Group 2 Analysis

R Group

2023-02-06

Contents

1.0 Loaded Packages

```r
knitr::opts_chunk$set(
    echo = TRUE,
    message = FALSE,
    warning = FALSE
)
library(pacman)
pacman::p_load(pacman,party,psych,rio,tidyverse)
library(datasets)
library(mice)
```

```
##
## Attaching package: 'mice'

## The following object is masked from 'package:stats':
##
##     filter

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```r
library(naniar)
library(ggplot2)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

2.0 Data Observations

2.1 Description of the data and response variable

Prior explore further into data processing, it is important for us to understand what data are available to us. There are total of 28 columns and 742 rows in the dataset retrieved. Below shows the description of the relevant column name.

“Job Title”: The title of the job posting

“Salary Estimate”: The range of the salary offer

“Job Description”: The description of the job posting

“Rating”: The rating of the company that posted job offer

“Company Name”: The name of the company that posted job offer

“Location”: The location of the company that posted job offer

“Headquarters”: The headquarter location of the company that posted job offer

“Size”: The size of the company that posted job offer

“Founded”: The year when the company that posted job offer was founded

“Type of ownership”: The company’s type of ownership

“Industry”: The industry of the company that posted job offer

“Sector”: The sector of the company that posted job offer

“Revenue”: The yearly revenue of the company that posted job offer

“Competitors”: The competitors of the company that posted job offer

“Hourly”: (Not defined)

“employer_provided”: (Not defined)

“min_salary”: The minimum salary range offered

“max_salary”: The maximum salary range offered

“avg_salary”: The average salary range offered

“company_txt”: The name of the company

“job_state”: The state where the job is located

“same_state”: Whether the state where the job is located is the same as the location of the job seeker

“age”: The age of the job seeker

“python_yn”: Whether the job seeker knows Python

“R_yn”: Whether the job seeker knows R

“spark”: Whether the job seeker knows Spark

“aws”: Whether the job seeker knows AWS

“excel”: Whether the job seeker knows Excel

In the dataset, there are 21 columns that are related to the job posting company: “Job Title”, “Salary Estimate”, “Job Description”, “Rating”, “Company Name”, “Location”, “Headquarters”, “Size”, “Founded”, “Type of ownership”, “Industry”, “Sector”, “Revenue”, “Competitors”, “hourly”, “employer_provided”, “min_salary”, “max_salary”, “avg_salary”, “company_txt” and “job_state”.

However, column "hourly" and "employer_provided" were not described and defined by the data provider. With the data "0" and "1", we are unable to predict what are the purpose of the data collected and their usage. Understanding this, both of the data should be considered to be removed from the analysis as we do not have enough information to interpret the result from these data.

On the other hand, there are 7 columns in the dataset that are relevant to the job seeker information: "same_state", "age", "python_yn", "R_yn", "spark", "aws" and "excel".

In this project, the aim is to predict data scientists' salary using the data retrieved from Glassdoor.com, hence the response variable would be the salary. There are few columns in the dataset that are seemed to be related to salary, like "Salary Estimate", "min_salary", "max_salary" and "avg_salary". "Salary Estimate" is he range of the salary offer, "min_salary" and "max_salary" indicate the minimum and maximum range of the salary offered while "avg_salary" informed the average. In this case, "Salary Estimate", "min_salary" and "max_salary" seems to be informing a replicated information. Eg. Salary Estimate: 53K-91K (Glassdoor est.), min_salary: 53, max_salary: 91, avg_salary: 72.

Hence, it can be considered to remove these redundant data and focus on the "avg_salary" for response variable.

2.2 Importing Our Dataset & Data Transformation

```
Salary <- import("Salary.csv")%>%
  as_tibble()%>%
  rename(ownership = `Type of ownership`)%>%
  mutate(ownership=as.factor(ownership))%>%
  mutate(Sector=as.factor(Sector))%>%
  mutate( job_state =as.factor(job_state))%>%
  mutate(Industry=as.factor(Industry))%>%
  mutate(Revenue=as.ordered(Revenue))%>%
  mutate(Size=as.ordered(Size))%>%
  mutate(python_yn=as.logical(python_yn))%>%
  mutate(R_yn=as.logical(R_yn))%>%
  mutate(spark=as.logical(spark))%>%
  mutate(aws=as.logical(aws))%>%
  mutate(excel=as.logical(excel))%>%
  print()
```

```
## # A tibble: 742 x 28
##    Job Ti~1 Salar~2 Job D~3 Rating Compa~4 Locat~5 Headq~6 Size  Founded owner~7
##    <chr>    <chr>   <chr>    <dbl> <chr>   <chr>   <chr>   <ord>   <int> <fct>
##  1 Data Sc~ $53K-$~ "Data ~    3.8 "Tecol~ Albuqu~ Goleta~ 501 ~    1973 Compan~
##  2 Healthc~ $63K-$~ "What ~    3.4 "Unive~ Linthi~ Baltim~ 1000~    1984 Other ~
##  3 Data Sc~ $80K-$~ "KnowB~    4.8 "KnowB~ Clearw~ Clearw~ 501 ~    2010 Compan~
##  4 Data Sc~ $56K-$~ "*Orga~    3.8 "PNNL\~ Richla~ Richla~ 1001~    1965 Govern~
##  5 Data Sc~ $86K-$~ "Data ~    2.9 "Affin~ New Yo~ New Yo~ 51 t~    1998 Compan~
##  6 Data Sc~ $71K-$~ "Cyrus~    3.4 "Cyrus~ Dallas~ Dallas~ 201 ~    2000 Compan~
##  7 Data Sc~ $54K-$~ "Job D~    4.1 "Clear~ Baltim~ Baltim~ 501 ~    2008 Compan~
##  8 Data Sc~ $86K-$~ "Advan~    3.8 "Logic~ San Jo~ Seattl~ 201 ~    2005 Compan~
##  9 Researc~ $38K-$~ "SUMMA~    3.3 "Roche~ Roches~ Roches~ 1000~    2014 Hospit~
## 10 Data Sc~ $120K-~ "isn't~    4.6 "<inte~ New Yo~ New Yo~ 51 t~    2009 Compan~
## # ... with 732 more rows, 18 more variables: Industry <fct>, Sector <fct>,
## #   Revenue <ord>, Competitors <chr>, hourly <int>, employer_provided <int>,
## #   min_salary <int>, max_salary <int>, avg_salary <dbl>, company_txt <chr>,
```

```
## #   job_state <fct>, same_state <int>, age <int>, python_yn <lgl>, R_yn <lgl>,
## #   spark <lgl>, aws <lgl>, excel <lgl>, and abbreviated variable names
## #   1: ‘Job Title‘, 2: ‘Salary Estimate‘, 3: ‘Job Description‘,
## #   4: ‘Company Name‘, 5: Location, 6: Headquarters, 7: ownership
```

We decided to transform the following attributes: "Sector", "Revenue", "Size", "Industry" as factor data type as these attributes were originally listed as character type. However, after observing the dataset, it can be deduced that it is a categorical variable with a limited number of categories. For example, job_state is an attribute that shows the US state of where the job is located. Furthermore, the following attributes: "python_yn", "R_yn", "spark", "aws" and "excel" have been transformed into logical/boolean data type as it was initially in double data type.

2.3 A Quick Glance At Our Data

```
summary(Salary)
```

```
##   Job Title         Salary Estimate    Job Description       Rating
##  Length:742         Length:742         Length:742         Min.   :-1.000
##  Class :character   Class :character   Class :character   1st Qu.: 3.300
##  Mode  :character   Mode  :character   Mode  :character   Median : 3.700
##                                                           Mean   : 3.619
##                                                           3rd Qu.: 4.000
##                                                           Max.   : 5.000
##
##  Company Name         Location         Headquarters
##  Length:742         Length:742         Length:742
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##                       Size       Founded
##  1001 to 5000 employees :150   Min.   :  -1
##  501 to 1000 employees  :134   1st Qu.:1939
##  10000+ employees       :130   Median :1988
##  201 to 500 employees   :117   Mean   :1837
##  51 to 200 employees    : 94   3rd Qu.:2007
##  5001 to 10000 employees: 76   Max.   :2019
##  (Other)                : 41
##                            ownership
##  Company - Private             :410
##  Company - Public              :193
##  Nonprofit Organization        : 55
##  Subsidiary or Business Segment: 34
##  Government                    : 15
##  Hospital                      : 15
##  (Other)                       : 20
##                                       Industry                          Sector
##  Biotech & Pharmaceuticals               :112   Information Technology   :180
##  Insurance Carriers                      : 63   Biotech & Pharmaceuticals:112
```

```
##  Computer Hardware & Software          : 59   Business Services        : 97
##  IT Services                           : 50   Insurance                : 69
##  Health Care Services & Hospitals      : 49   Health Care              : 49
##  Enterprise Software & Network Solutions: 42   Finance                  : 42
##  (Other)                               :367   (Other)                  :193
##                             Revenue    Competitors           hourly
##  Unknown / Non-Applicable        :203   Length:742         Min.   :0.00000
##  $10+ billion (USD)              :124   Class :character   1st Qu.:0.00000
##  $100 to $500 million (USD)      : 91   Mode  :character   Median :0.00000
##  $1 to $2 billion (USD)          : 60                      Mean   :0.03234
##  $500 million to $1 billion (USD): 57                      3rd Qu.:0.00000
##  $50 to $100 million (USD)       : 46                      Max.   :1.00000
##  (Other)                         :161
##  employer_provided   min_salary       max_salary      avg_salary
##  Min.   :0.00000   Min.   : 10.00   Min.   : 16.0   Min.   : 13.5
##  1st Qu.:0.00000   1st Qu.: 52.00   1st Qu.: 96.0   1st Qu.: 73.5
##  Median :0.00000   Median : 69.50   Median :124.0   Median : 97.5
##  Mean   :0.02291   Mean   : 74.07   Mean   :127.2   Mean   :100.6
##  3rd Qu.:0.00000   3rd Qu.: 91.00   3rd Qu.:155.0   3rd Qu.:122.5
##  Max.   :1.00000   Max.   :202.00   Max.   :306.0   Max.   :254.0
##
##  company_txt          job_state     same_state         age
##  Length:742         CA     :151   Min.   :0.000   Min.   : -1.00
##  Class :character   MA     :103   1st Qu.:0.000   1st Qu.: 11.00
##  Mode  :character   NY     : 72   Median :1.000   Median : 24.00
##                     VA     : 41   Mean   :0.558   Mean   : 46.59
##                     IL     : 40   3rd Qu.:1.000   3rd Qu.: 59.00
##                     MD     : 35   Max.   :1.000   Max.   :276.00
##                     (Other):300
##  python_yn          R_yn           spark            aws
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:350       FALSE:740       FALSE:575       FALSE:566
##  TRUE :392       TRUE :2         TRUE :167       TRUE :176
##
##
##
##
##    excel
##  Mode :logical
##  FALSE:354
##  TRUE :388
##
##
##
##
```

Using the summary() function, certain observations are seen to have “-1” in attributes such as “age”, “Founded”, “Rating”, “Industry” and more. We are going to assume that these are missing values for these specific cases. On the other hand, “Revenue” refers to missing values as “Unknown / Non-Applicable”. We will go further in detail on how we will deal with these missing values in the next section.

3.0 Data Pre-processing and Cleaning

3.1 Removing redundant variables

First, lets begin with removing redundant variables as we have 28 variables in total. As we are using "avg_salary" as our target variable, it is sensible to remove "min_salary", "max_salary" and "Salary Estimate" as they will not be required for our analysis. "Competitors" should also be removed because it contains 460 missing values to 742 observations. Other attributes will be removed for their redundancy: "Job Title", "Job Description", "Company Name", "Location", "Headquarters", "Industry", "hourly", "employer_provided", "company_txt" and "same_state".

```
#Removing our redundant variables
Salary_df = as_tibble(subset(Salary, select = -c(min_salary,max_salary,
                                       `Salary Estimate`, Competitors,
                                       `Job Title`, `Job Description`,
                                       `Company Name`, Location,
                                       Headquarters, Industry, hourly,
                                       employer_provided, company_txt,
                                       same_state)))
```

This shows that Sector, Age, Rating, Size and Revenue have 50, 10, 50, 11, 9 and 203 missing values respectively out of 742 observations. The impact of imputation on the first 4 columns could be said is negligible, however Revenue has around 27% of observations being missing which could create a bias and skewness when imputed.

To remove the missing values, we must standardise them all to NA values before imputation. We will use the "naniar" package to accomplish this, using the replace_with_na() function.

3.2 Standardising and counting all missing values to N/A values

3.3 Listwise deletion on NA values

We will use listwise deletion on NA values for "Rating", "Size", "Founded", "ownership" and "Sector".

```
Salary_df <- Salary_df[!is.na(Salary_df$Rating),]
Salary_df <- Salary_df[!is.na(Salary_df$Size),]
Salary_df <- Salary_df[!is.na(Salary_df$Founded),]
Salary_df <- Salary_df[!is.na(Salary_df$ownership),]
Salary_df <- Salary_df[!is.na(Salary_df$Sector),]

summary(Salary_df)
```

```
##      Rating                             Size        Founded
##  Min.   :1.900   1001 to 5000 employees :150   Min.   :1744
##  1st Qu.:3.400   501 to 1000 employees  :125   1st Qu.:1958
##  Median :3.700   10000+ employees       :124   Median :1992
##  Mean   :3.703   201 to 500 employees   :109   Mean   :1970
##  3rd Qu.:4.000   51 to 200 employees    : 88   3rd Qu.:2007
##  Max.   :5.000   5001 to 10000 employees: 75   Max.   :2019
##                  (Other)                : 18
##                          ownership                         Sector
##  Company - Private             :378   Information Technology   :171
```

```
##  Company - Public              :187   Biotech & Pharmaceuticals:106
##  Nonprofit Organization        : 47   Business Services        : 88
##  Subsidiary or Business Segment: 32   Insurance                : 67
##  Government                    : 15   Health Care              : 49
##  Hospital                      : 15   Finance                  : 42
##  (Other)                       : 15   (Other)                  :166
##                              Revenue      avg_salary      job_state
##  $10+ billion (USD)              :124   Min.   : 13.5   CA     :139
##  $100 to $500 million (USD)      : 85   1st Qu.: 73.0   MA     : 92
##  $1 to $2 billion (USD)          : 60   Median : 96.0   NY     : 71
##  $500 million to $1 billion (USD): 57   Mean   :100.2   VA     : 40
##  $50 to $100 million (USD)       : 45   3rd Qu.:123.5   IL     : 36
##  (Other)                         :138   Max.   :254.0   MD     : 35
##  NA's                            :180                   (Other):276
##       age        python_yn          R_yn           spark
##  Min.   :18.00   Mode :logical   Mode :logical   Mode :logical
##  1st Qu.:24.00   FALSE:317       FALSE:687       FALSE:529
##  Median :36.00   TRUE :372       TRUE :2         TRUE :160
##  Mean   :39.03
##  3rd Qu.:52.00
##  Max.   :78.00
##  NA's   :380
##     aws            excel
##  Mode :logical   Mode :logical
##  FALSE:528       FALSE:332
##  TRUE :161       TRUE :357
##
##
##
##
```

With this, the remaining NA values are on "Revenue" and "age" with 180 and 238 values respectively. These will be fixed with imputation techniques. By using the "Mice" package, we can use imputation techniques on "Revenue". For example, we will use "polr" for our imputation technique as Revenue is an ordered factor. However, for "age" we will use the mean for imputation.

3.4 Impute missing values

3.5 Removing Age & other inconsistent data

Based on our summary, the "age" column has 380 missing values. Due to this large number of NA values, the quality of this column is very poor and must not be used in the model.

```r
#Removing age
Salary_Final <- as_tibble(subset(Salary_Final, select = -c(age)))

#Inconsistent data
unique(Salary_Final$job_state)
```

```
##  [1] NM          MD          FL          WA          NY          TX
##  [7] CA          VA          MA          NJ          CO          IL
## [13] KY          OR          CT          MI          OH          AL
```

```
## [19] MO          PA          GA          IN          LA          WI
## [25] DC          NC          AZ          NE          MN          Los Angeles
## [31] TN          DE          UT          ID          RI          IA
## [37] SC          KS
## 38 Levels: AL AZ CA CO CT DC DE FL GA IA ID IL IN KS KY LA Los Angeles ... WI
```

```
Salary_Final["job_state"][Salary_Final["job_state"] == "Los Angeles"] <- "LA"
```

3.6 Removing outliers in "avg_salary"

The summary also displayed 3rd Quartile to be 123.5 while the max value is 254. To remove bias-ness and influence of skewness, we will filter the "avg_salary" to a reasonable range of below 200.

3.6 Summary of Cleaned Data Set

```
summary(Salary_Final)
```

```
##      Rating                          Size        Founded
##  Min.   :1.9   1001 to 5000 employees :149   Min.   :1744
##  1st Qu.:3.4   501 to 1000 employees  :123   1st Qu.:1958
##  Median :3.7   10000+ employees       :121   Median :1992
##  Mean   :3.7   201 to 500 employees   :109   Mean   :1970
##  3rd Qu.:4.0   51 to 200 employees    : 85   3rd Qu.:2007
##  Max.   :5.0   5001 to 10000 employees: 75   Max.   :2019
##                (Other)                : 18
##                          ownership                              Sector
##  Company - Private             :370   Information Technology   :167
##  Company - Public              :187   Biotech & Pharmaceuticals:106
##  Nonprofit Organization        : 47   Business Services        : 87
##  Subsidiary or Business Segment: 31   Insurance                : 64
##  Government                    : 15   Health Care              : 48
##  Hospital                      : 15   Finance                  : 42
##  (Other)                       : 15   (Other)                  :166
##                             Revenue      avg_salary       job_state
##  $10+ billion (USD)              :153   Min.   : 13.50   CA     :134
##  $100 to $500 million (USD)      :103   1st Qu.: 73.00   MA     : 92
##  $1 to $2 billion (USD)          : 98   Median : 95.75   NY     : 71
##  $500 million to $1 billion (USD): 85   Mean   : 98.50   VA     : 40
##  $50 to $100 million (USD)       : 58   3rd Qu.:121.62   MD     : 35
##  $25 to $50 million (USD)        : 51   Max.   :194.50   IL     : 33
##  (Other)                         :132                    (Other):275
##  python_yn          R_yn           spark            aws
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:317       FALSE:678       FALSE:523       FALSE:522
##  TRUE :363       TRUE :2         TRUE :157       TRUE :158
##
##
##
##
##    excel
```

```
##  Mode :logical
##  FALSE:325
##  TRUE :355
##
##
##
##
```

```
str(Salary_Final)
```

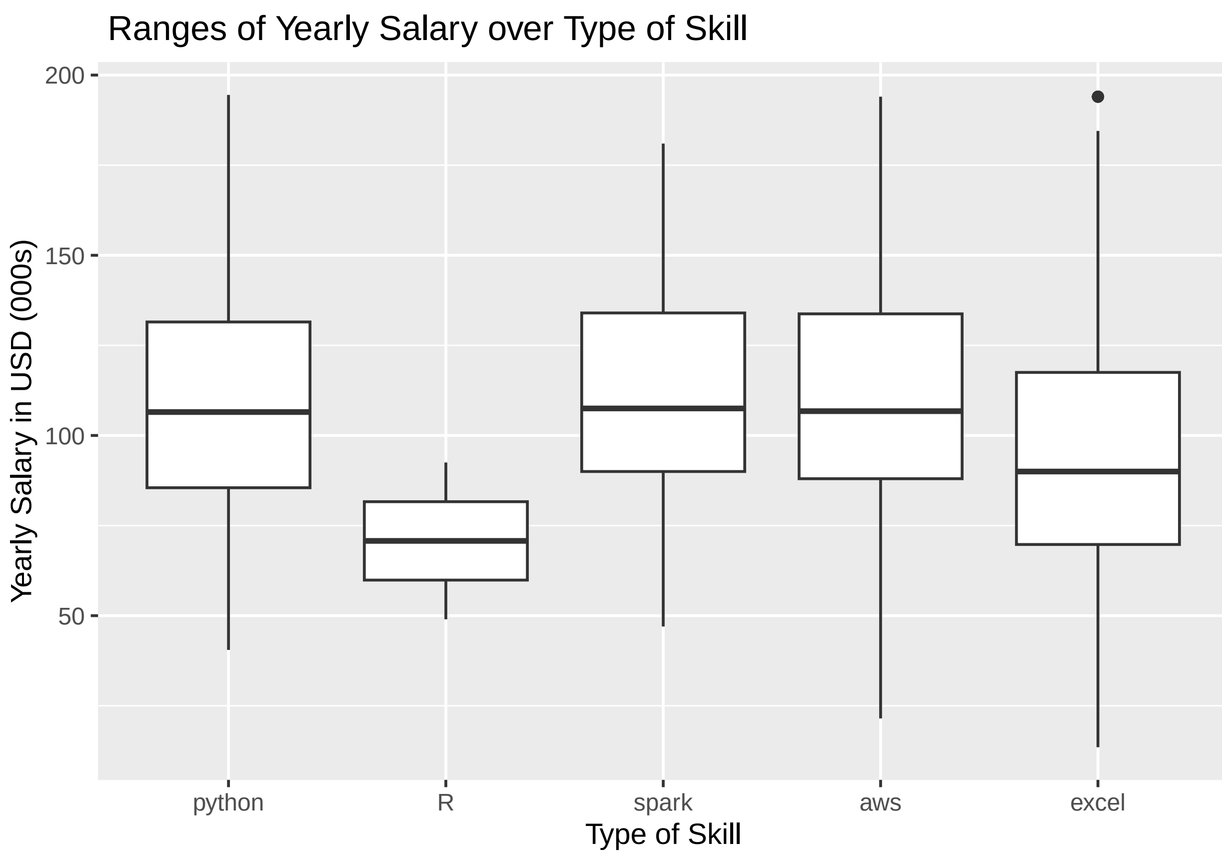
```
## tibble [680 x 13] (S3: tbl_df/tbl/data.frame)
##  $ Rating    : num [1:680] 3.8 3.4 4.8 3.8 2.9 3.4 4.1 3.8 3.3 4.6 ...
##  $ Size      : Ord.factor w/ 9 levels "-1"<"1 to 50 employees"<..: 7 3 7 4 8 5 7 5 3 8 ...
##  $ Founded   : int [1:680] 1973 1984 2010 1965 1998 2000 2008 2005 2014 2009 ...
##  $ ownership : Factor w/ 11 levels "-1","College / University",..: 3 8 3 5 3 4 3 3 6 3 ...
##  $ Sector    : Factor w/ 25 levels "-1","Accounting & Legal",..: 3 13 7 20 7 21 11 7 13 14 ...
##  $ Revenue   : Ord.factor w/ 14 levels "-1"<"$1 to $2 billion (USD)"<..: 11 7 6 12 5 2 12 8 12 6 ...
##  $ avg_salary: num [1:680] 72 87.5 85 76.5 114.5 ...
##  $ job_state : Factor w/ 38 levels "AL","AZ","CA",..: 26 19 8 37 27 34 19 3 27 27 ...
##  $ python_yn : logi [1:680] TRUE TRUE TRUE TRUE TRUE TRUE ...
##  $ R_yn      : logi [1:680] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ spark     : logi [1:680] FALSE FALSE TRUE FALSE FALSE FALSE ...
##  $ aws       : logi [1:680] FALSE FALSE FALSE FALSE FALSE TRUE ...
##  $ excel     : logi [1:680] TRUE FALSE TRUE FALSE TRUE TRUE ...
```

4.0 Exploratory Data Analysis (EDA)

4.1 Range of 'avg_salary' by skill sets (Boxplot)

```
# Filtering avg_salary by skill = TRUE & using melt() function to reshape

python1 <- Salary_Final %>%
  filter( python_yn == TRUE) %>%
  select(avg_salary)%>%
  rename(python = avg_salary)%>%
  melt()

R1 <- Salary_Final %>%
  filter( R_yn == TRUE) %>%
  select(avg_salary)%>%
  rename(R = avg_salary)%>%
  melt()


spark1 <- Salary_Final %>%
  filter( spark == TRUE) %>%
  select(avg_salary)%>%
  rename(spark = avg_salary)%>%
  melt()


aws1 <- Salary_Final %>%
  filter( aws == TRUE) %>%
  select(avg_salary)%>%
  rename(aws = avg_salary)%>%
  melt()


excel1 <- Salary_Final %>%
  filter( excel == TRUE) %>%
  select(avg_salary)%>%
  rename(excel = avg_salary)%>%
  melt()


#combining all into 1
df <- rbind(python1,R1,spark1,aws1,excel1)
```

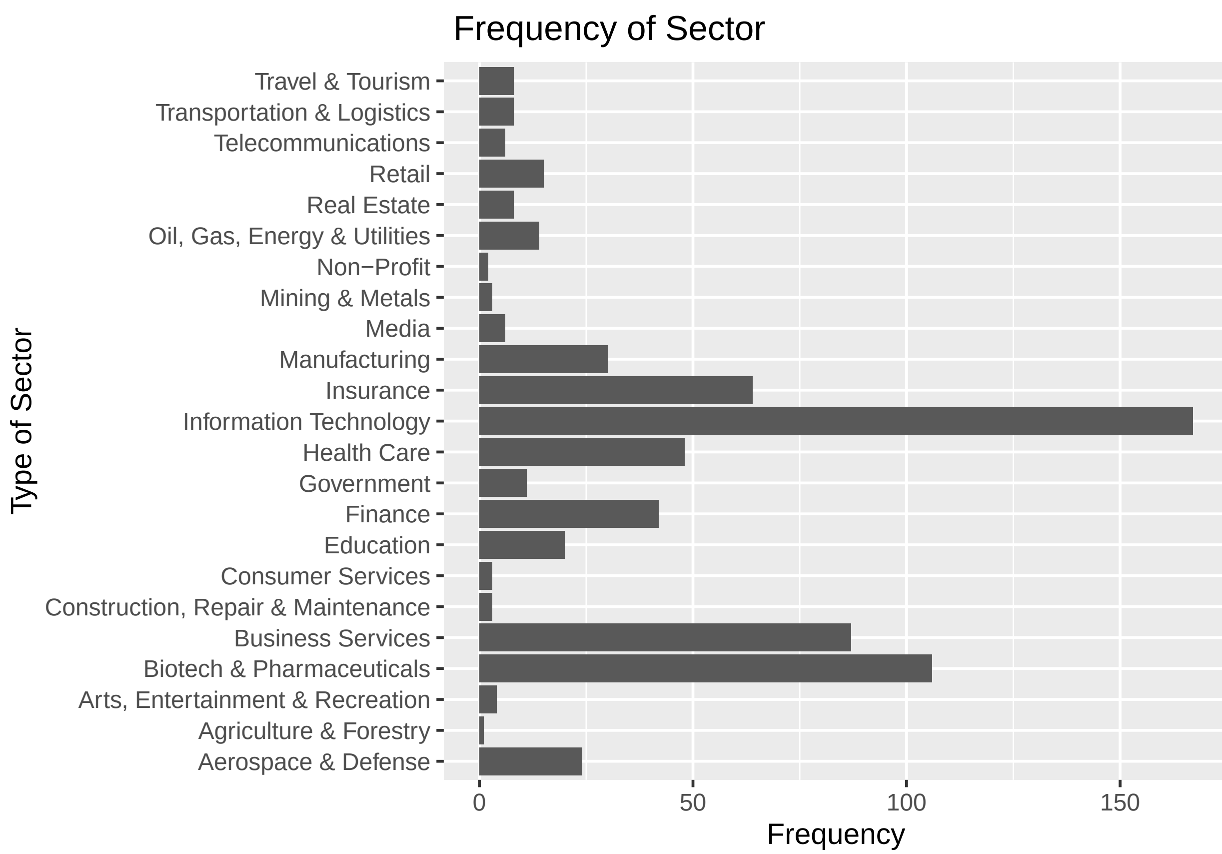
```
boxplot1 <- ggplot(df, aes(x = variable, y = value)) +           # Applying ggplot function
  geom_boxplot()+
  xlab("Type of Skill")+
  ylab("Yearly Salary in USD (000s)")+
  ggtitle(" Ranges of Yearly Salary over Type of Skill")

boxplot1
```



The box-plots in Fig. - compare the type of skill against the yearly salary in USD. Spark and Aws have the highest Salary range, more than 100000 USD, whereas there is a significant difference in the Salary range of people having skills in R programming, compared to Python, Spark, Aws and Excel. Excel is comparatively less demanding skill compared to Python, Spark, Aws as well.

4.2 Count of 'Sector' by 'Sector'

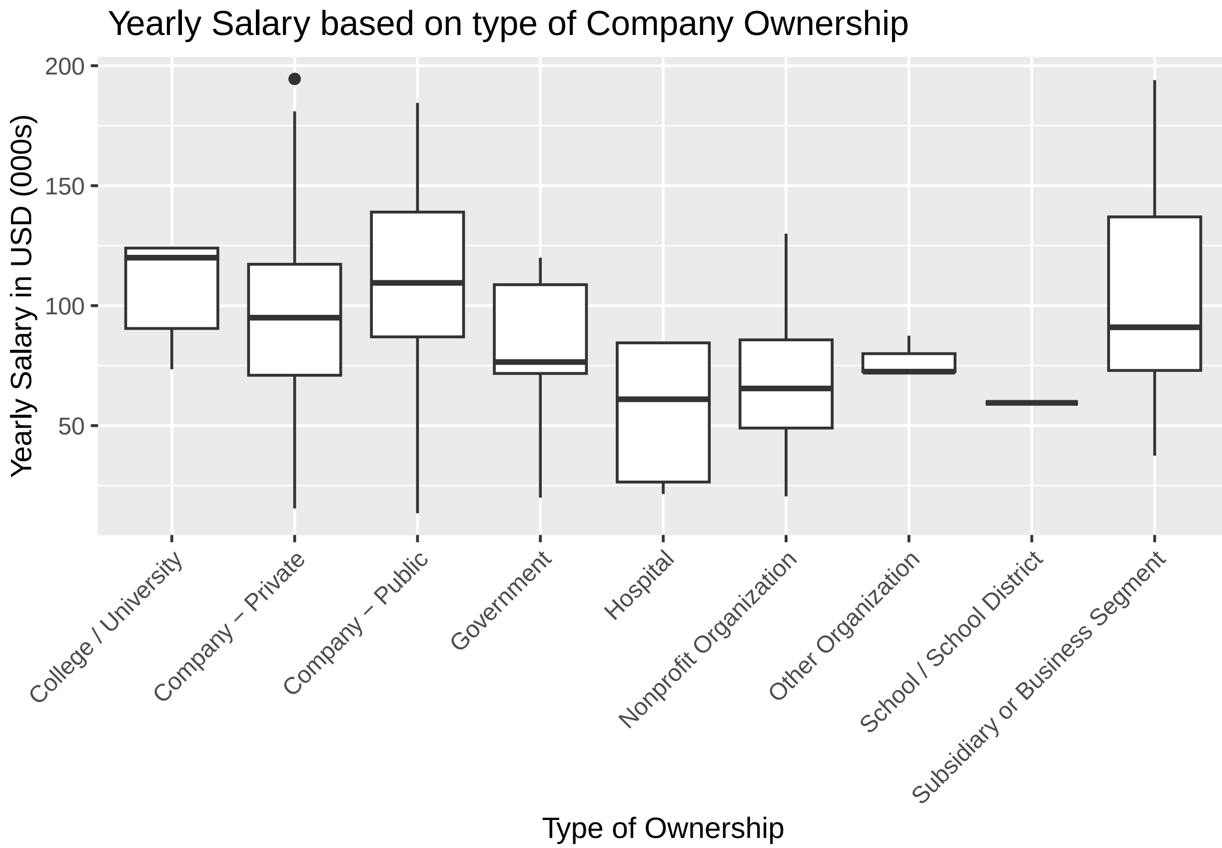
```
by_Sector <- Salary_Final %>% count(Sector, sort = TRUE)

barchart <- ggplot(by_Sector,aes(x = Sector, y = n))+
  geom_col()+
  xlab("Type of Sector")+
  ylab("Frequency")+
  coord_flip()+
  ggtitle(" Frequency of Sector")

barchart
```



The type of sector and frequency shows the frequency of jobs in each sector, which is imbalanced as the frequency of jobs in business sector, IT sector, Biotech and Pharmaceuticals is high compared to other job sectors. The frequency of jobs in agricultural sector is the lowest, which displays biasness, due to the distribution.

4.3 Average Salary by Type of Company Ownership

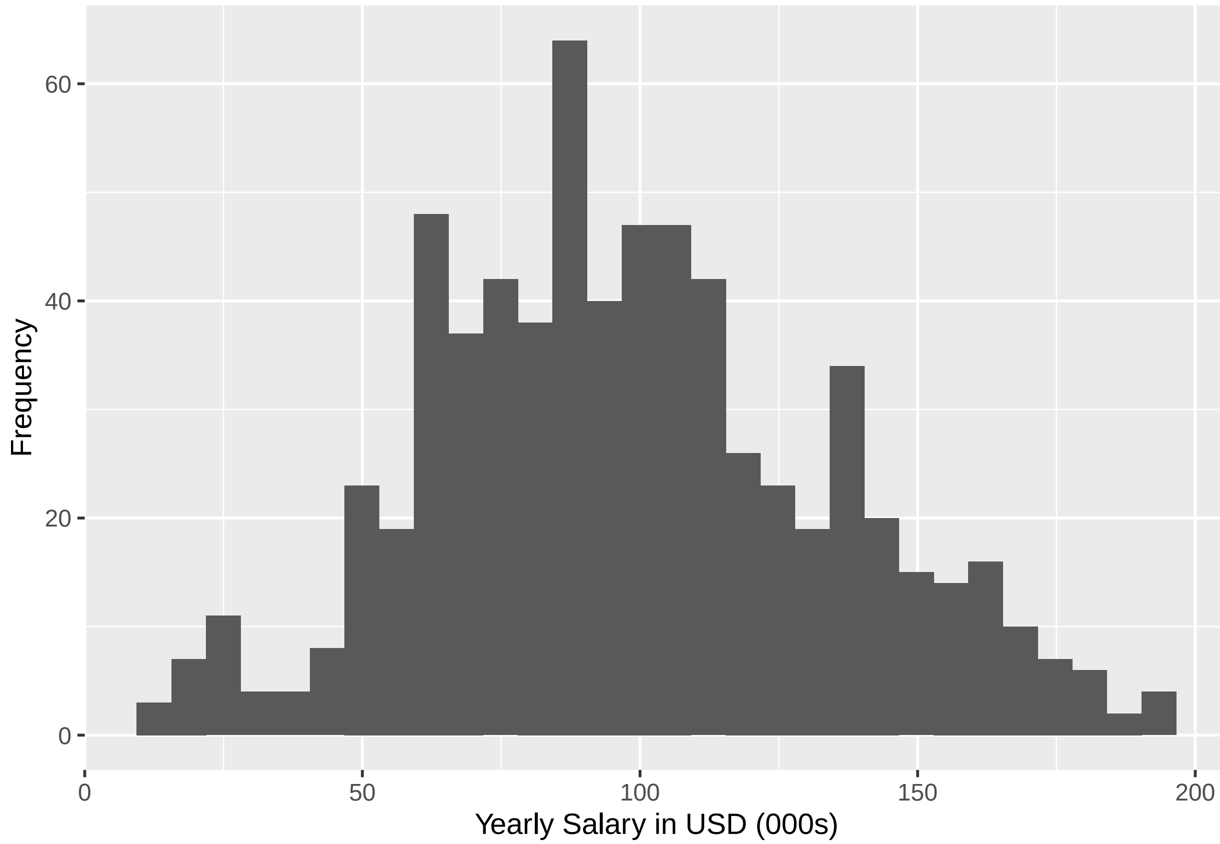
```
owner <- Salary_Final %>%
  select(avg_salary, ownership)%>%
  melt()

boxplot2 <- ggplot(owner, aes(x = ownership, y = value)) +            # Applying ggplot function
  geom_boxplot()+
  xlab("Type of Ownership")+
  ylab("Yearly Salary in USD (000s)")+
  theme(axis.text.x = element_text(size = 9, angle = 45, hjust = 1))+
  ggtitle(" Yearly Salary based on type of Company Ownership")

boxplot2
```



The box plot in Fig, shows a comparison of yearly salary in USD against the type of ownership of the company. In regards to the various types of ownerships, there are significant gaps in the range of salaries offered in different types of organizations, where Subsidiaries or Business Segment has the highest range of yearly salaries for data scientists and School/School District has the lowest range of salary.

4.4 Salary Distribution Histogram

```
sal_hist <- ggplot(Salary_Final, aes(x = avg_salary))+
  geom_histogram()+
  xlab("Yearly Salary in USD (000s)")+
  ylab("Frequency")

sal_hist
```

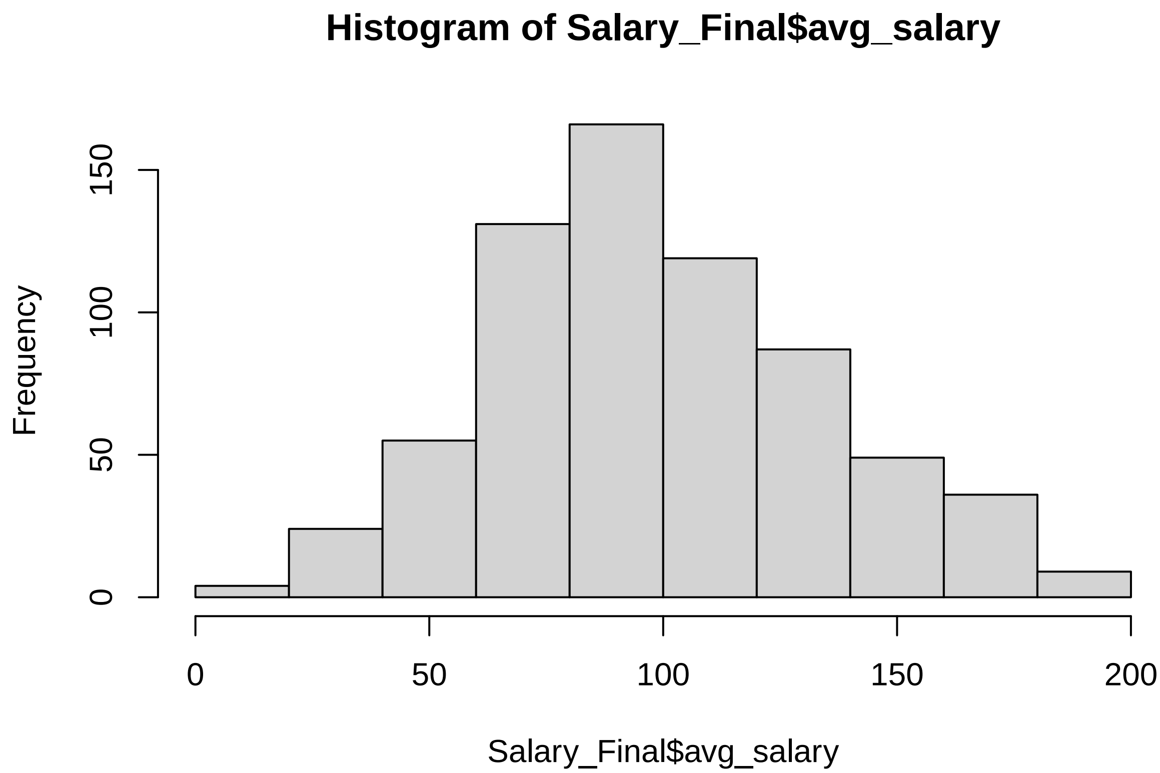


Our distribution of Yearly Salary very roughly follows a bell curve, having the most frequencies in between 50-150K in Salary. Although, there are some peaks in certain Salary levels e.g. 25, 50, 60, 80 and 140. This can create some level of bias to our data. The distribution is also very wide which can create lots of variance and errors in our model.

4.5 Converting continuous average salary to categorical variable

This is use for Feature Selection.

```
hist (Salary_Final$avg_salary)
```




```
summary (Salary_Final$avg_salary)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   13.50   73.00   95.75   98.50  121.62  194.50
```

```
sort (Salary_Final$avg_salary)
```

```
##   [1]  13.5  13.5  15.5  20.0  20.5  20.5  21.5  21.5  21.5  21.5  25.0  25.0
##  [13]  26.5  26.5  26.5  27.5  27.5  27.5  27.5  27.5  27.5  29.5  31.5  31.5
##  [25]  31.5  37.0  37.5  37.5  40.5  43.0  43.0  44.0  44.0  44.5  45.5  45.5
##  [37]  45.5  47.0  47.0  47.0  47.5  48.0  48.0  48.5  48.5  48.5  48.5  49.0
##  [49]  49.0  49.0  50.0  51.0  51.0  51.5  51.5  51.5  51.5  52.5  52.5  53.0
##  [61]  53.5  53.5  53.5  54.0  54.0  54.0  54.0  54.0  54.0  55.0  56.5  56.5
##  [73]  56.5  56.5  58.0  58.5  59.0  59.0  59.0  59.5  60.0  60.0  60.0  60.5
##  [85]  60.5  61.0  61.0  61.0  61.0  61.0  61.0  61.0  61.5  61.5  61.5  61.5
##  [97]  61.5  61.5  62.0  62.0  62.0  62.5  62.5  62.5  62.5  62.5  62.5  62.5
## [109]  63.0  63.0  63.5  63.5  64.0  64.0  64.0  64.0  64.5  64.5  65.0  65.0
## [121]  65.0  65.0  65.5  65.5  65.5  65.5  65.5  66.0  66.0  66.5  66.5  66.5
## [133]  66.5  66.5  67.0  67.0  67.0  68.0  68.5  68.5  68.5  68.5  68.5  68.5
## [145]  69.0  69.5  69.5  69.5  69.5  70.0  70.0  70.0  70.5  70.5  70.5  70.5
## [157]  70.5  70.5  71.0  71.0  71.5  71.5  71.5  71.5  72.0  72.5  72.5  72.5
## [169]  72.5  73.0  73.0  73.0  73.0  73.0  73.5  73.5  73.5  73.5  73.5  74.0
## [181]  74.0  74.5  74.5  75.5  75.5  75.5  75.5  75.5  76.0  76.0  76.5  76.5
## [193]  76.5  76.5  76.5  76.5  76.5  77.0  77.5  77.5  77.5  77.5  77.5  77.5
```

```
## [205]  77.5  78.0  79.0  79.5  79.5  80.0  80.0  80.0  80.0  80.0  80.5  80.5
## [217]  80.5  80.5  80.5  80.5  80.5  81.0  81.0  81.0  81.0  81.0  81.0  81.0
## [229]  81.0  81.0  81.0  81.0  81.5  81.5  82.0  82.0  82.5  83.0  83.0  83.5
## [241]  84.0  84.0  84.0  84.0  84.5  84.5  84.5  84.5  84.5  84.5  84.5  84.5
## [253]  84.5  84.5  85.0  85.0  85.0  85.0  85.0  85.0  85.0  85.0  85.0  85.0
## [265]  85.5  85.5  85.5  85.5  85.5  85.5  85.5  86.0  86.0  86.5  86.5  86.5
## [277]  86.5  87.0  87.0  87.0  87.0  87.0  87.0  87.0  87.0  87.5  87.5  87.5
## [289]  87.5  87.5  87.5  87.5  87.5  87.5  87.5  87.5  88.0  88.0  88.5  89.0
## [301]  89.5  90.0  90.0  90.0  90.0  90.5  90.5  90.5  91.0  91.0  91.5  92.0
## [313]  92.0  92.0  92.0  92.0  92.5  92.5  93.0  93.5  93.5  93.5  93.5  93.5
## [325]  94.0  94.5  94.5  94.5  94.5  94.5  94.5  95.0  95.0  95.0  95.0  95.0
## [337]  95.0  95.0  95.5  95.5  96.0  96.0  96.0  96.0  96.0  96.5  96.5  96.5
## [349]  97.0  97.0  97.5  97.5  97.5  97.5  98.0  98.0  98.0  98.0  98.0  98.5
## [361]  98.5  98.5  98.5  99.0  99.0  99.0  99.0  99.0  99.5  99.5  99.5  99.5
## [373]  99.5 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.5 100.5 100.5 100.5
## [385] 101.0 101.0 101.0 101.0 101.0 101.5 102.0 102.0 102.5 102.5 102.5 103.0
## [397] 103.0 103.5 103.5 103.5 103.5 103.5 104.5 104.5 104.5 105.0 105.5 106.0
## [409] 106.0 106.5 106.5 106.5 106.5 106.5 106.5 107.0 107.0 107.0 107.0 107.0
## [421] 107.0 107.0 107.0 107.5 107.5 107.5 107.5 107.5 107.5 107.5 107.5 107.5
## [433] 107.5 108.0 108.0 109.0 109.0 109.0 109.0 109.0 109.0 109.0 109.5 109.5
## [445] 109.5 109.5 110.0 110.0 110.0 110.5 110.5 110.5 111.0 111.5 111.5 112.0
## [457] 112.0 112.5 112.5 112.5 112.5 112.5 112.5 113.0 113.0 113.5 113.5 113.5
## [469] 114.0 114.0 114.0 114.0 114.0 114.5 114.5 114.5 114.5 114.5 114.5 114.5
## [481] 115.0 115.0 115.0 115.0 116.5 116.5 116.5 117.5 117.5 117.5 118.0 118.5
## [493] 119.0 119.0 119.5 120.0 120.0 120.0 120.0 120.5 120.5 120.5 121.0 121.0
## [505] 121.0 121.0 121.0 121.0 121.0 121.5 122.0 122.0 122.0 122.5 122.5 122.5
## [517] 123.5 123.5 123.5 124.0 124.0 124.0 124.0 124.0 124.0 124.5 124.5 124.5
## [529] 124.5 125.0 125.0 125.0 127.0 128.0 128.0 128.0 128.5 128.5 128.5 129.5
## [541] 129.5 130.0 130.0 130.0 130.0 132.5 132.5 133.0 133.0 133.0 133.5 134.0
## [553] 134.5 134.5 134.5 136.5 136.5 137.0 137.0 137.0 137.5 138.5 138.5 138.5
## [565] 138.5 139.0 139.0 139.0 139.0 139.0 139.5 139.5 139.5 139.5 139.5 140.0
## [577] 140.0 140.0 140.0 140.0 140.0 140.0 140.0 140.0 140.0 140.0 140.5 140.5
## [589] 140.5 142.0 142.0 142.0 142.5 142.5 142.5 142.5 143.0 143.0 143.5 143.5
## [601] 145.0 145.0 145.5 146.0 146.5 146.5 147.0 147.0 147.0 147.0 147.5 147.5
## [613] 148.0 148.0 149.5 149.5 150.5 150.5 150.5 151.5 151.5 153.0 153.0 153.0
## [625] 153.5 153.5 154.5 154.5 154.5 154.5 154.5 154.5 154.5 154.5 157.0 161.5
## [637] 161.5 161.5 161.5 162.0 162.0 162.0 162.0 162.5 163.0 163.5 164.0 164.0
## [649] 164.5 164.5 165.0 167.5 167.5 167.5 168.0 168.0 169.0 169.0 169.0 171.5
## [661] 171.5 172.0 172.0 173.0 173.0 173.0 174.0 177.0 179.5 180.0 180.0 181.0
## [673] 181.0 181.0 184.5 184.5 194.0 194.0 194.5 194.5
```

```
Salary_Final <- within(Salary_Final, {
  Salary_cat <- NA
  Salary_cat[avg_salary < 15.5] <- "Very Low"
  Salary_cat[avg_salary >= 15.5 & avg_salary < 73.0] <- "Low"
  Salary_cat[avg_salary >= 73.0 & avg_salary < 98.5] <- "Average"
  Salary_cat[avg_salary >= 98.5 & avg_salary < 121.62] <- "High"
  Salary_cat[avg_salary >= 121.62] <- "Very High"

})

Salary_Final$Salary_cat <- factor(Salary_Final$Salary_cat, levels = c("Very High","High", "Average", "Lo
str(Salary_Final)
```

```
## tibble [680 x 14] (S3: tbl_df/tbl/data.frame)
##  $ Rating    : num [1:680] 3.8 3.4 4.8 3.8 2.9 3.4 4.1 3.8 3.3 4.6 ...
##  $ Size      : Ord.factor w/ 9 levels "-1"<"1 to 50 employees"<..: 7 3 7 4 8 5 7 5 3 8 ...
##  $ Founded   : int [1:680] 1973 1984 2010 1965 1998 2000 2008 2005 2014 2009 ...
##  $ ownership : Factor w/ 11 levels "-1","College / University",..: 3 8 3 5 3 4 3 3 6 3 ...
##  $ Sector    : Factor w/ 25 levels "-1","Accounting & Legal",..: 3 13 7 20 7 21 11 7 13 14 ...
##  $ Revenue   : Ord.factor w/ 14 levels "-1"<"$1 to $2 billion (USD)"<..: 11 7 6 12 5 2 12 8 12 6 ...
##  $ avg_salary: num [1:680] 72 87.5 85 76.5 114.5 ...
##  $ job_state : Factor w/ 38 levels "AL","AZ","CA",..: 26 19 8 37 27 34 19 3 27 27 ...
##  $ python_yn : logi [1:680] TRUE TRUE TRUE TRUE TRUE TRUE ...
##  $ R_yn      : logi [1:680] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ spark     : logi [1:680] FALSE FALSE TRUE FALSE FALSE FALSE ...
##  $ aws       : logi [1:680] FALSE FALSE FALSE FALSE FALSE TRUE ...
##  $ excel     : logi [1:680] TRUE FALSE TRUE FALSE TRUE TRUE ...
##  $ Salary_cat: Factor w/ 5 levels "Very High","High",..: 4 3 3 3 2 3 3 2 4 1 ...
```

```
summary(Salary_Final$Salary_cat)
```

```
## Very High      High   Average       Low  Very Low
##       170       151       190       167         2
```

5.0 Feature Selection

5.1 Library

5.2 Random Sampling Data

```
data(Salary_Final)
str(Salary_Final)
```

```
## tibble [680 x 14] (S3: tbl_df/tbl/data.frame)
##  $ Rating    : num [1:680] 3.8 3.4 4.8 3.8 2.9 3.4 4.1 3.8 3.3 4.6 ...
##  $ Size      : Ord.factor w/ 9 levels "-1"<"1 to 50 employees"<..: 7 3 7 4 8 5 7 5 3 8 ...
##  $ Founded   : int [1:680] 1973 1984 2010 1965 1998 2000 2008 2005 2014 2009 ...
##  $ ownership : Factor w/ 11 levels "-1","College / University",..: 3 8 3 5 3 4 3 3 6 3 ...
##  $ Sector    : Factor w/ 25 levels "-1","Accounting & Legal",..: 3 13 7 20 7 21 11 7 13 14 ...
##  $ Revenue   : Ord.factor w/ 14 levels "-1"<"$1 to $2 billion (USD)"<..: 11 7 6 12 5 2 12 8 12 6 ...
##  $ avg_salary: num [1:680] 72 87.5 85 76.5 114.5 ...
##  $ job_state : Factor w/ 38 levels "AL","AZ","CA",..: 26 19 8 37 27 34 19 3 27 27 ...
##  $ python_yn : logi [1:680] TRUE TRUE TRUE TRUE TRUE TRUE ...
##  $ R_yn      : logi [1:680] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ spark     : logi [1:680] FALSE FALSE TRUE FALSE FALSE FALSE ...
##  $ aws       : logi [1:680] FALSE FALSE FALSE FALSE FALSE TRUE ...
##  $ excel     : logi [1:680] TRUE FALSE TRUE FALSE TRUE TRUE ...
##  $ Salary_cat: Factor w/ 5 levels "Very High","High",..: 4 3 3 3 2 3 3 2 4 1 ...
```

```
view(Salary_Final)
```

5.3 K-Fold Cross Validation

Before proceeding to cross-validation, it is important to split between the training and testing data of Salary_Final with a proportion of 80% : 20%.

```
RNGkind (sample.kind = "Rounding")
set.seed(100)

insample <- sample(nrow(Salary_Final), nrow(Salary_Final)*0.8)
RF_train <- Salary_Final[insample,]
RF_test <- Salary_Final[insample,]
```

Now, it is important to check on the proportion of the target class of the Salary_Final data.

```
prop.table(table(RF_train$Salary_cat))
```

```
##
##   Very High        High     Average         Low    Very Low
## 0.250000000 0.226102941 0.268382353 0.253676471 0.001838235
```

Model Fitting

```r
set.seed (100)
control <- trainControl(method = "repeatedcv", number = 5, repeats =3)

model_RF <- train (Salary_cat ~ ., data = RF_train, method = "rf", trainControl = control)

saveRDS(model_RF, "model_RF.RDS")
```

Read Model

```r
model_RF <- readRDS("model_RF.RDS")
model_RF
```

```
## Random Forest
##
## 544 samples
##  13 predictor
##   5 classes: 'Very High', 'High', 'Average', 'Low', 'Very Low'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 544, 544, 544, 544, 544, 544, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     2   0.5182657  0.3572051
##    51   0.9994132  0.9992130
##   100   1.0000000  1.0000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 100.
```

From the results, we can understand as the following:

1. 544 samples -> the number of rows on our data train used in creating the model
2. 13 predictor -> the number of predictor variables in our data train.
3. 5 classes -> the number of target class on our data
4. Summary of sample sizes -> the number of sample size on our data train based on the k-fold cross validation.
5. mtry and accuracy -> shows the number of mtry used and the number of accuracy based on each entry.

From the model summary, after doing several trials of mtry the number of mtry that we can choose is 100, which has the highest accuracy when tested in the test data from the boostrap sampling.

5.4 Out of Bag Error

The boostrap sampling produces unused data when making random forest. These data are called out-of-bag data and are considered as data test by the model. The model will then try to do prediction using those data and calculate the error. The error is called out-of-bag error.

```
model_RF$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, trainControl = ..1)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 100
##
##         OOB estimate of  error rate: 0.18%
## Confusion matrix:
##           Very High High Average Low Very Low class.error
## Very High       136    0       0   0        0           0
## High              0  123       0   0        0           0
## Average           0    0     146   0        0           0
## Low               0    0       0 138        0           0
## Very Low          0    0       0   1        0           1
```

From the above model, our out-of-bag error rate is 0.18%, which means our model's accuracy is 100%. Based on the model, let's see what are the predictors that highly affect salary of a person.

```
varImp(model_RF)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 100)
##
##                                    Overall
## avg_salary                        100.0000
## job_stateTN                         0.2337
## SectorBiotech & Pharmaceuticals     0.0000
## job_stateWI                         0.0000
## job_stateDC                         0.0000
## SectorEducation                     0.0000
## Revenue^11                          0.0000
## Revenue^4                           0.0000
## job_stateOH                         0.0000
## SectorManufacturing                 0.0000
## job_stateLA                         0.0000
## SectorInformation Technology        0.0000
## Founded                             0.0000
## job_stateLos Angeles                0.0000
## R_ynTRUE                            0.0000
## job_stateDE                         0.0000
## Size^7                              0.0000
## Revenue^9                           0.0000
## SectorConsumer Services             0.0000
## SectorOil, Gas, Energy & Utilities   0.0000
```

From the overall above, setting aside avg_salary as we have create a new column categorical class Salary_cat through avg_salary. So that we can predict the classes of range in average salary.

1. Company in job_stateTN is the most important predictor for determining one's salary.
2. Second: Rating of the company
3. Revenue.L: The company revenue is important even though it is indicate as Low compared to others.

5.5 Prediction and Model Evaluation

```
model_RF_test <- predict(model_RF, newdata = RF_test)
confusionMatrix(as.factor(model_RF_test), RF_test$Salary_cat)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Very High High Average Low Very Low
##   Very High       136    0       0   0        0
##   High              0  123       0   0        0
##   Average           0    0     146   0        0
##   Low               0    0       0 138        0
##   Very Low          0    0       0   0        1
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9932, 1)
##     No Information Rate : 0.2684
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Very High Class: High Class: Average Class: Low
## Sensitivity                      1.00      1.0000         1.0000     1.0000
## Specificity                      1.00      1.0000         1.0000     1.0000
## Pos Pred Value                   1.00      1.0000         1.0000     1.0000
## Neg Pred Value                   1.00      1.0000         1.0000     1.0000
## Prevalence                       0.25      0.2261         0.2684     0.2537
## Detection Rate                   0.25      0.2261         0.2684     0.2537
## Detection Prevalence             0.25      0.2261         0.2684     0.2537
## Balanced Accuracy                1.00      1.0000         1.0000     1.0000
##                      Class: Very Low
## Sensitivity                 1.000000
## Specificity                 1.000000
## Pos Pred Value              1.000000
## Neg Pred Value              1.000000
## Prevalence                  0.001838
## Detection Rate              0.001838
## Detection Prevalence        0.001838
## Balanced Accuracy           1.000000
```

6.0 Support Vector Machine

6.1 Import the necessary data attribute for SVM model

```
#install.packages("e1071")
library(e1071)

Salary_Final_SVM <- Salary_Final[, c("Rating", "avg_salary","python_yn")]
```

6.2 Change the True = 0 and False = 1

Also, change the data type for python_yn for SVM model. python_yn is now in "numeric" data type.

```
levels(Salary_Final_SVM$python_yn) <- c("0", "1")
Salary_Final_SVM$python_yn <- as.numeric(Salary_Final_SVM$python_yn)
class(Salary_Final_SVM$python_yn)
```

```
## [1] "numeric"
```

6.3 Import the necessary data attribute for SVM model

```
#install.packages("e1071")
library(e1071)

Salary_Final_SVM <- Salary_Final[, c("Rating", "avg_salary","python_yn")]
```

6.4 Encoding the target feature as factor

```
Salary_Final_SVM$python_yn <- as.logical(Salary_Final$python_yn, levels = c(0,1))
class(Salary_Final$python_yn)
```

```
## [1] "logical"
```

```
Salary_Final_SVM
```

```
## # A tibble: 680 x 3
##    Rating avg_salary python_yn
##     <dbl>      <dbl> <lgl>
##  1    3.8       72   TRUE
##  2    3.4       87.5 TRUE
##  3    4.8       85   TRUE
##  4    3.8       76.5 TRUE
##  5    2.9      114.  TRUE
##  6    3.4       95   TRUE
##  7    4.1       73.5 FALSE
```

```
## 8     3.8       114   TRUE
##  9     3.3        61   FALSE
## 10     4.6       140   TRUE
## # ... with 670 more rows
```

6.5 Splitting the dataset

```
#install.packages('caTools')
library(caTools)

set.seed(123)
split = sample.split(Salary_Final_SVM$python_yn, SplitRatio = 0.80)
split
```

```
##   [1]  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
##  [13]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
##  [25]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE
##  [37]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE
##  [49] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
##  [61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
##  [73]  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [85]  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [97]  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE
## [109]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [121]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [133]  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
## [145]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
## [157]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
## [169]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
## [181] FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [193]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
## [205]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [217]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
## [229]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE
## [241]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE
## [253]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [265]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
## [277]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
## [289]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE
## [301] FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
## [313] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE
## [325]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [337] FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
## [349]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
## [361]  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [373] FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
## [385]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
## [397] FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [409] FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
## [421]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
## [433] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [445] FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
```

```
## [457]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE
## [469]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE
## [481]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
## [493]  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE
## [505]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
## [517]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [529] FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE
## [541] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
## [553]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
## [565]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
## [577]  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
## [589]  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
## [601] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE
## [613]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [625]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE
## [637]  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
## [649]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
## [661]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [673] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
```

```
training_set = subset(Salary_Final_SVM, split == TRUE)
test_set = subset(Salary_Final_SVM, split == FALSE)
```

6.6 Feature Scaling

It is often necessary to perform feature scaling when using Support Vector Machines (SVM).

SVM is sensitive to the scale of the features, so if the features have different scales, the model may give more weight to the features with larger scales, and this can impact the model's performance.

Feature scaling, also known as normalization, addresses this issue by transforming the features so that they have the same scale.

6.7 Fitting SVM to the training set

```
#install.packages('e1071')
library(e1071)

classifier = svm(formula = python_yn ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'linear')

classifier
```

```
##
## Call:
## svm(formula = python_yn ~ ., data = training_set, type = "C-classification",
##     kernel = "linear")
##
##
```

```
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  436
```

6.8 Predicting the test set result

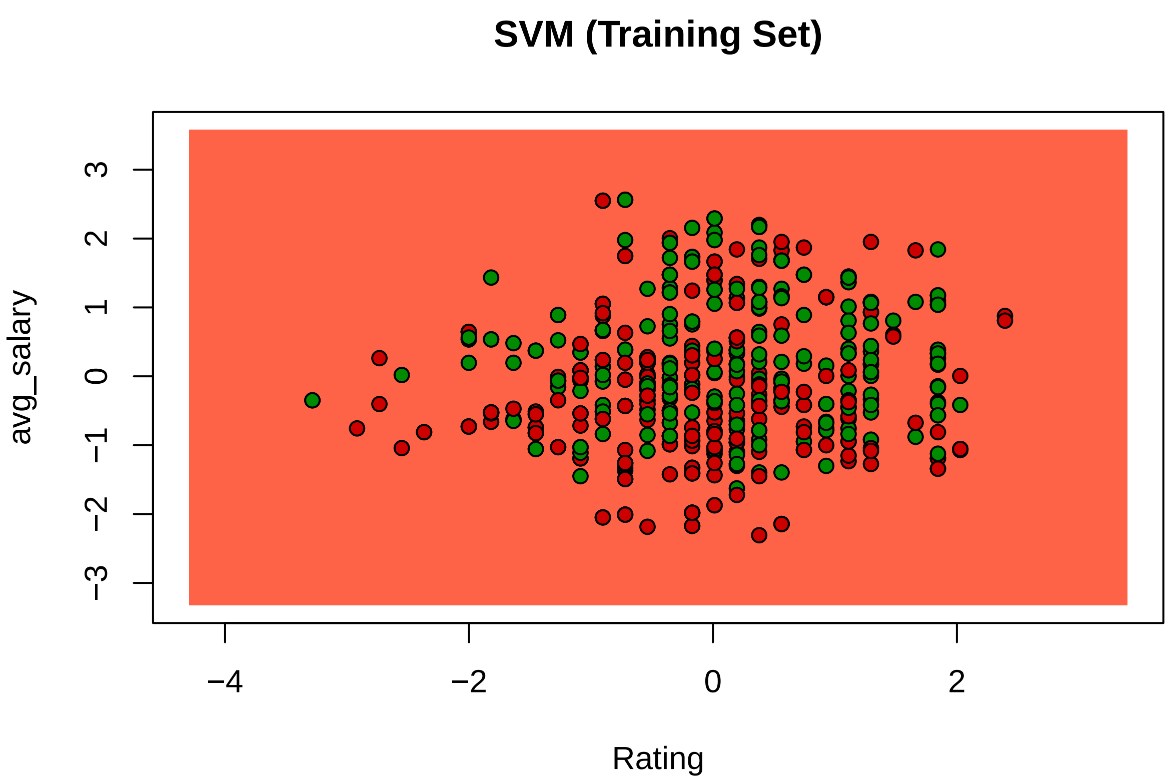
```r
y_pred = predict(classifier, newdata = test_set[-3])
y_pred
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12    13
## FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE
##    14    15    16    17    18    19    20    21    22    23    24    25    26
##  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE
##    27    28    29    30    31    32    33    34    35    36    37    38    39
##  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
##    40    41    42    43    44    45    46    47    48    49    50    51    52
## FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
##    53    54    55    56    57    58    59    60    61    62    63    64    65
## FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE
##    66    67    68    69    70    71    72    73    74    75    76    77    78
## FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
##    79    80    81    82    83    84    85    86    87    88    89    90    91
## FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE
##    92    93    94    95    96    97    98    99   100   101   102   103   104
## FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
##   105   106   107   108   109   110   111   112   113   114   115   116   117
##  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE
##   118   119   120   121   122   123   124   125   126   127   128   129   130
##  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
##   131   132   133   134   135   136
##  TRUE FALSE  TRUE FALSE  TRUE  TRUE
## Levels: FALSE TRUE
```

6.9 Visualizing the Training set results

```r
library(caret)

#install.packages('Rfast')
library('Rfast')
set  = training_set
X1 = seq(min(set[, 1]) -1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) -1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Rating', 'avg_salary')
prob_set = predict(classifier, type = 'response', newdata = grid_set)
y_grid = ifelse(prob_set==0, 1, 0)
```

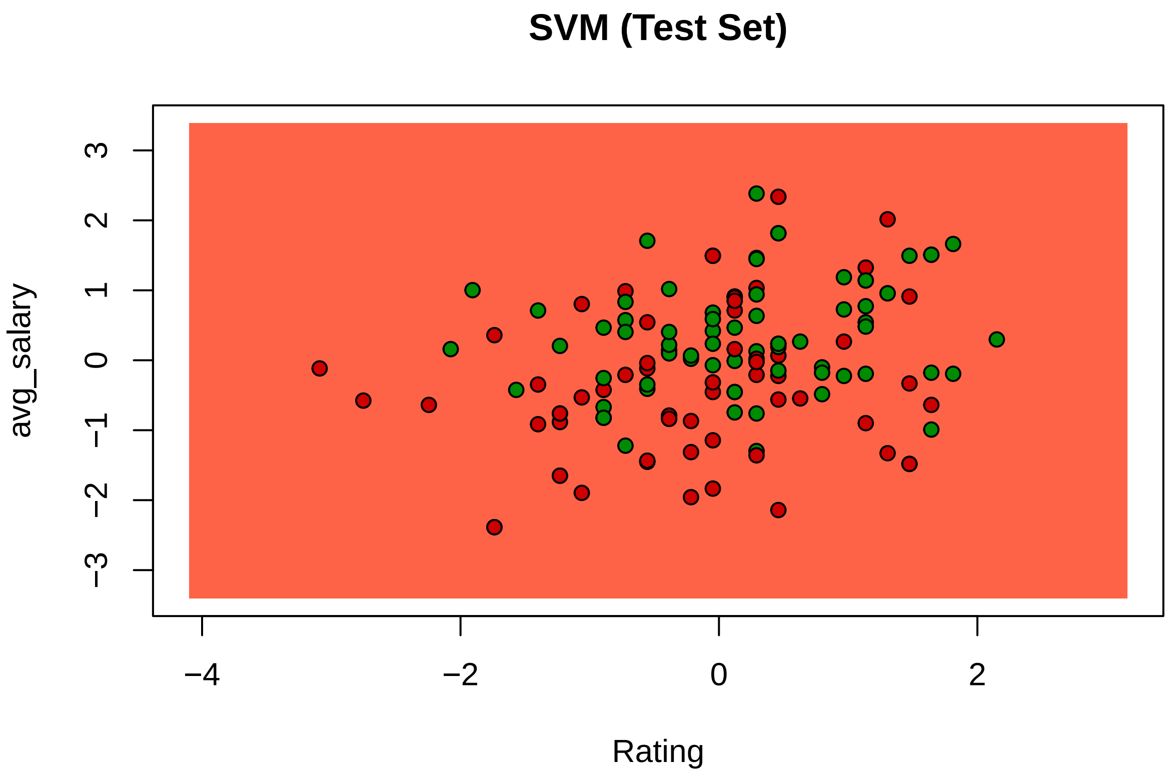
```
plot(set[, -3],
     main = 'SVM (Training Set)',
     xlab = 'Rating',
     ylab = 'avg_salary',
     xlim = range(X1),
     ylim = range(X2)
)
contour(X1, X2, matrix(as.numeric(y_grid),length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid==1, 'springgreen3', 'tomato') )
points(set, pch = 21, bg = ifelse(set[, 3]== 1, 'green4', 'red3'))
```



6.10 Visualizing the Test set results

```
# Visualizing the test set results
# install.packages('Rfast')
library('Rfast')
set  = test_set
X1 = seq(min(set[, 1]) -1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) -1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Rating', 'avg_salary')
prob_set = predict(classifier, type = 'response', newdata = grid_set)
y_grid = ifelse(prob_set == 1, 1, 0)
plot(set[, -3],
     main = 'SVM (Test Set)',
     xlab = 'Rating',
```

```r
     ylab = 'avg_salary',
     xlim = range(X1),
     ylim = range(X2)
)

contour(X1,X2, matrix(as.numeric(y_grid),length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid==1, 'springgreen3', 'tomato') )
points(set, pch = 21, bg = ifelse(set[, 3]== 1, 'green4', 'red3'))
```



6.11 Building SVM Regressor model for comparing Training and Testing

Caret package is used to perform SVM regression.

First, we will use the trainControl() function to define the method of cross validation to be carried out and search type i.e. "grid". Then train the model using train() function.

Syntax: train(formula, data = , method = , trControl = , tuneGrid =)

where:

formula = y~x1+x2+x3+. . . , where y is the independent variable and x1,x2,x3 are the dependent variables data = dataframe method = Type of the model to be built ("svmLinear" for SVM) trControl = Takes the control parameters. We will use trainControl function out here where we will specify the Cross validation technique. tuneGrid = takes the tuning parameters and applies grid search CV on them

```r
# specifying the CV technique which will be passed into the train() function later and number parameter
train_control_training = trainControl(method = "cv", number = 5)
```

```r
set.seed(50)

# training a Regression model while tuning parameters (Method = "rpart")
model_training = train(avg_salary~., data = training_set, method = "svmLinear", trControl = train_contr

# summarising the results
print(model_training)
```

```
## Support Vector Machines with Linear Kernel
##
## 544 samples
##   2 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 434, 436, 435, 437, 434
## Resampling results:
##
##   RMSE       Rsquared    MAE
##   0.9604208  0.09533904  0.7624588
##
## Tuning parameter 'C' was held constant at a value of 1
```

The root mean squared error (RMSE) for training set is 0.9604208.

6.12 Make predictions on the SVM testing model

```r
library(Metrics)


# Fit the SVM model on the training data
svm_model <- svm(avg_salary ~ ., data = training_set, type = "eps-regression")

# Make predictions on the testing data
testing_predictions <- predict(svm_model, newdata = test_set)

# Calculate the RMSE of the testing set
testing_rmse <- rmse(testing_predictions, test_set$avg_salary)

# Print the RMSE of the testing set
print(testing_rmse)
```

```
## [1] 0.9127156
```

The root mean squared error (RMSE) for testing set is 0.9127156.

6.13 Final Results for SVM Model

The testing set in an SVM model for our salary prediction data set is the BETTER FIT SVM model to the data.

As the RMSE Testing = 0.9127156 lower compare to RMSE Training = 0.9604208, it means a lower RMSE value for the testing set indicates that the model is generalizing well to new, unseen data. Overall, it is the best fit to the data. However in general, a good model should have a low MSE on both the training and testing set.

For our results, our training and testing are only 0.0477052 difference.

7.0 Multiple Linear Regression

We will be using the Multiple Linear Regression model as we want to predict the value of Yearly Salary. Instead of the normal linear regression with only 1 variable influencing the predicted label. However, to be more accurate, we will use Multiple linear regression to allow for multiple independent variables for the predictor. Furthermore, this would mean that we need to test and meet the assumptions of Multiple linear regression i.e. the relationship between the independent and dependent variables are linear which can be illustrated with by scatter plots. Secondly, the errors between observed and predicted values (e.g. residuals of regression) should be normally distributed. Another assumption to be tested is that there is no multicollinearity in the data. This occurs when our independent variables are too highly correlated to each other. We will test this with Variance Inflation Factor. Lastly, the final assumption is to check for homoscedasticity, the lacking of any patterns in the scatterplot of residuals against predicted values.

We will restrict the data set to the features identified by our feature selection process for this analysis: "Rating", "python_yn" and "job_state".

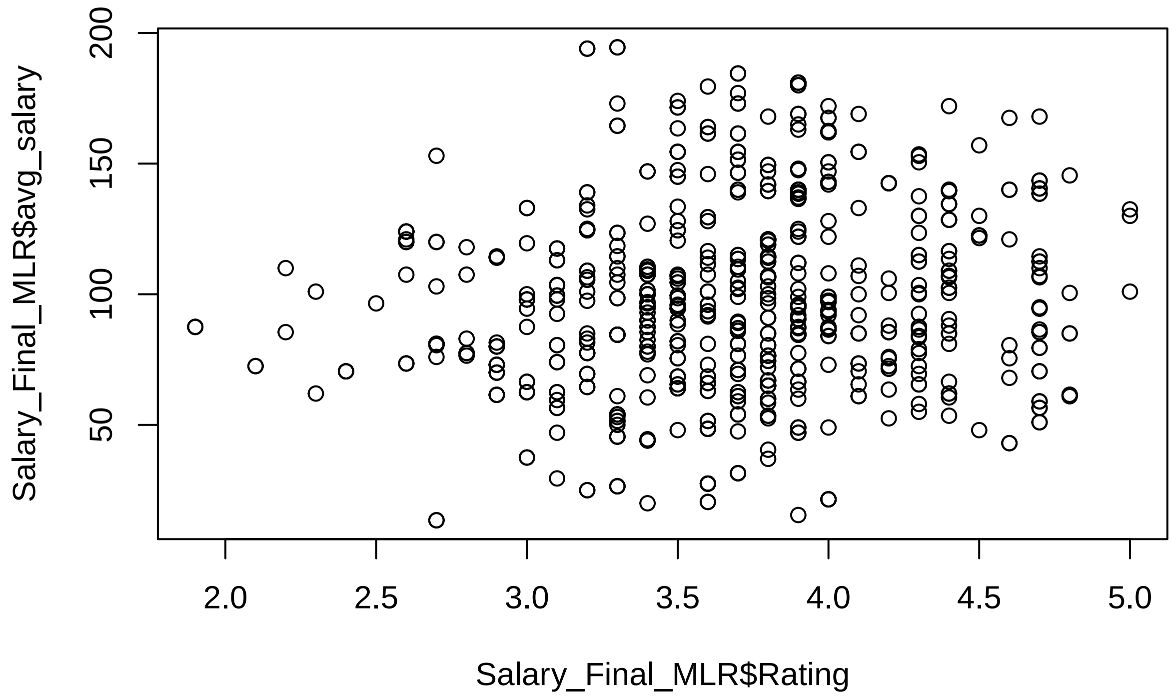
7.1 Generating Multiple Linear Regression model

Generating new Data set

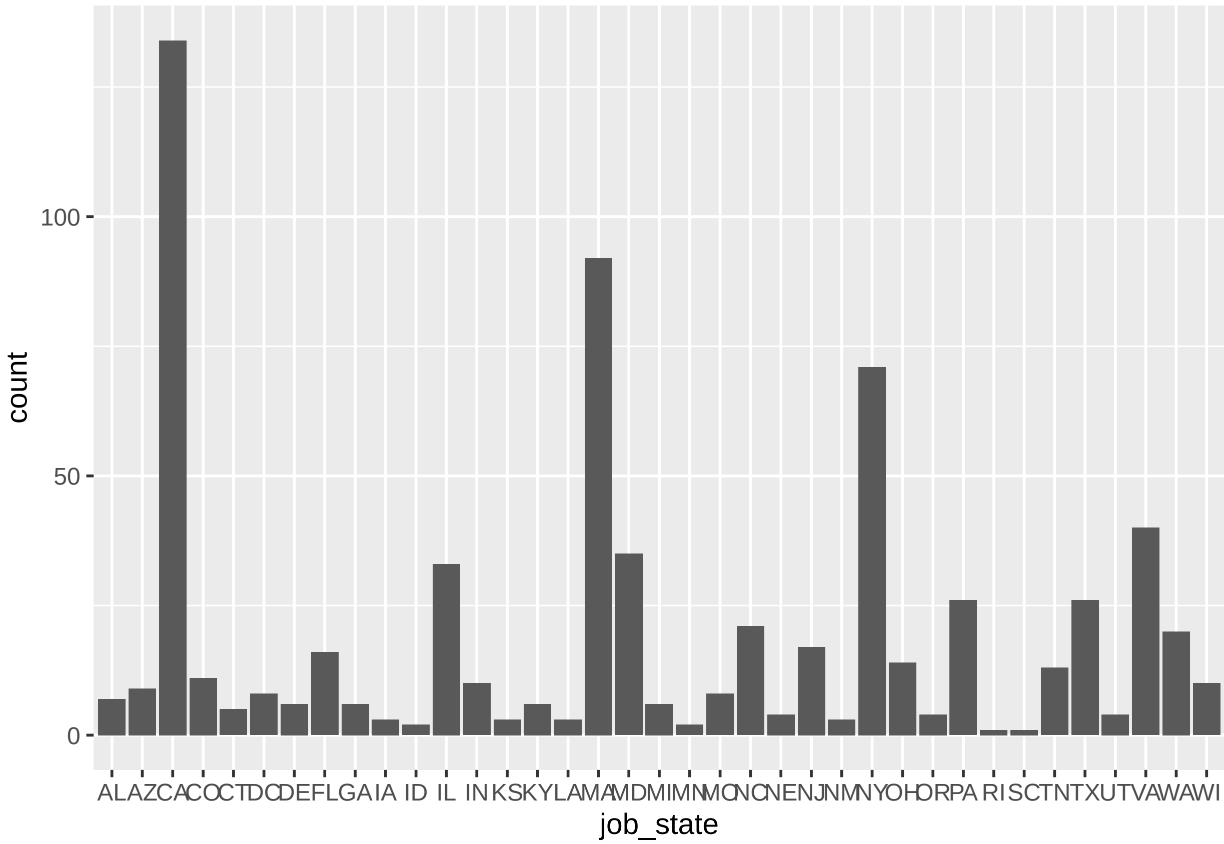
```
Salary_Final_MLR <- Salary_Final[, c("Rating", "avg_salary","python_yn","job_state")]
```

Univariate & Bivariate Analysis

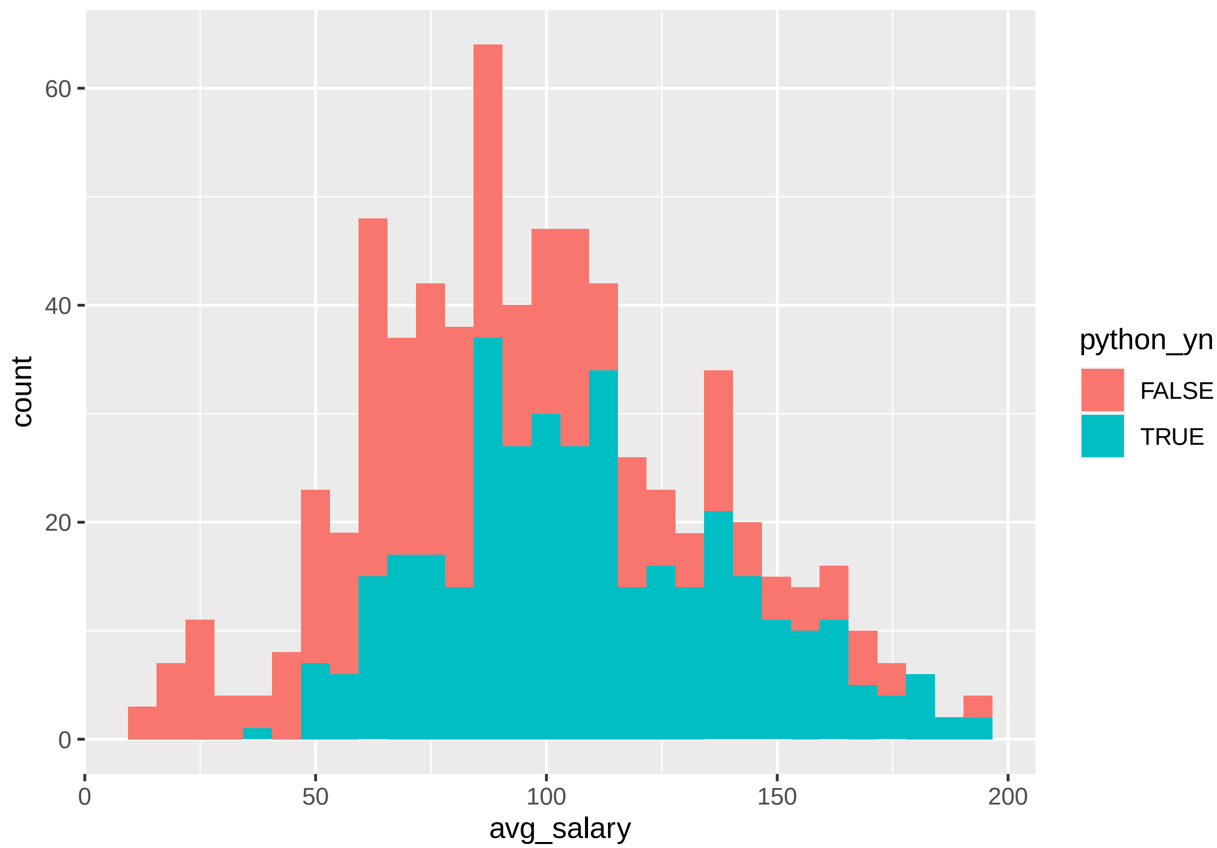
```
x <- plot(Salary_Final_MLR$Rating,Salary_Final_MLR$avg_salary)
```




```
z <- ggplot(Salary_Final_MLR, aes(x=job_state)) + geom_bar()
z
```




```
y <- ggplot(Salary_Final_MLR, aes(x= avg_salary, fill = python_yn)) + geom_histogram()
y
```



Based on our quick illustrations on our independent variables, occurrences of high salary occur most in companies with a rating between 3.0 and 4.0. Furthermore, jobs located in CA, MA, NY and VA have the most frequency, this can imply that the model may struggle with its accuracy for jobs located outside of these areas. Lastly, it can be stated that there are people without a python skill set than with, while also having more frequencies at higher salaries. This can cause bias and inaccuracy to the model due to the imbalance of class. Moreover, both TRUE or FALSE for python_yn is somewhat of a bellcurve for Salary which can mean indicate that it is sort of normally distributed.

However, we should also test whether the Multiple Linear Regression model would perform better on 2 predictors e.g. avg_salary ~ Rating + python_yn.

Creating dataset with 2 Predictors

```
Salary_Final_MLR2 <- subset(Salary_Final_MLR, select = -c(job_state))
```

Splitting Data set into Training and Test set

```
set.seed(123)

# Splitting 80% for training and 20% for testing
sampleset <- sample.split(Salary_Final_MLR$avg_salary, SplitRatio = 0.8)

reg_training1 <- subset(Salary_Final_MLR, sampleset == TRUE)
```

```
reg_test1 <- subset(Salary_Final_MLR, sampleset == FALSE)

reg_training2 <- subset(Salary_Final_MLR2, sampleset == TRUE)
reg_test2 <- subset(Salary_Final_MLR2, sampleset == FALSE)
```

Regression model & Graphical Output

```
MLR1 <- lm(avg_salary ~ Rating + python_yn + job_state, data = reg_training1)
MLR2 <- lm(avg_salary ~  python_yn + Rating, data = reg_training2)
#Rsquared of both models
summary(MLR1)$r.squared
```

```
## [1] 0.3012597
```

```
summary(MLR2)$r.squared
```

```
## [1] 0.1028335
```

```
#RSE of both models
summary(MLR1)$sigma
```

```
## [1] 31.63306
```

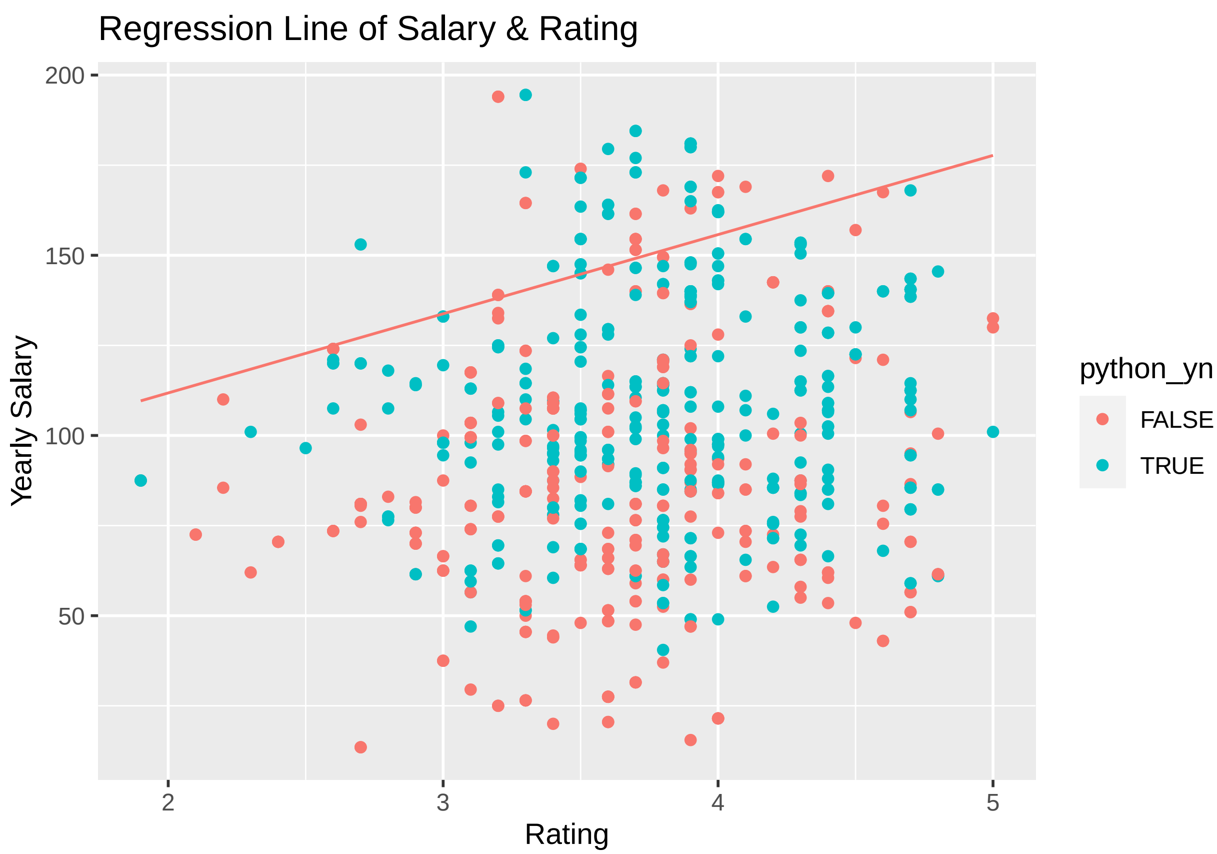
```
summary(MLR2)$sigma
```

```
## [1] 34.67087
```

We will refer to the regression model with 3 predictors as MLR1 and the model with 2 predictors as MLR2. Summary of MLR2 (model with 2 predictors) shows that the the R^2 statistic is 0.103. This means that 10.3% of the variation in avg_salary is accounted by python_yn and Rating whereas the R^2 statistic of MLR1 (model with 3 predictors) is 0.3013 or 30.1%. Therefore, it can be deduced that job_state accounts for 19.8% of variation. This inclusion of job_state increased the variation of almost 20% in Salary. Furthermore, it can be seen that MLR2 has higher RSE on higher degrees of freedom.This can imply MLR2 is less accurate in comparison to MLR1.

We can also explore the regression line of MLR2 as ggplot has the capability of plotting on with 1 continuous predictor and 1 nominal predictor.

```
equation1=function(x){coef(MLR2)[2]*x+coef(MLR2)[1]}


ggplot(Salary_Final_MLR2,aes(y=avg_salary,x=Rating,color=python_yn))+geom_point()+
        stat_function(fun=equation1,geom="line",color=scales::hue_pal()(2)[1])+
        ylab("Yearly Salary")+
        xlab("Rating")+
        ggtitle("Regression Line of Salary & Rating")
```



It can be seen that the regression line does not pass through many of the observations.

7.2 Testing Regression Model Assumptions

Standardized Beta Estimates

```
library(lm.beta)
lm.beta(MLR1)
```

```
##
## Call:
## lm(formula = avg_salary ~ Rating + python_yn + job_state, data = reg_training1)
##
## Standardized Coefficients::
##    (Intercept)         Rating python_ynTRUE    job_stateAZ    job_stateCA
##             NA    0.098919975    0.256778195    0.068914237    0.666229709
##    job_stateCO    job_stateCT    job_stateDC    job_stateDE    job_stateFL
##    0.062901679    0.060427686    0.076095734   -0.051892208    0.051482554
##    job_stateGA    job_stateIA    job_stateID    job_stateIL    job_stateIN
##    0.065204609    0.003438633   -0.006752461    0.291932760    0.082342209
##    job_stateKS    job_stateKY    job_stateLA    job_stateMA    job_stateMD
##    0.039692063    0.109584173    0.022937531    0.456962082    0.270579191
##    job_stateMI    job_stateMN    job_stateMO    job_stateNC    job_stateNE
##    0.066456918    0.037159802    0.080107238    0.218307730   -0.010386858
```

```
##   job_stateNJ   job_stateNM   job_stateNY   job_stateOH   job_stateOR
##   0.208655426   0.011366930   0.302306745   0.103490312   0.062377996
##   job_statePA   job_stateRI   job_stateSC   job_stateTN   job_stateTX
##   0.173583750   0.061945236   0.008659455   0.107943547   0.161668824
##   job_stateUT   job_stateVA   job_stateWA   job_stateWI
##   0.108848558   0.214441190   0.185773301   0.085214174
```

```
lm.beta(MLR2)
```

```
##
## Call:
## lm(formula = avg_salary ~ python_yn + Rating, data = reg_training2)
##
## Standardized Coefficients::
##   (Intercept) python_ynTRUE        Rating
##            NA    0.30045881    0.07615114
```

In regards to MLR1, the beta values of Rating and python_ynTRUE are 0.10 and 0.26 respectively while each job_state are varying e.g. 0.07, -0.05 and 0.666. Beta values indicate the number of standard deviations, changing the outcome depending on how much standard deviation change in the predictor. This means that the higher the beta, the higher the change in Salary. For example, certain states have higher betas in comparison to Rating and python_ynTRUE, jobs in California, New York or Massachusetts have higher influence in generating a higher Salary.

On the other hand, MLR2 only has python_ynTRUE and Rating. Their beta values are 0.30 and 0.08 respectively, this implies that without job_state, python_ynTRUE has higher effects of changing the outcome.

Confidence Intervals

```
confint(MLR1)
```

```
##                     2.5 %    97.5 %
## (Intercept)    -9.786283  57.20799
## Rating          1.371038  11.94182
## python_ynTRUE  13.074331  24.47513
## job_stateAZ   -12.459926  54.93571
## job_stateCA    34.927599  87.25420
## job_stateCO   -14.544802  51.13045
## job_stateCT   -14.097867  66.58117
## job_stateDC    -9.004752  63.06519
## job_stateDE   -58.003787  17.65580
## job_stateFL   -18.281455  42.39928
## job_stateGA   -12.814194  59.13729
## job_stateIA   -42.332347  45.77784
## job_stateID   -54.880723  46.60158
## job_stateIL    23.323171  80.06193
## job_stateIN    -8.215035  58.96724
## job_stateKS   -24.236444  64.00771
## job_stateKY     7.262602  87.91462
## job_stateLA   -32.656897  55.64020
## job_stateMA    22.849077  75.82127
```

```
## job_stateMD    15.994359  71.88973
## job_stateMI   -10.659412  77.24894
## job_stateMN   -27.981343  73.54270
## job_stateMO    -8.881992  58.25666
## job_stateNC    14.800676  73.39041
## job_stateNE   -44.961590  35.94028
## job_stateNJ    18.287930  79.46002
## job_stateNM   -43.942670  57.87958
## job_stateNY     8.488801  61.96416
## job_stateOH    -3.653541  60.80962
## job_stateOR   -12.932322  75.43479
## job_statePA     5.460037  64.66363
## job_stateRI   -13.627817 120.94240
## job_stateSC   -59.699674  74.70141
## job_stateTN    -3.388529  60.28127
## job_stateTX     2.876775  60.83855
## job_stateUT    15.849645 117.60877
## job_stateVA     4.440925  59.50257
## job_stateWA    11.980625  72.17367
## job_stateWI    -8.222225  57.78553
```

```
confint(MLR2)
```

```
##                     2.5 %   97.5 %
## (Intercept)   48.0749676 87.65470
## python_ynTRUE 16.1511542 27.78586
## Rating        -0.2295715 10.47815
```

MLR1 has very poor confidence intervals, the best predictors in this case are Rating and python_ynTRUE which have tighter confidence intervals in comparison to the job_state intervals where it crosses zero. Meanwhile, MLR2 has better confidence intervals in the intercept but very slightly wider in Rating and python_ynTRUE. Specifically, Rating also crosses zero. Overall, python_ynTRUE and Rating show significance tight intervals in comparison to job_state.

Comparing Models

```
anova(MLR2,MLR1)
```

```
## Analysis of Variance Table
##
## Model 1: avg_salary ~ python_yn + Rating
## Model 2: avg_salary ~ Rating + python_yn + job_state
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    559 671957
## 2    523 523340 36    148617 4.1256 2.384e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value is signficantly smaller than 0.01. This means that the alternative hypothesis is accept and that there is a difference between including job_state and not including it, F(36,523) = 4.12.

Assesing the assumption of independence

```
library(car)
durbinWatsonTest(MLR1)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.04241608      2.084813   0.326
##  Alternative hypothesis: rho != 0
```

```
durbinWatsonTest(MLR2)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.03411993      2.065844   0.466
##  Alternative hypothesis: rho != 0
```

Both models have a D-W Statistic that is very close to 2 but very slightly above. This means that our residuals are marginally negative autocorrelated while both p-values are above 0.05. With this result, we can say it passes the assumption of independence.

Assessing the assumption of no multicollinearity

```
v1 <- vif(MLR1)
v1
```

```
##               GVIF Df GVIF^(1/(2*Df))
## Rating    1.196512  1        1.093852
## python_yn 1.178846  1        1.085747
## job_state 1.360840 36        1.004288
```

```
v2 <- vif(MLR2)
v2
```

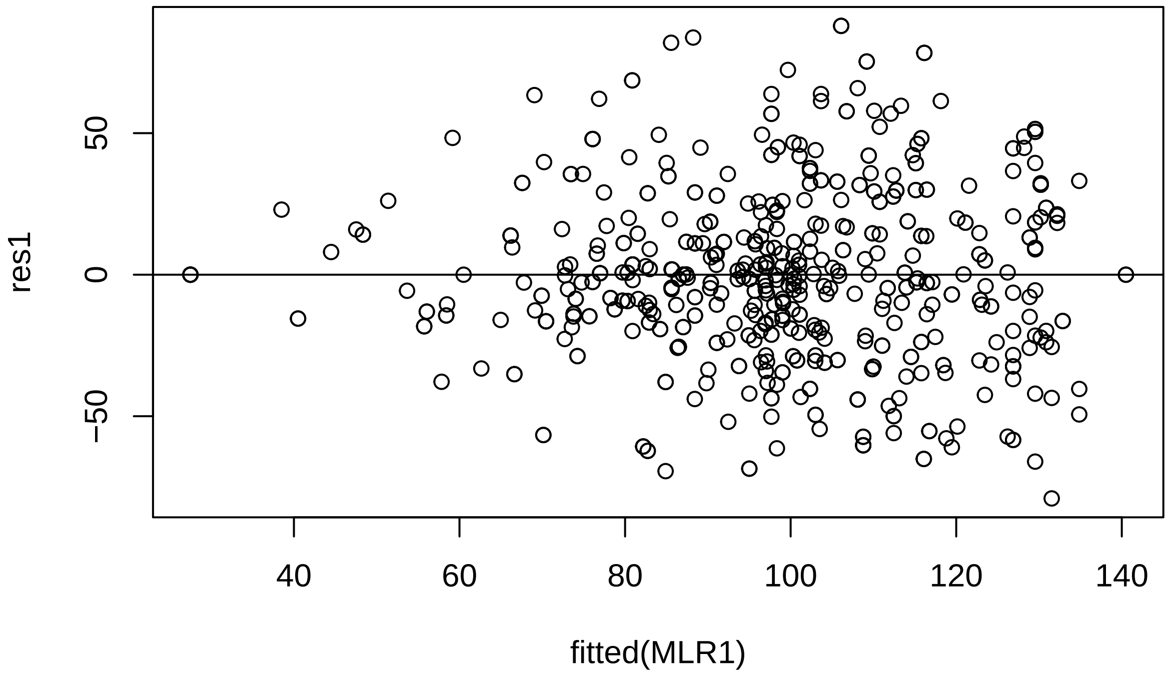
```
## python_yn    Rating
##  1.022303  1.022303
```

The Variance inlfation factor (vif) is a measure of the amount of multicollinearity in the regression model. When multicollinearity exists, it means that there is a correlation between the multiple independent variables, in this case it is a test of how correlated Rating, python_yn and job_state are. In this case, because we are using a polynomial variable for MLR1 , GVIF is generated. Despite that, in both models, all the VIF scores are very close to 1 which can indicate the absence of collinearity between each of the predictors and variables. This suggests that both models also pass the assumption of no multicollinearity.

Assessing our Assumptions of Homoscedasticity (Residuals and Linearity)

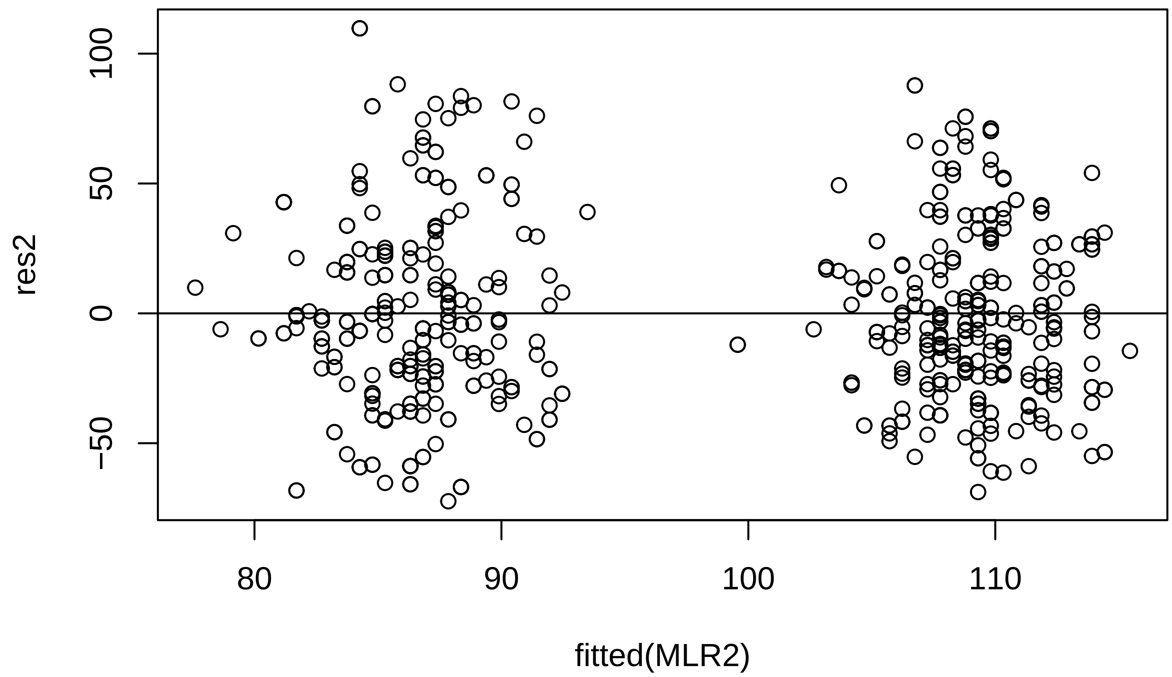
```
#Storing each model's residuals
res1 <- resid(MLR1)
res2 <- resid(MLR2)
```

```r
#produce residual vs. fitted plot
plot(fitted(MLR1), res1)

#add a horizontal line at 0
abline(0,0)
```



In the case for MLR1, the plotting has resulted in somewhat of a funnel-shape. This could imply heteroscedasticity in our model indicating an increase of variance going across the residuals.

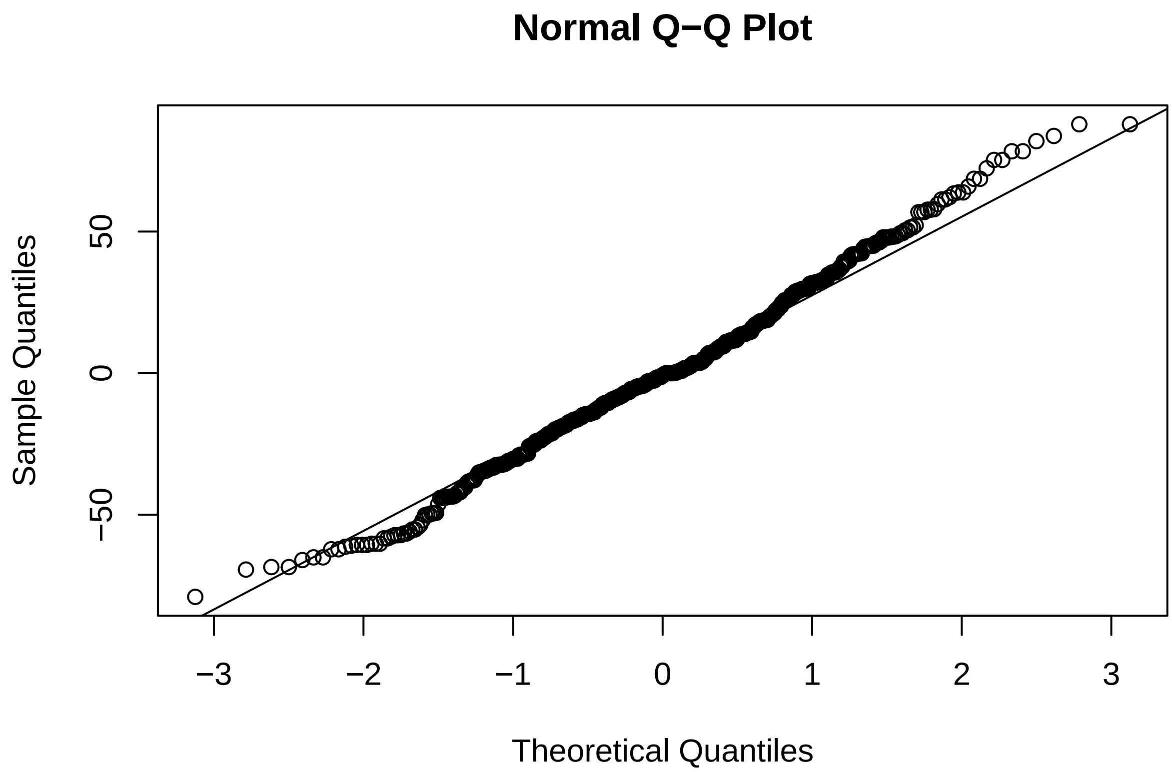
```r
#produce residual vs. fitted plot
plot(fitted(MLR2), res2)

#add a horizontal line at 0
abline(0,0)
```



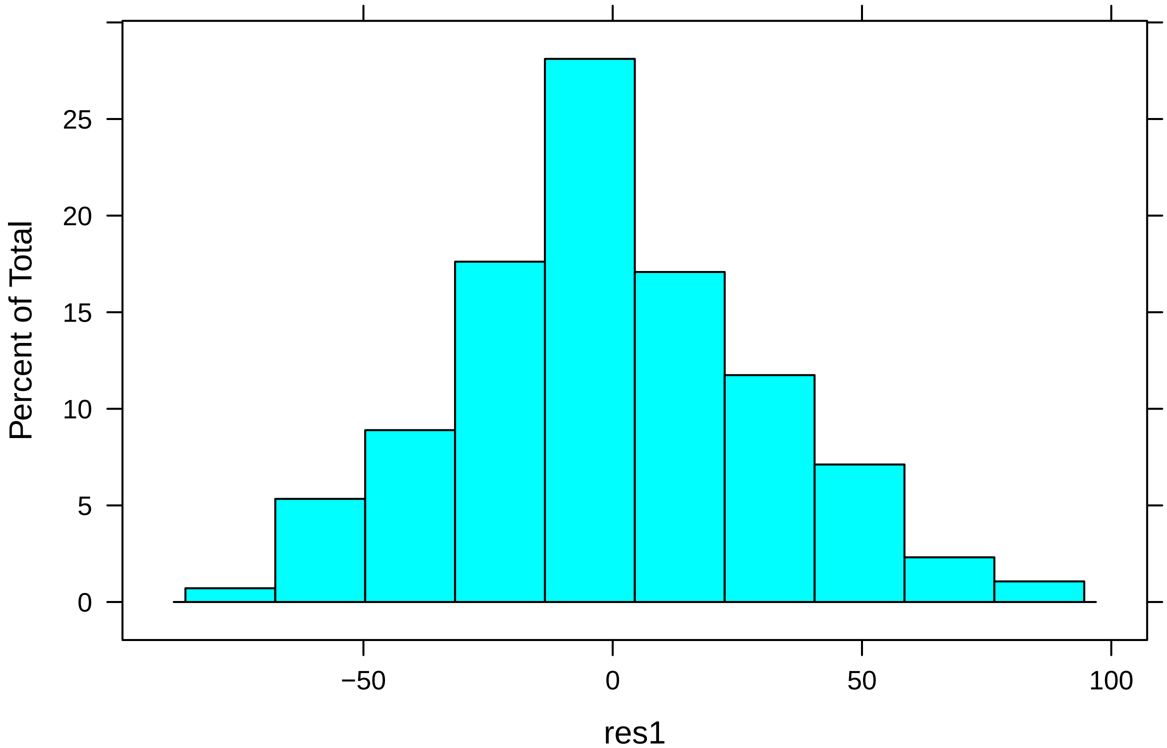
On the other hand, the MLR2 model has generated 2 clusters but show no violation of the assumption of homoscedasticity.

Finally, lets also generate Q-Q and histogram plots to dig deeper into our residuals.

```
#create Q-Q plot for residuals
qqnorm(res1)

#add a straight diagonal line to the plot
qqline(res1)
```



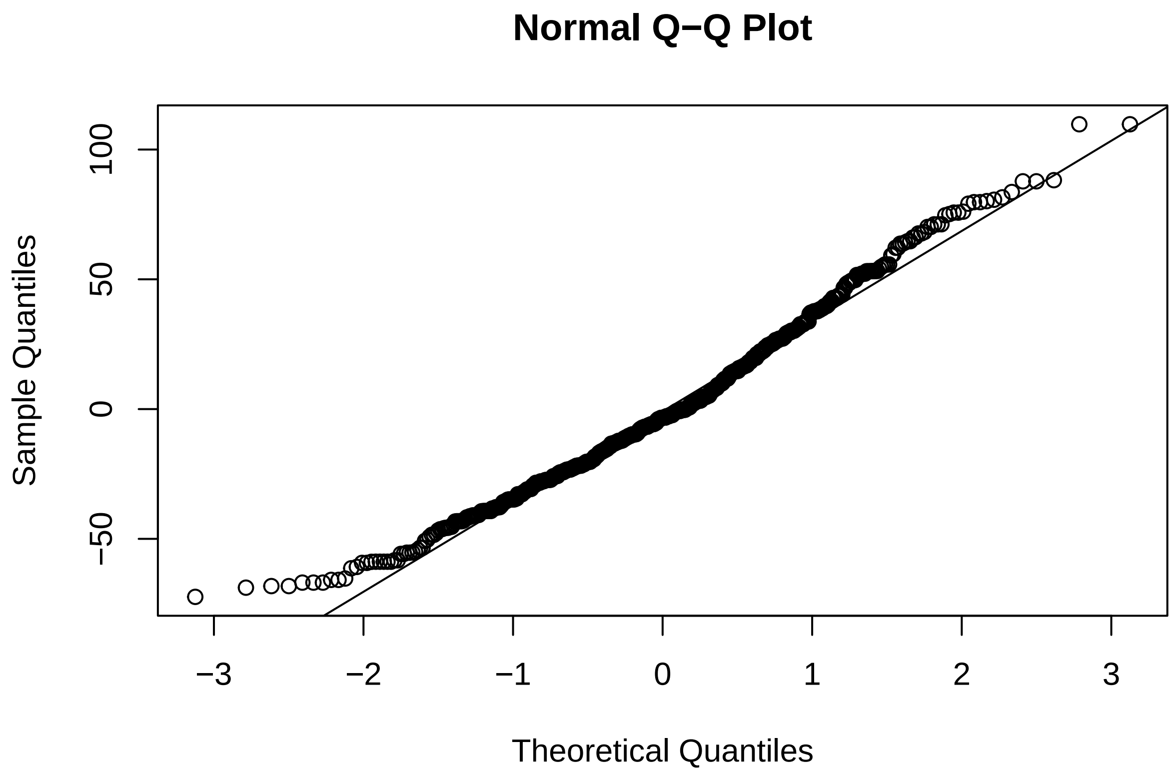

```
#Create density plot of residuals
plot(histogram(res1))
```



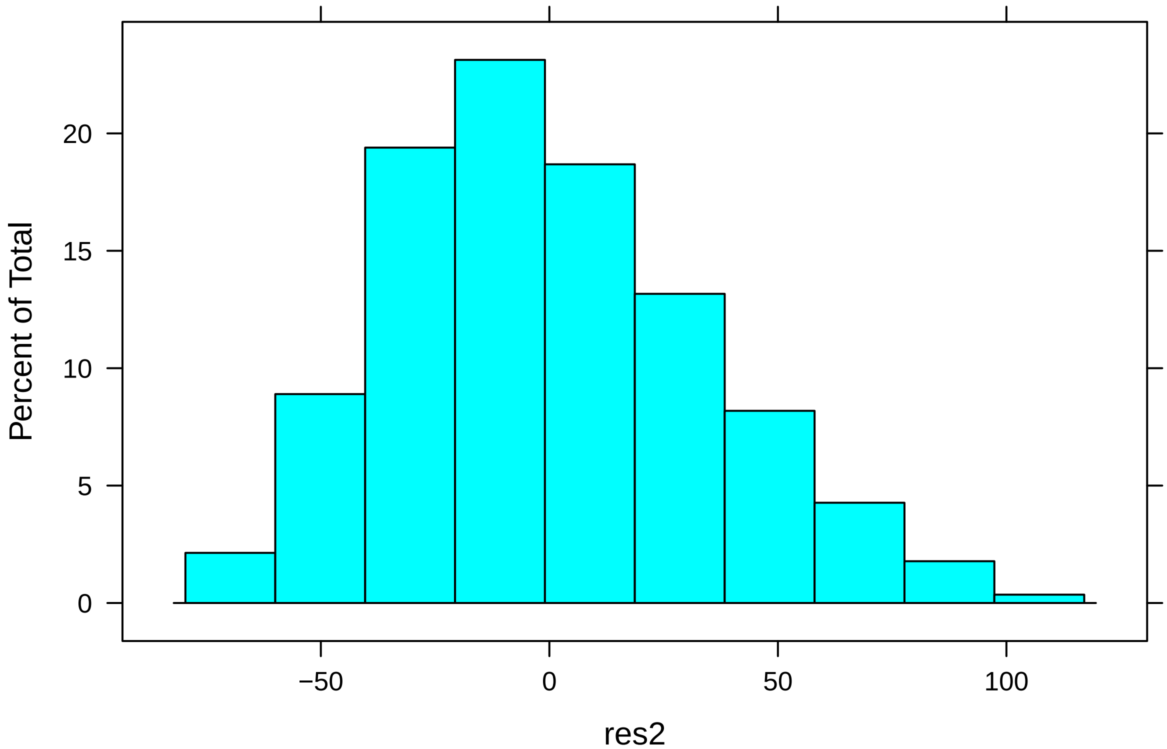
For our MLR1 model, most residuals remain on the plotted line until lower end and higher end of the plot. In this case, the data points only follow the straight line from -1 to 1 of the theoretical Quantities which might indicate that our data is not normally distributed.

However, the Histogram of our residuals indicated that it follows the bell curve quite well except for the fact that the most centered area is too high. Overall, MLR1 roughly follows the bell-shaped symmetry and could be assumed that the data is normally distributed

```
#create Q-Q plot for residuals
qqnorm(res2)

#add a straight diagonal line to the plot
qqline(res2)
```




```
#Create density plot of residuals
plot(histogram(res2))
```



Taking a look into MLR2, the Q-Q plot illustrates that the data points are straying away much further from the plotted line, indicating that the data points are not normally distributed. This is further cemented by the histogram which show that the residuals are skewed to the left, indicating that it is not following a bell shape. This means that MLR2 has completely violated the assumption of normality.

With all tests of our assumptions completed, it can be concluded that MLR1 is better than MLR2 in most cases, being more accurate for the sample and generalisable to the population. However, the possibility of heteroscedasticity should be verified but for the sake of this project, we will proceed with MLR1 for Salary predictions.

7.3 Predictions

```
#Making predictions
reg_pred <- predict(MLR1, newdata = reg_test1)

reg_ci <- predict(MLR1, newdata = reg_test1, interval = "confidence", level=.95)
head(reg_ci)
```

```
##         fit       lwr       upr
## 1  96.97510  82.55653 111.39368
## 2  76.86908  62.37492  91.36325
## 3 118.69683  87.04878 150.34488
## 4 109.85716  93.52981 126.18450
## 5  99.74349  63.54971 135.93727
## 6 117.11519 109.14522 125.08515
```

After we predicted the Salary on our test set, we also generated the confidence intervals for each predicted Salary. As seen from the head(reg_ci), many of the values have very wide upper and lower bounds for the interval, this means that the range of the predicted Salary being incorrect is very high.

The evaluation metrics are generated below. Just for comparison, we also generated predictions from MLR2 to illustrate the changes in performance from 2 predictors to 3.

```
Mdl1 <- postResample(pred = reg_pred, obs = reg_test1$avg_salary)
Mdl1
```

```
##       RMSE   Rsquared        MAE
## 30.6054975  0.2037674 24.2671234
```

```
Mdl2
```

```
##       RMSE   Rsquared        MAE
## 32.0012512  0.1167421 25.5095447
```

The first results are MLR1 and the second is MLR2. Based on the output above, MLR1 has a lower RMSE than MLR2 which means that the average difference between values predicted by MLR1 is smaller than MLR2. Secondly, Rsquared values are much higher with MLR1, implying that the regression model can explain 20% of the variability of the Salary. Lastly, MLR1's mean absolute error is slightly smaller meaning that MLR1 is slightly more accurate in generating predictions.

7.4 Overall Results

In conclusion, the overall regression model performance was not as expected. With a MAE value of 24.2, this means that on average, the actual value will be in the range of 24.2 plus/minus outside the fitted line. Furthermore, with the Rsquared value being 0.2, only 20% of the variability could be explained by the model. This can mean that not enough variables have been added to the model, causing our regression model to be under-fit. This may also be due to the lack of other continuous variables in the dataset, leaving only categorical variables to be used. If the quality of the data set was much better, age not having 380 missing values and consisting more of continuous variables, the model performance would be much better. As the saying goes, "garbage in, garbage out". This could be due to the fact that our dependent variable (avg_salary) is very influenced by scale and no common pattern in regards to the tech industry. For example, many tech jobs are present in certain states such as New York and Los Angeles. Variables such as this heavily impact the model performance.

7.5 Generating a Second Iteration

Changes should be made for improvements: we will be testing whether increasing the number of variables from 3 to 5 would be a significant improvement to the model. Furthermore, we can try to resolve the potentiality of heteroscedasticity if it is present but due to the fact that majority of variables are factor or categorical, it will be difficult to attempt to log transformation our dataset.

Building a new model

```r
# Data set with 5 variables

Salary_Final_MLR3 <- Salary_Final[, c("Rating", "avg_salary","python_yn","job_state",
                                      "Revenue","Sector")]

# Splitting dataset
reg_training3 <- subset(Salary_Final_MLR3, sampleset == TRUE)
reg_test3 <- subset(Salary_Final_MLR3, sampleset == FALSE)

# Model
MLR3 <- lm(avg_salary ~ Rating + python_yn + job_state + Revenue + Sector, data = reg_training3)

summary(MLR3)$r.squared
```

```
## [1] 0.40394
```

```r
summary(MLR3)$sigma
```

```
## [1] 30.15354
```

As shown by the summary, MLR3's residual standard error is 30.15 while MLR2 is 31.63. This means that despite adding new variables, the error as not significantly improved. This would mean that we need to attempt a different way to improve our model performance. As previously mentioned, we identified the potentiality of heteroscedasticity in our model, let's try to diagnose this issue with the Breusch-Pagan test.

```r
library(lmtest)

bptest(MLR3)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  MLR3
## BP = 136.19, df = 70, p-value = 3.728e-06
```

Using the common level of significance as 0.05, with our resulting p-value as 0.000003728, the null hypothesis of homoscedasticity is rejected. This means that heteroscedasticity is present.

Weighted Least Squares Regression

As heteroscedasticity is present, performing weighted least squares by defining weights would lower the variance of our observations and hopefully improve the performance.

```r
# defining weights to use
wt <- 1 / lm(abs(MLR3$residuals) ~ MLR3$fitted.values)$fitted.values^2

#creating WLS model
wls_salary <- lm(avg_salary ~ Rating + python_yn + job_state + Revenue + Sector,
                 data = reg_training3, weights=wt)

#Rsquared value
summary(wls_salary)$r.squared
```

```
## [1] 0.5313016
```

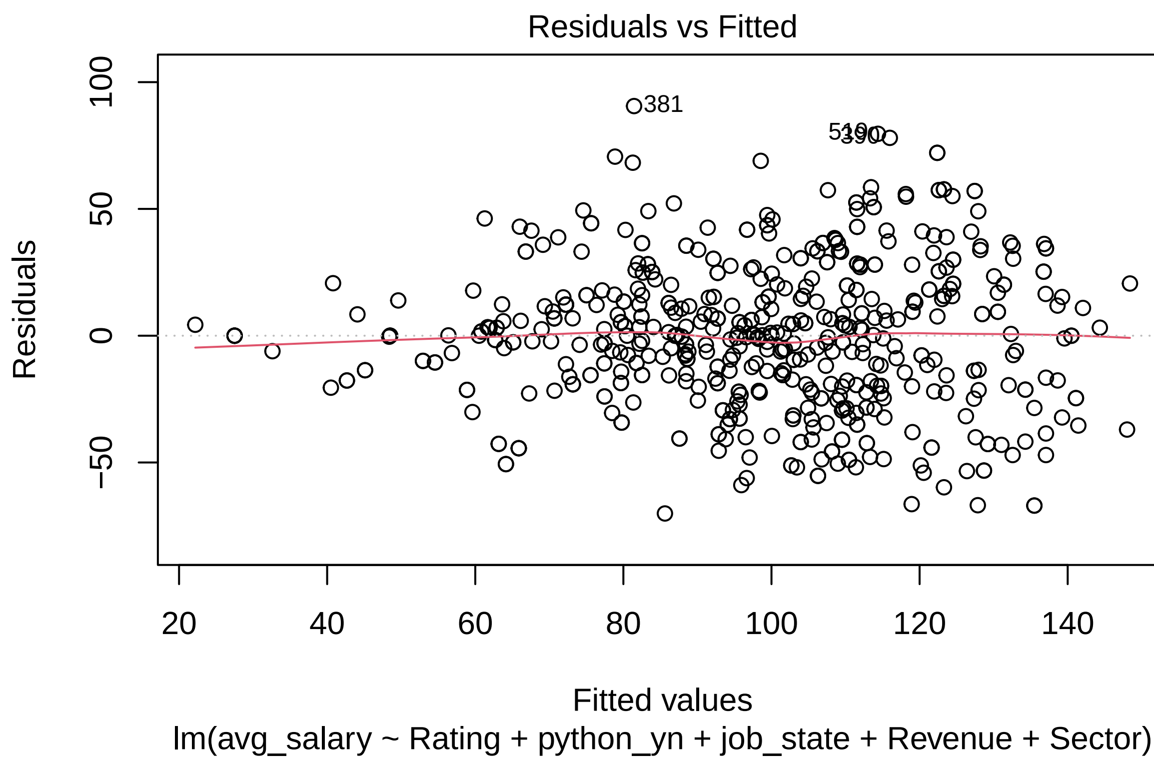
```
summary(wls_salary)$sigma
```

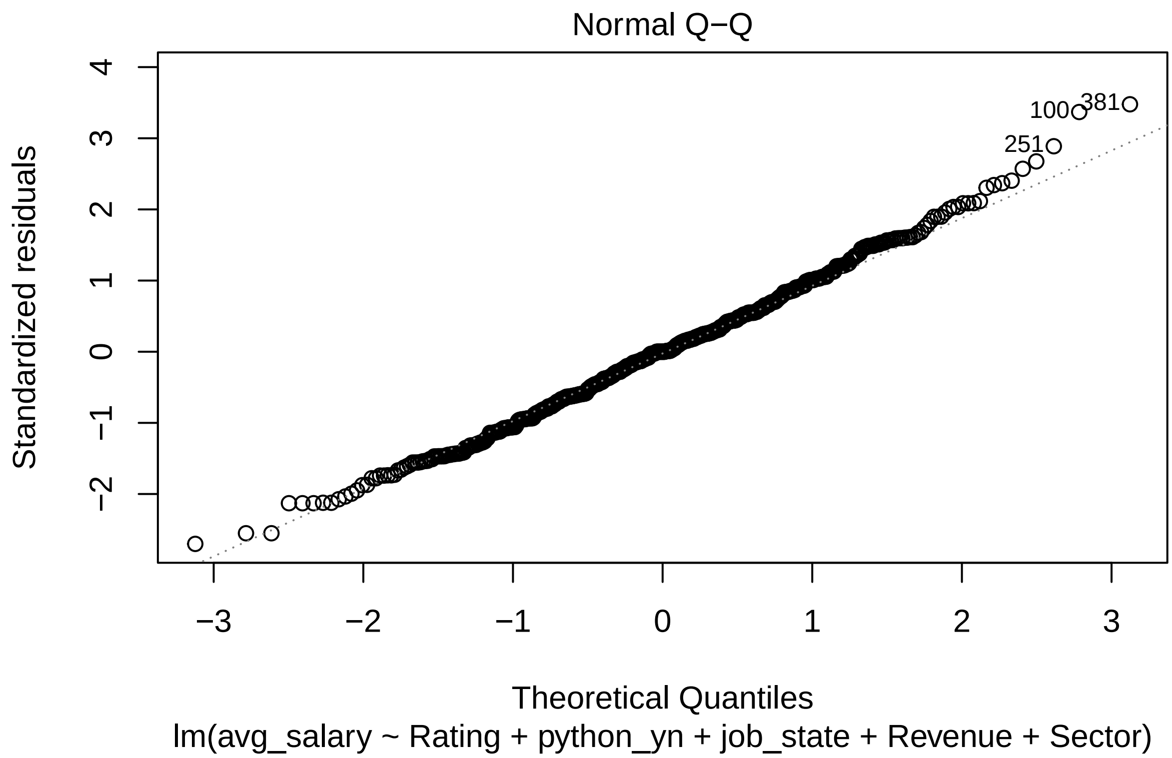
```
## [1] 1.351442
```

The result of our WLS model has significantly improved in regards to the Residual standard error (1.351 from 30.15). Furthermore, the Rsquared value has also improved significantly from 0.4039 to 0.5313, meaning that an increase of 13% of the variance in salary could be explained by our WLS regression model.

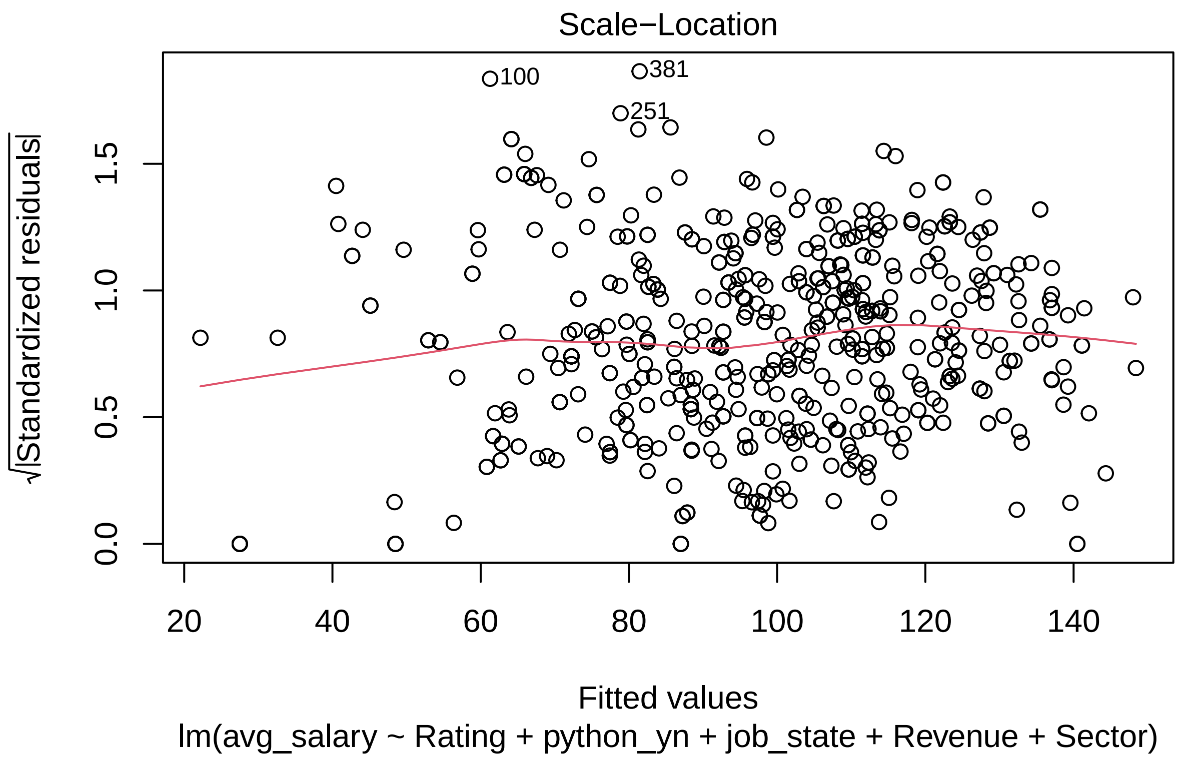
Checking for assumptions

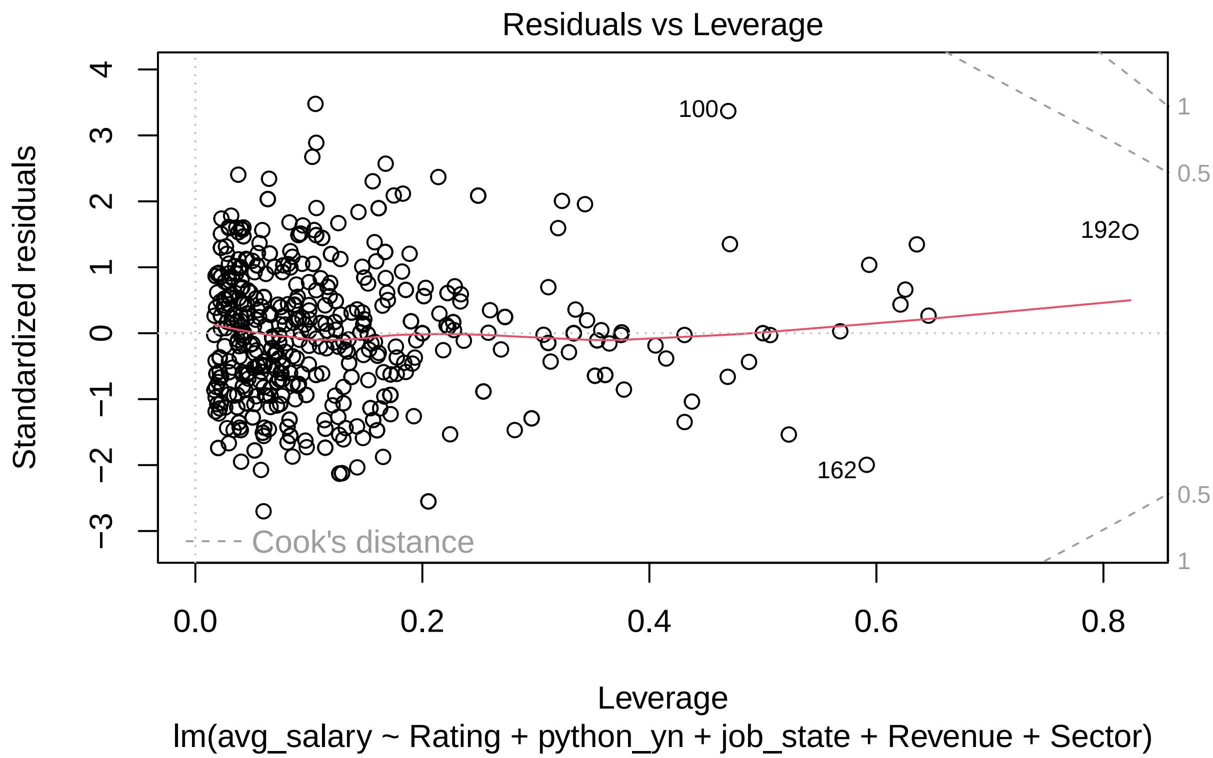
```
# Test for Heteroscedasticity
plot(wls_salary)
```





Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(avg_salary ~ Rating + python_yn + job_state + Revenue + Sector)
100
381
251


Scale−Location
100
381
251
√|Standardized residuals|
0.0
0.5
1.0
1.5
20
40
60
80
100
120
140
Fitted values
lm(avg_salary ~ Rating + python_yn + job_state + Revenue + Sector)




```
bptest(wls_salary)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  wls_salary
## BP = 0.53461, df = 70, p-value = 1
```

```
hist(wls_salary$residuals)
```

As our p value is well above 0.05, we can say that our data no longer violates the assumption of homoscedasticity. Furthermore, the assumptions of linearity and normality have also been met with our plots and histogram.

```
# Durbin-Watson test
dwt(wls_salary)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.04539854      2.090322   0.296
##  Alternative hypothesis: rho != 0
```

```
# VIF, Tolerance and Mean VIF
vif(wls_salary)
```

```
##                   GVIF Df GVIF^(1/(2*Df))
## Rating        1.559193  1        1.248677
## python_yn     1.584831  1        1.258901
## job_state 3780.224493 36        1.121212
## Revenue     38.069453 10        1.199579
## Sector    1794.956977 22        1.185648
```

```
1/vif(wls_salary)
```

```
##                     GVIF         Df GVIF^(1/(2*Df))
## Rating      0.6413573676 1.00000000       0.8008479
```

```
## python_yn 0.6309821810 1.00000000       0.7943439
## job_state 0.0002645346 0.02777778       0.8918920
## Revenue   0.0262677797 0.10000000       0.8336258
## Sector    0.0005571164 0.04545455       0.8434206
```

```
mean(c(1.248677,1.258901,1.121212,1.199579,1.185648))
```

```
## [1] 1.202803
```

Our Durbin-Watson test shows that the DW statistic is within the acceptable range, being very close to 2.0, furthermore the p-value is well above 0.05 which confirms our conclusion. This meets the assumption of independence. Regarding our VIF and tolerance, we will be referring to the GVIF^(1/(2*Df)) values as we have many nominal values. The VIF scores show that all are well below 10 while the tolerance is well above 0.2. Furthermore, the mean VIF is quite close to 1. Based on these values, we can safely conclude that there is no collinearity within our data.

Overall, all assumptions have been met by our model meaning that it can be accurate and be used to generalise the population. We will proceed with the predictions to see evaluate the model performance.

WLS Model Predictions

```
#Making predictions
reg_pred3 <- predict(wls_salary, newdata = reg_test3)

reg_ci3 <- predict(wls_salary, newdata = reg_test3, interval = "confidence", level=.95)
head(reg_ci)
```

```
##         fit       lwr       upr
## 1  96.97510  82.55653 111.39368
## 2  76.86908  62.37492  91.36325
## 3 118.69683  87.04878 150.34488
## 4 109.85716  93.52981 126.18450
## 5  99.74349  63.54971 135.93727
## 6 117.11519 109.14522 125.08515
```

Similar to the previous model, the Salary is highly variable and in most cases have very high ranges between the intervals. Let's dig deeper to evaluate the model performance.

```
Mdl3 <- postResample(pred = reg_pred3, obs = reg_test3$avg_salary)
## Comparing our 2 models
# First model
Mdl1
```

```
##       RMSE   Rsquared        MAE
## 30.6054975  0.2037674 24.2671234
```

```
# Second iteration
Mdl3
```

```
##       RMSE   Rsquared        MAE
## 26.5738273  0.3999875 20.9530351
```

Based on these measures, our WLS model has improved on all metrics. For example, the Root Mean Squared error has improved with a decrease of 4. Similarly, the Mean Absolute Error has also decreased by 4. Both of these metrics imply that our model has improved in becoming more accurate in predicting Salary. Lastly, Rsquared value has significantly increased from 0.20 to 0.39 which is an increase of 19% in being able to explain the variability of Salary in our model.