Data Science

Statistical Data Modelling

Task 1:

a. Compare amongst various types of predictive models. Use comparison paragraph/chart/table with specific similarities and differences supported by examples.

Linear Regression	**Logistic Regression**	**Decision Trees**	**Random Forest**	**Neural Networks**	**K-Nearest Neighbors (KNN)**	**Naive Bayes**
use one or more independent variables to forecast a continuous dependent variable. However, it is a statistical technique that uses a set of input values to predict an output value.	Problems involving binary classification, where the possible answers are 0 or 1, yes or no. However, it is a statistical technique that estimates the likelihood that an output value from a collection of categorical	Regression and classification problems. Decision trees use a set of queries or conditions to determine their decisions.	Multiple decision trees are used in an ensemble learning technique to increase accuracy and decrease overfitting.	Models with deep learning capabilities can extract complicated trends from data.	Regression and classification using the average of the k-nearest data points or the majority class as the basis.	Problems with classification, particularly in spam filtering and natural language processing.

	variables will fall into a particular category.					
Estimating the cost of a home by taking into account variables like square footage, number of bedrooms, etc.	Determining whether or not an email is spam.	Estimating a customer's likelihood of churn based on a number of variables.	Estimating the course of a disease from a variety of medical characteristics.	Recognition of images and natural language processing.	Estimating an individual's income by looking at the incomes of their k-nearest neighbors.	Determining whether an email contains spam based on the frequency of particular phrases.
A straight line that represents a linear relationship.	An S-shaped curve represents a sigmoidal or logistic relationship.				A non-parametric technique for making predictions is KNN regression, which uses neighbors as a basis.	
Linear Regression	**Logistic Regression**	**Decision Trees**	**Random Forest**	**Neural Networks**	**K-Nearest Neighbors (KNN)**	**Naive Bayes**
One kind of supervised	Among the various kinds of supervised					

learning is regression.	learning is classification.					
Distribution type is Normal or Gaussian.	Distribution type is Binomial.					
Ideal for tasks where a continuous dependent variable from a scale needs to be expected.	Ideal for tasks involving for estimating the probability that a categorical dependent variable will fall into one of a fixed set of categories.					
Linear Regression	**Logistic Regression**	**Decision Trees**	**Random Forest**	**Neural Networks**	**K-Nearest Neighbors (KNN)**	**Naive Bayes**
		Decision Trees perform better than Logistic Regressions				

		when the data set is vast and the interactions between the many features and the target variable are complex and non-linear.				
	Because logistic regression is a linear model that relies on a linear relationship between the predictors and the response variable's log-odds, it is impacted by outliers. The approach	Compared to logistic regression, decision trees are far more efficient at handling missing values and anomalies in the data. Outliers have no effect on a decision tree because it divides the				

	minimizes the sum of squared errors to find the line that fits the best. Overfitting can occur as a result of outliers, which can significantly affect the line. The model's ability to generalize to new data may be hindered by the presence of outliers, which might draw the line towards them.	data according to feature values. Outliers are simply regarded by the tree as additional data points with particular feature values, and the data will be divided appropriately.				

	Furthermore, the logistic regression approach estimates the model's parameters using Maximum Likelihood Estimation (MLE), which is susceptible to outliers.					
Linear Regression	**Logistic Regression**	**Decision Trees**	**Random Forest**	**Neural Networks**	**K-Nearest Neighbors (KNN)**	**Naive Bayes**
	Unlike decision trees, however, logistic regression	Decision trees can handle data that has a large number of missing or zero				

	requires that all features have non-zero values, which makes it ineffective with sparse data. It makes the assumption that the data are Missing Completely at Random (MCAR), which denotes that there is no relationship between the chance of missing data and either observed or	values for each attribute, which is known as sparse data.				

	unobserved variable. However, this is frequently not the case in practice, and since the missing data may be connected to the outcome variable, the estimates and predictions made by the model may be biased. Another problem is that, as a parametric model, logistic regression					

	requires a high sample size for accurate parameter estimation and makes assumptions about the data's distribution.					
Linear Regression	**Logistic Regression**	**Decision Trees**	**Random Forest**	**Neural Networks**	**K-Nearest Neighbors (KNN)**	**Naive Bayes**
	All features are used simultaneously in logistic regression, and as the	Decision trees are more resilient to high-dimensional data because				

	number of features rises, the model gets more complicated and challenging to understand.	they divide the feature space into smaller sections. This is so that patterns and predictions may be found more easily because the splitting lowers the effective dimensionality of the data.				
Linear Regression	**Logistic Regression**	**Decision Trees**	**Random Forest**	**Neural Networks**	**K-Nearest Neighbors (KNN)**	**Naive Bayes**
A global estimator based on the linear relationship between variables is					A neighborhood-based local estimator is KNN regression.	

called a linear regression.						
To create predictions, linear regression fits a line to the data.					Using KNN regression, values are predicted according to how similar they are to neighboring data points.	
					KNN is not a linear classifier.	Naive Bayes is a linear classifier.
					Because more calculations are needed for each successive step in the process, KNN is typically slower when dealing with larger data sets.	Naïve Bayes is faster than the KNN.
Linear Regression	**Logistic Regression**	**Decision Trees**	**Random Forest**	**Neural Networks**	**K-Nearest Neighbors (KNN)**	**Naive Bayes**
					When k in K-NN rises, the error rate falls until it hits the optimal Bayes value	Applying Naive Bayes to large data sets gives very accurate results.

					(for k→∞). This results in a high accuracy for KNN.	
					One tuning parameter for K-NN is the "k," or number of neighbors.	The alpha and beta hyperparameters are the two that Naive Bayes gives you to adjust for smoothing. A hyperparameter is an earlier parameter that is optimized by fine-tuning it on the training set.
					The K-NN approach is the most effective if conditional independence will significantly hinder classification.	The zero probability problem affects Naive Bayes, meaning that it is unable to generate a reliable forecast when the conditional probability of a given attribute is zero.

Linear Regression	Logistic Regression	Decision Trees	Random Forest	Neural Networks	K-Nearest Neighbors (KNN)	Naive Bayes
						Only when the decision boundary is parabolic, elliptic, or linear can Naive Bayes work.
					Understanding the underlying probability distributions is not necessary to use K-NN.	Knowing the underlying probability distributions for categories is necessary in order to use Naive Bayes. All other classifiers are measured against this ideal by the algorithm.
Linear Regression	**Logistic Regression**	**Decision Trees**	**Random Forest**	**Neural Networks**	**K-Nearest Neighbors (KNN)**	**Naive Bayes**
					K-NN requires no training; it just needs to be loaded with the	Naive Bayes does require training.

					dataset to begin operating.	
		Decision trees perform poorly when it comes to infrequent events since they nearly always remove significant classes from the model.			KNN models perform better than decision trees for uncommon events.	Naive Bayes models perform better than decision trees in situations involving rare occurrences.
		Decision trees don't need to be designed beforehand; they can operate straight from a table of data.			KNN classifiers require previous design work and cannot function straight from a table of data.	Naive Bayes techniques require previous design work and cannot operate immediately from a table of data.
Linear Regression	**Logistic Regression**	**Decision Tree**	**Random Forest**	**Neural Networks**	**K-Nearest Neighbors (KNN)**	**Naive Bayes**
		Construct	The number of rows chosen at random			

		multiple decision trees and determine the result.	during the building process of a random forest.			
		Presents accurate results	Presents less accurate results.			
		Decision trees are susceptible to overfitting, which is an inaccuracy brought on by bias or variance.	Using more than one tree lowers the likelihood of overfitting.			
		Because of its simplicity, the decision tree is simple to read and understand.	Random forest interpretation is more difficult. However, it plays an important role to show hidden patterns behind the data.			
Linear Regression	**Logistic Regression**	**Decision Tree**	**Random Forest**	**Neural Networks**	**K-Nearest Neighbors (KNN)**	**Naive Bayes**

		Decision trees process quickly, making implementation quick.	In a random forest, the generation, processing, and analysis of trees must be done slowly.			
		Decision trees has less computation.	Because random forests have n number of decision trees, more computation results from having more decision trees.			
Linear Regression	**Logistic Regression**	**Decision Tree**	**Random Forest**	**Neural Networks**	**K-Nearest Neighbors (KNN)**	**Naive Bayes**
		Neural network simplifications include decision trees and, as opposed to SVM		Decision trees and the problem solving method are similar.		

		or Logistic Regression, discover a single, complicated boundary that can divide a dataset; instead, they address problems gradually.				
		To maximize the amount of information gained, decision trees gradually divide the feature space along different features (using vertical and horizontal lines).		Neural networks work similarly, but they do so by adjusting the activation function's structure.		

		Decision Trees approach the piece-by-piece fitting of the model from a deterministic perspective. Deterministic models, such as decision trees, are therefore continually better suited to represent structured or tabular data.		A probabilistic approach is used by neural networks to piece-by-piece model fitting. Text and pixel values, for example, require the subtlety of probabilistic computation, and these types of data are well suited for neural networks. Neural network modifications like the recurrent and convolution layers are all quite clever.		

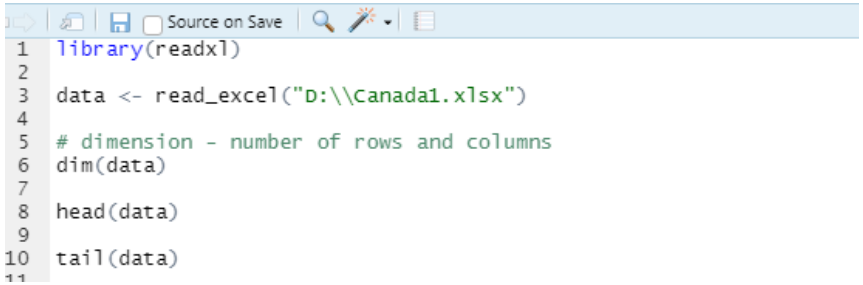
b. Use the data set to prepare a Histogram using R Programming. Use function to take values to plot the Histogram with two parameters. You can level and colour the histograms.

Histogram:

The histogram is a visual depiction of a dataset distribution that makes it simple to determine which factors—frequency distributions across continuous (numeric) variables—have the most and the least data. Histograms do, in fact, accept both grouped and ungrouped data. While ungrouped data requires the formation of the grouped frequency distribution, grouped data histograms are built by taking class boundaries into account.

Data source: www.kaggle.com

Viewing dimension (number of rows and columns), first 6 rows and last 6 rows.



```
Console  Terminal  Jobs
R 4.1.3 · ~/
> data <- read_excel("D:\\Canada1.xlsx")
> # dimension - number of rows and columns
> dim(data)
[1] 197  43
> tail(data)
# A tibble: 6 x 43
  Type    Coverage OdName  AREA AreaName   REG RegName   DEV DevName `1980` `1981` `1982` `1983` `1984`
  <chr>   <chr>    <chr>  <dbl> <chr>    <dbl> <chr>   <dbl> <chr>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 Immigr~ Foreign~ Weste~   903 Africa     912 Northe~   902 Develo~      0      0      0      0      0
2 Immigr~ Foreign~ Yemen    935 Asia       922 Wester~   902 Develo~      1      2      1      6      0
3 Immigr~ Foreign~ Zambia   903 Africa     910 Easter~   902 Develo~     11     17     11      7     16
4 Immigr~ Foreign~ Zimba~   903 Africa     910 Easter~   902 Develo~     72    114    102     44     32
5 Immigr~ Foreign~ Unkno~   999 World      999 World     999 World    44000  18078  16904  13635  14855
6 Immigr~ Both     Total    999 World      999 World     999 World   143137 128641 121175  89185  88272
# i 29 more variables: `1985` <dbl>, `1986` <dbl>, `1987` <dbl>, `1988` <dbl>, `1989` <dbl>,
#   `1990` <dbl>, `1991` <dbl>, `1992` <dbl>, `1993` <dbl>, `1994` <dbl>, `1995` <dbl>, `1996` <dbl>,
#   `1997` <dbl>, `1998` <dbl>, `1999` <dbl>, `2000` <dbl>, `2001` <dbl>, `2002` <dbl>, `2003` <dbl>,
#   `2004` <dbl>, `2005` <dbl>, `2006` <dbl>, `2007` <dbl>, `2008` <dbl>, `2009` <dbl>, `2010` <dbl>,
#   `2011` <dbl>, `2012` <dbl>, `2013` <dbl>
> tail(data)
# A tibble: 6 x 43
  Type    Coverage OdName  AREA AreaName   REG RegName   DEV DevName `1980` `1981` `1982` `1983` `1984`
  <chr>   <chr>    <chr>  <dbl> <chr>    <dbl> <chr>   <dbl> <chr>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 Immigr~ Foreign~ Weste~   903 Africa     912 Northe~   902 Develo~      0      0      0      0      0
2 Immigr~ Foreign~ Yemen    935 Asia       922 Wester~   902 Develo~      1      2      1      6      0
3 Immigr~ Foreign~ Zambia   903 Africa     910 Easter~   902 Develo~     11     17     11      7     16
4 Immigr~ Foreign~ Zimba~   903 Africa     910 Easter~   902 Develo~     72    114    102     44     32
5 Immigr~ Foreign~ Unkno~   999 World      999 World     999 World    44000  18078  16904  13635  14855
6 Immigr~ Both     Total    999 World      999 World     999 World   143137 128641 121175  89185  88272
# i 29 more variables: `1985` <dbl>, `1986` <dbl>, `1987` <dbl>, `1988` <dbl>, `1989` <dbl>,
#   `1990` <dbl>, `1991` <dbl>, `1992` <dbl>, `1993` <dbl>, `1994` <dbl>, `1995` <dbl>, `1996` <dbl>,
#   `1997` <dbl>, `1998` <dbl>, `1999` <dbl>, `2000` <dbl>, `2001` <dbl>, `2002` <dbl>, `2003` <dbl>,
```

```
> print(data)
# A tibble: 197 x 43
   Type   Coverage OdName  AREA AreaName   REG RegName   DEV DevName `1980` `1981` `1982` `1983` `1984`
   <chr>  <chr>    <chr>  <dbl> <chr>    <dbl> <chr>   <dbl> <chr>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
 1 Immig~ Foreign~ Afgha~   935 Asia      5501 Southe~   902 Develo~     16     39     39     47     71
 2 Immig~ Foreign~ Alban~   908 Europe     925 Southe~   901 Develo~      1      0      0      0      0
 3 Immig~ Foreign~ Alger~   903 Africa     912 Northe~   902 Develo~     80     67     71     69     63
 4 Immig~ Foreign~ Ameri~   909 Oceania    957 Polyne~   902 Develo~      0      1      0      0      0
 5 Immig~ Foreign~ Andor~   908 Europe     925 Southe~   901 Develo~      0      0      0      0      0
 6 Immig~ Foreign~ Angola   903 Africa     911 Middle~   902 Develo~      1      3      6      6      4
 7 Immig~ Foreign~ Antig~   904 Latin A~   915 Caribb~   902 Develo~      0      0      0      0     42
 8 Immig~ Foreign~ Argen~   904 Latin A~   931 South ~   902 Develo~    368    426    626    241    237
 9 Immig~ Foreign~ Armen~   935 Asia       922 Wester~   902 Develo~      0      0      0      0      0
10 Immig~ Foreign~ Austr~   909 Oceania    927 Austra~   901 Develo~    702    639    484    317    317
# i 187 more rows
# i 29 more variables: `1985` <dbl>, `1986` <dbl>, `1987` <dbl>, `1988` <dbl>, `1989` <dbl>,
#   `1990` <dbl>, `1991` <dbl>, `1992` <dbl>, `1993` <dbl>, `1994` <dbl>, `1995` <dbl>, `1996` <dbl>,
#   `1997` <dbl>, `1998` <dbl>, `1999` <dbl>, `2000` <dbl>, `2001` <dbl>, `2002` <dbl>, `2003` <dbl>,
#   `2004` <dbl>, `2005` <dbl>, `2006` <dbl>, `2007` <dbl>, `2008` <dbl>, `2009` <dbl>, `2010` <dbl>,
#   `2011` <dbl>, `2012` <dbl>, `2013` <dbl>
# i Use `print(n = ...)` to see more rows
>
```

Viewing names of the columns, number of rows and columns.

```
> names(data)
 [1] "Type"     "Coverage" "OdName"   "AREA"     "AreaName" "REG"      "RegName"  "DEV"      "DevName"
[10] "1980"     "1981"     "1982"     "1983"     "1984"     "1985"     "1986"     "1987"     "1988"
[19] "1989"     "1990"     "1991"     "1992"     "1993"     "1994"     "1995"     "1996"     "1997"
[28] "1998"     "1999"     "2000"     "2001"     "2002"     "2003"     "2004"     "2005"     "2006"
[37] "2007"     "2008"     "2009"     "2010"     "2011"     "2012"     "2013"
> # no of rows
> nrow(data)
[1] 197
> print(nrow(data))
[1] 197
> # no of columns
> ncol(data)
[1] 43
```

Finding data types of each column.

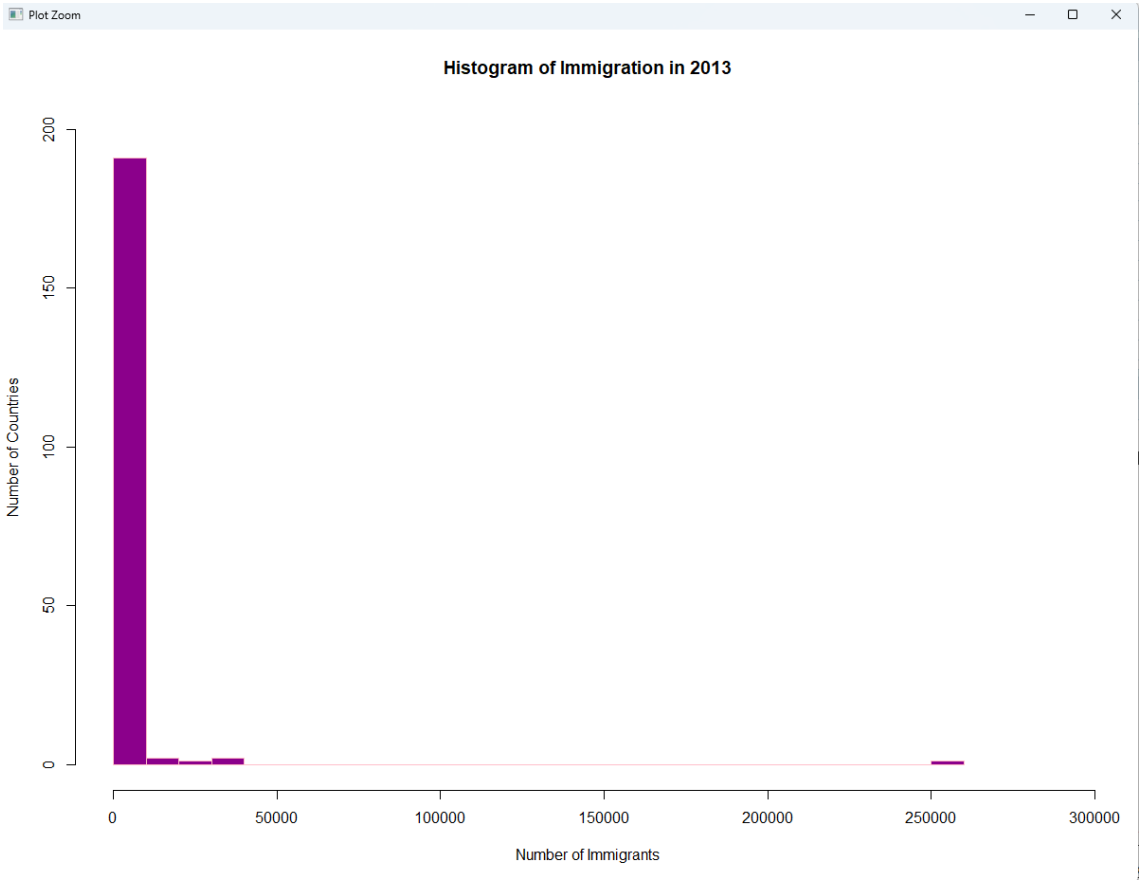
```
> # data types
> str(data)
tibble [197 x 43] (S3: tbl_df/tbl/data.frame)
 $ Type    : chr [1:197] "Immigrants" "Immigrants" "Immigrants" "Immigrants" ...
 $ Coverage: chr [1:197] "Foreigners" "Foreigners" "Foreigners" "Foreigners" ...
 $ OdName  : chr [1:197] "Afghanistan" "Albania" "Algeria" "American Samoa" ...
 $ AREA    : num [1:197] 935 908 903 909 908 903 904 904 935 909 ...
 $ AreaName: chr [1:197] "Asia" "Europe" "Africa" "Oceania" ...
 $ REG     : num [1:197] 5501 925 912 957 925 ...
 $ RegName : chr [1:197] "Southern Asia" "Southern Europe" "Northern Africa" "Polynesia" ...
 $ DEV     : num [1:197] 902 901 902 902 901 902 902 902 902 901 ...
 $ DevName : chr [1:197] "Developing regions" "Developed regions" "Developing regions" "Developing region
s" ...
 $ 1980    : num [1:197] 16 1 80 0 0 1 0 368 0 702 ...
 $ 1981    : num [1:197] 39 0 67 1 0 3 0 426 0 639 ...
 $ 1982    : num [1:197] 39 0 71 0 0 6 0 626 0 484 ...
 $ 1983    : num [1:197] 47 0 69 0 0 6 0 241 0 317 ...
 $ 1984    : num [1:197] 71 0 63 0 0 4 42 237 0 317 ...
 $ 1985    : num [1:197] 340 0 44 0 0 3 52 196 0 319 ...
 $ 1986    : num [1:197] 496 1 69 0 2 5 51 213 0 356 ...
 $ 1987    : num [1:197] 741 2 132 1 0 5 61 519 0 467 ...
 $ 1988    : num [1:197] 828 2 242 0 0 11 34 374 0 410 ...
 $ 1989    : num [1:197] 1076 3 434 1 0 ...
 $ 1990    : num [1:197] 1028 3 491 2 3 ...
 $ 1991    : num [1:197] 1378 21 872 0 0 ...
 $ 1992    : num [1:197] 1170 56 795 0 1 ...
 $ 1993    : num [1:197] 713 96 717 0 0 ...
 $ 1994    : num [1:197] 858 71 595 0 0 8 18 366 66 702 ...
 $ 1995    : num [1:197] 1537 63 1106 0 0 ...
 $ 1996    : num [1:197] 2212 113 2054 0 0 ...
 $ 1997    : num [1:197] 2555 307 1842 0 0 ...
 $ 1998    : num [1:197] 1999 574 2292 0 2 ...
 $ 1999    : num [1:197] 2395 1264 2389 0 0 ...
 $ 2000    : num [1:197] 3326 1816 2867 0 0 ...
 $ 2001    : num [1:197] 4067 1602 3418 0 1 ...
 $ 2002    : num [1:197] 3697 1021 3406 0 0 ...
 $ 2003    : num [1:197] 3479 853 3072 0 2 ...
 $ 2004    : num [1:197] 2978 1450 3616 0 0 ...
 $ 2005    : num [1:197] 3436 1223 3626 0 0 ...
 $ 2006    : num [1:197] 3009 856 4807 1 1 ...
 $ 2007    : num [1:197] 2652 702 3623 0 1 ...
 $ 2008    : num [1:197] 2111 560 4005 0 0 ...
 $ 2009    : num [1:197] 1746 716 5393 0 0 ...
 $ 2010    : num [1:197] 1758 561 4752 0 0 ...
 $ 2011    : num [1:197] 2203 539 4325 0 0 ...
 $ 2012    : num [1:197] 2635 620 3774 0 1 ...
 $ 2013    : num [1:197] 2004 603 4331 0 1 ...
>
```

Finding the minimum and maximum value of the column ‘2013’

```
> # min value
> min(data$`2013`)
[1] 0
> # max value
> max(data$`2013`)
[1] 259021
>
```

Plotting a Histogram using the column ‘2013’

```
# breaks = specify the number of cells
# limits for x and y axis
# labels for x & y axis
# Name of the graph / header
hist(data$`2013`,breaks = 20, xlim = c(0, 290000), ylim = c(0, 200),
     xlab = 'Number of Immigrants', ylab = 'Number of Countries',
     main = 'Histogram of Immigration in 2013', col = "darkmagenta", border = "pink")
```



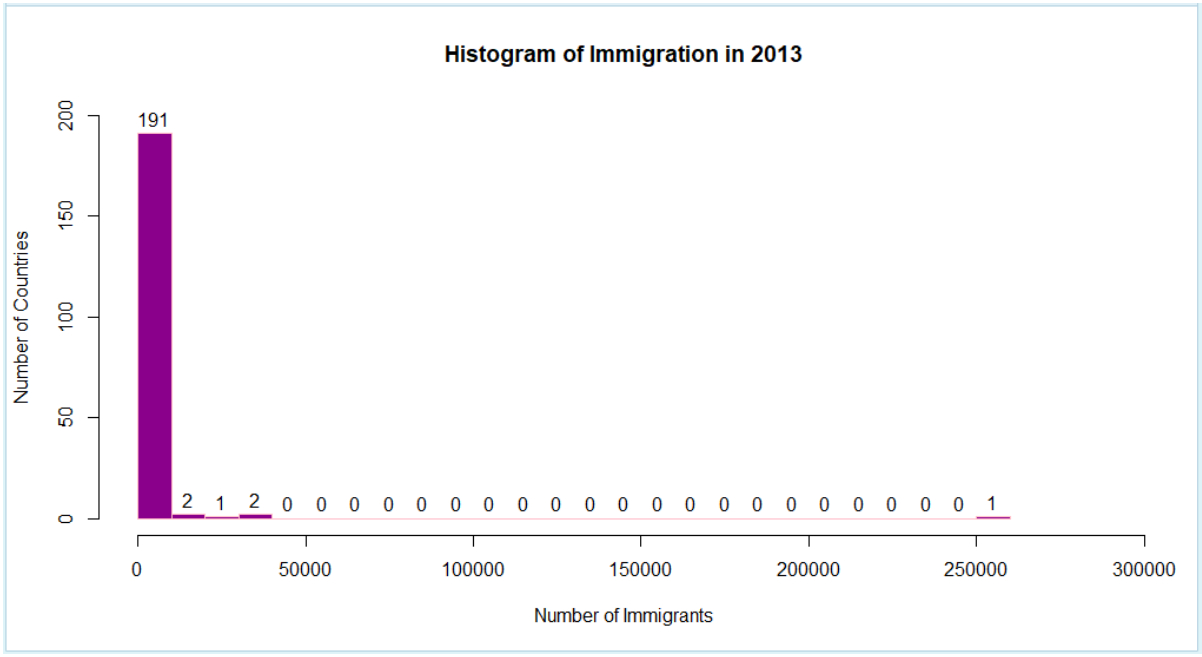
Using the hist() function to get the components; breaks, counts, density, mids, xname, equidist and attr.

```
56
57  # The hist() function returns a list with 6 components.
58  h <- hist(data$`2013`,breaks = 20, xlim = c(0, 290000), ylim = c(0, 200),
59            xlab = 'Number of Immigrants', ylab = 'Number of Countries',
60            main = 'Histogram of Immigration in 2013', col = "darkmagenta", border = "pink")
61  print(h)
62
```

```
> print(h)
$breaks
 [1]      0  10000  20000  30000  40000  50000  60000  70000  80000  90000 100000 110000 120000 130000
[15] 140000 150000 160000 170000 180000 190000 200000 210000 220000 230000 240000 250000 260000

$counts
 [1] 191   2   1   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
[25]   0   1

$density
 [1] 9.695431e-05 1.015228e-06 5.076142e-07 1.015228e-06 0.000000e+00 0.000000e+00 0.000000e+00
 [8] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
[15] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
[22] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 5.076142e-07

$mids
 [1]   5000  15000  25000  35000  45000  55000  65000  75000  85000  95000 105000 115000 125000 135000
[15] 145000 155000 165000 175000 185000 195000 205000 215000 225000 235000 245000 255000

$xname
[1] "data$`2013`"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
>
```

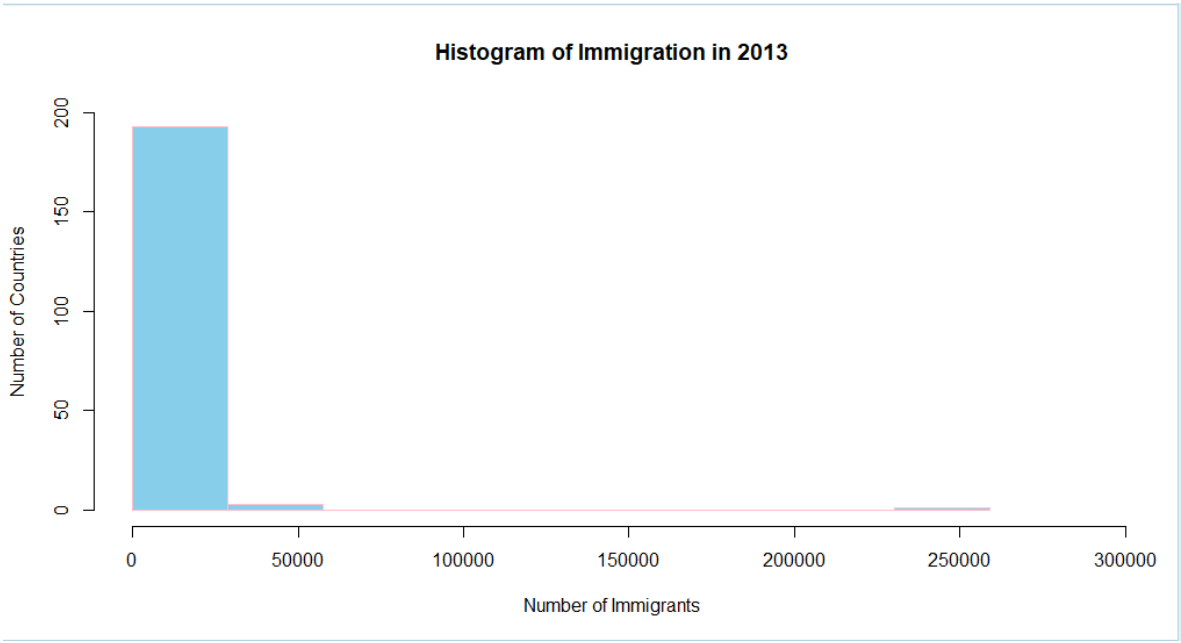
Placing the count on top of each cell.

```
64  # place the counts on top of each cell
65  text(h$mids,h$counts,labels=h$counts, adj=c(0.5, -0.5))
66
```



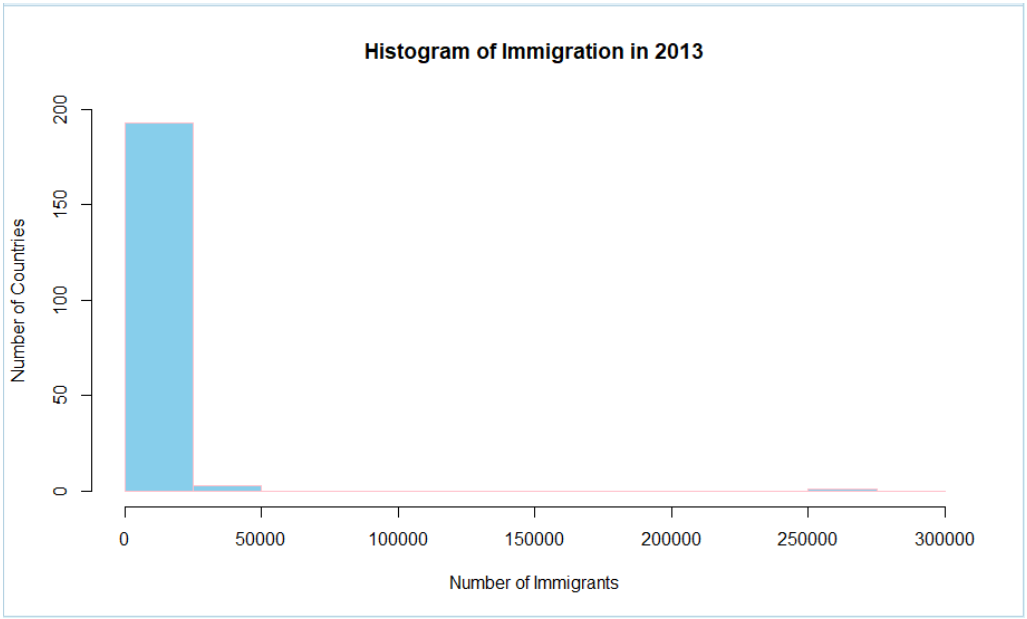
Using the break parameter using the min max values of the column ‘2013’

```
# breaks = specify the number of cells
hist(data$`2013`,breaks = seq(min(data$`2013`),max(data$`2013`), length.out = 10),
     xlim = c(0, 290000), ylim = c(0, 200),
     xlab = 'Number of Immigrants', ylab = 'Number of Countries',
     main = 'Histogram of Immigration in 2013', col = "skyblue", border = "pink")
```



Using a vector to define breakpoints.

```
# breaks = specify the number of cells
# defining breakpoints between the cells as a vector
hist(data$`2013`,breaks = c(0,25000,50000,75000,100000,125000,150000,175000,200000,225000,250000,275000,300000),
     xlim = c(0, 300000), ylim = c(0, 200),
     xlab = 'Number of Immigrants', ylab = 'Number of Countries',
     main = 'Histogram of Immigration in 2013', col = "skyblue", border = "pink")
```



c. Build linear model with the given set of data.

Use R programming for initial data analysis that explores the numerical and graphical characteristics of the data. Perform variable selection to choose the best model.

Guidelines:

1- Use comparison paragraph/chart/table with specific similarities and differences supported by examples.

2- You need to download and install R and R Studio.

Sample R Studio screen:

3- You need to refer to the relevant dataset for finding solutions for each case (part (b) and (c)

4- Program Output Screen pasted in same word file.

5- You need to refer to the relevant dataset for finding solutions for each case.

Report should include the following:

1. Comparisons paragraph/chart/table with similarities and differences amongst various predictive models.
2. Solution of the problem should be sufficiently explanatory.
3. Program code with proper code alignment in a word file. Program Output Screen pasted in same word file.
4. Share the program file.
5. References (Adhering to Harvard Referencing Format)

Linear Regression:

A statistical method for determining if there is a link between an independent variable (explanatory variable or predictor variable) and a dependent variable (response variable or outcome variable) is called linear regression (LR). A causal relationship between the independent variable (X) and the dependent variable (Y) cannot be established using regression models. Put differently, regression analysis does not allow us to claim that the value of Y determines the value of X or that a change in the value of X causes a change in the value of Y. All it can be proved is that the variation in Y's value is related to the variation in X's value. According to LR, the dependent variable and the regression coefficient have a functional relationship that is linear and there is only one independent variable in the model.

Data source: www.kaggle.com

Reading source data, Viewing dimensions of the data, Displaying the 1st five and last five rows and Viewing column names.

```
R 4.1.3 · ~/
> library(dplyr)
> datacsv <- read.csv("D:\\Placement_Data_Full_Class.csv")
> # dimension
> dim(datacsv)
[1] 215  15
> # 1st 5 rows
> head(datacsv, 5)
  sl_no gender ssc_p    ssc_b hsc_p    hsc_b     hsc_s degree_p  degree_t workex etest_p
1     1      M 67.00   Others 91.00   Others Commerce    58.00   Sci&Tech     No    55.0
2     2      M 79.33 Central 78.33   Others  Science    77.48   Sci&Tech    Yes    86.5
3     3      M 65.00 Central 68.00 Central     Arts    64.00 Comm&Mgmt     No    75.0
4     4      M 56.00 Central 52.00 Central  Science    52.00   Sci&Tech     No    66.0
5     5      M 85.80 Central 73.60 Central Commerce    73.30 Comm&Mgmt     No    96.8
  specialisation mba_p     status salary
1         Mkt&HR 58.80     Placed 270000
2        Mkt&Fin 66.28     Placed 200000
3        Mkt&Fin 57.80     Placed 250000
4         Mkt&HR 59.43 Not Placed     NA
5        Mkt&Fin 55.50     Placed 425000
> # last 5 rows
> tail(datacsv, 5)
    sl_no gender ssc_p   ssc_b hsc_p  hsc_b     hsc_s degree_p  degree_t workex etest_p
211   211      M  80.6  Others    82 Others Commerce     77.6 Comm&Mgmt     No      91
212   212      M  58.0  Others    60 Others  Science     72.0  Sci&Tech     No      74
213   213      M  67.0  Others    67 Others Commerce     73.0 Comm&Mgmt    Yes      59
214   214      F  74.0  Others    66 Others Commerce     58.0 Comm&Mgmt     No      70
215   215      M  62.0 Central    58 Others  Science     53.0 Comm&Mgmt     No      89
    specialisation mba_p     status salary
211        Mkt&Fin 74.49     Placed 400000
212        Mkt&Fin 53.62     Placed 275000
213        Mkt&Fin 69.72     Placed 295000
214         Mkt&HR 60.23     Placed 204000
215         Mkt&HR 60.22 Not Placed     NA
> # column names
> names(datacsv)
 [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"
>
```

Get summary statistics of the dataset.

```
> describe(datacsv)
               vars   n      mean       sd median   trimmed      mad       min       max     range  skew kurtosis      se
sl_no             1 215    108.00    62.21    108    108.00    80.06      1.00    215.00    214.00  0.00    -1.22    4.24
gender*           2 215      1.65     0.48      2      1.68     0.00      1.00      2.00      1.00 -0.61    -1.64    0.03
ssc_p             3 215     67.30    10.83     67     67.45    10.67     40.89     89.40     48.51 -0.13    -0.64    0.74
ssc_b*            4 215      1.46     0.50      1      1.45     0.00      1.00      2.00      1.00  0.16    -1.98    0.03
hsc_p             5 215     66.33    10.90     65     66.21     8.90     37.00     97.70     60.70  0.16     0.38    0.74
hsc_b*            6 215      1.61     0.49      2      1.64     0.00      1.00      2.00      1.00 -0.44    -1.81    0.03
hsc_s*            7 215      2.37     0.58      2      2.40     0.00      1.00      3.00      2.00 -0.28    -0.74    0.04
degree_p          8 215     66.37     7.36     66     66.20     7.56     50.00     91.00     41.00  0.24     0.00    0.50
degree_t*         9 215      1.60     0.89      1      1.50     0.00      1.00      3.00      2.00  0.87    -1.18    0.06
workex*          10 215      1.34     0.48      1      1.31     0.00      1.00      2.00      1.00  0.65    -1.58    0.03
etest_p          11 215     72.10    13.28     71     71.56    16.31     50.00     98.00     48.00  0.28    -1.11    0.91
specialisation*  12 215      1.44     0.50      1      1.43     0.00      1.00      2.00      1.00  0.23    -1.95    0.03
mba_p            13 215     62.28     5.83     62     62.07     6.23     51.21     77.89     26.68  0.31    -0.51    0.40
status*          14 215      1.69     0.46      2      1.73     0.00      1.00      2.00      1.00 -0.81    -1.35    0.03
salary           15 148 288655.41 93457.45 265000 272958.33 51891.00 200000.00 940000.00 740000.00  3.50    17.60 7682.16
> |
```

Data cleaning and formatting.

```
> # Data Cleaning & Formatting#######
> # Handling missing values
> colSums(is.na(datacsv))
         sl_no         gender          ssc_p          ssc_b          hsc_p          hsc_b          hsc_s       degree_p
            0              0              0              0              0              0              0              0
     degree_t         workex       etest_p specialisation         mba_p         status         salary
            0              0              0              0              0              0             67
> |
```

As per the above screen, 67 missing values are present in the salary column.

Imputing missing values in the salary column by mean.

```
> library(imputeTS)
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
> datacsv$salary <- imputeTS::na.mean(datacsv$salary)
Warning message:
na.mean will be replaced by na_mean.
    Functionality stays the same.
    The new function name better fits modern R code style guidelines.
    Please adjust your code accordingly.
> colSums(is.na(datacsv))
         sl_no         gender          ssc_p          ssc_b         hsc_p          hsc_b          hsc_s       degree_p
             0              0              0              0             0              0              0              0
     degree_t         workex       etest_p specialisation         mba_p         status         salary
             0              0              0              0             0              0              0
> |
```

Viewing total missing values.

```
> # Seeing missing values
> sum(colSums(is.na(datacsv)))
[1] 0
> |
```

Standardization.

```
#Standardization#########################

standardized_datacsv <- data.frame(datacsv)

colnames(datacsv)

# Create a new variable
standardized_sl_no <- scale(standardized_datacsv$sl_no)

# Print the first few values
head(standardized_sl_no)
```

```
> head(standardized_sl_no)
           [,1]
[1,] -1.719999
[2,] -1.703925
[3,] -1.687850
[4,] -1.671775
[5,] -1.655700
[6,] -1.639626
>
```

Similarly, the above same can be done for all the columns.

```
> standardized_ssc_p <- scale(standardized_datacsv$ssc_p)
> standardized_hsc_p <- scale(standardized_datacsv$hsc_p)
> standardized_degree_p <- scale(standardized_datacsv$degree_p)
> standardized_etest_p <- scale(standardized_datacsv$etest_p)
> standardized_mba_p <- scale(standardized_datacsv$mba_p)
> standardized_salary <- scale(standardized_datacsv$salary)
>
```

Combining newly created columns to a new data frame.

```
> updated_datacsv <- NULL
> updated_datacsv <- cbind(datacsv, standardized_sl_no)
> updated_datacsv <- cbind(updated_datacsv, standardized_ssc_p)
> updated_datacsv <- cbind(updated_datacsv, standardized_hsc_p)
> updated_datacsv <- cbind(updated_datacsv, standardized_degree_p)
> updated_datacsv <- cbind(updated_datacsv, standardized_etest_p)
> updated_datacsv <- cbind(updated_datacsv, standardized_mba_p)
> updated_datacsv <- cbind(updated_datacsv, standardized_salary)
> head(updated_datacsv)
  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 standardized_sl_no standardized_ssc_p standardized_hsc_p
1     1      M 67.00   Others 91.00   Others Commerce    58.00   Sci&Tech     No    55.0         Mkt&HR 58.80     Placed 270000.0          -1.719999         -0.02802158          2.2635298
2     2      M 79.33 Central 78.33   Others  Science    77.48   Sci&Tech    Yes    86.5        Mkt&Fin 66.28     Placed 200000.0          -1.703925          1.11077644          1.1008788
3     3      M 65.00 Central 68.00 Central     Arts    64.00 Comm&Mgmt     No    75.0        Mkt&Fin 57.80     Placed 250000.0          -1.687850         -0.21274145          0.1529558
4     4      M 56.00 Central 52.00 Central  Science    52.00   Sci&Tech     No    66.0         Mkt&HR 59.43 Not Placed 288655.4          -1.671775         -1.04398087         -1.3152696
5     5      M 85.80 Central 73.60 Central Commerce    73.30 Comm&Mgmt     No    96.8        Mkt&Fin 55.50     Placed 425000.0          -1.655700          1.70834523          0.6668347
6     6      M 55.00   Others 49.80   Others  Science    67.25   Sci&Tech    Yes    55.0        Mkt&Fin 51.58 Not Placed 288655.4          -1.639626         -1.13634081         -1.5171506
  standardized_degree_p standardized_etest_p standardized_mba_p standardized_salary
1            -1.1374478           -1.2880848         -0.5962552          -0.2408457
2             1.5097434            1.0846256          0.6860192          -1.1445625
3            -0.3220911            0.2183980         -0.7676823          -0.4990505
4            -1.9528044           -0.4595193         -0.4882562           0.0000000
5             0.9417116            1.8604642         -1.1619645           1.7602413
6             0.1195604           -1.2880848         -1.8339586           0.0000000
>
```

Normalization.

```
> normalized_datacsv <- data.frame(updated_datacsv)
> normalized_sl_no <- (normalized_datacsv$sl_no - min(normalized_datacsv$sl_no)) / (max(normalized_datacsv$sl_no) - min(normalized_datacsv$sl_no))
> # Print the first few values
> head(normalized_sl_no)
[1] 0.000000000 0.004672897 0.009345794 0.014018692 0.018691589 0.023364486
>
```

```
> normalized_ssc_p <- (normalized_datacsv$ssc_p - min(normalized_datacsv$ssc_p)) / (max(normalized_datacsv$ssc_p) - min(normalized_datacsv$ssc_p))
> normalized_hsc_p <- (normalized_datacsv$hsc_p - min(normalized_datacsv$hsc_p)) / (max(normalized_datacsv$hsc_p) - min(normalized_datacsv$hsc_p))
> normalized_degree_p <- (normalized_datacsv$degree_p - min(normalized_datacsv$degree_p)) / (max(normalized_datacsv$degree_p) - min(normalized_datacsv$degree_p))
> normalized_etest_p <- (normalized_datacsv$etest_p - min(normalized_datacsv$etest_p)) / (max(normalized_datacsv$etest_p) - min(normalized_datacsv$etest_p))
> normalized_mba_p <- (normalized_datacsv$mba_p - min(normalized_datacsv$mba_p)) / (max(normalized_datacsv$mba_p) - min(normalized_datacsv$mba_p))
> normalized_salary <- (normalized_datacsv$salary - min(normalized_datacsv$salary)) / (max(normalized_datacsv$salary) - min(normalized_datacsv$salaryo))
```

Combining newly created columns to a newly created data frame.

```
> updated_datacsv_normalized <- NULL
> updated_datacsv_normalized <- cbind(updated_datacsv, normalized_sl_no)
```

```
> updated_datacsv_normalized <- cbind(updated_datacsv, normalized_ssc_p)
> updated_datacsv_normalized <- cbind(updated_datacsv, normalized_hsc_p)
> updated_datacsv_normalized <- cbind(updated_datacsv, normalized_degree_p)
> updated_datacsv_normalized <- cbind(updated_datacsv, normalized_etest_p)
> updated_datacsv_normalized <- cbind(updated_datacsv, normalized_mba_p)
> updated_datacsv_normalized <- cbind(updated_datacsv, normalized_salary)
> head(updated_datacsv_normalized)
  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 standardized_sl_no standardized_ssc_p standardized_hsc_p
1     1      M 67.00  Others 91.00  Others Commerce    58.00  Sci&Tech     No    55.0         Mkt&HR 58.80     Placed 270000.0          -1.719999         -0.02802158           2.2635298
2     2      M 79.33 Central 78.33  Others  Science    77.48  Sci&Tech    Yes    86.5        Mkt&Fin 66.28     Placed 200000.0          -1.703925          1.11077644           1.1008788
3     3      M 65.00 Central 68.00 Central     Arts    64.00 Comm&Mgmt     No    75.0        Mkt&Fin 57.80     Placed 250000.0          -1.687850         -0.21274145           0.1529558
4     4      M 56.00 Central 52.00 Central  Science    52.00  Sci&Tech     No    66.0         Mkt&HR 59.43 Not Placed 288655.4          -1.671775         -1.04398087          -1.3152696
5     5      M 85.80 Central 73.60 Central Commerce    73.30 Comm&Mgmt     No    96.8        Mkt&Fin 55.50     Placed 425000.0          -1.655700          1.70834523           0.6668347
6     6      M 55.00  Others 49.80  Others  Science    67.25  Sci&Tech    Yes    55.0        Mkt&Fin 51.58 Not Placed 288655.4          -1.639626         -1.13634081          -1.5171506
  standardized_degree_p standardized_etest_p standardized_mba_p standardized_salary normalized_salary
1            -1.1374478           -1.2880848         -0.5962552          -0.2408457        0.09459459
2             1.5097434            1.0846256          0.6860192          -1.1445625        0.00000000
3            -0.3220911            0.2183980         -0.7676823          -0.4990505        0.06756757
4            -1.9528044           -0.4595193         -0.4882562           0.0000000        0.11980460
5             0.9417116            1.8604642         -1.1619645           1.7602413        0.30405405
6             0.1195604           -1.2880848         -1.8339586           0.0000000        0.11980460
>
```

Checking data types of newly added columns.

```
> # data types
> str(updated_datacsv_normalized)
'data.frame':   215 obs. of  23 variables:
 $ sl_no                : int  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               : chr  "No" "Yes" "No" "No" ...
 $ 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 288655 425000 ...
 $ standardized_sl_no   : num  -1.72 -1.7 -1.69 -1.67 -1.66 ...
 $ standardized_ssc_p   : num  -0.028 1.111 -0.213 -1.044 1.708 ...
 $ standardized_hsc_p   : num  2.264 1.101 0.153 -1.315 0.667 ...
 $ standardized_degree_p: num  -1.137 1.51 -0.322 -1.953 0.942 ...
 $ standardized_etest_p : num  -1.288 1.085 0.218 -0.46 1.86 ...
 $ standardized_mba_p   : num  -0.596 0.686 -0.768 -0.488 -1.162 ...
 $ standardized_salary  : num  -0.241 -1.145 -0.499 0 1.76 ...
 $ normalized_salary    : num  0.0946 0 0.0676 0.1198 0.3041 ...
>
```

One hot encoding: converting categorical variables as binary vectors.

```
> # Create a new data frame with the one-hot encoded variables
> encoded_datacsv_ohe <- NULL
> encoded_datacsv_ohe_gender <- as.data.frame(model.matrix(~ gender - 1, updated_datacsv_normalized))
> # Add the new columns to the encoded data frame
> encoded_datacsv_ohe <- cbind(encoded_datacsv_ohe_gender, updated_datacsv_normalized)
> # Print the first few rows of the encoded data frame
> head(encoded_datacsv_ohe)
  genderF genderM 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 standardized_sl_no standardized_ssc_p
1       0       1     1      M 67.00  Others 91.00  Others Commerce    58.00  Sci&Tech     No    55.0         Mkt&HR 58.80     Placed 270000.0          -1.719999        -0.02802158
2       0       1     2      M 79.33 Central 78.33  Others  Science    77.48  Sci&Tech    Yes    86.5        Mkt&Fin 66.28     Placed 200000.0          -1.703925         1.11077644
3       0       1     3      M 65.00 Central 68.00 Central     Arts    64.00 Comm&Mgmt     No    75.0        Mkt&Fin 57.80     Placed 250000.0          -1.687850        -0.21274145
4       0       1     4      M 56.00 Central 52.00 Central  Science    52.00  Sci&Tech     No    66.0         Mkt&HR 59.43 Not Placed 288655.4          -1.671775        -1.04398087
5       0       1     5      M 85.80 Central 73.60 Central Commerce    73.30 Comm&Mgmt     No    96.8        Mkt&Fin 55.50     Placed 425000.0          -1.655700         1.70834523
6       0       1     6      M 55.00  Others 49.80  Others  Science    67.25  Sci&Tech    Yes    55.0        Mkt&Fin 51.58 Not Placed 288655.4          -1.639626        -1.13634081
  standardized_hsc_p standardized_degree_p standardized_etest_p standardized_mba_p standardized_salary normalized_salary
1          2.2635298            -1.1374478           -1.2880848         -0.5962552          -0.2408457        0.09459459
2          1.1008788             1.5097434            1.0846256          0.6860192          -1.1445625        0.00000000
3          0.1529558            -0.3220911            0.2183980         -0.7676823          -0.4990505        0.06756757
4         -1.3152696            -1.9528044           -0.4595193         -0.4882562           0.0000000        0.11980460
5          0.6668347             0.9417116            1.8604642         -1.1619645           1.7602413        0.30405405
6         -1.5171506             0.1195604           -1.2880848         -1.8339586           0.0000000        0.11980460
```

Similarly, it is possible to apply the same mechanism for other categories as well.

```
> encoded_datacsv_ohe_ssc_b <- as.data.frame(model.matrix(~ ssc_b - 1, updated_datacsv_normalized))
> encoded_datacsv_ohe <- cbind(encoded_datacsv_ohe_ssc_b, encoded_datacsv_ohe)
> encoded_datacsv_ohe_hsc_b <- as.data.frame(model.matrix(~ hsc_b - 1, updated_datacsv_normalized))
> encoded_datacsv_ohe <- cbind(encoded_datacsv_ohe_hsc_b, encoded_datacsv_ohe)
> encoded_datacsv_ohe_hsc_s <- as.data.frame(model.matrix(~ hsc_s - 1, updated_datacsv_normalized))
> encoded_datacsv_ohe <- cbind(encoded_datacsv_ohe_hsc_s, encoded_datacsv_ohe)
> encoded_datacsv_ohe_degree_t <- as.data.frame(model.matrix(~ degree_t - 1, updated_datacsv_normalized))
> encoded_datacsv_ohe <- cbind(encoded_datacsv_ohe_degree_t, encoded_datacsv_ohe)
> encoded_datacsv_ohe_workex <- as.data.frame(model.matrix(~ workex - 1, updated_datacsv_normalized))
> encoded_datacsv_ohe <- cbind(encoded_datacsv_ohe_workex, encoded_datacsv_ohe)
> encoded_datacsv_ohe_specialisation <- as.data.frame(model.matrix(~ specialisation - 1, updated_datacsv_normalized))
> encoded_datacsv_ohe <- cbind(encoded_datacsv_ohe_specialisation, encoded_datacsv_ohe)
> encoded_datacsv_ohe_status <- as.data.frame(model.matrix(~ status - 1, updated_datacsv_normalized))
> encoded_datacsv_ohe <- cbind(encoded_datacsv_ohe_status, encoded_datacsv_ohe)
> head(encoded_datacsv_ohe)
  statusNot Placed statusPlaced specialisationMkt&Fin specialisationMkt&HR workexNo workexYes degree_tComm&Mgmt degree_tOthers degree_tSci&Tech hsc_sArts hsc_sCommerce hsc_sScience
1                0            1                     0                    1        1         0                 0              0                1         0              1             0
2                0            1                     1                    0        0         1                 0              0                1         0              0             1
3                0            1                     1                    0        1         0                 1              0                0         1              0             0
4                1            0                     0                    1        1         0                 0              0                1         0              0             1
5                0            1                     1                    0        1         0                 1              0                0         0              1             0
6                1            0                     1                    0        0         1                 0              0                1         0              0             1
  hsc_bCentral hsc_bOthers ssc_bCentral ssc_bOthers genderF genderM sl_no gender ssc_p   ssc_b hsc_p   hsc_b    hsc_s degree_p  degree_t workex etest_p specialisation mba_p     status
1            0           1            0           1       0       1     1      M 67.00  Others 91.00  Others Commerce    58.00  Sci&Tech     No    55.0         Mkt&HR 58.80     Placed
2            0           1            1           0       0       1     2      M 79.33 Central 78.33  Others  Science    77.48  Sci&Tech    Yes    86.5        Mkt&Fin 66.28     Placed
3            1           0            1           0       0       1     3      M 65.00 Central 68.00 Central     Arts    64.00 Comm&Mgmt     No    75.0        Mkt&Fin 57.80     Placed
4            1           0            1           0       0       1     4      M 56.00 Central 52.00 Central  Science    52.00  Sci&Tech     No    66.0         Mkt&HR 59.43 Not Placed
5            1           0            1           0       0       1     5      M 85.80 Central 73.60 Central Commerce    73.30 Comm&Mgmt     No    96.8        Mkt&Fin 55.50     Placed
6            0           1            0           1       0       1     6      M 55.00  Others 49.80  Others  Science    67.25  Sci&Tech    Yes    55.0        Mkt&Fin 51.58 Not Placed
    salary standardized_sl_no standardized_ssc_p standardized_hsc_p standardized_degree_p standardized_etest_p standardized_mba_p standardized_salary normalized_salary
1 270000.0          -1.719999        -0.02802158          2.2635298            -1.1374478          -1.2880848        -0.5962552          -0.2408457        0.09459459
2 200000.0          -1.703925         1.11077644          1.1008788             1.5097434           1.0846256         0.6860192          -1.1445625        0.00000000
3 250000.0          -1.687850        -0.21274145          0.1529558            -0.3220911           0.2183980        -0.7676823          -0.4990505        0.06756757
4 288655.4          -1.671775        -1.04398087         -1.3152696            -1.9528044          -0.4595193        -0.4882562           0.0000000        0.11980460
5 425000.0          -1.655700         1.70834523          0.6668347             0.9417116           1.8604642        -1.1619645           1.7602413        0.30405405
6 288655.4          -1.639626        -1.13634081         -1.5171506             0.1195604          -1.2880848        -1.8339586           0.0000000        0.11980460
>
```

Ordinal encoding - retain the ordinal relationship among the categories when converting categorical data with ordered levels into numerical values.

Using ordinal encoding technique with specialization column.

```
> unique(encoded_datacsv_ohe$specialisation)
[1] "Mkt&HR"  "Mkt&Fin"
> # Create a copy of the original data frame
> encoded_datacsv_ohe_oe <- NULL
> encoded_datacsv_ohe_oe <- encoded_datacsv_ohe
> # Define the mapping of categories to numerical values
> mapping <- c("Mkt&HR" = 0, "Mkt&Fin" = 1)
> # Apply the ordinal encoding
> encoded_datacsv_ohe_oe$specialisation_oe <- NULL
> encoded_datacsv_ohe_oe$specialisation_oe <- mapping[as.character(encoded_datacsv_ohe_oe$specialisation)]
```

```
> # adding new column
> encoded_datacsv_ohe <- cbind(encoded_datacsv_ohe_oe$specialisation_oe, encoded_datacsv_ohe_oe)
> # Print the first few rows of the encoded data frame
> head(encoded_datacsv_ohe)
  encoded_datacsv_ohe_oe$specialisation_oe statusNot Placed statusPlaced specialisationMkt&Fin specialisationMkt&HR workexNo workexYes degree_tComm&Mgmt degree_tOthers
1                                        0                0           1                     0                     1        1         0                    0              0
2                                        1                0           1                     1                     0        0         1                    0              0
3                                        1                0           1                     1                     0        1         0                    1              0
4                                        0                1           0                     0                     1        1         0                    0              0
5                                        1                0           1                     1                     0        1         0                    1              0
6                                        1                1           0                     1                     0        0         1                    0              0
  degree_tSci&Tech hsc_sArts hsc_sCommerce hsc_sScience hsc_bCentral hsc_bOthers ssc_bCentral ssc_bOthers genderF genderM sl_no gender ssc_p    ssc_b hsc_p   hsc_b    hsc_s degree_p
1                1         0             1           0           0           1           0           1       0       1     1      M 67.00  Others 91.00  Others Commerce    58.00
2                1         0             0           1           0           1           1           0       0       1     2      M 79.33 Central 78.33  Others  Science    77.48
3                0         1             0           0           1           0           1           0       0       1     3      M 65.00 Central 68.00 Central     Arts    64.00
4                1         0             0           1           1           0           1           0       0       1     4      M 56.00 Central 52.00 Central  Science    52.00
5                0         0             1           0           1           0           1           0       0       1     5      M 85.80 Central 73.60 Central Commerce    73.30
6                1         0             0           1           0           1           0           1       0       1     6      M 55.00  Others 49.80  Others  Science    67.25
   degree_t workex etest_p specialisation mba_p     status   salary standardized_sl_no standardized_ssc_p standardized_hsc_p standardized_degree_p standardized_etest_p
1  Sci&Tech     No    55.0         Mkt&HR 58.80     Placed 270000.0          -1.719999        -0.02802158          2.2635298           -1.1374478          -1.2880848
2  Sci&Tech    Yes    86.5        Mkt&Fin 66.28     Placed 200000.0          -1.703925         1.11077644          1.1008788            1.5097434           1.0846256
3 Comm&Mgmt     No    75.0        Mkt&Fin 57.80     Placed 250000.0          -1.687850        -0.21274145          0.1529558           -0.3220911           0.2183980
4  Sci&Tech     No    66.0         Mkt&HR 59.43 Not Placed 288655.4          -1.671775        -1.04398087         -1.3152696           -1.9528044          -0.4595193
5 Comm&Mgmt     No    96.8        Mkt&Fin 55.50     Placed 425000.0          -1.655700         1.70834523          0.6668347            0.9417116           1.8604642
6  Sci&Tech    Yes    55.0        Mkt&Fin 51.58 Not Placed 288655.4          -1.639626        -1.13634081         -1.5171506            0.1195604          -1.2880848
  standardized_mba_p standardized_salary normalized_salary specialisation_oe
1         -0.5962552          -0.2408457        0.09459459                0
2          0.6860192          -1.1445625        0.00000000                1
3         -0.7676823          -0.4990505        0.06756757                1
4         -0.4882562           0.0000000        0.11980460                0
5         -1.1619645           1.7602413        0.30405405                1
6         -1.8339586           0.0000000        0.11980460                1
>
```

Finding the length of the data points (both lengths should be equal) and set the x and y variables.

```
> # check the length of the data points
> length(encoded_datacsv_ohe$degree_p)
[1] 215
> length(encoded_datacsv_ohe$mba_p)
[1] 215
> # x = independant variable
> # y = dependant variable
> x <- encoded_datacsv_ohe$degree_p
> y <- encoded_datacsv_ohe$mba_p
>
```

Finding the correlation using the Peason's method,

```
> # correlation
> # the default correlation method (Pearson) is been used because the two parameters contain numerical data
> cor(x,y)
[1] 0.4023638
>
```

Correlation Methods:

Pearson's correlation: This is a widely used correlation statistic to measure the degree of the relationship between linearly related variables.

Formular of the Pearson's correlation:

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}$$

r_{xy} = Pearson r correlation coefficient between x and y

n = number of observations

x_i = value of x (for ith observation)

y_i = value of y (for ith observation)

Image Source: www.statisticssolutions.com

Spearman's correlation:

This non-parametric test assesses the strength of the relationship between two variables. When the variables are measured on a scale that is at least ordinal, the Spearman rank correlation test is the proper correlation analysis because it makes no assumptions about the distribution of the data.

Formular of the Spearman's correlation:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

Image Source: www.statisticssolutions.com

P = Spearman correlation

di = the difference between the ranks of corresponding variables

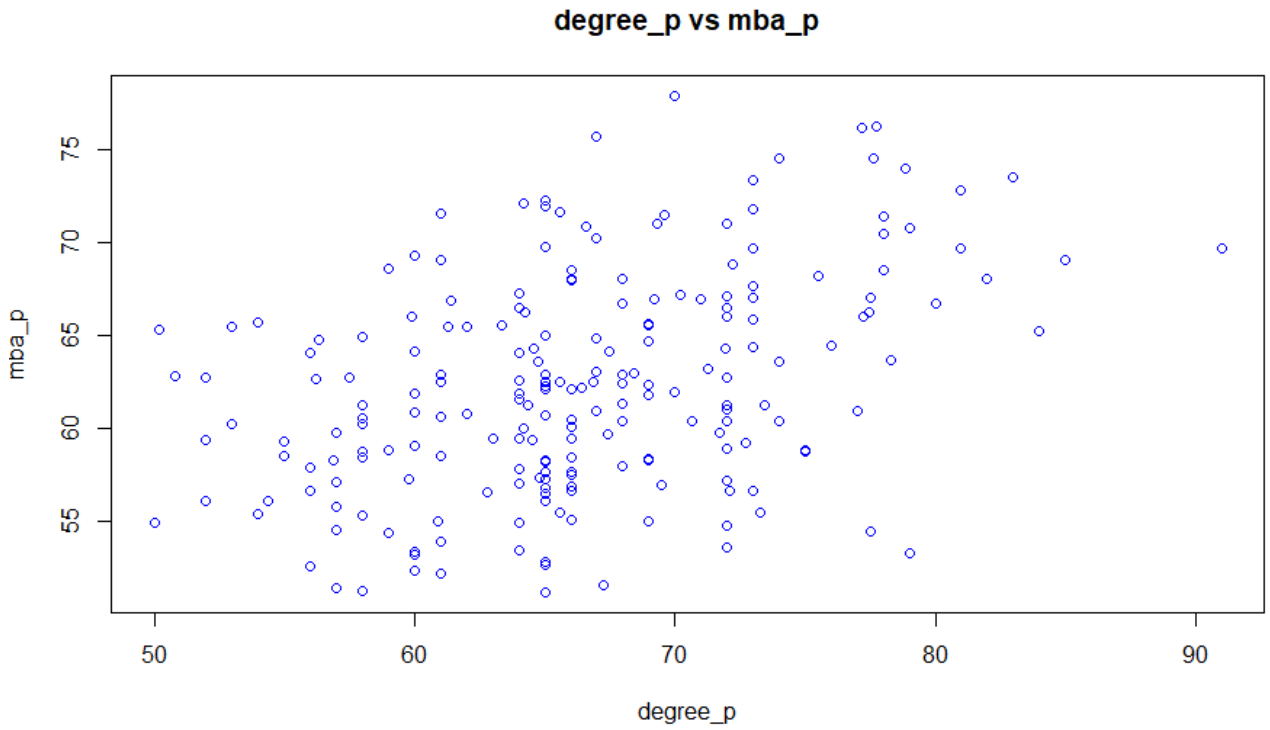
n = number of observations

Assumptions of the two correlation methods:

Pearson's correlation	Spearman's correlation
For quantitative data.	For ordinal data.
It is expected that both variables have a normal distribution, which is characterized by a bell-shaped curve.	Both variables are not normally distributed.
There is homoscedasticity, or equal distribution of the data around the regression line.	Data is not equally distributed.
There should be a straight-line, or linear, relationship between two variables.	One variable's scores must have a monotonic relationship with the other. (and the two variables' non-linear relationship)

Plotting the model.

```
> # scatter plot
> plot(x,y, main="degree_p vs mba_p", xlab="degree_p", ylab="mba_p", col="blue", cex=1)
>
```



Printing coefficients.

```
> print(model_lr)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     41.109        0.319

>
```

Viewing the statistical summary of the model.

```
> # summary of the model
> summary(model_lr)

Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-13.0166  -3.9567  -0.0328   3.6580  14.4540

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 41.10877    3.32040  12.381  < 2e-16 ***
x            0.31896    0.04973   6.414 8.99e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.353 on 213 degrees of freedom
Multiple R-squared:  0.1619,    Adjusted R-squared:  0.158
F-statistic: 41.15 on 1 and 213 DF,  p-value: 8.993e-10
```

Summary explanation:

Call Formular:

y = target/response variable / dependent variable / "degree_p"

x = predictor / independent variable / "mba_p"

Residuals:

In essence, residuals represent the discrepancy between the response values that the model anticipated and the actual observed response values. The model output is broken down into 5 summary points in the Residuals section. It is advised to search for a symmetrical distribution across these points on the mean value zero (0) when evaluating how well the model fit the data. The distribution of the residuals does not seem to be significantly symmetrical, as seen in the screenshot above. In other words, the model forecasts some points that deviate significantly from the actual observed spots.

Coefficients:

The intercept and slope terms in the linear model are represented by the two unknown constants in simple linear regression, which are called coefficients.

Estimate:

This contains two rows.

1. **Intercept**: 41.10877

   When we consider the average mba_p of every row in the dataset, this is basically the predicted value of the degree_p.

2. **Slope:** 0.31896

   According to this model's slope term, the required degree_p increases by 0.31896 for every 1 mba_p increase.

Standard Error:

The average difference between the coefficient estimates and the response variable's actual average value (degree_p) is measured by the standard error. It would be nice to have a smaller standard error in relation to its coefficients. One may say that there is a 0.04973 variation in the necessary degree_p. Additionally, the Standard Errors can be used to statistically test the hypothesis that there is a link between x and y (mba_p and degree_p) and to compute confidence intervals.

t value:

The number of standard deviations that separate our coefficient estimate from zero is shown by the coefficient t-value. A t number that is far from zero would be preferable since it would suggest that the null hypothesis may be rejected. This indicates that a relationship between x and y (mba_p and degree_p) can be declared. The screenshot above indicates that the t values are not very close to zero. In general, p-values are also calculated using t-values.

Pr(> |t|):

This shows that there is a chance of seeing any value that is larger than or equal to t. A low p-value suggests that there is little chance of a relationship between the response (y or degree_p) and the predictor (x or mba_p) variables being observed. A decent cut-off point is usually a p-value of 5% (0.05) or less. The p-values in the model mentioned above are quite near to zero. Each estimate's "signif. Codes" state that a p-value with three stars (or asterisks) indicates that it is very significant. Thus, a small p-value for the slope and the intercept suggests that the null hypothesis can be rejected, allowing for the conclusion that x and y, or mba_p and degree_p, are related.

Residual Standard Error:

This measures how well a linear regression fit was made. It is typically considered that every linear regression model has an error term (E), and that this error term makes it challenging to predict the response variable (degree_p) from the predictor (mba_p) with perfect accuracy. One can use the Residual Standard Error to get out of this predicament. The degree_p represents the average deviation of the response from the actual regression line. The actual degree_p of deviation from the correct regression line, on average, is 5.353, as per the model mentioned above, with the Residual Standard error. Stated differently, assuming that the residual standard error is 5.353 and the mean distance is 41.10877. Additionally, 213 degrees of freedom are used to determine the residual standard error. This indicates that 213 data points were utilized to estimate the parameters after intercept and slope factors were taken into account.

Multiple R-squared, Adjusted R-squared:

An indicator of how well the model fits the real data is the R-squared. It is a measurement of the linear relationship that exists between the response/target variable (degree_p) and the predictor variable (mba_p). R-squared, on the other hand, shows how each data point's variation from the mean or from the dependent variable (called degree_p) is explained by the independent variable (called mba_p). It is always in the range of 0 to 1. When the response variable's observed variation is well explained by a regression, a regression that is close to 0 is represented by a value that is close to 1. According to the summary above, the predictor variable (mba_p) could be accountable for roughly 16% of the variance seen in the responder variable (degree_p), or R-squared = 0.16. But in this model, the R-squared value is not all that strong.

F-Statistic:

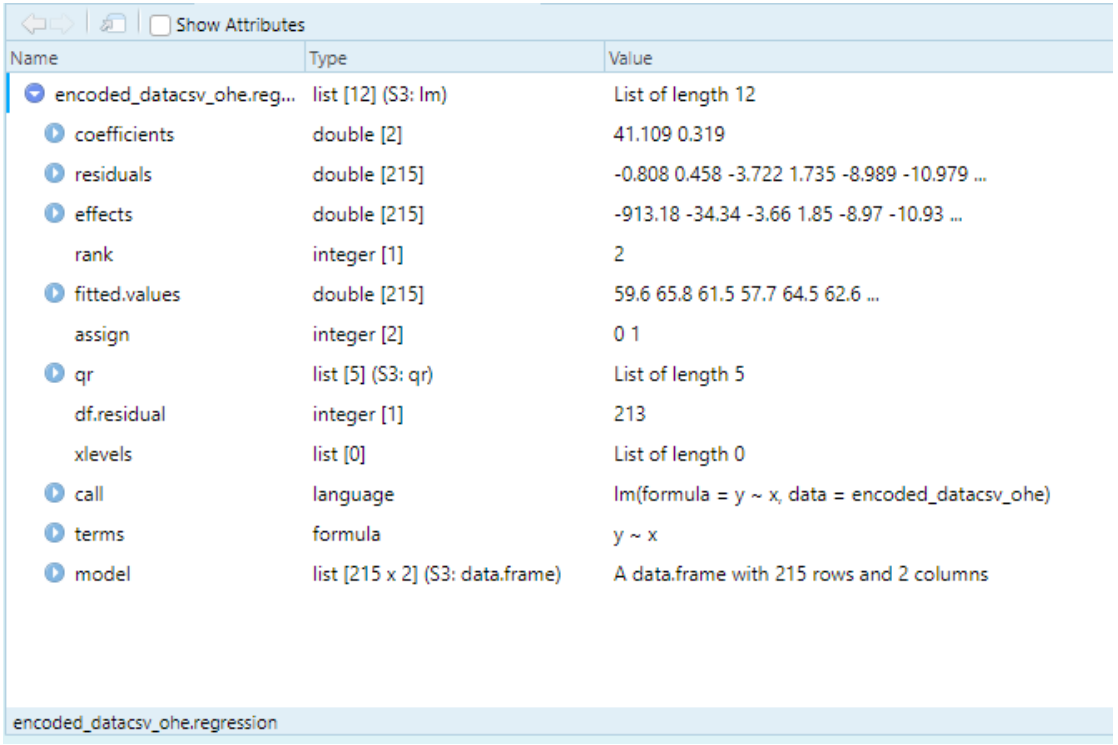
This is a useful tool for determining whether the response variables and the predictor have a relationship; the closer the F-statistic is to 1, the stronger the connection is. However, the number of data points and predictors determines how much greater the F-statistic needs to be. An F-statistic that is only marginally greater than 1 is typically enough to reject the null hypothesis when there are a lot of data points. On the other hand, a high F-Statistic value is necessary to determine whether or not there may be a relationship between the predictor and response variables when there are few datapoints. Considering the amount of our data, the F-statistic in the model mentioned above is 41.15, which is comparatively greater than 1.

Using data frame,

```
> # linear regression model / data fram
> encoded_datacsv_ohe.regression <- lm(y ~ x, data = encoded_datacsv_ohe)
> # summary
> summary(encoded_datacsv_ohe.regression)

Call:
lm(formula = y ~ x, data = encoded_datacsv_ohe)

Residuals:
     Min       1Q   Median       3Q      Max
-13.0166  -3.9567  -0.0328   3.6580  14.4540

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 41.10877    3.32040  12.381  < 2e-16 ***
x            0.31896    0.04973   6.414 8.99e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.353 on 213 degrees of freedom
Multiple R-squared:  0.1619,    Adjusted R-squared:  0.158
F-statistic: 41.15 on 1 and 213 DF,  p-value: 8.993e-10

>
```

Showing attributes’ details,

```
> # view attribute details
> View(encoded_datacsv_ohe.regression)
>
```



For instance, this data frame, which measures 215 by 2, has fitted values, which are the predicted values, residuals, which are the differences between the actual and projected values, etc.

Displaying only the two columns; degree_p (x) and mba_p (y),

```
> # data frame
> encoded_datacsv_ohe.data <- data.frame(y, x)
> # viewing only the data frame / x and y columns
> View(encoded_datacsv_ohe.data)
>
```

Filter

	y	x
1	58.80	58.00
2	66.28	77.48
3	57.80	64.00
4	59.43	52.00
5	55.50	73.30
6	51.58	67.25
7	53.29	79.00
8	62.14	66.00
9	61.29	72.00
10	52.21	61.00
11	60.85	60.00
12	63.70	78.30
13	65.04	65.00
14	68.63	59.00
15	54.96	50.00

Showing 1 to 16 of 215 entries, 2 total columns

Adding the residuals column,

```
> # adding the residuals column
> encoded_datacsv_ohe.data2 <- encoded_datacsv_ohe.data
> encoded_datacsv_ohe.data2$residuals <- encoded_datacsv_ohe.regression$residuals
> # viewing 3 columns
> head(encoded_datacsv_ohe.data2)
      y     x   residuals
1 58.80 58.00  -0.8084339
2 66.28 77.48   0.4582309
3 57.80 64.00  -3.7221922
4 59.43 52.00   1.7353243
5 55.50 73.30  -8.9885175
6 51.58 67.25 -10.9788112
>
```

Adding predicted values (fitted values) column,

```
> # adding predicted values column
> encoded_datacsv_ohe.data2$fitted.values <- encoded_datacsv_ohe.regression$fitted.values
> # viewing 4 columns
> head(encoded_datacsv_ohe.data2)
      y     x   residuals fitted.values
1 58.80 58.00  -0.8084339      59.60843
2 66.28 77.48   0.4582309      65.82177
3 57.80 64.00  -3.7221922      61.52219
4 59.43 52.00   1.7353243      57.69468
5 55.50 73.30  -8.9885175      64.48852
6 51.58 67.25 -10.9788112      62.55881
> |
```

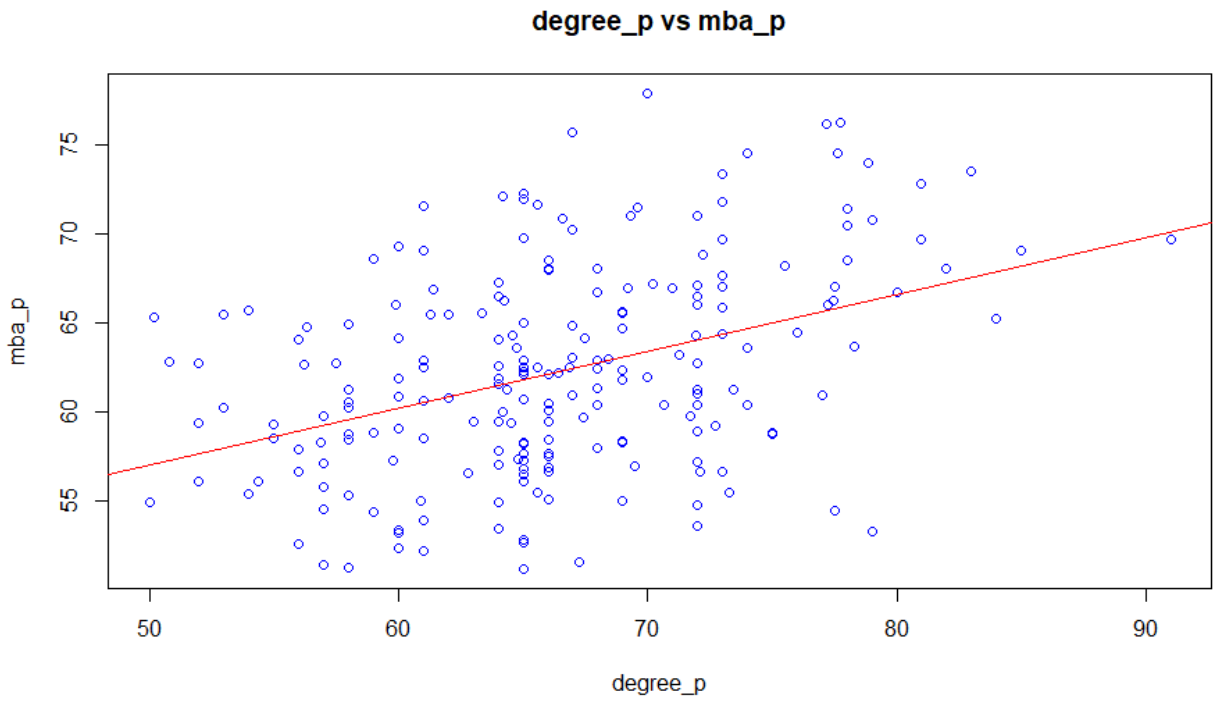
Finding the confidence levels and confidence intervals,

```
> # check confidance intervals
> confint(encoded_datacsv_ohe.regression)
                 2.5 %     97.5 %
(Intercept) 34.5637265 47.6538152
x            0.2209433  0.4169761
> |
```

It can be observed that the slope will lie from 0.2209433 to 0.4169761 for 97.5 percentage of time.

Adding the regression line.

```
> # add regression line
> abline(encoded_datacsv_ohe.regression, col="red")
> |
```



The majority of the data points are off the regression line when looking at the above result. As a result, the model is lacking the capacity to generate predictions.

Re-generating the model with another variables: hsc_p and ssc_p

```
> # check the length of the data points
> length(encoded_datacsv_ohe$hsc_p)
[1] 215
> length(encoded_datacsv_ohe$ssc_p)
[1] 215
> # x = independant variable
> # y = dependant variable
> x <- encoded_datacsv_ohe$hsc_p
> y <- encoded_datacsv_ohe$ssc_p
>
```

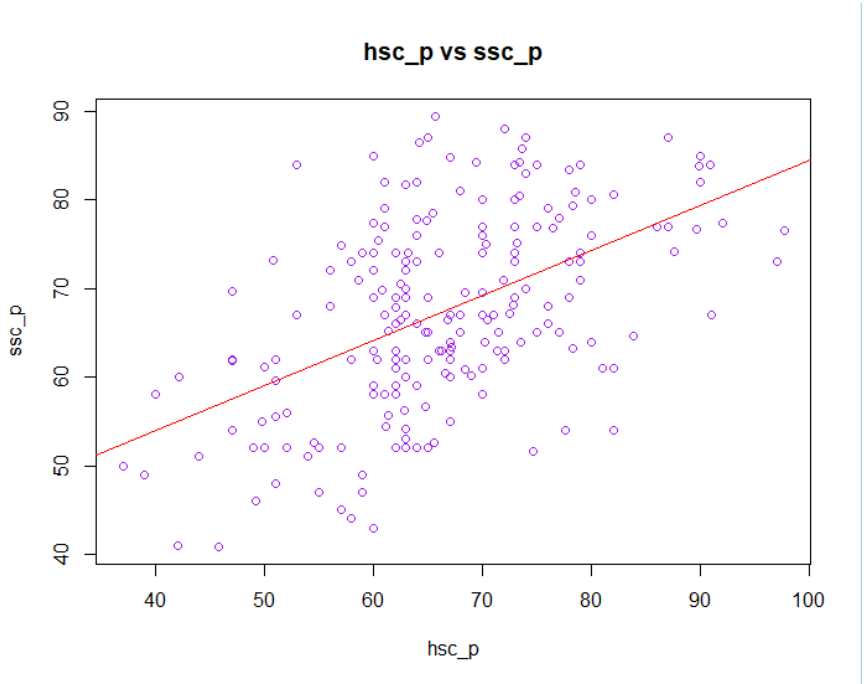
Viewing the correlation,

```
> # correlation
> # the default correlation method (Pearson) is been used because the two parameters contain numerical data
> cor(x,y)
[1] 0.5114721
>
```

Here the correlation is higher than the previous model.

Generating the scatter plot and adding the regression line,

```
> # scatter plot
> plot(x,y, main="hsc_p vs ssc_p", xlab="hsc_p", ylab="ssc_p", col="purple", cex=1)
> # add regression line
> abline(encoded_datacsv_ohe.regression, col="red")
>
```



Here the correlation between the two new variables (hsc_p and ssc_p) are stronger than the previous model's variables.

Displaying coefficients,

```
> # linear regression model
> # applying the linear regression function
> # y = target/response variable
> # x = predictor
> model_lr <- lm(y ~ x)
> print(model_lr)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
    33.5947       0.5082

> |
```

Generating the summary of the model,

```
> # summary of the model
> summary(model_lr)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-21.265  -6.593  -1.041   6.431  23.432

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  33.5947     3.9322   8.544  2.5e-15 ***
x             0.5082     0.0585   8.687  9.9e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.326 on 213 degrees of freedom
Multiple R-squared:  0.2616,    Adjusted R-squared:  0.2581
F-statistic: 75.46 on 1 and 213 DF,  p-value: 9.902e-16

> |
```

Adding the residuals column,

```
> # data frame
> encoded_datacsv_ohe.data <- data.frame(y, x)
> # adding the residuals column
> encoded_datacsv_ohe.data2 <- encoded_datacsv_ohe.data
> encoded_datacsv_ohe.data2$residuals <- encoded_datacsv_ohe.regression$residuals
> head(encoded_datacsv_ohe.data2)
      y     x  residuals
1 67.00 91.00 -12.838401
2 79.33 78.33   5.930143
3 65.00 68.00  -3.150436
4 56.00 52.00  -4.019677
5 85.80 73.60  14.803798
6 55.00 49.80  -3.901698
> 
```

Adding the predicted values column,

```
> # adding predicted values column
> encoded_datacsv_ohe.data2$fitted.values <- encoded_datacsv_ohe.regression$fitted.values
> # viewing 4 columns
> head(encoded_datacsv_ohe.data2)
      y     x  residuals fitted.values
1 67.00 91.00 -12.838401      79.83840
2 79.33 78.33   5.930143      73.39986
3 65.00 68.00  -3.150436      68.15044
4 56.00 52.00  -4.019677      60.01968
5 85.80 73.60  14.803798      70.99620
6 55.00 49.80  -3.901698      58.90170
> 
```

Checking confidence intervals,

```
> # check confidance intervals
> confint(encoded_datacsv_ohe.regression)
                  2.5 %     97.5 %
(Intercept) 25.8437796 41.3456446
x            0.3928625  0.6234824
> 
```

By taking into account all these observations, it can be considered as the 2nd model is much more efficient than the 1st model.

References

Alex. (2019, Jun 1). *Linear Regression Summary(lm): Interpretting in R*. Retrieved from boostedml.com: https://boostedml.com/2019/06/linear-regression-in-r-interpreting-summarylm.html

amazon.com. (n.d.). *What's the Difference Between Linear Regression and Logistic Regression?* Retrieved from amazon.com: https://aws.amazon.com/compare/the-difference-between-linear-regression-and-logistic-regression/

Arvind Shukla. (2023, Aug 7). *Neural Networks are Decision Trees*. Retrieved from www.linkedin.com: https://www.linkedin.com/pulse/neural-networks-decision-trees-arvind-shukla/

developers.google.com. (n.d.). *Classification: ROC Curve and AUC*. Retrieved from developers.google.com: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Dr. Osman Dag. (2022, Mar 4). *How to Remove Outliers from Data in R* . Retrieved from universeofdatascience.com: https://universeofdatascience.com/how-to-remove-outliers-from-data-in-r/

Gustav Willig. (2023, Jan 17). *Decision Tree vs Logistic Regression*. Retrieved from gustavwillig.medium.com: https://gustavwillig.medium.com/decision-tree-vs-logistic-regression-1a40c58307d0

John . (2020, Jan 19). *How to Remove Outliers in R*. Retrieved from www.r-bloggers.com: https://www.r-bloggers.com/2020/01/how-to-remove-outliers-in-r/

makemeanalyst.com. (n.d.). *Normal Probability Plot in R using ggplot2*. Retrieved from makemeanalyst.com: https://makemeanalyst.com/statistics-with-r/normal-probability-plot-in-r-using-ggplot2/

methodenlehre.github.io. (n.d.). *Graphics with ggplot2*. Retrieved from methodenlehre.github.io: https://methodenlehre.github.io/SGSCLM-R-course/graphics-with-ggplot2.html

Niam Zaki Zamani. (2021, Jan 18). *Linear Regression on Student Grade Prediction*. Retrieved from rpubs.com: https://rpubs.com/niamzaki/student_grade_prediction

Peter Bruce, Andrew Bruce. (n.d.). *Chapter 4. Regression and Prediction*. Retrieved from www.oreilly.com: https://www.oreilly.com/library/view/practical-statistics-for/9781491952955/ch04.html

Rebecca C. Steorts. (n.d.). *Comparison of Linear Regression with K-Nearest*. Retrieved from www2.stat.duke.edu: https://www2.stat.duke.edu/~rcs46/lectures_2017/03-lr/03-knn.pdf

Rohit Kundu. (2022, Sep 13). *Confusion Matrix: How To Use It & Interpret Results [Examples]*. Retrieved from www.v7labs.com: https://www.v7labs.com/blog/confusion-matrix-guide

Safa Mulani. (2022, Aug 3). *Outlier Analysis in R - Detect and Remove Outliers*. Retrieved from www.digitalocean.com: https://www.digitalocean.com/community/tutorials/outlier-analysis-in-r

stackoverflow.com. (2023, May 3). *Difference between Logistic Regression and Decision Trees*. Retrieved from stackoverflow.com: https://stackoverflow.com/questions/76161673/difference-between-logistic-regression-and-decision-trees

typeset.io. (n.d.). *What is the difference between KNN regression and linear regression?* . Retrieved from typeset.io: https://typeset.io/questions/what-is-the-difference-between-knn-regression-and-linear-1pad331c0a

www.ibm.com. (n.d.). *What is logistic regression?* . Retrieved from www.ibm.com: https://www.ibm.com/topics/logistic-regression

www.rdocumentation.org. (n.d.). *plot: Generic X-Y Plotting*. Retrieved from www.rdocumentation.org: https://www.rdocumentation.org/packages/graphics/versions/3.6.2/topics/plot

www.statisticssolutions.com. (n.d.). *Correlation (Pearson, Kendall, Spearman)*. Retrieved from www.statisticssolutions.com: https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/correlation-pearson-kendall-spearman/

www.sthda.com. (n.d.). *QQ-plots: Quantile-Quantile plots - R Base Graphs* . Retrieved from www.sthda.com: http://www.sthda.com/english/wiki/qq-plots-quantile-quantile-plots-r-base-graphs