AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Science and Technology

Project

Assignment Title:	Mid Term Project			
Assignment No:	01		Date of Submission:	26 April 2025
Course Title:	INTRODUCTION TO DATA SCIENCE			
Course Code:	CSC4180		Section:	E
Semester:	Spring	2024-25	Course Teacher:	**DR. ABDUS SALAM**

Declaration and Statement of Authorship:

1. I/we hold a copy of this Assignment/Case-Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case-Study is my/our original work and no part of it has been copied from any other student's work or from any other source except where due acknowledgement is made.
3. No part of this Assignment/Case-Study has been written for me/us by any other person except where such collaboration has been authorized by the concerned teacher and is clearly acknowledged in the assignment.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared and archived for the purpose of detecting plagiarism.
6. I/we give permission for a copy of my/our marked work to be retained by the faculty for review and comparison, including review by external examiners.
7. I/we understand that Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a formofcheatingandisaveryseriousacademicoffencethatmayleadtoexpulsionfromtheUniversity. Plagiarized material can be drawn from, and presented in, written, graphic and visual form, including electronic data, and oral presentations. Plagiarism occurs when the origin of their arterial used is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or to copy my/our work.

* *Student(s) must complete all details except the faculty use part.*
** Please submit all assignments to your course teacher or the office of the concerned teacher.

Group Name/No.: 9

No	Name	ID	Program	Signature
1	MAHMUDUS SAMI MAAHI	22-46446-1	BSc [CSE]	
2	SABIHA IQBAL ETU	22-47227-1	BSc [CSE]	
3	MUKSHIT SAFI OWASI	22-47251-1	BSc [CSE]	
4				

Faculty use only

FACULTY COMMENTS		
	Marks Obtained	
	Total Marks	

Introduction:

This report is based on the placement_Data_Full_Class – modified.csv dataset. It contains student academic and placement-related information and includes 216 student records and 16 attributes, where each represents detailed information about education, work experiences, and placement outcomes. The dataset helps to understand whether things affect whether a student gets a job after finishing their studies. The dataset is also useful for data preparation tasks like handling missing or invalid values, normalization, data balancing and statistical analysis.

Feature Description:

sl_no: Serial number (unique identifier for each student)
gender: Student's gender (M/F)
ssc_p: Secondary Education (10th grade) percentage
ssc_b: Board of Secondary Education (Central/Others)
hsc_p: Higher Secondary (12th grade) percentage
hsc_b: Board of Higher Secondary (Central/Others)
hsc_s: HSC specialization (Commerce, Science, Arts)
degree_p: Undergraduate degree percentage
degree_t: Type of undergraduate degree (Sci&Tech, Comm&Mgmt, Others)
workex: Work experience (Yes/No)
etest_p: Employability test percentage
specialisation: MBA Specialization (Mkt&Fin or Mkt&HR)
mba_p: MBA percentage
status: Placement status (Placed/Not Placed)
salary: Salary offered (only for placed students)
class: Additional attribute

Data exploration:

Required library –

```
library(dplyr)
```

dplyr is a package used for data manipulation in R. This library is included for further use.

Importing the dataset –

```
> file <- read.csv("F:/spring/DataScienceProject/Placement_Data_Full_Class - modified.csv")
> file
   sl_no gender ssc_p    ssc_b hsc_p   hsc_b    hsc_s degree_p  degree_t workex etest_p specialisation
1      1      M 67.00  Others 91.00  Others Commerce    58.00  Sci&Tech      0   55.00         Mkt&HR
2      2      M 79.33 Central 78.33  Others  Science    77.48  Sci&Tech      1   86.50        Mkt&Fin
3      3      M 65.00 Central 68.00 Central     Arts    64.00 Comm&Mgmt      0   75.00        Mkt&Fin
4      4      M 56.00 Central 52.00 Central  Science    52.00  Sci&Tech      0   66.00         Mkt&HR
5      5      M 85.80 Central 73.60 Central Commerce    73.30 Comm&Mgmt      0   96.80        Mkt&Fin
6      6      M 55.00  Others 49.80  Others  Science    67.25  Sci&Tech      1   55.00        Mkt&Fin
7      7      F 46.00  Others 49.20  Others Commerce    79.00 Comm&Mgmt      0   74.28        Mkt&Fin
8      8      M 82.00 Central 64.00 Central  Science    66.00  Sci&Tech      1   67.00        Mkt&Fin
9      9      M 73.00 Central 79.00 Central Commerce    72.00 Comm&Mgmt      0   91.34        Mkt&Fin
10    10      M 58.00 Central 70.00 Central Commerce    61.00 Comm&Mgmt      0   54.00        Mkt&Fin
11    11      M 58.00 Central 61.00 Central Commerce    60.00 Comm&Mgmt      1   62.00         Mkt&HR
12    12      M 69.60 Central 68.40 Central Commerce    78.30 Comm&Mgmt      1   60.00        Mkt&Fin
```

The read.csv() function reads a CSV file and loads the data into a data frame.

Dataset explanation –

```
> dim(file)
[1] 216  16
>
```

It shows the dimension of the dataset. (216 rows, 16 columns)

```
> names(file)
 [1] "sl_no"          "gender"         "ssc_p"          "ssc_b"          "hsc_p"
 [6] "hsc_b"          "hsc_s"          "degree_p"       "degree_t"       "workex"
[11] "etest_p"        "specialisation" "mba_p"          "status"         "salary"
[16] "class"
>
```

names(file) is used to display column names.

```
> head(file)
  sl_no gender ssc_p    ssc_b hsc_p   hsc_b    hsc_s degree_p  degree_t workex etest_p specialisation
1     1      M 67.00  Others 91.00  Others Commerce    58.00  Sci&Tech      0    55.0         Mkt&HR
2     2      M 79.33 Central 78.33  Others  Science    77.48  Sci&Tech      1    86.5        Mkt&Fin
3     3      M 65.00 Central 68.00 Central     Arts    64.00 Comm&Mgmt      0    75.0        Mkt&Fin
4     4      M 56.00 Central 52.00 Central  Science    52.00  Sci&Tech      0    66.0         Mkt&HR
5     5      M 85.80 Central 73.60 Central Commerce    73.30 Comm&Mgmt      0    96.8        Mkt&Fin
6     6      M 55.00  Others 49.80  Others  Science    67.25  Sci&Tech      1    55.0        Mkt&Fin
  mba_p     status salary class
1 58.80     Placed 270000     0
2 66.28     Placed 200000     0
3 57.80     Placed 250000     0
4 59.43 Not Placed     NA     0
5 55.50     Placed 425000     1
6 51.58 Not Placed     NA     1
```

head(files) show the first 6 rows of the dataset.

```
> summary(file)
     sl_no          gender             ssc_p           ssc_b              hsc_p           hsc_b
 Min.   :  1.00   Length:216         Min.   : 0.76   Length:216         Min.   :  37.00   Length:216
 1st Qu.: 54.75   Class :character   1st Qu.:60.36   Class :character   1st Qu.:  60.95   Class :character
 Median :108.50   Mode  :character   Median :67.00   Mode  :character   Median :  65.00   Mode  :character
 Mean   :108.50                      Mean   :66.95                      Mean   :  97.06
 3rd Qu.:162.25                      3rd Qu.:75.25                      3rd Qu.:  73.05
 Max.   :216.00                      Max.   :89.40                      Max.   :6680.00

    hsc_s             degree_p        degree_t           workex          etest_p      specialisation
 Length:216         Min.   :50.00   Length:216         Min.   :0.0000   Min.   :50.00   Length:216
 Class :character   1st Qu.:61.00   Class :character   1st Qu.:0.0000   1st Qu.:60.00   Class :character
 Mode  :character   Median :66.00   Mode  :character   Median :0.0000   Median :70.50   Mode  :character
                    Mean   :66.33                      Mean   :0.3442   Mean   :72.02
                    3rd Qu.:72.00                      3rd Qu.:1.0000   3rd Qu.:83.25
                    Max.   :91.00                      Max.   :1.0000   Max.   :98.00
                                                       NA's   :1
     mba_p          status              salary           class
 Min.   :51.21   Length:216         Min.   :200000   Min.   :0.0000
 1st Qu.:57.97   Class :character   1st Qu.:240000   1st Qu.:0.0000
 Median :61.95   Mode  :character   Median :265000   Median :1.0000
 Mean   :62.26                      Mean   :288530   Mean   :0.5556
 3rd Qu.:66.24                      3rd Qu.:300000   3rd Qu.:1.0000
 Max.   :77.89                      Max.   :940000   Max.   :1.0000
                                    NA's   :67
```

The summary() function provides a statistical summary of an object, which includes measures like minimum, maximum, median, and quartiles for each variable.

Data pre-processing :

Handling duplicate values -

```
> original_rows <- nrow(file)
> file <- unique(file)
> new_rows <- nrow(file)
> print(paste("Original rows:", original_rows))
[1] "Original rows: 216"
> print(paste("After removing duplicates:", new_rows))
[1] "After removing duplicates: 216"
> print(paste("Duplicates removed:", original_rows - new_rows))
[1] "Duplicates removed: 0"
>
```

To ensure the dataset was clean, duplicate rows were identified and removed using the unique() function. First, the total number of original rows was recorded. After removing duplicates, the new number of rows was compared with the original to find out how many duplicates were eliminated. Removing duplicates is important because repeated records can affect the accuracy of data analysis.

Finding missing values -

```
> missing_values <- colSums(is.na(file))
> print("Missing values in each column:")
[1] "Missing values in each column:"
> print(missing_values)
         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              1              1              0              0              0              0
       salary          class
           67              0
>
```

To find missing values in the dataset, the is.na() function was used to check each cell for missing (NA) entries. Then, colSums(is.na(file)) was applied to count how many missing values were present in each column. Finally, the missing values were printed to clearly see which columns had missing data and how many entries were missing. Finding missing values is important because missing data can affect the accuracy and quality of the analysis.

Handling missing values –

```
> median_salary <- median(file$salary, na.rm = TRUE)
> print(paste("Median salary:", median_salary))
[1] "Median salary: 265000"
> file
```

In this step, the median salary was calculated using the median() function, while ignoring any missing values (na.rm = TRUE). The calculated median was then printed to check its value. This was done because the salary column had some missing entries, and using the median is a good way to fill in missing numeric values without being affected by outliers. Finding the median first helps to prepare for replacing missing salary values in the next step, which ensures the dataset stays complete and ready for analysis.

```
› file$salary[is.na(file$salary)] <- median_salary
› file
♯ A tibble: 216 × 16
   sl_no gender ssc_p ssc_b    hsc_p hsc_b    hsc_s degree_p degree_t workex etest_p specialisation mba_p status salary
   <dbl> <chr>  <dbl> <chr>    <dbl> <chr>    <chr>    <dbl> <chr>     <dbl>   <dbl> <chr>          <dbl> <chr>   <dbl>
 1     1 M       67   Others    91   Others   Comm…     58   Sci&Tech      0    55   Mkt&HR          58.8 Placed 270000
 2     2 M       79.3 Central   78.3 Others   Scie…     77.5 Sci&Tech      1    86.5 Mkt&Fin         66.3 Placed 200000
 3     3 M       65   Central   68   Central  Arts      64   Comm&Mg…      0    75   Mkt&Fin         57.8 Placed 250000
 4     4 M       56   Central   52   Central  Scie…     52   Sci&Tech      0    66   Mkt&HR          59.4 Not P… 265000
 5     5 M       85.8 Central   73.6 Central  Comm…     73.3 Comm&Mg…      0    96.8 Mkt&Fin         55.5 Placed 425000
 6     6 M       55   Others    49.8 Others   Scie…     67.2 Sci&Tech      1    55   Mkt&Fin         51.6 Not P… 265000
 7     7 F       46   Others    49.2 Others   Comm…     79   Comm&Mg…      0    74.3 Mkt&Fin         53.3 Not P… 265000
 8     8 M       82   Central   64   Central  Scie…     66   Sci&Tech      1    67   Mkt&Fin         62.1 Placed 252000
```

After calculating the median salary, this line was used to fill in the missing (NA) values in the salary column. The is.na(file$salary) part checks which rows have missing salaries, and those values are replaced with the previously calculated median. This ensures that the dataset has no empty salary entries.

Checking for invalid values –

```
> invalid_ssc <- sum(file$ssc_p < 0 | file$ssc_p > 100, na.rm=TRUE)
> invalid_hsc <- sum(file$hsc_p < 0 | file$hsc_p > 100, na.rm=TRUE)
> invalid_degree <- sum(file$degree_p < 0 | file$degree_p > 100, na.rm=TRUE)
> invalid_mba <- sum(file$mba_p < 0 | file$mba_p > 100, na.rm=TRUE)
>
> print(paste("Invalid ssc_p values:", invalid_ssc))
[1] "Invalid ssc_p values: 0"
> print(paste("Invalid hsc_p values:", invalid_hsc))
[1] "Invalid hsc_p values: 1"
> print(paste("Invalid degree_p values:", invalid_degree))
[1] "Invalid degree_p values: 0"
> print(paste("Invalid mba_p values:", invalid_mba))
[1] "Invalid mba_p values: 0"
```

In this part, it was checked that if there were any wrong values in the percentage columns like ssc_p, hsc_p, degree_p, and mba_p. A percentage should always be between 0 and 100. So, a condition was used to find if any values were less than 0 or greater than 100. Then, it was counted how many wrong values there were and printed them.

Correcting invalid percentage values by replacing them with the median -

```
ssc_median <- median(file$ssc_p, na.rm = TRUE)
hsc_median <- median(file$hsc_p, na.rm = TRUE)
degree_median <- median(file$degree_p, na.rm = TRUE)
mba_median <- median(file$mba_p, na.rm = TRUE)

file$ssc_p[file$ssc_p < 0 | file$ssc_p > 100] <- ssc_median
file$hsc_p[file$hsc_p < 0 | file$hsc_p > 100] <- hsc_median
file$degree_p[file$degree_p < 0 | file$degree_p > 100] <- degree_median
file$mba_p[file$mba_p < 0 | file$mba_p > 100] <- mba_median
```

In this step, we first corrected invalid percentage values in the columns ssc_p, hsc_p, degree_p, and mba_p. Any value less than 0 or greater than 100 was considered invalid and replaced with the median of that respective column, instead of removing the record.

Converting ssc_b from categorical to numeric –

```
  argument "x" is missing, with no default
> print(paste("Missing values in ssc_b:", sum(is.na(file$ssc_b))))
[1] "Missing values in ssc_b: 0"
> file$ssc_b[is.na(file$ssc_b)] <- "Central"
>
>
> file$ssc_b[file$ssc_b == "Central"] <- 0
> file$ssc_b[file$ssc_b == "Others"] <- 1
> file$ssc_b <- as.numeric(file$ssc_b)
> file
```

We handled missing values in the ssc_b column, which represents the type of board (either Central or Others). First, we checked for missing values and found any occurrences of NA. We then replaced these missing values with Central as a default value, assuming it's the most common category. After handling the missing values, we converted the ssc_b column from categorical values to numeric values for easier analysis and modeling. We assigned "Central" a value of 0 and "Others" a value of 1, then converted the entire column to a numeric format.

Normalizing –

```
> min_salary <- min(file$salary, na.rm = TRUE)
> max_salary <- max(file$salary, na.rm = TRUE)
> print(paste("Min salary:", min_salary))
[1] "Min salary: 2e+05"
> print(paste("Max salary:", max_salary))
[1] "Max salary: 940000"
>
>
> file$salary <- (file$salary - min_salary) / (max_salary - min_salary)
```

Normalizing salary value using min-max

In this part we normalized the salary column using the Min-Max normalization technique to scale the salary values between 0 and 1. First, we calculated the minimum and maximum salary values in the dataset. Then, using the Min-Max formula, we transformed the salary values by subtracting the minimum salary and dividing by the range (maximum salary - minimum salary). This ensures that all salary values are on the same scale.

Balancing dataset –

```
> placed_count <- sum(file$class == 1)
> not_placed_count <- sum(file$class == 0)
>
> if (placed_count > not_placed_count) {
+   majority_class <- 1
+   minority_class <- 0
+   minority_count <- not_placed_count
+ } else {
+   majority_class <- 0
+   minority_class <- 1
+   minority_count <- placed_count
+ }
>
> print(paste("Majority class:", majority_class, "with", max(placed_count, not_placed_count), "instances"))
[1] "Majority class: 1 with 120 instances"
> print(paste("Minority class:", minority_class, "with", min(placed_count, not_placed_count), "instances"))
[1] "Minority class: 0 with 96 instances"
```

In this step, we checked how many students were placed (class = 1) and not placed (class = 0) by counting their numbers using the sum() function. Then, we compared the two counts using an if-else condition to identify which group had more students (majority class) and which had fewer (minority class). We saved the

counts into variables for further balancing. This step was important because we needed to balance the dataset later, and first identifying the majority and minority classes was necessary to do that correctly.

Converting workex to categorical –

```
>
> file$workex[file$workex == 0] <- "No"
> file$workex[file$workex == 1] <- "Yes"
>
```

In this step, we converted the workex attribute from numeric values (0 and 1) to categorical values (No and Yes). We replaced 0 with No and 1 with Yes.

```
> set.seed(123)
> majority_indices <- which(file$class == majority_class)
> sampled_indices <- sample(majority_indices, minority_count)
> keep_indices <- c(sampled_indices, which(file$class == minority_class))
>
> file <- file[keep_indices, ]
>
```

In this step, we first used set.seed(123) to make sure the random sampling gives the same result every time the code is run. Then, we found the rows where students belonged to the majority class using the which() function. From the majority of students, we randomly selected a number of samples equal to the number of minority students using sample(). We combined these sampled majority students with all minority students using c(), and finally updated the dataset by keeping only these selected rows. This was done to create a balanced dataset.

Split into training and test sets –

```
>
> set.seed(123)
> total_rows <- nrow(file)
> train_size <- round(0.8 * total_rows)
> train_indices <- sample(1:total_rows, train_size)
>
> train_data <- file[train_indices, ]
> test_data  <- file[-train_indices, ]
>
> print(paste("Total balanced data rows:", total_rows))
[1] "Total balanced data rows: 192"
> print(paste("Training set rows:", nrow(train_data)))
[1] "Training set rows: 154"
> print(paste("Testing set rows:", nrow(test_data)))
[1] "Testing set rows: 38"
>
```

In this step, we split the balanced dataset into a training set and a testing set. First, we set a seed using set.seed(123) to make sure the random split is the same every time the code is run. We calculated 80% of the total rows for training and selected random rows using the sample() function. The selected rows became the train_data, and the remaining rows became the test_data. This split is important to train the model on one part of the data and test its performance on unseen data to check how well it generalizes.

```
>
> print("Training set class distribution:")
[1] "Training set class distribution:"
> print(table(train_data$class))

 0  1
77 77
> print("Testing set class distribution:")
[1] "Testing set class distribution:"
> print(table(test_data$class))

 0  1
19 19
```

In this step, we checked the class distribution in both the training and testing sets by using the table() function on the class column. We printed how many students were placed and not placed in each set. This check was important to confirm that even after splitting, both the training and testing sets remained balanced, so that the model would learn and perform properly without being biased toward any class.

Central tendency measure for ssc_p –

```
> ssc_mean <- mean(file$ssc_p, na.rm = TRUE)
> ssc_median <- median(file$ssc_p, na.rm = TRUE)
> ssc_table <- table(file$ssc_p)
> ssc_mode <- as.numeric(names(ssc_table)[which.max(ssc_table)])
>
> print("Central tendency for SSC Percentage:")
[1] "Central tendency for SSC Percentage:"
> print(paste("Mean:", round(ssc_mean, 2)))
[1] "Mean: 66.75"
> print(paste("Median:", ssc_median))
[1] "Median: 67"
> print(paste("Mode:", ssc_mode))
[1] "Mode: 62"
>
```

We calculated the central tendency measures (mean, median, and mode) for the ssc_p (SSC Percentage) to understand the typical performance of students in this column. The mean was calculated using the mean() function, the median using median(), and the mode by finding the most frequent value using table() and which.max(). These measures help us summarize the data and understand the general trend of SSC percentages.

Spread measure for ssc_p –

```
> ssc_range <- max(file$ssc_p, na.rm = TRUE) - min(file$ssc_p, na.rm = TRUE)
> ssc_iqr <- IQR(file$ssc_p, na.rm = TRUE)
> ssc_variance <- var(file$ssc_p, na.rm = TRUE)
> ssc_sd <- sd(file$ssc_p, na.rm = TRUE)
>
> print("Spread measures for SSC Percentage:")
[1] "Spread measures for SSC Percentage:"
> print(paste("Range:", round(ssc_range, 2)))
[1] "Range: 88.64"
> print(paste("IQR:", round(ssc_iqr, 2)))
[1] "IQR: 14.02"
> print(paste("Variance:", round(ssc_variance, 2)))
[1] "Variance: 137.62"
> print(paste("Standard Deviation:", round(ssc_sd, 2)))
[1] "Standard Deviation: 11.73"
```

We calculated the spread measures for ssc_p (SSC Percentage) to understand the variability in the data. The range was computed by subtracting the minimum value from the maximum value. The Interquartile Range (IQR) was calculated using the 25th and 75th percentiles (Q1 and Q3), showing the spread of the middle

50% of the data. Variance and standard deviation were calculated using var() and sd() to measure how spread out the values are from the mean. These measures help us understand the data distribution and identify potential outliers or variations in student performance.

Central tendency and spread for MBA percentage –

```
> mba_mean <- mean(file$mba_p, na.rm = TRUE)
> mba_median <- median(file$mba_p, na.rm = TRUE)
> mba_table <- table(file$mba_p)
> mba_mode <- as.numeric(names(mba_table)[which.max(mba_table)])
>
> print("Central tendency for MBA Percentage:")
[1] "Central tendency for MBA Percentage:"
> print(paste("Mean:", round(mba_mean, 2)))
[1] "Mean: 62.16"
> print(paste("Median:", mba_median))
[1] "Median: 61.7"
> print(paste("Mode:", mba_mode))
[1] "Mode: 56.7"
>
> mba_range <- max(file$mba_p, na.rm = TRUE) - min(file$mba_p, na.rm = TRUE)
> mba_iqr <- IQR(file$mba_p, na.rm = TRUE)
> mba_variance <- var(file$mba_p, na.rm = TRUE)
> mba_sd <- sd(file$mba_p, na.rm = TRUE)
>
> print("Spread measures for MBA Percentage:")
[1] "Spread measures for MBA Percentage:"
> print(paste("Range:", round(mba_range, 2)))
[1] "Range: 26.68"
> print(paste("IQR:", round(mba_iqr, 2)))
[1] "IQR: 8.33"
> print(paste("Variance:", round(mba_variance, 2)))
[1] "Variance: 33.27"
> print(paste("Standard Deviation:", round(mba_sd, 2)))
[1] "Standard Deviation: 5.77"
```

We calculated the central tendency measures (mean, median, and mode) to determine the typical performance of the students. The mean provides an average score, the median shows the middle value when sorted, and the mode identifies the most common score. To understand the variability in the data, we calculated the spread measures, including range, interquartile range (IQR), variance, and standard deviation. These measures highlight the extent to which the MBA percentages deviate from the average. By using functions like max(), min(), IQR(), var(), and sd(), we gained insight into the consistency and dispersion of MBA percentages.