AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH

(AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ENGINEERING

INTRODUCTION TO DATA SCIENCE

Spring 2024-2025

Section: E

Report name:

Project Report on Data Analysis

<u>Supervised By:</u>

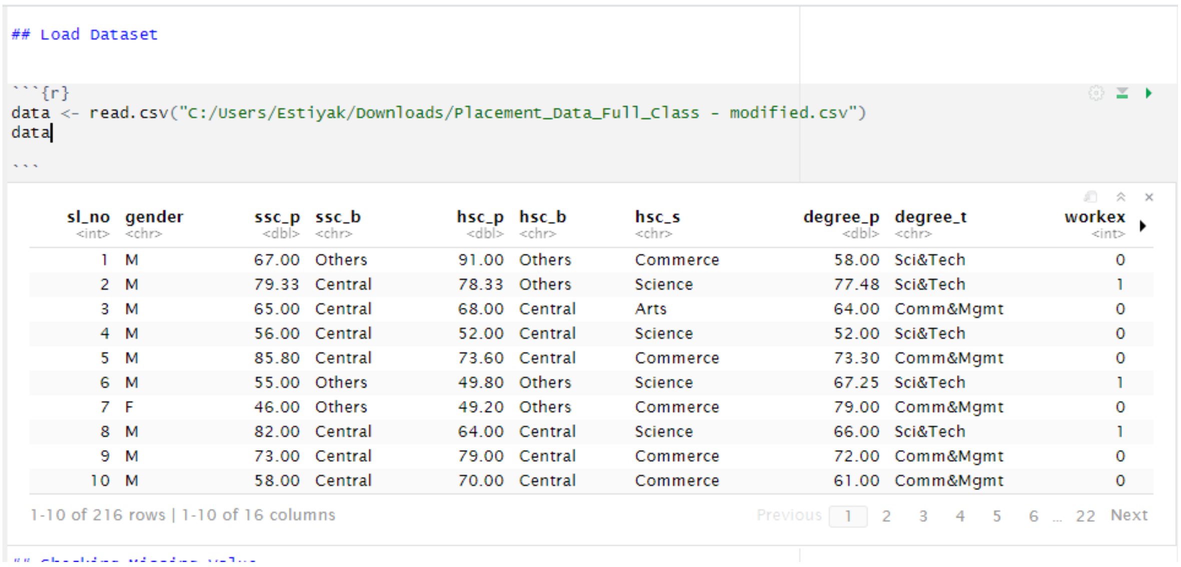
Abdus Salam

<u>Submitted By</u>:

NAME	ID
1. Estiyak Rubaiat	22-47210-1
2. Jahir Uddin Mohammad Babar	22-47213-1
3. MD. Rifat-ul-Khan	22-46016-1
4. Sabbir Ahmmed Shuvo	22-47181-1

Date of Submission: April 26, 2025

Data Description



~~~
## Load Dataset

```{r}
data <- read.csv("C:/Users/Estiyak/Downloads/Placement_Data_Full_Class - modified.csv")
data
```
~~~

sl_no <int>	gender <chr>	ssc_p <dbl>	ssc_b <chr>	hsc_p <dbl>	hsc_b <chr>	hsc_s <chr>	degree_p <dbl>	degree_t <chr>	workex <int>
1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	0
2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	1
3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	0
4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	0
5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	0
6	M	55.00	Others	49.80	Others	Science	67.25	Sci&Tech	1
7	F	46.00	Others	49.20	Others	Commerce	79.00	Comm&Mgmt	0
8	M	82.00	Central	64.00	Central	Science	66.00	Sci&Tech	1
9	M	73.00	Central	79.00	Central	Commerce	72.00	Comm&Mgmt	0
10	M	58.00	Central	70.00	Central	Commerce	61.00	Comm&Mgmt	0

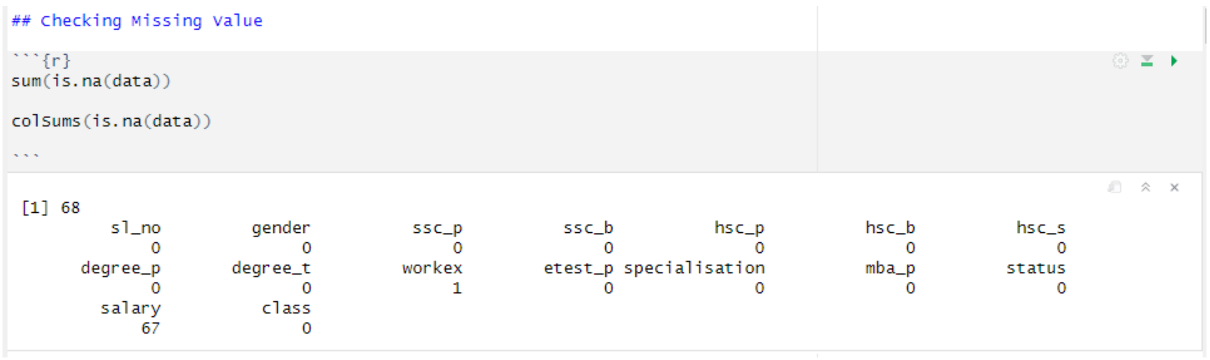
1-10 of 216 rows | 1-10 of 16 columns

The dataset contains information about the academic and placement records of 216 students. It includes 16 attributes such as gender, scores from secondary (SSC) and higher secondary (HSC) education, specialization streams, degree types, work experience, employability test percentages, MBA scores, and placement status. Additionally, it records the offered salary for placed students and a "class" label, which likely categorizes the students based on their placement outcomes. This dataset is typically useful for analyzing the factors that influence student placement success and salary offers, and it can be used for predictive modeling in education and career counseling domains.

This image features a snippet of R code that loads a CSV file named "Placement_Data_Full_class - modified.csv" using the read.csv() function. The dataset, which is displayed below the code, includes student details such as gender, educational background (SSC, HSC, Degree), and work experience. There are a total of 16 columns, including sl_no, gender, ssc_p, ssc_b, hsc_p, hsc_b, hsc_s, degree_p, degree_t, and workex. The preview reveals the first 10 rows out of a total of 216 records. This data appears to be prepared for analysis related to placements or employment.

Features:

- **readr** is used to quickly and cleanly read CSV files.
- **dplyr** is used for data manipulation like filtering, selecting, and summarizing data.
- **caTools** is used to split the dataset into training and testing sets.
- **modeest** is used to accurately calculate the mode of numeric and categorical variables.
- **ggplot2** is used to create professional graphs and visualizations.
- **tinytex** is used to install LaTeX support for generating PDF reports from R Markdown.

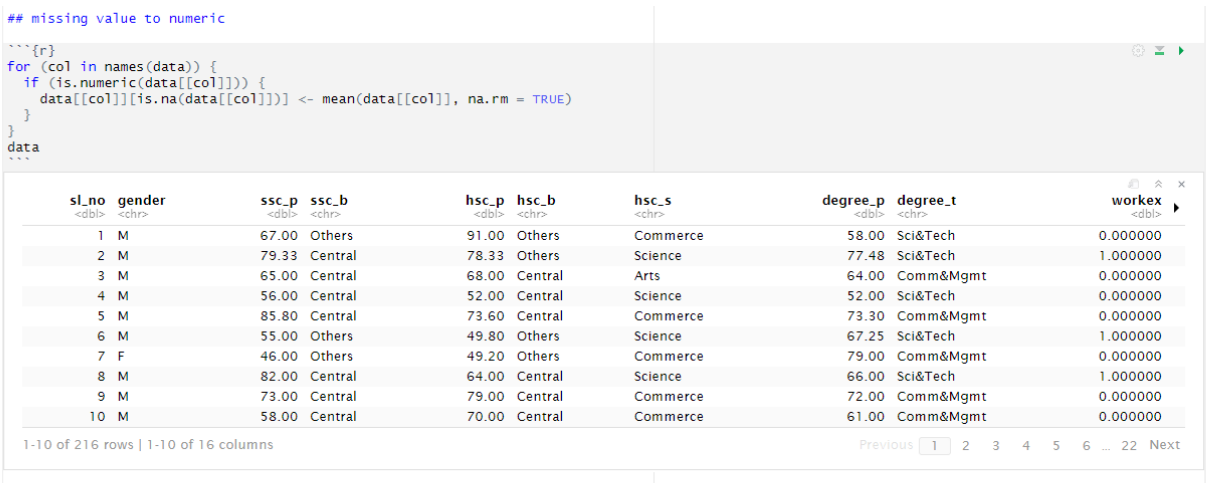


~~~~
## Checking Missing Value

```{r}
sum(is.na(data))

colSums(is.na(data))

```
~~~~

```
[1] 68
        sl_no        gender         ssc_p         ssc_b         hsc_p         hsc_b         hsc_s 
            0             0             0             0             0             0             0 
     degree_p      degree_t        workex       etest_p specialisation         mba_p        status 
            0             0             1             0             0             0             0 
       salary         class 
           67             0 
```

This image displays R code that is used to identify missing values in the dataset named data. The function sum(is.na(data)) indicates that there are a total of 68 missing entries. The function colSums(is.na(data)) provides a detailed count of the missing values for each column. It shows that the salary column contains 67 missing values, while the workex column has 1 missing value. All other columns (such as sl_no, gender, ssc_p, etc.) are free from any missing data.



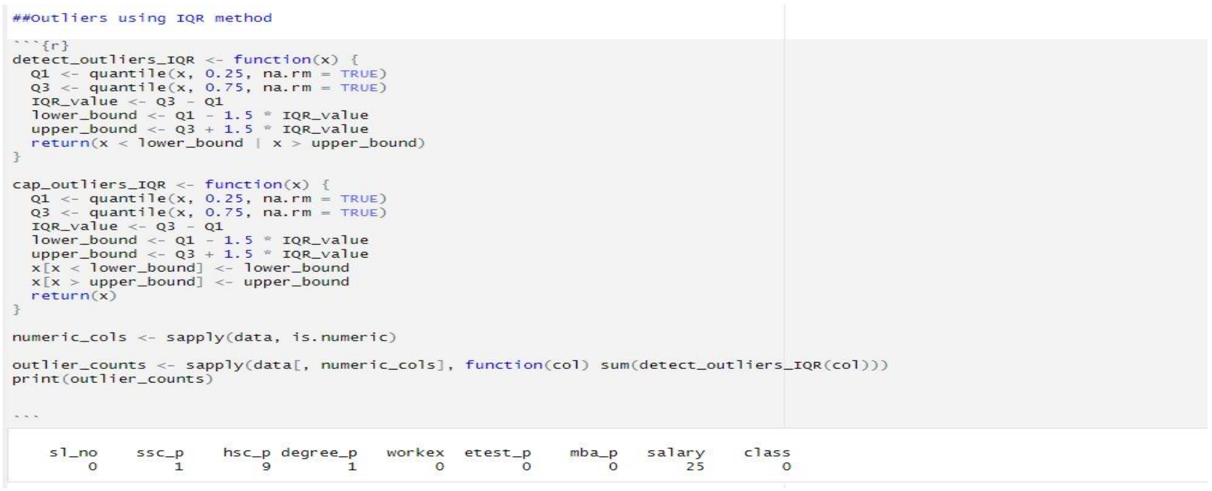
~~~~
## missing value to numeric

```{r}
for (col in names(data)) {
  if (is.numeric(data[[col]])) {
    data[[col]][is.na(data[[col]])] <- mean(data[[col]], na.rm = TRUE)
  }
}
data
```
~~~~

sl_no <dbl>	gender <chr>	ssc_p <dbl>	ssc_b <chr>	hsc_p <dbl>	hsc_b <chr>	hsc_s <chr>	degree_p <dbl>	degree_t <chr>	workex <dbl>
1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	0.000000
2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	1.000000
3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	0.000000
4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	0.000000
5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	0.000000
6	M	55.00	Others	49.80	Others	Science	67.25	Sci&Tech	1.000000
7	F	46.00	Others	49.20	Others	Commerce	79.00	Comm&Mgmt	0.000000
8	M	82.00	Central	64.00	Central	Science	66.00	Sci&Tech	1.000000
9	M	73.00	Central	79.00	Central	Commerce	72.00	Comm&Mgmt	0.000000
10	M	58.00	Central	70.00	Central	Commerce	61.00	Comm&Mgmt	0.000000

1-10 of 216 rows | 1-10 of 16 columns

This picture demonstrates how to use R code to look for missing values in the dataset. There are 68 missing values in all, according to the sum(is.na(data)) function. The colSums(is.na(data)) function breaks down missing values column-wise. It reveals that the salary column has 67 missing values, and workex has 1 missing value. All other columns (sl_no, gender, ssc_p, etc.) have no missing data.

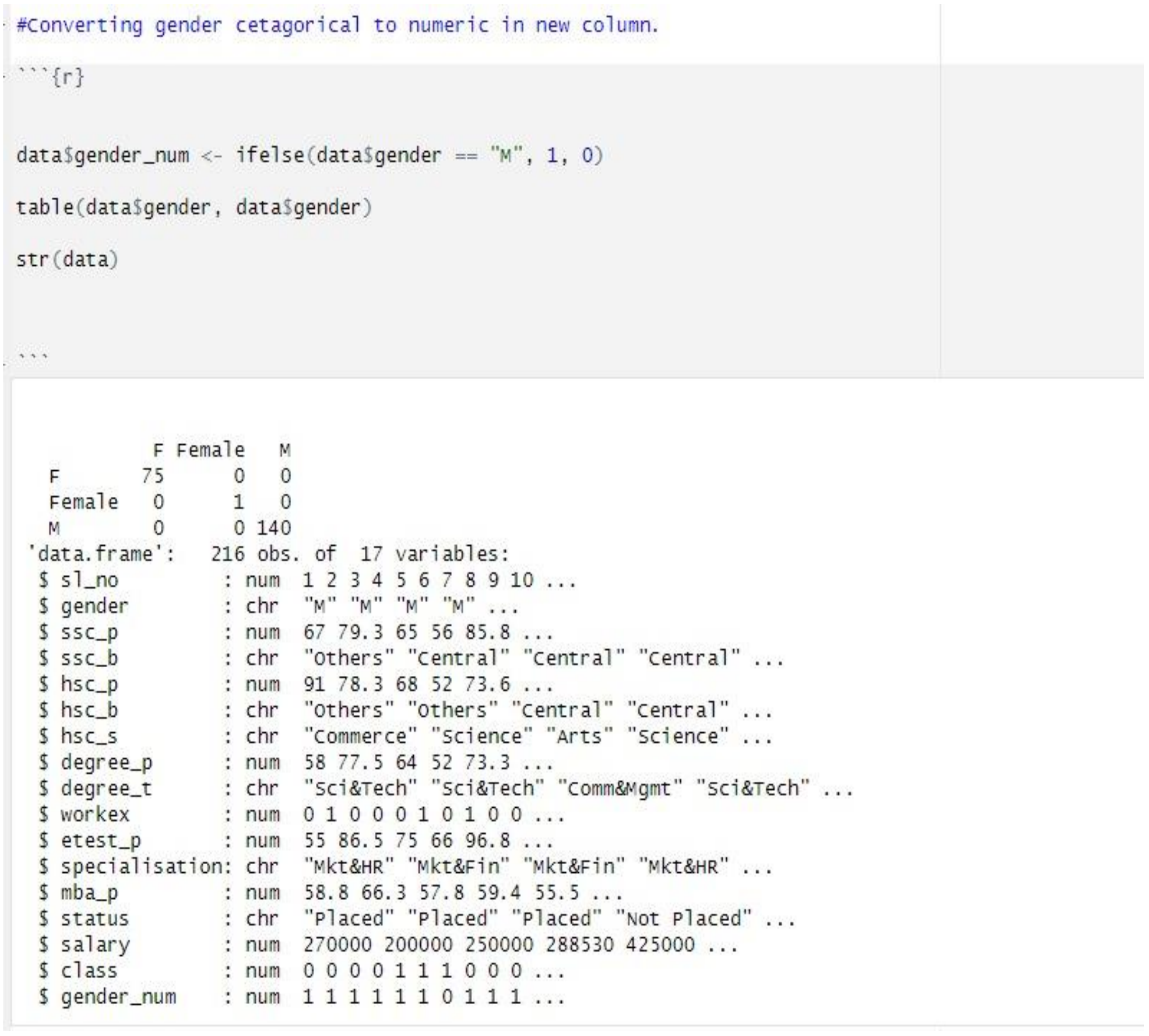


~~~~
##Outliers using IQR method

```{r}
detect_outliers_IQR <- function(x) {
  Q1 <- quantile(x, 0.25, na.rm = TRUE)
  Q3 <- quantile(x, 0.75, na.rm = TRUE)
  IQR_value <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR_value
  upper_bound <- Q3 + 1.5 * IQR_value
  return(x < lower_bound | x > upper_bound)
}

cap_outliers_IQR <- function(x) {
  Q1 <- quantile(x, 0.25, na.rm = TRUE)
  Q3 <- quantile(x, 0.75, na.rm = TRUE)
  IQR_value <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR_value
  upper_bound <- Q3 + 1.5 * IQR_value
  x[x < lower_bound] <- lower_bound
  x[x > upper_bound] <- upper_bound
  return(x)
}

numeric_cols <- sapply(data, is.numeric)

outlier_counts <- sapply(data[, numeric_cols], function(col) sum(detect_outliers_IQR(col)))
print(outlier_counts)


```
~~~~

```
    sl_no     ssc_p     hsc_p  degree_p    workex   etest_p     mba_p    salary     class 
        0         1         9         1         0         0         0        25         0 
```

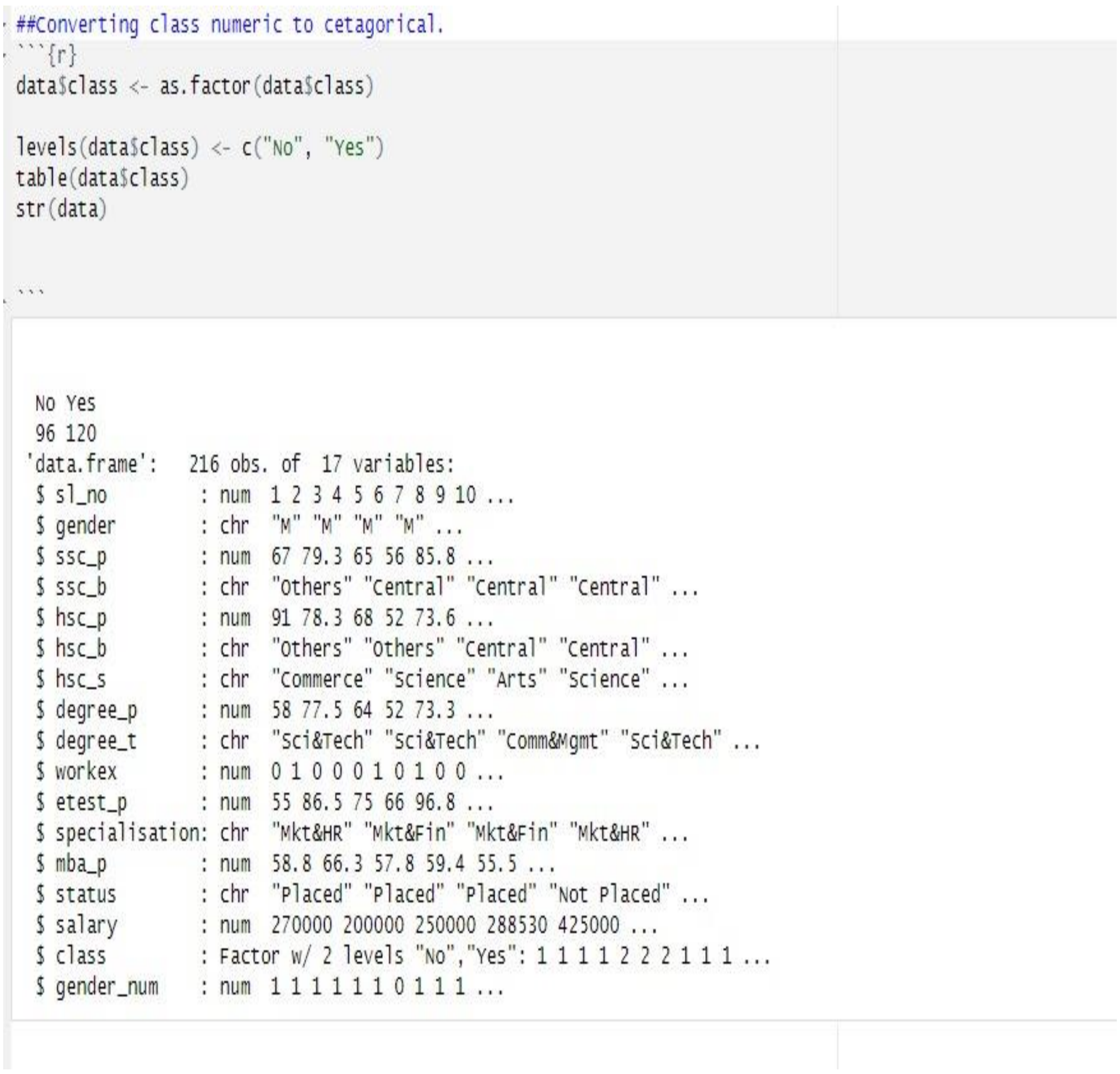
This picture demonstrates how R code uses the IQR (Interquartile Range) approach to identify and cap outliers. To identify and modify outliers, two custom functions are defined: detect_outliers_IQR and cap_outliers_IQR.To identify numerical columns, use sapply() with is.numeric.Each numeric column's outlier count is computed; for instance, salary has 25 outliers and degree_p has 9.No outliers have been found in columns like sl_no, workex, and class.



~~~
#Converting gender cetagorical to numeric in new column.

```{r}

data$gender_num <- ifelse(data$gender == "M", 1, 0)

table(data$gender, data$gender)

str(data)


```
~~~

```
          F Female   M
  F      75      0   0
  Female  0      1   0
  M       0      0 140
'data.frame':	216 obs. of  17 variables:
 $ sl_no         : num  1 2 3 4 5 6 7 8 9 10 ...
 $ gender        : chr  "M" "M" "M" "M" ...
 $ ssc_p         : num  67 79.3 65 56 85.8 ...
 $ ssc_b         : chr  "Others" "Central" "Central" "Central" ...
 $ hsc_p         : num  91 78.3 68 52 73.6 ...
 $ hsc_b         : chr  "Others" "Others" "Central" "Central" ...
 $ hsc_s         : chr  "Commerce" "Science" "Arts" "Science" ...
 $ degree_p      : num  58 77.5 64 52 73.3 ...
 $ degree_t      : chr  "Sci&Tech" "Sci&Tech" "Comm&Mgmt" "Sci&Tech" ...
 $ workex        : num  0 1 0 0 0 1 0 1 0 0 ...
 $ etest_p       : num  55 86.5 75 66 96.8 ...
 $ specialisation: chr  "Mkt&HR" "Mkt&Fin" "Mkt&Fin" "Mkt&HR" ...
 $ mba_p         : num  58.8 66.3 57.8 59.4 55.5 ...
 $ status        : chr  "Placed" "Placed" "Placed" "Not Placed" ...
 $ salary        : num  270000 200000 250000 288530 425000 ...
 $ class         : num  0 0 0 0 1 1 1 0 0 0 ...
 $ gender_num    : num  1 1 1 1 1 1 0 1 1 1 ...
```

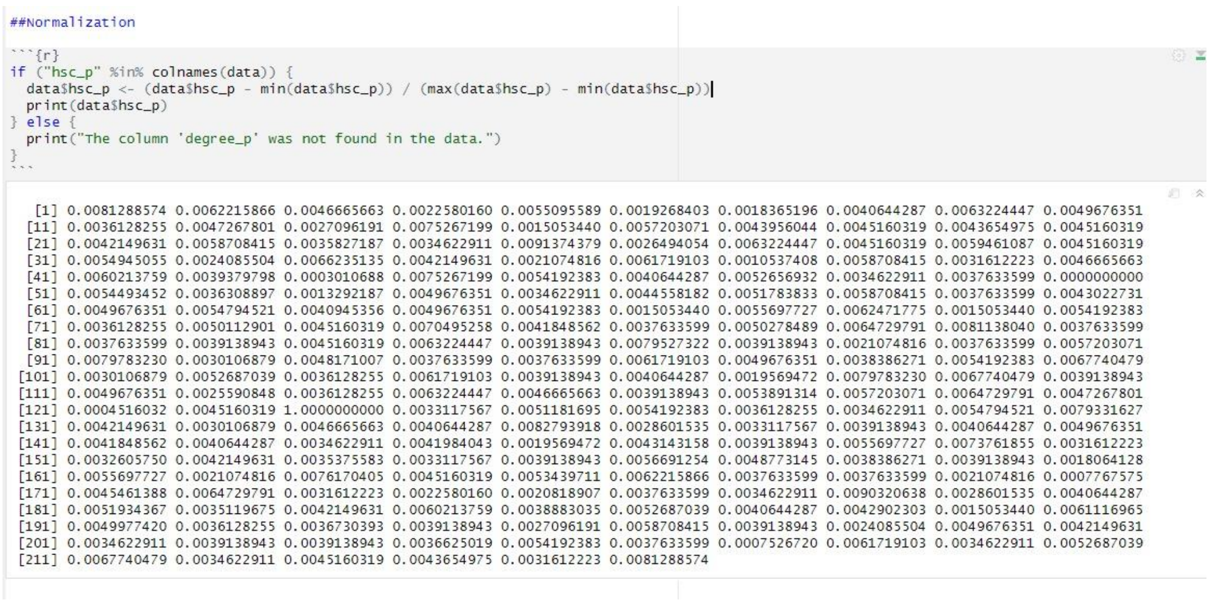
This image shows R code where the gender column, originally categorical ("M" and "F"), is converted into a numeric column gender_num.The ifelse() function assigns 1 for "M" and 0 for "F", effectively encoding gender.The table() function displays the frequency distribution between the old and new gender values.The str(data) command reveals the structure of the data frame, showing it now has 17 variables and 216 observations.New column gender_num is confirmed as numeric (num) along with other variables like sl_no, ssc_p, and salary.The original gender column remains unchanged as character data (chr).This transformation is useful for machine learning models that require numeric inputs.

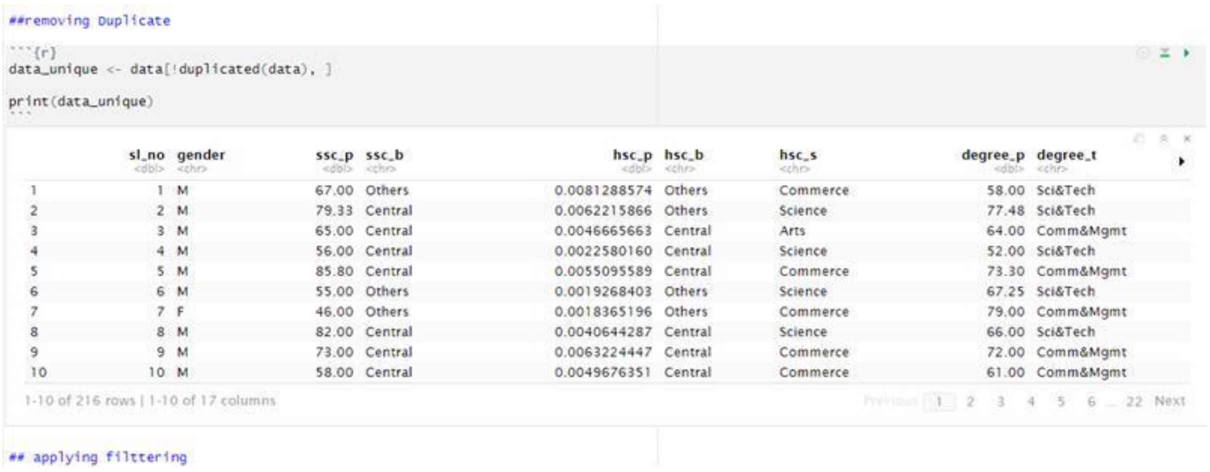


~~~
##Converting class numeric to cetagorical.
```{r}
data$class <- as.factor(data$class)

levels(data$class) <- c("No", "Yes")
table(data$class)
str(data)


```
~~~

```
 No Yes
 96 120
'data.frame':	216 obs. of  17 variables:
 $ sl_no         : num  1 2 3 4 5 6 7 8 9 10 ...
 $ gender        : chr  "M" "M" "M" "M" ...
 $ ssc_p         : num  67 79.3 65 56 85.8 ...
 $ ssc_b         : chr  "Others" "Central" "Central" "Central" ...
 $ hsc_p         : num  91 78.3 68 52 73.6 ...
 $ hsc_b         : chr  "Others" "Others" "Central" "Central" ...
 $ hsc_s         : chr  "Commerce" "Science" "Arts" "Science" ...
 $ degree_p      : num  58 77.5 64 52 73.3 ...
 $ degree_t      : chr  "Sci&Tech" "Sci&Tech" "Comm&Mgmt" "Sci&Tech" ...
 $ workex        : num  0 1 0 0 0 1 0 1 0 0 ...
 $ etest_p       : num  55 86.5 75 66 96.8 ...
 $ specialisation: chr  "Mkt&HR" "Mkt&Fin" "Mkt&Fin" "Mkt&HR" ...
 $ mba_p         : num  58.8 66.3 57.8 59.4 55.5 ...
 $ status        : chr  "Placed" "Placed" "Placed" "Not Placed" ...
 $ salary        : num  270000 200000 250000 288530 425000 ...
 $ class         : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 2 2 1 1 1 ...
 $ gender_num    : num  1 1 1 1 1 1 0 1 1 1 ...
```

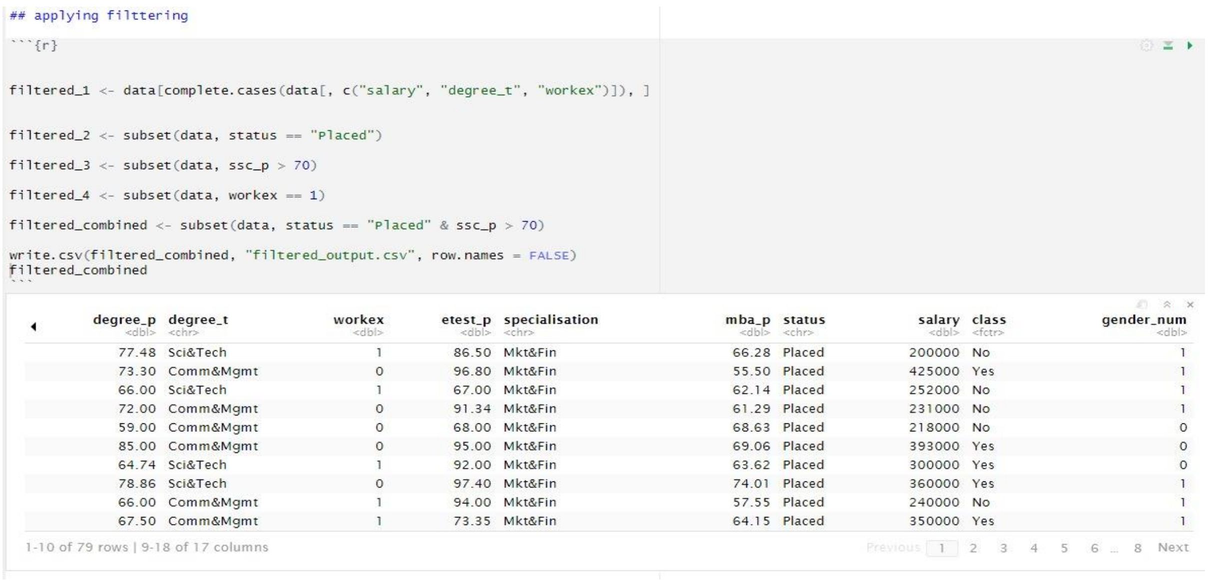
The R code that changes the class column's numeric format to categorical (factor) format is displayed in this image.The class variable is transformed into a factor with two levels: "No" and "Yes" using as.factor().The factor levels are renamed using the levels() function, where "No" most likely denotes students who were not placed and "Yes" denotes students who were placed.A frequency table shows 96 observations labeled "No" and 120 labeled "Yes".The str(data) output confirms that class is now a factor with two levels, and the dataset still contains 17 variables and 216 observations. Other variables like salary, workex, and gender_num remain numeric, while columns like status are still characters. This transformation prepares the class variable for classification modelling or further analysis.

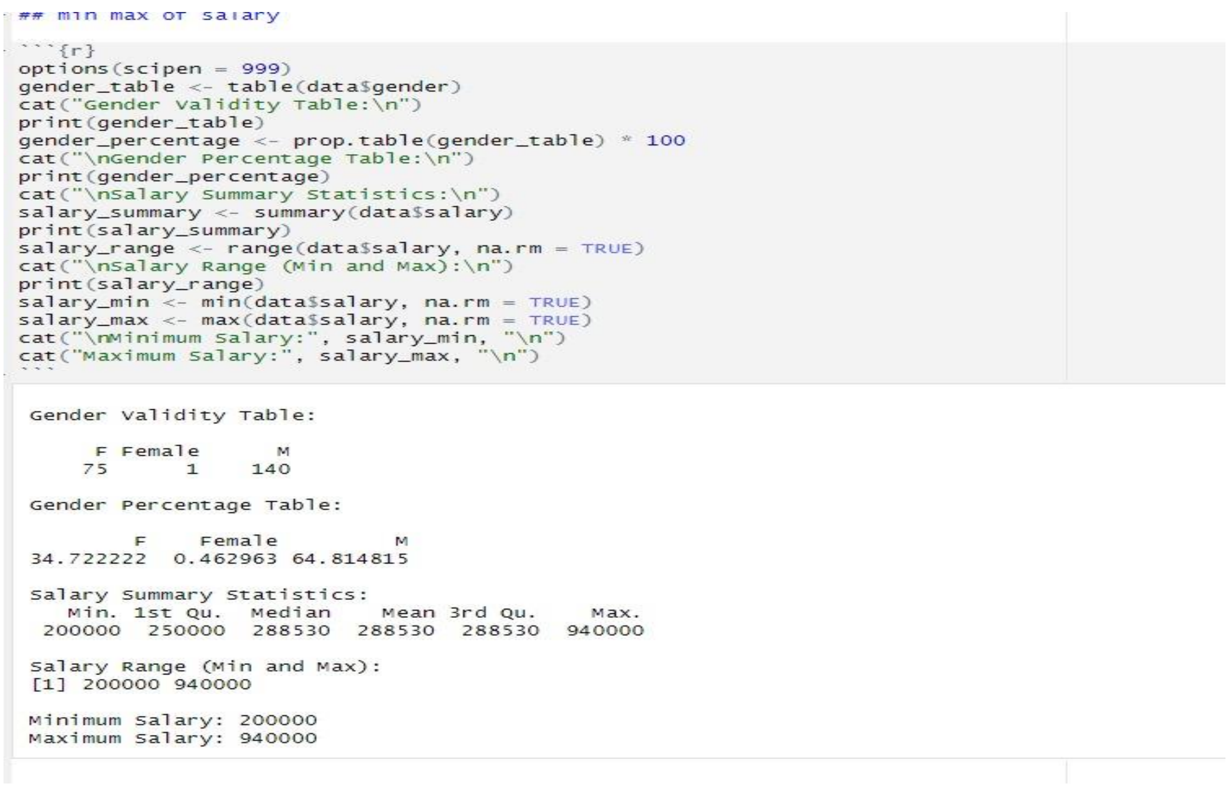


~~~
##Normalization

```{r}
if ("hsc_p" %in% colnames(data)) {
  data$hsc_p <- (data$hsc_p - min(data$hsc_p)) / (max(data$hsc_p) - min(data$hsc_p))
  print(data$hsc_p)
} else {
  print("The column 'degree_p' was not found in the data.")
}
~~~
```

```
  [1] 0.0081288574 0.0062215866 0.0046665663 0.0022580160 0.0055095589 0.0019268403 0.0018365196 0.0040644287 0.0063224447 0.0049676351
 [11] 0.0036128255 0.0047267801 0.0027096191 0.0075267199 0.0015053440 0.0057203071 0.0043956044 0.0045160319 0.0043654975 0.0045160319
 [21] 0.0042149631 0.0058708415 0.0035827187 0.0034622911 0.0091374379 0.0026494054 0.0063224447 0.0045160319 0.0059461087 0.0045160319
 [31] 0.0054945055 0.0024085504 0.0066235135 0.0042149631 0.0021074816 0.0061719103 0.0010537408 0.0058708415 0.0031612223 0.0046665663
 [41] 0.0060213759 0.0039379798 0.0003010688 0.0075267199 0.0054192383 0.0040644287 0.0052656932 0.0034622911 0.0037633599 0.0000000000
 [51] 0.0054493452 0.0036308897 0.0013292187 0.0049676351 0.0034622911 0.0044558182 0.0051783833 0.0058708415 0.0037633599 0.0043022731
 [61] 0.0049676351 0.0054794521 0.0040945356 0.0049676351 0.0054192383 0.0015053440 0.0055697727 0.0062471775 0.0015053440 0.0054192383
 [71] 0.0036128255 0.0050112901 0.0045160319 0.0070495258 0.0041848562 0.0037633599 0.0050278489 0.0064729791 0.0081138040 0.0037633599
 [81] 0.0037633599 0.0039138943 0.0045160319 0.0063224447 0.0039138943 0.0079527322 0.0039138943 0.0021074816 0.0037633599 0.0057203071
 [91] 0.0079783230 0.0030106879 0.0048171007 0.0037633599 0.0037633599 0.0061719103 0.0049676351 0.0038386271 0.0054192383 0.0067740479
[101] 0.0030106879 0.0052687039 0.0036128255 0.0061719103 0.0039138943 0.0040644287 0.0019569472 0.0079783230 0.0067740479 0.0039138943
[111] 0.0049676351 0.0025590848 0.0036128255 0.0063224447 0.0046665663 0.0039138943 0.0053891314 0.0057203071 0.0064729791 0.0047267801
[121] 0.0004516032 0.0045160319 1.0000000000 0.0033117567 0.0051181695 0.0054192383 0.0036128255 0.0034622911 0.0054794521 0.0079331627
[131] 0.0042149631 0.0030106879 0.0046665663 0.0040644287 0.0082793918 0.0028601535 0.0033117567 0.0039138943 0.0040644287 0.0049676351
[141] 0.0041848562 0.0040644287 0.0034622911 0.0041984043 0.0019569472 0.0043143158 0.0039138943 0.0055697727 0.0073761855 0.0031612223
[151] 0.0032605750 0.0042149631 0.0035375583 0.0033117567 0.0039138943 0.0056691254 0.0048773145 0.0038386271 0.0039138943 0.0018064128
[161] 0.0055697727 0.0021074816 0.0076170405 0.0045160319 0.0053439711 0.0062215866 0.0037633599 0.0037633599 0.0021074816 0.0007767575
[171] 0.0045461388 0.0064729791 0.0031612223 0.0022580160 0.0020818907 0.0037633599 0.0034622911 0.0090320638 0.0028601535 0.0040644287
[181] 0.0051934367 0.0035119675 0.0042149631 0.0060213759 0.0038883035 0.0052687039 0.0040644287 0.0042902303 0.0015053440 0.0061116965
[191] 0.0049977420 0.0036128255 0.0036730393 0.0039138943 0.0027096191 0.0058708415 0.0039138943 0.0024085504 0.0049676351 0.0042149631
[201] 0.0034622911 0.0039138943 0.0039138943 0.0036625019 0.0054192383 0.0037633599 0.0007526720 0.0061719103 0.0034622911 0.0052687039
[211] 0.0067740479 0.0034622911 0.0045160319 0.0043654975 0.0031612223 0.0081288574
```

This image shows the normalization of the hsc_p column in an R dataset. It checks if hsc_p exists in the dataset using an if condition, and if found, scales the values between 0 and 1 using Min-Max normalization. The normalization formula is applied:

$(x−\text{min})/(\text{max}−\text{min})$

(x−min)/(max−min).

The normalized values of hsc_p are then printed, showing a list of small decimal numbers. If hsc_p were not found, an error message would have been printed instead. The printed output confirms that normalization was successful across all rows. This preprocessing step helps in preparing the data for machine learning models where scaled features improve performance.



```
##removing Duplicate

```{r}
data_unique <- data[!duplicated(data), ]

print(data_unique)
```
```

	sl_no <dbl>	gender <chr>	ssc_p <dbl>	ssc_b <chr>	hsc_p <dbl>	hsc_b <chr>	hsc_s <chr>	degree_p <dbl>	degree_t <chr>
1	1	M	67.00	Others	0.0081288574	Others	Commerce	58.00	Sci&Tech
2	2	M	79.33	Central	0.0062215866	Others	Science	77.48	Sci&Tech
3	3	M	65.00	Central	0.0046665663	Central	Arts	64.00	Comm&Mgmt
4	4	M	56.00	Central	0.0022580160	Central	Science	52.00	Sci&Tech
5	5	M	85.80	Central	0.0055095589	Central	Commerce	73.30	Comm&Mgmt
6	6	M	55.00	Others	0.0019268403	Others	Science	67.25	Sci&Tech
7	7	F	46.00	Others	0.0018365196	Others	Commerce	79.00	Comm&Mgmt
8	8	M	82.00	Central	0.0040644287	Central	Science	66.00	Sci&Tech
9	9	M	73.00	Central	0.0063224447	Central	Commerce	72.00	Comm&Mgmt
10	10	M	58.00	Central	0.0049676351	Central	Commerce	61.00	Comm&Mgmt

1-10 of 216 rows | 1-10 of 17 columns

Previous
1
2
3
4
5
6
...
22
Next

```
## applying filttering
```

This illustration demonstrates how to use R to eliminate duplicate rows from a dataset. Only unique rows are kept by data[!duplicated(data),] after duplicate entries are found using the duplicated() function.Data_unique is used to hold the resultant dataset, which is subsequently printed to show the modified data.Fields like sl_no, gender, ssc_p, hsc_p, and degree_p are among the first 10 rows across 17 columns that are displayed in the table preview.The hsc_p column's normalization, which was previously seen, is represented here with tiny decimal
```

values.After eliminating duplicates, the bottom displays that there are now 216 rows.To further refine the data, a remark indicates that filtering operations will be conducted next.



~~~
## applying filttering
```{r}

filtered_1 <- data[complete.cases(data[, c("salary", "degree_t", "workex")]), ]

filtered_2 <- subset(data, status == "Placed")

filtered_3 <- subset(data, ssc_p > 70)

filtered_4 <- subset(data, workex == 1)

filtered_combined <- subset(data, status == "Placed" & ssc_p > 70)

write.csv(filtered_combined, "filtered_output.csv", row.names = FALSE)
filtered_combined
~~~
```

degree_p <dbl>	degree_t <chr>	workex <dbl>	etest_p <dbl>	specialisation <chr>	mba_p <dbl>	status <chr>	salary <dbl>	class <fctr>	gender_num <dbl>
77.48	Sci&Tech	1	86.50	Mkt&Fin	66.28	Placed	200000	No	1
73.30	Comm&Mgmt	0	96.80	Mkt&Fin	55.50	Placed	425000	Yes	1
66.00	Sci&Tech	1	67.00	Mkt&Fin	62.14	Placed	252000	No	1
72.00	Comm&Mgmt	0	91.34	Mkt&Fin	61.29	Placed	231000	No	1
59.00	Comm&Mgmt	0	68.00	Mkt&Fin	68.63	Placed	218000	No	0
85.00	Comm&Mgmt	0	95.00	Mkt&Fin	69.06	Placed	393000	Yes	0
64.74	Sci&Tech	1	92.00	Mkt&Fin	63.62	Placed	300000	Yes	0
78.86	Sci&Tech	0	97.40	Mkt&Fin	74.01	Placed	360000	Yes	1
66.00	Comm&Mgmt	1	94.00	Mkt&Fin	57.55	Placed	240000	No	1
67.50	Comm&Mgmt	1	73.35	Mkt&Fin	64.15	Placed	350000	Yes	1

1-10 of 79 rows | 9-18 of 17 columns   Previous 1 2 3 4 5 6 ... 8 Next

This picture shows how to use a variety of R data filtering methods. To choose rows with non-missing values, "placed" status, high ssc_p scores, or previous work experience, various filters are developed. Candidates who are "placed" and have ssc_p greater than 70 are chosen using a combination filter.A CSV file called filtered_output.csv is created from the 79-row filtered result. Important fields such as degree_p, degree_t, workex, etest_p, specialization, pay, and class are displayed in the table preview. This procedure aids in reducing the dataset so that qualified persons can be the subject of more focused investigation.



```
## min max of salary
```{r}
options(scipen = 999)
gender_table <- table(data$gender)
cat("Gender Validity Table:\n")
print(gender_table)
gender_percentage <- prop.table(gender_table) * 100
cat("\nGender Percentage Table:\n")
print(gender_percentage)
cat("\nSalary Summary Statistics:\n")
salary_summary <- summary(data$salary)
print(salary_summary)
salary_range <- range(data$salary, na.rm = TRUE)
cat("\nSalary Range (Min and Max):\n")
print(salary_range)
salary_min <- min(data$salary, na.rm = TRUE)
salary_max <- max(data$salary, na.rm = TRUE)
cat("\nMinimum Salary:", salary_min, "\n")
cat("Maximum Salary:", salary_max, "\n")
```
```

```
Gender Validity Table:

     F Female   M
    75   1   140

Gender Percentage Table:

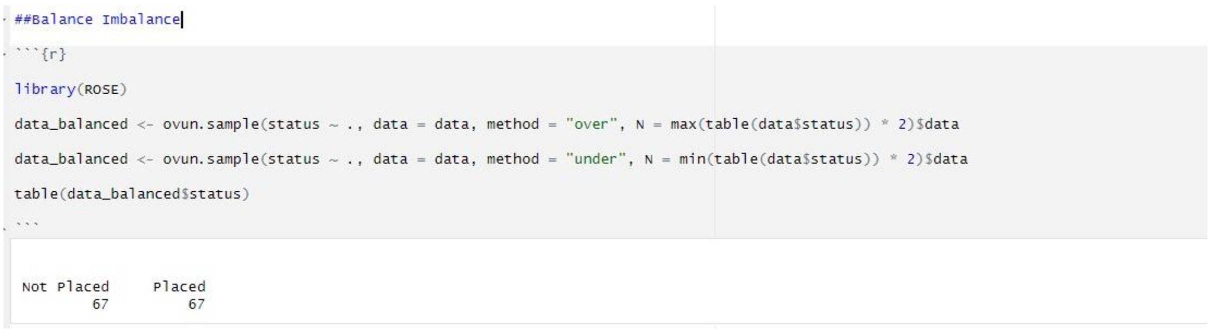
        F   Female   M
34.722222 0.462963 64.814815

Salary Summary Statistics:
   Min. 1st Qu. Median   Mean 3rd Qu.   Max.
 200000 250000 288530 288530 288530 940000

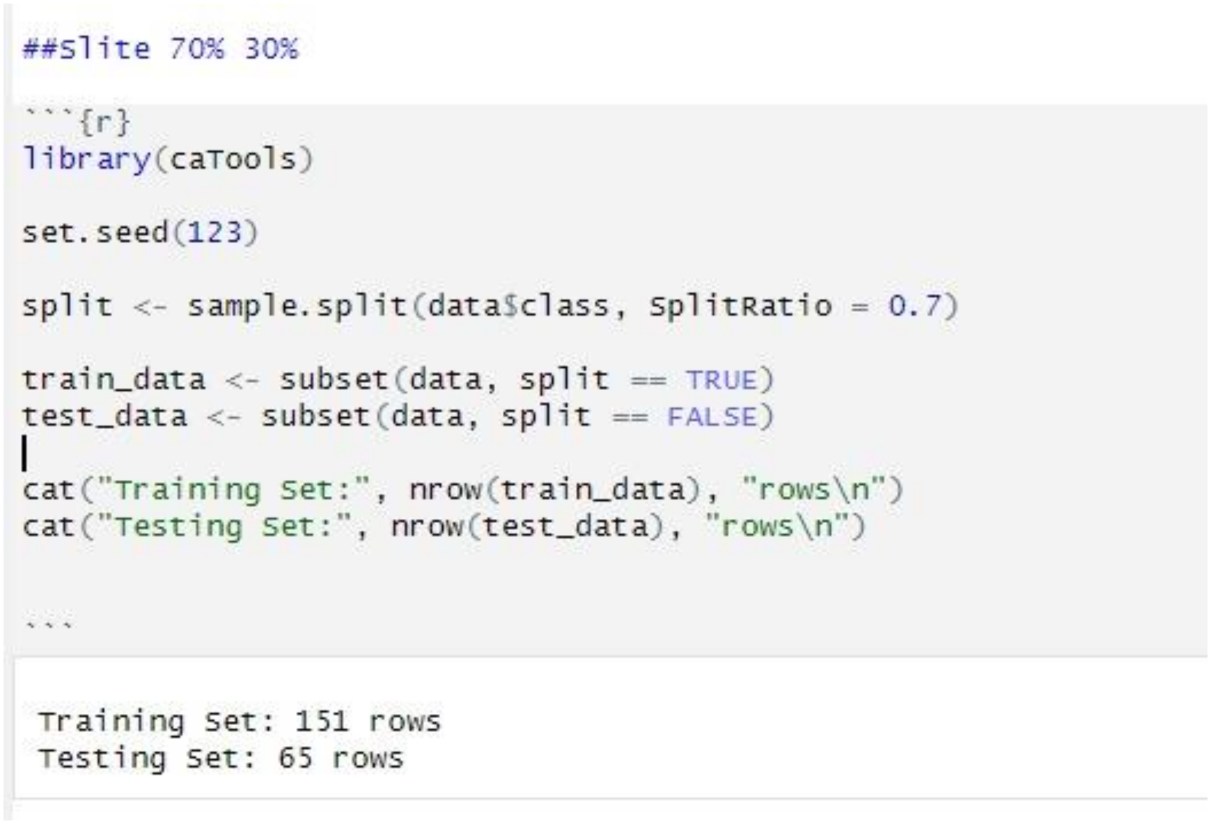
Salary Range (Min and Max):
[1] 200000 940000

Minimum Salary: 200000
Maximum Salary: 940000
```
```

This image shows R code and output for analysing gender distribution and salary statistics. First, it creates a table showing the count and percentage of males and females in the dataset. The gender split is approximately 65% male and 35% female. It also generates summary statistics of the salary variable, including mean, median, and quartiles. The salary range is calculated, showing a minimum salary of 200,000 and a maximum of 940,000.Overall, the code provides a quick demographic and financial overview of the dataset.



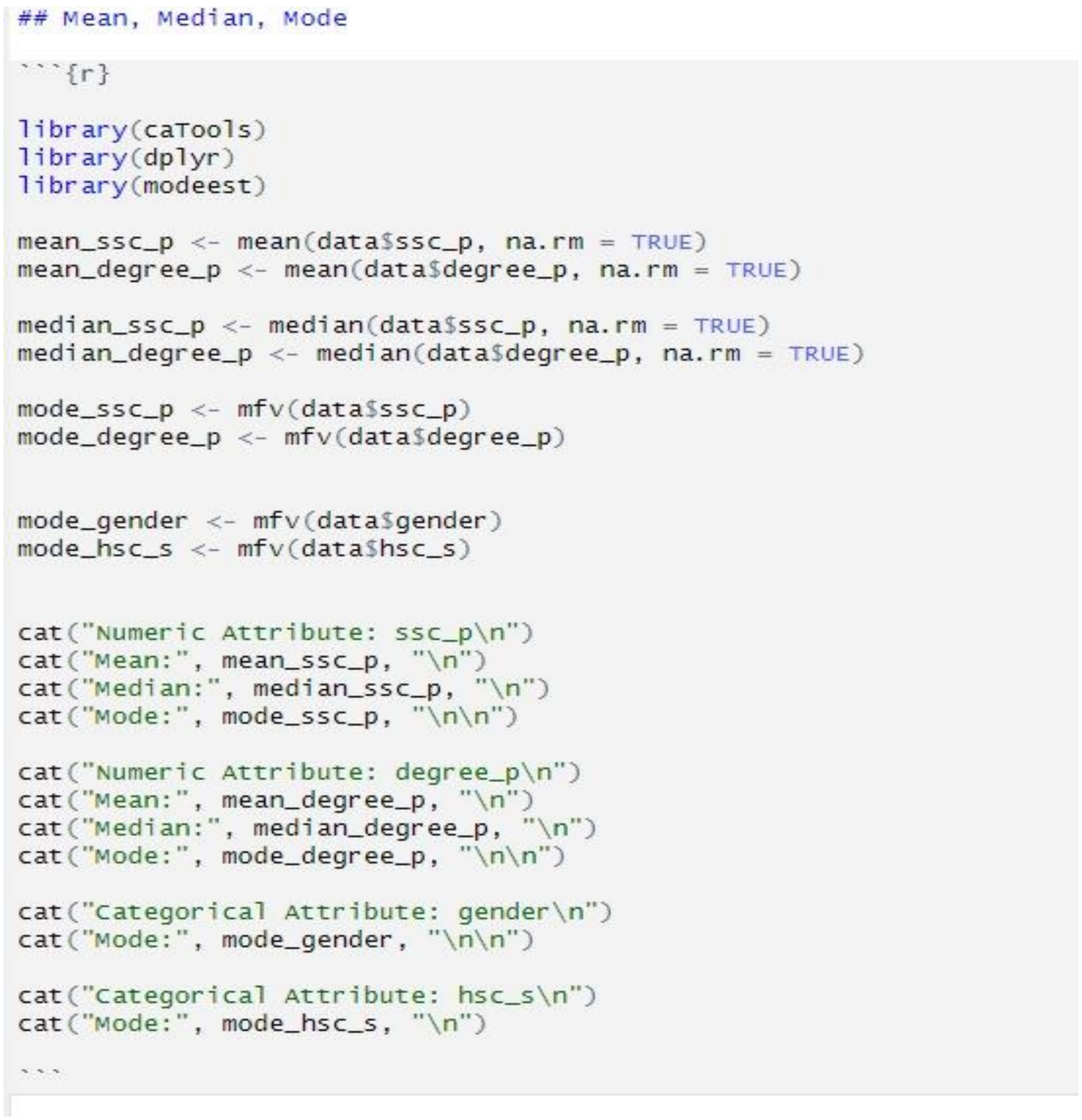
~~~
##Balance Imbalance
```{r}
library(ROSE)
data_balanced <- ovun.sample(status ~ ., data = data, method = "over", N = max(table(data$status)) * 2)$data
data_balanced <- ovun.sample(status ~ ., data = data, method = "under", N = min(table(data$status)) * 2)$data
table(data_balanced$status)
~~~

Not Placed   Placed
        67   67
```

The R code used to address class imbalance in a dataset is displayed in this image. The status variable can be oversampled and under sampled using the ROSE library. While under sampling shrinks the majority class, oversampling expands the minority class to equal the size of the majority class. The "Placed" and "Not Placed" categories each have 67 instances in the final balanced dataset. This guarantees that during training, the model won't be skewed toward either class.



```
##slite 70% 30%

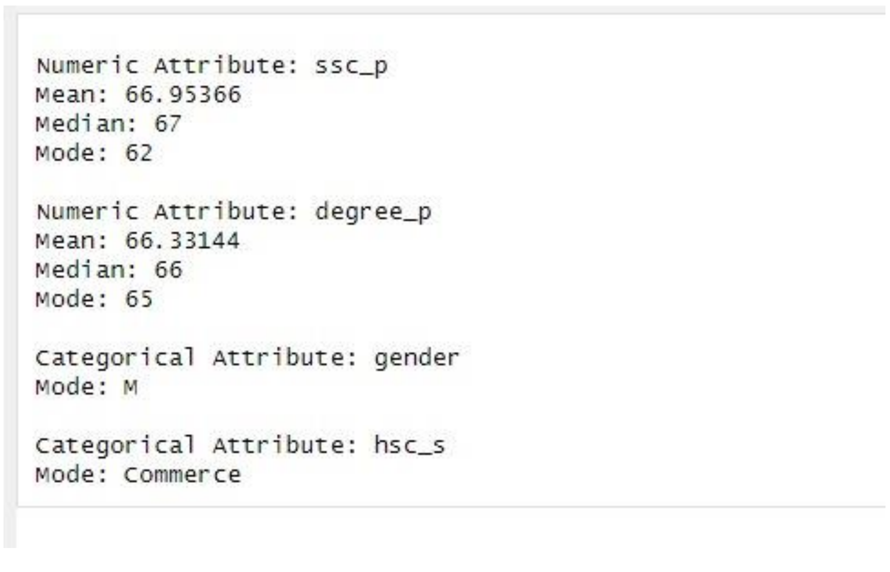
```{r}
library(caTools)

set.seed(123)

split <- sample.split(data$class, SplitRatio = 0.7)

train_data <- subset(data, split == TRUE)
test_data <- subset(data, split == FALSE)

cat("Training Set:", nrow(train_data), "rows\n")
cat("Testing Set:", nrow(test_data), "rows\n")


```

 Training Set: 151 rows
 Testing Set: 65 rows
```

This picture displays R code that uses the caTools library to divide a dataset into training and testing sets.For reproducibility, a random seed (set.seed(123)) is used. According to the class column, the dataset is divided into 70% training and 30% testing. According to the output,
```

train_data has 151 rows and test_data has 65 rows.The data is ready for model construction and assessment thanks to this separation.

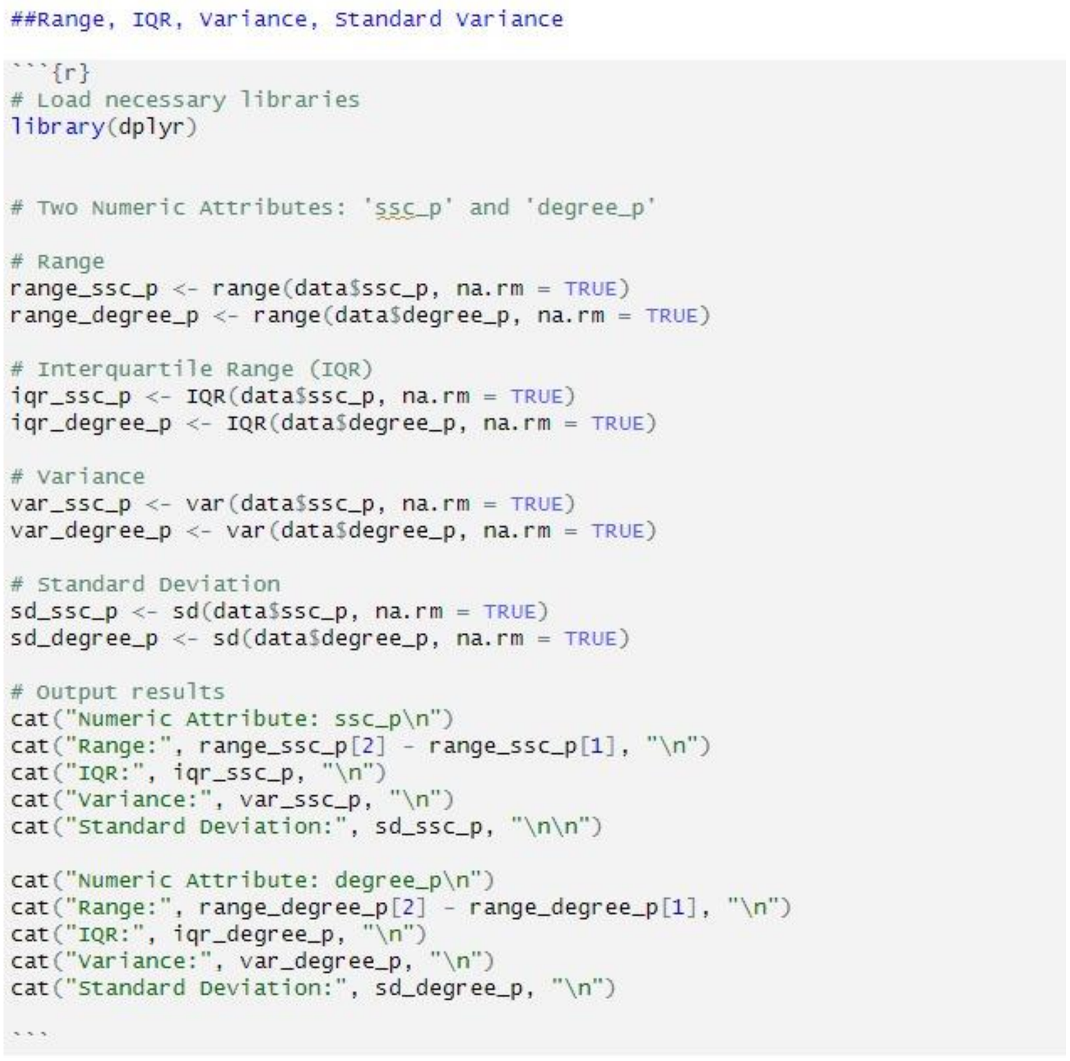


~~~
## Mean, Median, Mode

```{r}

library(caTools)
library(dplyr)
library(modeest)

mean_ssc_p <- mean(data$ssc_p, na.rm = TRUE)
mean_degree_p <- mean(data$degree_p, na.rm = TRUE)

median_ssc_p <- median(data$ssc_p, na.rm = TRUE)
median_degree_p <- median(data$degree_p, na.rm = TRUE)

mode_ssc_p <- mfv(data$ssc_p)
mode_degree_p <- mfv(data$degree_p)


mode_gender <- mfv(data$gender)
mode_hsc_s <- mfv(data$hsc_s)


cat("Numeric Attribute: ssc_p\n")
cat("Mean:", mean_ssc_p, "\n")
cat("Median:", median_ssc_p, "\n")
cat("Mode:", mode_ssc_p, "\n\n")

cat("Numeric Attribute: degree_p\n")
cat("Mean:", mean_degree_p, "\n")
cat("Median:", median_degree_p, "\n")
cat("Mode:", mode_degree_p, "\n\n")

cat("Categorical Attribute: gender\n")
cat("Mode:", mode_gender, "\n\n")

cat("Categorical Attribute: hsc_s\n")
cat("Mode:", mode_hsc_s, "\n")

```
~~~

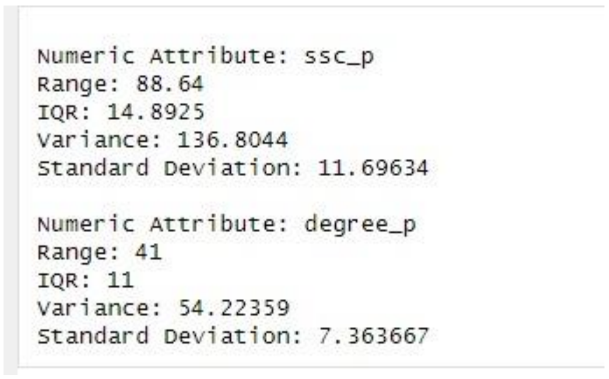


```
Numeric Attribute: ssc_p
Mean: 66.95366
Median: 67
Mode: 62

Numeric Attribute: degree_p
Mean: 66.33144
Median: 66
Mode: 65

Categorical Attribute: gender
Mode: M

Categorical Attribute: hsc_s
Mode: Commerce
```

The R code and its results for determining the mean, median, and mode of both numerical and categorical attributes are displayed in this graphic. The code calculates statistics for gender, hsc_s (High School Specialization), degree_p (Degree Percentage), and ssc_p (Secondary School Percentage) using the caTools, dplyr, and modeest libraries. Whereas just the mode is shown for categorical data, the mean, median, and mode are computed and presented for numeric attributes. The results indicate that "M" and "Commerce" are the most prevalent values (modes) for gender and hsc_s, respectively, and that the mean and median of ssc_p and degree_p are around 66. This aids in comprehending how the data properties are distributed.



~~~
##Range, IQR, Variance, Standard Variance

```{r}
# Load necessary libraries
library(dplyr)


# Two Numeric Attributes: 'ssc_p' and 'degree_p'

# Range
range_ssc_p <- range(data$ssc_p, na.rm = TRUE)
range_degree_p <- range(data$degree_p, na.rm = TRUE)

# Interquartile Range (IQR)
iqr_ssc_p <- IQR(data$ssc_p, na.rm = TRUE)
iqr_degree_p <- IQR(data$degree_p, na.rm = TRUE)

# Variance
var_ssc_p <- var(data$ssc_p, na.rm = TRUE)
var_degree_p <- var(data$degree_p, na.rm = TRUE)

# Standard Deviation
sd_ssc_p <- sd(data$ssc_p, na.rm = TRUE)
sd_degree_p <- sd(data$degree_p, na.rm = TRUE)

# Output results
cat("Numeric Attribute: ssc_p\n")
cat("Range:", range_ssc_p[2] - range_ssc_p[1], "\n")
cat("IQR:", iqr_ssc_p, "\n")
cat("Variance:", var_ssc_p, "\n")
cat("Standard Deviation:", sd_ssc_p, "\n\n")

cat("Numeric Attribute: degree_p\n")
cat("Range:", range_degree_p[2] - range_degree_p[1], "\n")
cat("IQR:", iqr_degree_p, "\n")
cat("Variance:", var_degree_p, "\n")
cat("Standard Deviation:", sd_degree_p, "\n")

```
~~~



```
Numeric Attribute: ssc_p
Range: 88.64
IQR: 14.8925
Variance: 136.8044
Standard Deviation: 11.69634

Numeric Attribute: degree_p
Range: 41
IQR: 11
Variance: 54.22359
Standard Deviation: 7.363667
```

The R code and results for determining statistical measures for the two numeric qualities, ssc_p and degree_p, are shown in this graphic. After loading the dplyr library, the code calculates the variance, standard deviation, range, and interquartile range (IQR) for both characteristics while accounting for missing values. It prints the results elegantly and makes use of procedures like range(), IQR(), var(), and sd(). In comparison to degree_p, the output indicates that ssc_p has a greater spread (range: 88.64) and variability (variance: 136.8044). All things considered, this excerpt offers a simple statistical overview to comprehend the consistency and dispersion of the two characteristics.