorkExpYes                    FALSE      FALSE
EmpTestPercent                FALSE      FALSE
MBAspecMkt&HR                 FALSE      FALSE
MBApercent                    FALSE      FALSE
1 subsets of each size up to 13
Selection Algorithm: backward
```

```
> coef(datafit.bkd, 7)
       (Intercept)            GenderM SecEducationPercent HigherSecEduPercent       DegreePercent  DegreeTypeSci&Tech
        1.60964401         0.10826521          0.24925404          0.08230178          0.12656824         -0.18375425
         WorkExpYes          MBApercent
        0.18847339        -0.14316827
```

3. Best subset selection

```
> #Best Subset Selection
> datafit.full <- regsubsets(Status ~., data = train_data, nvmax=13)
> summary(datafit.full)
Subset selection object
Call: regsubsets.formula(Status ~ ., data = train_data, nvmax = 13)
14 Variables  (and intercept)
                           Forced in Forced out
GenderM                        FALSE      FALSE
SecEducationPercent            FALSE      FALSE
SecBoardofEducationOthers      FALSE      FALSE
HigherSecEduPercent            FALSE      FALSE
HigherSecBoardofEduOthers      FALSE      FALSE
HigherSecSpecialCommerce       FALSE      FALSE
HigherSecSpecialScience        FALSE      FALSE
DegreePercent                  FALSE      FALSE
DegreeTypeOthers               FALSE      FALSE
DegreeTypeSci&Tech             FALSE      FALSE
WorkExpYes                     FALSE      FALSE
EmpTestPercent                 FALSE      FALSE
MBAspecMkt&HR                  FALSE      FALSE
MBApercent                     FALSE      FALSE
1 subsets of each size up to 13
Selection Algorithm: exhaustive
```

```
> coef(datafit.full, 7)
 (Intercept)      GenderM SecEducationPercent HigherSecEduPercent      DegreePercent DegreeTypeSci&Tech
  1.60964401   0.10826521          0.24925404          0.08230178         0.12656824        -0.18375425
  WorkExpYes   MBApercent
  0.18847339  -0.14316827
>
```

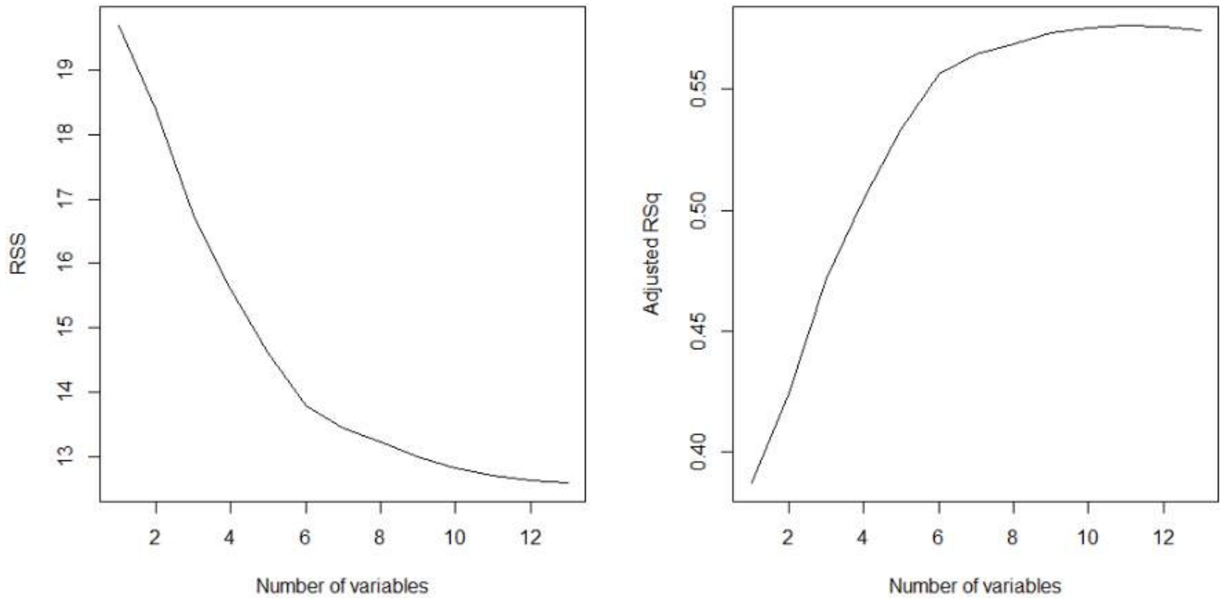
For this dataset, the best one-variable through seven-variable models are identical for forward selection, backward selection, and best subset.

EVALUATING SUBSET SELECTION MODELS

RSS and Adjusted R-squared

However, plotting the RSS and Adjusted R-squared for the forward selection models shows that the best model uses 11 to 13 variables.

```
> par(mfrow = c(1,2))
> plot(fwd.summary$rss, xlab = 'Number of variables',
+   ylab = 'RSS', type = 'l')
> plot(fwd.summary$adjr2, xlab = 'Number of variables',
+   ylab = 'Adjusted RSq', type = 'l')
```



We will printed out the RSS and Adjusted R-Squared values for the three models for comparison.

RSS values

```
> #Forward
> fwd.summary <- summary(datafit.fwd)
> fwd.summary$rss
 [1] 19.69887 18.38458 16.75791 15.60831 14.61266 13.78693 13.43211 13.21887 12.98568 12.82341 12.70744 12.63121 12.59218
> #Backward
> bkd.summary <- summary(datafit.bkd)
> bkd.summary$rss
 [1] 19.69887 18.69423 17.05732 15.78206 14.58517 13.78693 13.43211 13.21887 12.98568 12.82341 12.70744 12.63121 12.59218
> #Best Subset
> full.summary <- summary(datafit.full)
> full.summary$rss
 [1] 19.69887 18.38458 16.75791 15.60831 14.58517 13.78693 13.43211 13.21887 12.98568 12.82341 12.70744 12.63121 12.59218
> 
```

Adjusted R-Squared values

```
> #Forward
> fwd.summary <- summary(datafit.fwd)
> fwd.summary$adjr2
 [1] 0.3873789 0.4243890 0.4717500 0.5046183 0.5330199 0.5563485 0.5647435 0.5686371 0.5732410 0.5755638 0.5763761 0.5758663
[13] 0.5740904
> #Backward
> bkd.summary <- summary(datafit.bkd)
> bkd.summary$adjr2
 [1] 0.3873789 0.4146941 0.4623119 0.4991036 0.5338985 0.5563485 0.5647435 0.5686371 0.5732410 0.5755638 0.5763761 0.5758663
[13] 0.5740904
> #Best Subset
> full.summary <- summary(datafit.full)
> full.summary$adjr2
 [1] 0.3873789 0.4243890 0.4717500 0.5046183 0.5338985 0.5563485 0.5647435 0.5686371 0.5732410 0.5755638 0.5763761 0.5758663
[13] 0.5740904
> 
```

The above output shows that the three models predicted similar models. The least RSS is in 13 variable model and the highest adjusted r-squared is in 11 variable model. Hence, we will be using the 12 predictors for my prediction.

CHAPTER 4: PREDICTIONS

4.1. Resampling

We will perform a 10-fold Cross Validation, use each of the 10 parts as a testing set for the and train on the remaining 9 parts.

```
> #Resampling
> #10-fold Cross Validation
> set.seed(100)
> ctrl <- trainControl(method = 'cv', number =10, savePredictions = TRUE)
> 
```

4.2. Logistic Regression

We fit a logistic regression model to predict the Status variable using all the 12 predictors. We will use the function glm to fit the model and specify family = binomial so that R can can fit a logistic regression.

```
> #Logistics Regression
> log_model <- train(Status ~ ., data=train_data, method='glm', trControl=ctrl, tuneLength=5, family = binomial)
> #Summary
> summary(log_model)

Call:
NULL

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.09833  -0.05867   0.02457   0.21189   2.13569

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)                 3.1755     2.8419   1.117  0.26383
GenderM                     1.8103     1.0452   1.732  0.08327 .
SecEducationPercent         4.0205     1.0152   3.960 7.48e-05 ***
SecBoardofEducationOthers   0.2644     1.0234   0.258  0.79613
HigherSecEduPercent         1.2866     0.6341   2.029  0.04247 *
HigherSecBoardofEduOthers   1.0552     1.1840   0.891  0.37282
HigherSecSpecialCommerce   -3.1349     2.7658  -1.133  0.25702
HigherSecSpecialScience    -1.4814     2.7784  -0.533  0.59390
DegreePercent               1.6390     0.6787   2.415  0.01574 *
DegreeTypeOthers           -2.0816     2.6797  -0.777  0.43728
`DegreeTypeSci&Tech`       -3.7373     1.4342  -2.606  0.00916 **
WorkExpYes                  2.8383     1.0415   2.725  0.00642 **
EmpTestPercent             -0.4384     0.4770  -0.919  0.35805
`MBAspecMkt&HR`             0.4294     0.8406   0.511  0.60945
MBApercent                 -1.5310     0.5394  -2.839  0.00453 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 187.271  on 150  degrees of freedom
Residual deviance:  52.299  on 136  degrees of freedom
AIC: 82.299

Number of Fisher Scoring iterations: 8
```

The smallest p-value is SecEducationPercent with a positive coefficient. It is an indication that students that usually get placed have a high score in secondary education.

To check the accuracy of our model on the training set.

```
> #Accuracy
> log_model$results
  parameter  Accuracy     Kappa AccuracySD   KappaSD
1      none 0.8791667 0.7044773 0.04689261 0.1560664
```

Our accuracy on the training set is 0.8792, kappa is 0.7044.

We will now apply the model to make prediction using the test set, we then obtain the probability between being Placed and Not Placed, add the logistic regression prediction to the test_data and output the confusion matrix for the prediction.

Logistic Regression accuracy on the test_data set is 0.8281 and kappa is 0.6054

```
> #add prediction column to test dataset
> test_data$log <- predict(log_model, newdata=test_data)
> view(test_data)
> #get probabilities
> head(predict(log_model, newdata = test_data, type = 'prob'))
    Not.Placed      Placed
1 0.9703796373 0.029620363
2 0.0019991698 0.998000830
3 0.9984182647 0.001581735
4 0.0002783922 0.999721608
5 0.5225734953 0.477426505
6 0.0010791641 0.998920836
> log_preds <- predict(log_model, newdata = test_data)
> log_cm <- confusionMatrix(log_preds, test_data$Status)
> log_cm
Confusion Matrix and Statistics

            Reference
Prediction   Not.Placed Placed
  Not.Placed         15      6
  Placed              5     38

               Accuracy : 0.8281
                 95% CI : (0.7132, 0.911)
    No Information Rate : 0.6875
    P-Value [Acc > NIR] : 0.008412

                  Kappa : 0.6054

 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.7500
            Specificity : 0.8636
         Pos Pred Value : 0.7143
         Neg Pred Value : 0.8837
             Prevalence : 0.3125
         Detection Rate : 0.2344
   Detection Prevalence : 0.3281
      Balanced Accuracy : 0.8068

       'Positive' Class : Not.Placed

> 
```

4.3. Linear Discriminant Analysis

```
> #Linear Discrimeanant Analysis
> lda_model <- train(Status ~ ., data = train_data, method = 'lda', trControl = ctrl, tuneLength=5)
> #Summary
> summary(lda_model)
            Length Class      Mode
prior        2     -none-     numeric
counts       2     -none-     numeric
means       28     -none-     numeric
scaling     14     -none-     numeric
lev          2     -none-     character
svd          1     -none-     numeric
N            1     -none-     numeric
call         3     -none-     call
xNames      14     -none-     character
problemType  1     -none-     character
tuneValue    1     data.frame list
obsLevels    2     -none-     character
param        0     -none-     list
> #names(lda_model)
> lda_model$results
  parameter  Accuracy     Kappa AccuracySD   KappaSD
1      none 0.8940476 0.7443699 0.09786876 0.2439987
> |
```

Accuracy on training set is 0.8940, Kappa is 0.7444

We will now apply the model to make prediction using the test set, we then obtain the probability between being Placed and Not Placed, add the LDA prediction to the test_data and output the confusion matrix for the prediction.

```
> test_data$lda <- predict(lda_model, newdata = test_data)
> view(test_data)
> head(predict(lda_model, newdata=test_data, type='prob'))
   Not.Placed      Placed
1 0.850727392 0.149272608
2 0.003579500 0.996420500
3 0.997794418 0.002205582
4 0.002283610 0.997716390
5 0.717115328 0.282884672
6 0.005529019 0.994470981
> lda_prob <- predict(lda_model, newdata=test_data, type='prob')*100
> lda_pred <- predict(lda_model, newdata = test_data)
> lda_cm <- confusionMatrix(lda_pred, test_data$Status)
> lda_cm
Confusion Matrix and Statistics

            Reference
Prediction   Not.Placed Placed
  Not.Placed         16      5
  Placed              4     39

               Accuracy : 0.8594
                 95% CI : (0.7498, 0.9336)
    No Information Rate : 0.6875
    P-Value [Acc > NIR] : 0.001317

                  Kappa : 0.6771

 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.8000
            Specificity : 0.8864
         Pos Pred Value : 0.7619
         Neg Pred Value : 0.9070
             Prevalence : 0.3125
         Detection Rate : 0.2500
   Detection Prevalence : 0.3281
      Balanced Accuracy : 0.8432

       'Positive' Class : Not.Placed
```

LDA model accuracy on the test set is 0.8594 and kappa is 0.6771. It is higher than the logistic regression accuracy.

4.4. Generalize Linear Model

```
> #Generalized Linear Model
> glm_model <- train(Status ~ ., data=train_data, method='glm', trControl=ctrl, tuneLength=5)
> #Summary
> summary(glm_model)

Call:
NULL

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.09833  -0.05867   0.02457   0.21189   2.13569

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)                 3.1755     2.8419   1.117  0.26383
GenderM                     1.8103     1.0452   1.732  0.08327 .
SecEducationPercent         4.0205     1.0152   3.960 7.48e-05 ***
SecBoardofEducationOthers   0.2644     1.0234   0.258  0.79613
HigherSecEduPercent         1.2866     0.6341   2.029  0.04247 *
HigherSecBoardofEduOthers   1.0552     1.1840   0.891  0.37282
HigherSecSpecialCommerce   -3.1349     2.7658  -1.133  0.25702
HigherSecSpecialScience    -1.4814     2.7784  -0.533  0.59390
DegreePercent               1.6390     0.6787   2.415  0.01574 *
DegreeTypeOthers           -2.0816     2.6797  -0.777  0.43728
`DegreeTypeSci&Tech`       -3.7373     1.4342  -2.606  0.00916 **
WorkExpYes                  2.8383     1.0415   2.725  0.00642 **
EmpTestPercent             -0.4384     0.4770  -0.919  0.35805
`MBAspecMkt&HR`             0.4294     0.8406   0.511  0.60945
MBApercent                 -1.5310     0.5394  -2.839  0.00453 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 187.271  on 150  degrees of freedom
Residual deviance:  52.299  on 136  degrees of freedom
AIC: 82.299

Number of Fisher Scoring iterations: 8
```

The training accuracy was 0.8761, kappa was 0.7138 and testing accuracy was 0.8281, kappa was 0.6054.

```
> #Accuracy
> glm_model$results
  parameter Accuracy     Kappa AccuracySD   KappaSD
1      none 0.876131 0.7138233 0.08681251 0.1824966
> #add prediction column to test dataset
> test_data$glm <- predict(glm_model, newdata=test_data)
> view(test_data)
> #get probabilities
> head(predict(glm_model, newdata = test_data, type = 'prob'))
    Not.Placed      Placed
1 0.9703796373 0.029620363
2 0.0019991698 0.998000830
3 0.9984182647 0.001581735
4 0.0002783922 0.999721608
5 0.5225734953 0.477426505
6 0.0010791641 0.998920836
> glm_preds <- predict(glm_model, newdata = test_data)
> glm_cm <- confusionMatrix(glm_preds, test_data$Status)
> glm_cm
Confusion Matrix and Statistics

            Reference
Prediction   Not.Placed Placed
  Not.Placed         15      6
  Placed              5     38

               Accuracy : 0.8281
                 95% CI : (0.7132, 0.911)
    No Information Rate : 0.6875
    P-Value [Acc > NIR] : 0.008412

                  Kappa : 0.6054

 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.7500
            Specificity : 0.8636
         Pos Pred Value : 0.7143
         Neg Pred Value : 0.8837
             Prevalence : 0.3125
         Detection Rate : 0.2344
   Detection Prevalence : 0.3281
      Balanced Accuracy : 0.8068

       'Positive' Class : Not.Placed

>
```

4.5. Support Vector Machine: Linear

We will try one more linear algorithm to see if our result will improve.

```
> #Support Vector Machine: Linear
> svm_model <- train(Status ~ ., data = train_data, method = 'svmLinear', trControl = ctrl, tuneLength=5)
> #Summary
> summary(svm_model)
Length  Class   Mode
     1   ksvm     S4
> #names(svm_model)
> svm_model$results
  C  Accuracy     Kappa AccuracySD   KappaSD
1 1 0.8882738 0.729275 0.08234153 0.2109681
>
```

Training accuracy is 0.8883 and Kappa is 0.7293

Testing accuracy is 0.8594 and Kappa is 0.6771

The Linear SVM model and LDA model both have the same accuracy on the testing set.

```
> test_data$svm <- predict(svm_model, newdata = test_data)
> view(test_data)
> head(predict(svm_model, newdata=test_data, type='prob'))
  Not.Placed Placed
1         NA     NA
2         NA     NA
3         NA     NA
4         NA     NA
5         NA     NA
6         NA     NA
Warning message:
In method$prob(modelFit = modelFit, newdata = newdata, submodels = param) :
  kernlab class probability calculations failed; returning NAs
> svm_prob <- predict(svm_model, newdata=test_data, type='prob')*100
Warning message:
In method$prob(modelFit = modelFit, newdata = newdata, submodels = param) :
  kernlab class probability calculations failed; returning NAs
> svm_pred <- predict(svm_model, newdata = test_data)
> svm_cm <- confusionMatrix(svm_pred, test_data$Status)
> svm_cm
Confusion Matrix and Statistics

            Reference
Prediction   Not.Placed Placed
  Not.Placed         16      5
  Placed              4     39

               Accuracy : 0.8594
                 95% CI : (0.7498, 0.9336)
    No Information Rate : 0.6875
    P-Value [Acc > NIR] : 0.001317

                  Kappa : 0.6771

 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.8000
            Specificity : 0.8864
         Pos Pred Value : 0.7619
         Neg Pred Value : 0.9070
             Prevalence : 0.3125
         Detection Rate : 0.2500
   Detection Prevalence : 0.3281
      Balanced Accuracy : 0.8432

       'Positive' Class : Not.Placed

>
```

To improve the accuracy on the testing data, we shall use more flexible models.

4.6.Quadratic Linear Discriminant

The result of the model on the training set shows that QDA performed poorly when compared to the linear models.

```
> #Quadratic Linear Discriminant
> qda_model <- train(Status ~ ., data=train_data, method='qda', trControl=ctrl, tuneLe
ngth=5)
> #Summary
> summary(qda_model)
            Length Class      Mode
prior         2    -none-     numeric
counts        2    -none-     numeric
means        28    -none-     numeric
scaling     392    -none-     numeric
ldet          2    -none-     numeric
lev           2    -none-     character
N             1    -none-     numeric
call          3    -none-     call
xNames       14    -none-     character
problemType   1    -none-     character
tuneValue     1    data.frame list
obsLevels     2    -none-     character
param         0    -none-     list
> #Accuracy
> qda_model$results
  parameter Accuracy     Kappa AccuracySD   KappaSD
1      none 0.801369 0.5094769 0.07660018 0.1972876
>
```

Let’s see how the model does on the testing set. We could say that the QDA did not capture the true representation of our data.

```
> #add prediction column to test dataset
> test_data$qda <- predict(qda_model, newdata=test_data)
> view(test_data)
> #get probabilities
> head(predict(qda_model, newdata = test_data, type = 'prob'))
    Not.Placed    Placed
1 5.041577e-02 0.9495842
2 2.659177e-04 0.9997341
3 7.966635e-01 0.2033365
4 5.843813e-07 0.9999994
5 4.559530e-01 0.5440470
6 1.052395e-08 1.0000000
> qda_preds <- predict(qda_model, newdata = test_data)
> qda_cm <- confusionMatrix(qda_preds, test_data$Status)
> qda_cm
Confusion Matrix and Statistics

            Reference
Prediction   Not.Placed Placed
  Not.Placed         12      4
  Placed              8     40

               Accuracy : 0.8125
                 95% CI : (0.6954, 0.8992)
    No Information Rate : 0.6875
    P-Value [Acc > NIR] : 0.01825

                  Kappa : 0.5385

 Mcnemar's Test P-Value : 0.38648

            Sensitivity : 0.6000
            Specificity : 0.9091
         Pos Pred Value : 0.7500
         Neg Pred Value : 0.8333
             Prevalence : 0.3125
         Detection Rate : 0.1875
   Detection Prevalence : 0.2500
      Balanced Accuracy : 0.7545

       'Positive' Class : Not.Placed

>
```

4.7. K-Nearest Neighbors

KNN printed the accuracy of the model on the training set for different values of K at 5,7,9,11,13 since we set tunelength equals 5. The average will give us the result on our training set.

```
> #KNN
> knn_model <- train(Status ~ ., data = train_data, method = 'knn', trControl = ctrl,
 tuneLength=5)
> #Summary
> summary(knn_model)
            Length Class      Mode
learn        2     -none-     list
k            1     -none-     numeric
theDots      0     -none-     list
xNames      14     -none-     character
problemType  1     -none-     character
tuneValue    1     data.frame list
obsLevels    2     -none-     character
param        0     -none-     list
> #names(knn_model)
> knn_model$results
   k  Accuracy     Kappa AccuracySD   KappaSD
1  5 0.8352381 0.5807363 0.09001589 0.2249911
2  7 0.8481548 0.6150729 0.07994355 0.2086614
3  9 0.8552381 0.6205100 0.10438160 0.2728386
4 11 0.8548214 0.6199530 0.10154717 0.2651208
5 13 0.8481548 0.5970392 0.09276571 0.2507970
```

```
> test_data$knn <- predict(knn_model, newdata = test_data)
> view(test_data)
> head(predict(knn_model, newdata=test_data, type='prob'))
  Not.Placed    Placed
1  0.4444444 0.5555556
2  0.1111111 0.8888889
3  0.8888889 0.1111111
4  0.0000000 1.0000000
5  0.3333333 0.6666667
6  0.0000000 1.0000000
> knn_prob <- predict(knn_model, newdata=test_data, type='prob')*100
> knn_pred <- predict(knn_model, newdata = test_data)
> knn_cm <- confusionMatrix(knn_pred, test_data$Status)
> knn_cm
Confusion Matrix and Statistics

            Reference
Prediction   Not.Placed Placed
  Not.Placed         10      1
  Placed             10     43

               Accuracy : 0.8281
                 95% CI : (0.7132, 0.911)
    No Information Rate : 0.6875
    P-Value [Acc > NIR] : 0.008412

                  Kappa : 0.544

 Mcnemar's Test P-Value : 0.015861

            Sensitivity : 0.5000
            Specificity : 0.9773
         Pos Pred Value : 0.9091
         Neg Pred Value : 0.8113
             Prevalence : 0.3125
         Detection Rate : 0.1562
   Detection Prevalence : 0.1719
      Balanced Accuracy : 0.7386

       'Positive' Class : Not.Placed

>
```

The accuracy on the testing set is 0.8281 and Kappa is 0.544

4.8.Random Forest (Ranger)

Grow a random forest on the training data. For each observation of interest (test data), the weights of all training observations are computed by counting the number of trees in which both observations are in the same terminal node. For each test observation, grow a weighted random forest on the training data, using the weights obtained in step 2. Predict the outcome of the test observation as usual. In total, n+1 random forests are grown, where n is the number observations in the test dataset.

```
> #Random Forest-Ranger
> rf_model <- train(Status ~ ., data=train_data, method='ranger', trControl=ctrl, tuneLength=5)
> #Summary
> summary(rf_model)
                          Length Class         Mode     
predictions               151    factor        numeric  
num.trees                   1    -none-        numeric  
num.independent.variables   1    -none-        numeric  
mtry                        1    -none-        numeric  
min.node.size               1    -none-        numeric  
prediction.error            1    -none-        numeric  
forest                      9    ranger.forest list     
confusion.matrix            4    table         numeric  
splitrule                   1    -none-        character
treetype                    1    -none-        character
call                        9    -none-        call     
importance.mode             1    -none-        character
num.samples                 1    -none-        numeric  
replace                     1    -none-        logical  
xNames                     14    -none-        character
problemType                 1    -none-        character
tuneValue                   3    data.frame    list     
obsLevels                   2    -none-        character
param                       0    -none-        list     
> #names(rf_model)
> #Accuracy
> rf_model$results
   mtry min.node.size  splitrule  Accuracy     Kappa AccuracySD   KappaSD
1     2             1       gini 0.8536905 0.6283743 0.10303784 0.2542627
2     2             1 extratrees 0.8269643 0.5293028 0.08496422 0.2468516
3     5             1       gini 0.8741667 0.6903266 0.11523259 0.2816700
4     5             1 extratrees 0.8541071 0.6180482 0.06147109 0.1864771
5     8             1       gini 0.8804167 0.7075417 0.10352674 0.2498006
6     8             1 extratrees 0.8479167 0.6293548 0.07810571 0.1922923
7    11             1       gini 0.8541667 0.6444837 0.09854262 0.2358136
8    11             1 extratrees 0.8679167 0.6782037 0.07076658 0.1775010
9    14             1       gini 0.8604167 0.6617522 0.10193442 0.2443406
10   14             1 extratrees 0.8741667 0.6932516 0.09197759 0.2197050
```

The Random Forest model was evaluated using 5 different number of predictors 'mtry = (2,5,8,11,14) until m=p which is Bagging. The highest accuracy on the training set when all the predictors were considered m=8 meaning that the Random Forest approach will produce a better predictor than Bagging.

The accuracy on the test set is 0.8281 and Kappa is 0.5707.

For an ensemble model, it performed worse than the linear models of LDA and SVM.

```
> #add prediction column to test dataset
> test_data$rf <- predict(rf_model, newdata=test_data)
> view(test_data)
> #get probabilities
> #head(predict(rf_model, newdata = test_data, type = 'prob'))
> rf_preds <- predict(rf_model, newdata = test_data)
> rf_cm <- confusionMatrix(rf_preds, test_data$Status)
> rf_cm
Confusion Matrix and Statistics

            Reference
Prediction   Not.Placed Placed
  Not.Placed         12      3
  Placed              8     41

               Accuracy : 0.8281
                 95% CI : (0.7132, 0.911)
    No Information Rate : 0.6875
    P-Value [Acc > NIR] : 0.008412

                  Kappa : 0.5707

 Mcnemar's Test P-Value : 0.227800

            Sensitivity : 0.6000
            Specificity : 0.9318
         Pos Pred Value : 0.8000
         Neg Pred Value : 0.8367
             Prevalence : 0.3125
         Detection Rate : 0.1875
   Detection Prevalence : 0.2344
      Balanced Accuracy : 0.7659

       'Positive' Class : Not.Placed
```

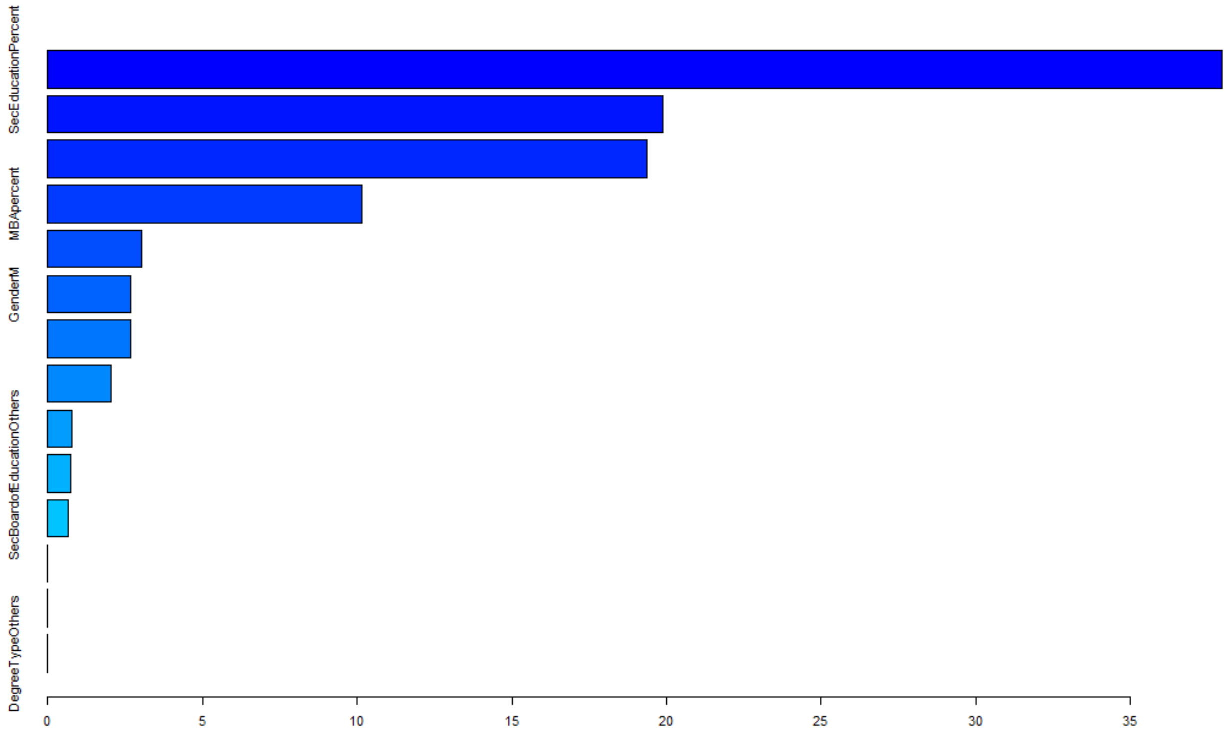
4.9. Gradient Boosting

```
> #Summary
> summary(gbm_model)
                                                     var    rel.inf
SecEducationPercent                    SecEducationPercent 38.9053360
DegreePercent                                DegreePercent 17.6202474
HigherSecEduPercent                    HigherSecEduPercent 14.0515425
MBApercent                                      MBApercent 10.6556823
EmpTestPercent                              EmpTestPercent  4.1207294
DegreeTypeSci&Tech                      DegreeTypeSci&Tech  4.0397762
GenderM                                            GenderM  3.4114123
WorkExpYes                                      WorkExpYes  3.0920035
SecBoardofEducationOthers SecBoardofEducationOthers  1.8523784
HigherSecSpecialCommerce     HigherSecSpecialCommerce  0.7427997
MBAspecMkt&HR                                MBAspecMkt&HR  0.6994353
HigherSecBoardofEduOthers HigherSecBoardofEduOthers  0.6266165
HigherSecSpecialScience       HigherSecSpecialScience  0.1820404
DegreeTypeOthers                          DegreeTypeOthers  0.0000000
> #names(gbm_model)
> #Accuracy
> gbm_model$results
  shrinkage interaction.depth n.minobsinnode n.trees  Accuracy     Kappa AccuracySD    KappaSD
1       0.1                 1             10      50 0.8456938 0.6186345 0.04360550 0.09854818
4       0.1                 2             10      50 0.8471413 0.6246483 0.03748883 0.09631077
7       0.1                 3             10      50 0.8468106 0.6245563 0.03217273 0.08282468
2       0.1                 1             10     100 0.8501893 0.6350566 0.04311432 0.10053509
5       0.1                 2             10     100 0.8586452 0.6582930 0.04105881 0.10147860
8       0.1                 3             10     100 0.8504322 0.6423413 0.03416463 0.07601724
3       0.1                 1             10     150 0.8546182 0.6508954 0.04564875 0.10368580
6       0.1                 2             10     150 0.8543839 0.6462755 0.03201343 0.08187106
9       0.1                 3             10     150 0.8458648 0.6337265 0.03528724 0.07404415
```

```
> #Accuracy
> gbm_model$results
  shrinkage interaction.depth n.minobsinnode n.trees  Accuracy     Kappa AccuracySD    KappaSD
1       0.1                 1             10      50 0.8456938 0.6186345 0.04360550 0.09854818
4       0.1                 2             10      50 0.8471413 0.6246483 0.03748883 0.09631077
7       0.1                 3             10      50 0.8468106 0.6245563 0.03217273 0.08282468
2       0.1                 1             10     100 0.8501893 0.6350566 0.04311432 0.10053509
5       0.1                 2             10     100 0.8586452 0.6582930 0.04105881 0.10147860
8       0.1                 3             10     100 0.8504322 0.6423413 0.03416463 0.07601724
3       0.1                 1             10     150 0.8546182 0.6508954 0.04564875 0.10368580
6       0.1                 2             10     150 0.8543839 0.6462755 0.03201343 0.08187106
9       0.1                 3             10     150 0.8458648 0.6337265 0.03528724 0.07404415
> #add prediction column to test dataset
> test_data$gbm <- predict(gbm_model, newdata=test_data)
> view(test_data)
> #get probabilities
> #head(predict(gbm_model, newdata = test_data, type = 'prob'))
> gbm_preds <- predict(gbm_model, newdata = test_data)
> gbm_cm <- confusionMatrix(gbm_preds, test_data$Status)
> gbm_cm
Confusion Matrix and Statistics

            Reference
Prediction   Not.Placed Placed
  Not.Placed         15      6
  Placed              5     38

               Accuracy : 0.8281
                 95% CI : (0.7132, 0.911)
    No Information Rate : 0.6875
    P-Value [Acc > NIR] : 0.008412

                  Kappa : 0.6054

 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.7500
            Specificity : 0.8636
         Pos Pred Value : 0.7143
         Neg Pred Value : 0.8837
             Prevalence : 0.3125
         Detection Rate : 0.2344
   Detection Prevalence : 0.3281
      Balanced Accuracy : 0.8068

       'Positive' Class : Not.Placed
```

Gradient boosting is an ensemble learning method, the accuracy on the test set is 0.8281 kappa is 0.6054.

The plot of the summary of the result show that the top 4 variables are Secondary education, Higher secondary education, University degree percentage and MBA percentage.

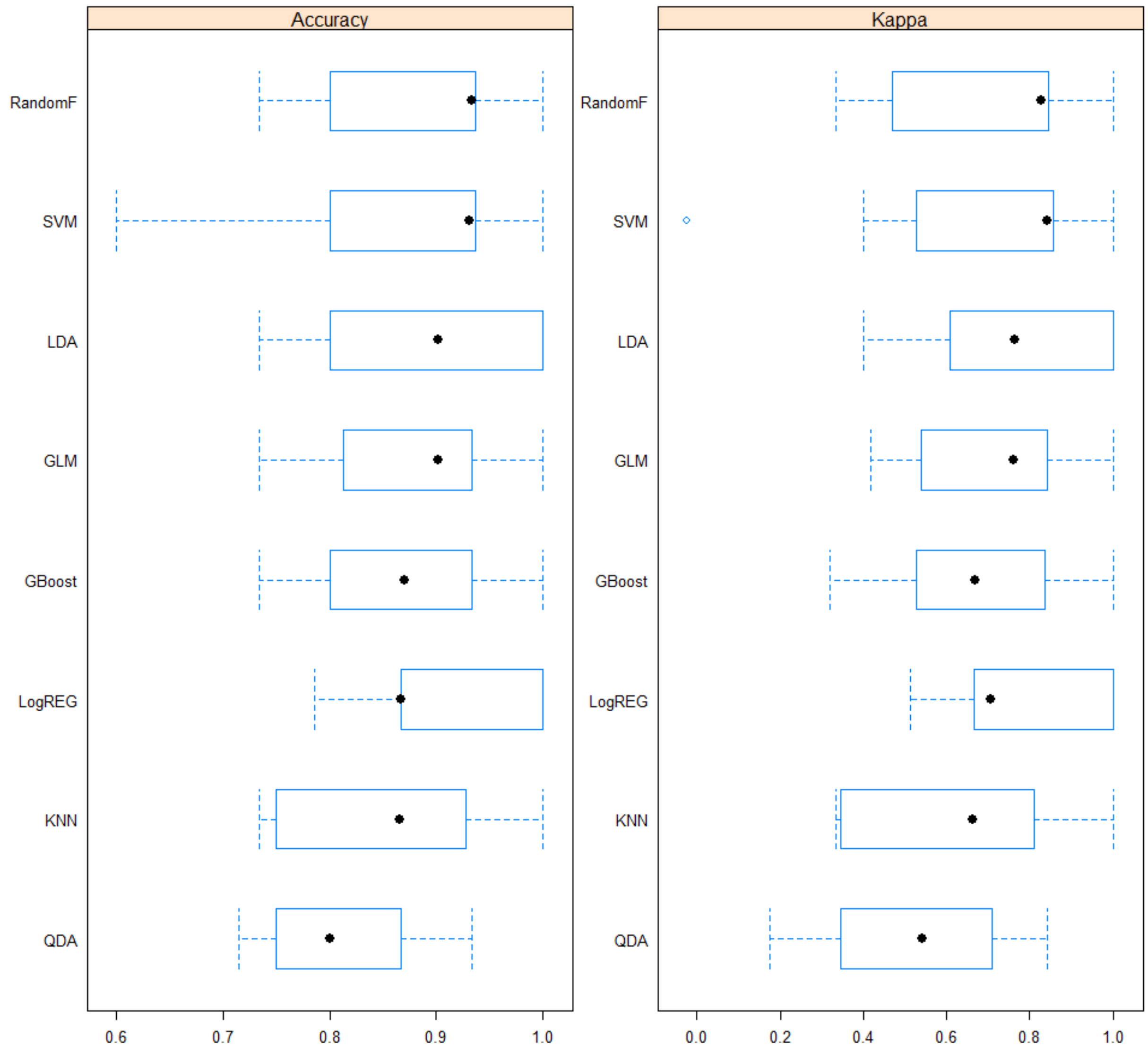


5. EVALUATION

We evaluated our models by comparing the accuracy and the kappa. Accuracy is the percentage of accurate predictions out of all samples and kappa is the accuracy that would be generated by chance. We also considered the amount of false positive and false negative in the model.

5.1. Model Accuracy, Kappa, and Confusion Matrix

```
> models <- resamples(list(GBoost = gbm_model, RandomF = rf_model,KNN = knn_model,
+           QDA = qda_model,SVM = svm_model,LDA = lda_model,LogREG = log_model,
+           GLM = glm_model))
>
> #Scaling
> scales <- list(x=list(relation="free"), y=list(relation="free"))
>
> #draw a boxplot to compare models
> bwplot(models, scales=scales)
>
> summary(models)

Call:
summary.resamples(object = models)

Models: GBoost, RandomF, KNN, QDA, SVM, LDA, LogREG, GLM
Number of resamples: 10

Accuracy
             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
GBoost  0.7333333 0.8000000 0.8708333 0.8620238 0.9321429 1.0000000    0
RandomF 0.7333333 0.8000000 0.9333333 0.8804167 0.9364583 1.0000000    0
KNN     0.7333333 0.7625000 0.8660714 0.8605952 0.9285714 1.0000000    0
QDA     0.7142857 0.7589286 0.8000000 0.8141667 0.8666667 0.9333333    0
SVM     0.6000000 0.8142857 0.9309524 0.8727381 0.9375000 1.0000000    0
LDA     0.7333333 0.8166667 0.9020833 0.9004167 1.0000000 1.0000000    0
LogREG  0.7857143 0.8666667 0.8666667 0.8927381 0.9687500 1.0000000    0
GLM     0.7333333 0.8260417 0.9017857 0.8761310 0.9321429 1.0000000    0

Kappa
               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
GBoost   0.31818182 0.5375940 0.6700680 0.6597130 0.8316107 1.0000000    0
RandomF  0.33333333 0.4845201 0.8284600 0.7075417 0.8451417 1.0000000    0
KNN      0.33333333 0.3778511 0.6617347 0.6429285 0.8108108 1.0000000    0
QDA      0.17647059 0.3778511 0.5415282 0.5369128 0.6984848 0.8421053    0
SVM     -0.02272727 0.5572368 0.8416816 0.6940248 0.8543956 1.0000000    0
LDA      0.40000000 0.6231884 0.7643678 0.7648542 1.0000000 1.0000000    0
LogREG   0.51162791 0.6750000 0.7074866 0.7546856 0.9318182 1.0000000    0
GLM      0.41818182 0.5705128 0.7599509 0.7138233 0.8342817 1.0000000    0

> |
```



The accuracy of the model indicates that most of the models have similar performance. But to determine which models' prediction error is costlier than the other, we will look at the confusion matrix and identify the amount of false negative and false positive in each prediction. False Negative happens when the model incorrectly predicts that someone would not be placed, while False is when the model incorrectly predicts that someone would be placed.

- The model with the least False Negative is KNN having a prediction accuracy of 82.8%

(10 false positive and 1 false negative). With KNN model a student, the likely hood that a student will miss the opportunity of getting placed due to model error is low compared to other models. This model favors the students.

- The model with the least False Positive is LDA. It has a prediction accuracy of 85.9% (4 false positive and 5 false negative). This model maximizes accuracy, but the output could be costly on students.

Using the tidy function to compare the two models.

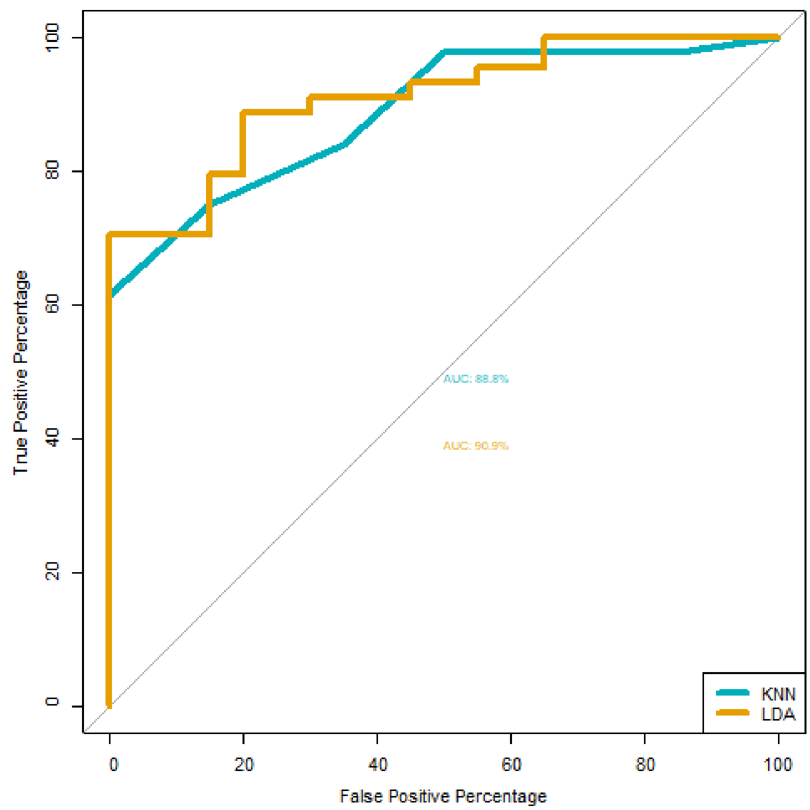
Comparing the statistics of KNN and LDA models

estimate	statistic	p.value	parameter	conf.low	conf.high	method	alternative
-0.0398214	-0.9514339	0.3662343	9	-0.134502	0.0548592	One Sample t-test	two.sided

Looking at the p-value, we can assume that there is no statistical reasoning to assume that one model is better than the other.

5.2. ROC Curve

Lastly, let us compare the ROC curve of the two models. From the ROC curve, we can conclude that the LDA is a better model.



6. CONCLUSION

From the result of our statistical analysis, we could conclude the following.

- Secondary education percent is the most important variable with influence on the placement of a student. Students that did well in secondary education got placed.
- The topmost important features are secondary education percent, higher secondary education percent, university degree percent and MBA percentage score.
- Employability test score does not correlate with the academic score. That is the employability test score is a practical test.
- As the education level increases, female students got higher score than the male students.
- The LDA model favors the students because if has the least false negative model errors and the KNN favors the recruiters because if has the least false positive errors.

7. REFRENCES

- https://www.kaggle.com/benroshan/factors-affecting-campus-placement
- 10.5281/zenodo.5109070 ISBN:978-93-5426-386-6@2021 MCA, Amal Jyothi College of Engineering Kanjirappally, Kottayam. Proceedings of the National Conference on Emerging Computer Applications (NCECA) -2021 Vol.3, Issue.1
- https://cran.r-project.org/
- https://www.analyticsvidhya.com/blog/2021/03/introduction-to-k-fold-cross-validation-in-r/