# PROFESSIONAL TRAINING REPORT

**at**

**Sathyabama Institute of Science and Technology (Deemed to be University)**

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering

By

## CHAKALI GANGADHAR

**REG. NO. 39110202**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,**

**CHENNAI – 600119, TAMILNADU**

**APRIL 2022**

|  |  |  |
| --- | --- | --- |
|  | **SATHYABAMA**  **INSTITUTE OF SCIENCE AND TECHNOLOGY** (DEEMED TO BE UNIVERSITY) **Accredited with Grade "A" by NAAC**  (Established under Section 3 of UGC Act, 1956)  JEPPIAAR NAGAR, RAJIV GANDHI SALAI  CHENNAI– 600119  [**www.sathyabama.ac.in**](http://www.sathyabama.ac.in) |  |

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **CHAKALI GANGADHAR (Reg. No: 39110202),** who carried out the project entitled "**Student Placement Prediction Using Naive Bayes Algorithm**" under my supervision from February 2022 to April 2022.

## Internal Guide

## Dr. A. Pravin, M.E., Ph.D.

**Head of the Department**

**Dr. S. Vigneshwari, M.E., Ph.D.**

**Dr. L. Lakshmanan, M.E., Ph.D.**

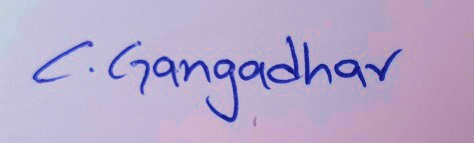


## Submitted for Viva-voce Examination held on

**InternalExaminer ExternalExaminer**

**DECLARATION**

I, **CHAKALI GANGADHAR,** hereby declare that the project report entitled **Student Placement Prediction Using Naive Bayes Algorithm** done by me under the guidance of **Dr. A. Pravin** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

****

## DATE:

**PLACE:** Chennai, Tamil Nadu **SIGNATURE OF THE CANDIDATE**

**ACKNOWLEDGEMENT**

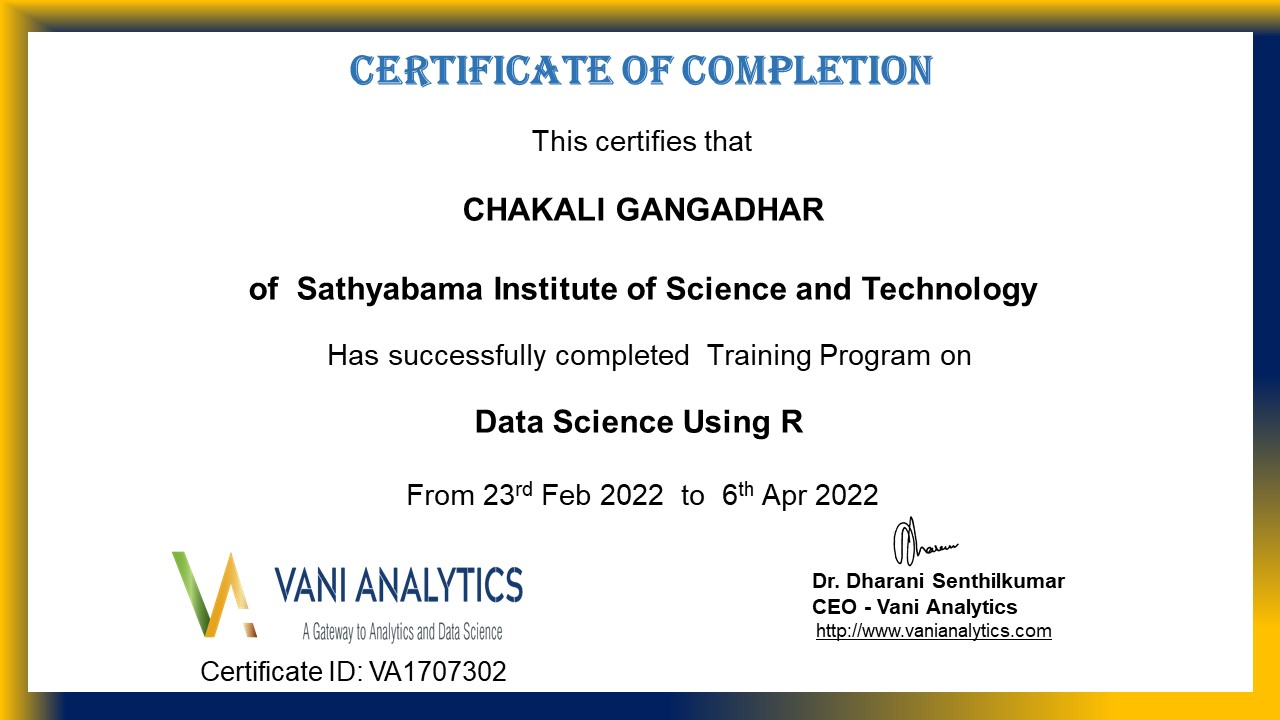
I am pleased to acknowledge my sincere thanks to the **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D.**, **Dean**, School of Computing, **Dr. S. Vigneshwari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of **Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.A. Pravin, M.E., Ph.D.,** for his valuable guidance, suggestions, and constant encouragement that paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project

**TRAINING CERTIFICATE**

****



# ABSTRACT

One of the most significant difficulties that higher education institutions encounter today is improving student placement performance. Placements are the essential factor in determining admission and a school's reputation. As a result, every university strives to improve its placement rate for every department. Every student desires to have a job offer before graduating from college. Educational institutions seek more efficient technology to aid in improved administration and decision-making procedures and help them develop new strategies. A placement prediction system can forecast which type of student can be placed. It allows students to see where they stand and what they need to do to get a good placement.

This project proposes a student placement prediction model that uses the Naive Bayes method to determine the likelihood of students being placed. This methodology aids an organization's placement cell in identifying prospective students and paying attention to and improve their technical and interpersonal abilities. Students can put more effort into this approach to be placed in organizations with more significant hierarchies.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| CHAPTER No. | TITLE | PAGE No |
|  | ABSTRACT | i |
|  | LIST OF FIGURES | iv |
|  | LIST OF TABLES | v |
|  | LIST OF ABBREVIATIONS | vi |
| 1. | INTRODUCTION | 1 |
|  | 1.1 Area of research | 2 |
| 2. | AIM AND SCOPE OF THE PRESENT INVESTIGATION | 3 |
|  | 2.1 Aim of the project | 3 |
|  | 2.2 Scope of the project | 3 |
| 3. | MATERIALS AND METHODS, ALGORITHMS USED | 4 |
|  | 3.1 Understanding Dataset | 5 |
|  | *3.1.1 Understanding dataset via tables* | 6 |
|  | *3.1.2 Understanding dataset via Graphs* | 9 |
|  | 3.2 Data Preprocessing | 15 |
|  | *3.2.1 Label Encoding* | 16 |
|  | 3.3 Naive Bayes Algorithm | 17 |
|  | 3.4 Understanding the model algorithm | 19 |
|  | 3.5 Building and testing the model | 19 |
|  | *3.5.1 Random Sampling Experiment* | 21 |
|  | 3.6 Performance metrics | 21 |
|  | *3.6.1 Confusion Matrix* | 22 |
|  | *3.6.2 Log loss* | 25 |
|  | *3.6.3 Accuracy score* | 25 |
|  | *3.6.4 ROC AUC curve* | 25 |
|  | *3.6.5 ROC AUC score* | 26 |
| 4. | RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS | 27 |
|  | 4.1 Analyzing performance | 27 |
|  | 4.2 Final result | 30 |
| 5 | SUMMARY AND CONCLUSIONS | 31 |
|  | 5.1 Summary of the Project | 31 |
|  | 5.2 Conclusion of the Project | 33 |
|  | REFERENCES | 34 |
|  | APPENDIX | 36 |
|  | A. SCREENSHOTS | 36 |
|  | B. SOURCE CODE | 74 |

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| FIGURE NO. | FIGURE NAME | PAGE NO. |
| 3.1 | Architecture map of the project | 4 |
| 3.2 | Correlation among the Data Points | 10 |
| 3.3 | Bar Graph of the prediction class | 10 |
| 3.4 | Bar Graph of gender class | 11 |
| 3.5 | Bar Graph of SSC Board class | 11 |
| 3.6 | Bar Graph of HSC Board class | 12 |
| 3.7 | Histograms of SSC and HSC Board pass percentages | 12 |
| 3.8 | Bar Graph of Branches in degree | 13 |
| 3.9 | Bar Graph of Work experience | 13 |
| 3.10 | Bar Graph of Work experience Placed vs. Not Placed | 14 |
| 3.11 | Bar Graph of Etest Placed vs. Not Placed | 14 |
| 3.12 | Label encoding process of categorical data | 16 |
| 3.13 | Naive Bayes frequency Look-up Table | 18 |
| 3.14 | Model complexity factors | 20 |
| 3.15 | Structure of confusion matrix for binary classification | 22 |
| 3.16 | Confusion matrix for testing data | 23 |
| 3.17 | ROC curve of NB model | 26 |
| 4.1 | Bar graphs for performance of model | 28 |
| 4.2 | Line graphs for performance of models | 29 |
| 4.3 | Line graphs for K-fold performance of models | 29 |

# LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| TABLE NO. | TABLE NAME | PAGE NO. |
| 3.1 | Features of student placement dataset | 11 |
| 3.2 | Sample records of students at the beginning of the dataset | 12 |
| 3.3 | Sample records of students in the ending of the dataset | 12 |
| 3.4 | Unique values in the dataset. | 13 |
| 3.5 | Information about the dataset | 13 |
| 3.6 | Description of the Dataset | 14 |
| 3.7 | Description of Placed Class Numerical Data Points | 14 |
| 3.8 | Description of Not Placed Class Numerical Data Points | 15 |
| 3.9 | Correlation Table of Data Points | 15 |
| 3.10 | Performance Measurements | 30 |
| 4.1 | Different Algorithms Model performances | 34 |
| 4.2 | Parameters of Naive Bayes model | 36 |
| 4.3 | Results of Naive Bayes model | 36 |

# LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| **ABBREVIATION** | **EXPANSION** |
| ML | Machine Learning |
| SSC | Secondary School Certificate |
| HSC | Higher School Certificate |
| RF | Random Forest |
| NB | Naive Bayes |
| K-NN | K-Nearest Neighbours |
| SVM | Support Vector Machine |
| R | R Programming |
| b/w | between |
| No. | Number of |
| min | Minimum |
| max | Maximum |
| Vs | Versus |
| int | integer |
| chr | character |
| tp | True Positives |
| fp | False Positive |
| tn | True Negatives |
| fn | False Negatives |

Chapter-1

**INTRODUCTION**

The majority of students in higher education enroll in an educational curriculum program to improve their chances of landing a good job. As a result, making an informed professional decision about where to work after completing a course is critical in a student's life. One of the variables examined in establishing the institution's excellence is placement. The admissions rate to an educational institution is based on placements, which is a well-known truth worldwide. As a result, every institution works hard to improve student placements.

In India, 1.5 million engineers graduate each year, according to data. The demand for qualified graduates in the IT business is increasing. However, most students are unaware of the importance of the IT business. The number of graduates who meet the company's requirements and quality standards is relatively low. An institute's placement cell and instructors should take the necessary procedures to generate a group of students that fit each company's needs. In each academic group, campus placement plays a critical role in assisting college students in achieving their objectives. Because of the enormous number of students, managing placement and education information in a large organization is difficult; differentiation and classification into distinct categories become tedious in this case. Before they leave college, every student hopes to find a job that they can do with their hands.

In this project, the Naive Bayes algorithm predicts student placement using factors such as gender, ssc\_p, ssc\_b, hsc\_p, hsc\_b, hsc\_s degree\_p, degree\_t, workex, etest\_p, specialization, mba\_p, status. A placement probability predictor gives college students an idea of where they stand and what they need to do to get a good job. The information gained from this can help students better identify their weak areas and how to improve them. Working in such areas allows college students to secure more placements in a college environment.

**1.1 Area of Research**

Machine learning (ML) investigates computer science algorithmic calculations that can work naturally through experience and data. It is part of artificial intelligence. Machine learning algorithms fabricate a model dependent on example data, known as "Training data," to make predictions or decisions without programming for every application. Machine learning algorithms are utilized in various applications, such as medication, email sorting, speech recognition, and computer vision. It is troublesome or unworkable to develop ordinary algorithms to perform the required tasks. A subset of machine learning is rigidly identified with computational statistics, which centers around making predictions utilizing computers, yet not all machine learning is statistical learning. Data mining is a related field of study, focusing on exploring data analysis through unsupervised learning. The investigation of mathematical optimization conveys strategies, hypotheses, and application areas to the field of machine learning. Some executions of machine learning use data and neural networks that impersonate the working of an actual brain. In its application across business issues, machine learning is likewise referred to as prescient analysis.

Chapter-2

**AIM AND SCOPE OF THE PRESENT INVESTIGATION**

**2.1 Aim of the Project**

The student placement dataset is collected and analyzed to predict whether the students get placed or not. Placement opportunities are abundant in the market due to many emerging technologies. Skill development of students should be done from a placement point of view. For prediction, Data Science techniques of the Naive Bayes algorithm have been applied. Data Science is an interdisciplinary field that incorporates computer science, mathematics, statistics, and domain knowledge. The Naive Bayes Algorithm is applied to the placement data, and efficient prediction accuracy is achieved by delivering the result of the placement chance of students.

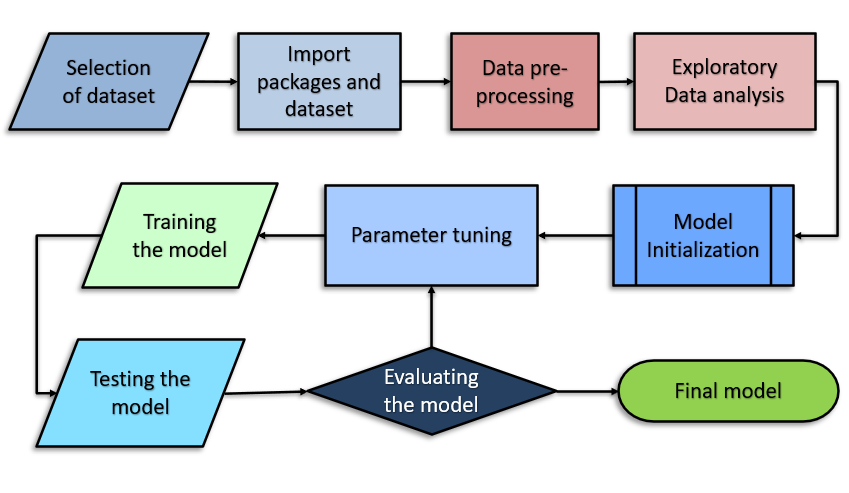
**2.2 Scope of the Project**

The ML model we are building is used by universities where there is a need for students' placement prediction based on their academic performance. Many ML Algorithms can solve this problem, but we adopt the Naive Bayes algorithm to predict student placement in this project, one of the better performing ML algorithms. Using source code and compatible software for machines makes it possible to make any pc to predict student placement, which helps educational management concentrate and focus on students who are predicted as Not Placed by an algorithm.

Chapter-3

**MATERIALS AND METHODS, ALGORITHMS USED**

In my study, the data points were acquired from students' academic data from different universities and noted. Before developing the model, we need to understand the dataset, examine each feature, find the relations b/w the features, how the relations impact results of the model, and extract valuable features of the dataset. This phase is called understanding data. After understanding the data, we need to clean the dataset, drop unnecessary values, fill the missing values, and deal with categorical data. This phase is called data preprocessing. This data preprocessing helps to make compatible data structure for the model. After this preprocessing data phase, we need to understand the model, how it works, the advantages and disadvantages of the algorithm, and its hyperparameters that help in performance. After understanding the model, we need to build the model with studied hyperparameters and test them with different variables of hyperparameters. This phase is called testing and validation. In this phase, we also need to understand the performance metrics of this problem statement, which helps in validating the model to reach my goal. These project phases are explained in detail in upcoming chapters and subchapters.



***Fig:3.1:*** *Architecture map of the Project*

**3.1 Understanding Dataset**

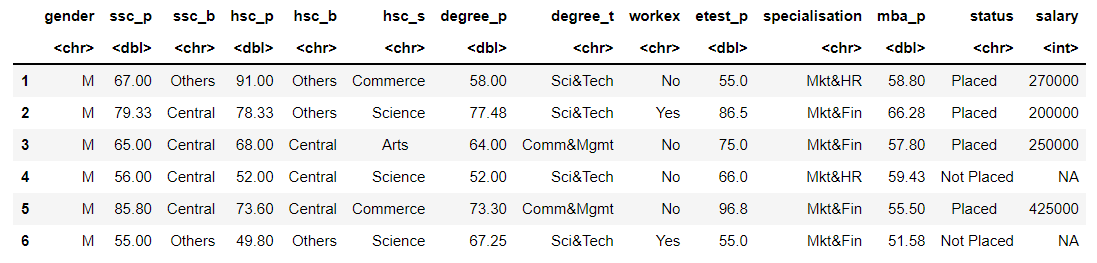
The features of student placement data collected from different universities have been noted in the dataset as numerical and categorical data representing students' performance. For understanding data with R integrated modules, we need to import the necessary libraries for working with a dataset and import the dataset to the appropriate Jupyter notebook or RStudio. Dplyr [[3]](https://www.rdocumentation.org/packages/dplyr/versions/0.7.8) module is designed to work with data manipulation that provides a uniform set of verbs, helping to resolve the most frequent data manipulation challenges. After we go through the features in the dataset, the best valuable features and characteristics utilized in feature extraction for student placement data points are given in Table 3.1.

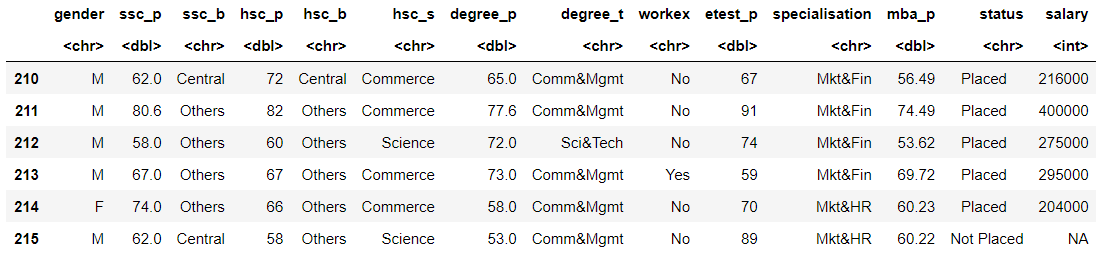
**Table 3.1:** Features of student placement dataset

|  |  |  |
| --- | --- | --- |
| S.no | Name | Characteristics |
| 1 | gender | Student gender in each datapoint. |
| 2 | ssc\_p | Secondary Education pass percentage-10th Grade. |
| 3 | ssc\_b | Board of Education- Central/ Others. |
| 4 | hsc\_p | Higher Secondary Education percentage- 12th Grade. |
| 5 | hsc\_b | Board of Education- Central/ Others. |
| 6 | hsc\_s | Specialization in Higher Secondary Education |
| 7 | degree\_p | Degree Percentage. |
| 8 | degree\_t | Under Graduation(Degree type)- Field of degree education. |
| 9 | workex | Work Experience |
| 10 | etest\_p | Employability test percentage ( conducted by the college) |
| 11 | specialisation | Post Graduation(MBA)- Specialization |
| 12 | mba\_p | MBA percentage |
| 13 | status | Status of placement- Placed/Not placed |
| 14 | salary | Salary offered by corporate to candidates |

***3.1.1 Understanding dataset via tables***

These features used in the dataset are understood by definition. Still, we also need to understand the dataset structure, how data is represented in the tabular form, find out the distribution of values, and the data types of values. This data is described in the below tables.

**Table 3.2:** Sample records of students at the beginning of the dataset

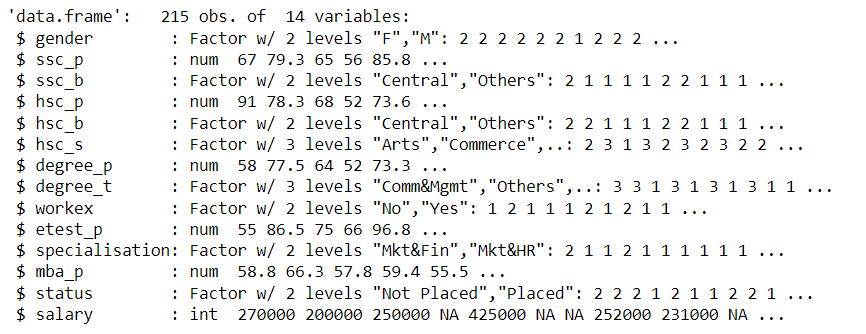
**Table 3.3:** Sample records of students at the ending of the dataset

In this review, the distribution of 215 data points of student performance was acquired by data collection among universities. We need to build the description of the dataset and unique data types for data preprocessing. The no. prediction class values in the dataset are represented in Table 3.4.

**Table 3.4:** Unique values in the dataset.

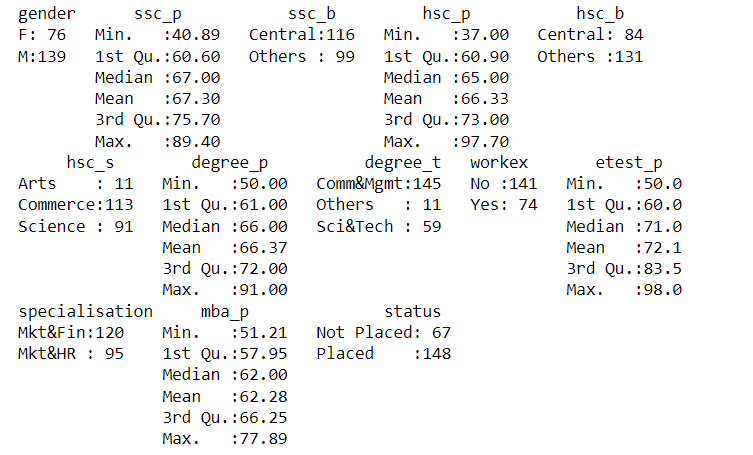
|  |  |  |
| --- | --- | --- |
| S.no | Name | No. of values |
| 1 | Placed | 148 |
| 2 | Not Placed | 67 |
| Total |  | 215 |

The Data types of features, levels of categorical data sample numeric values are represented in Table 3.5

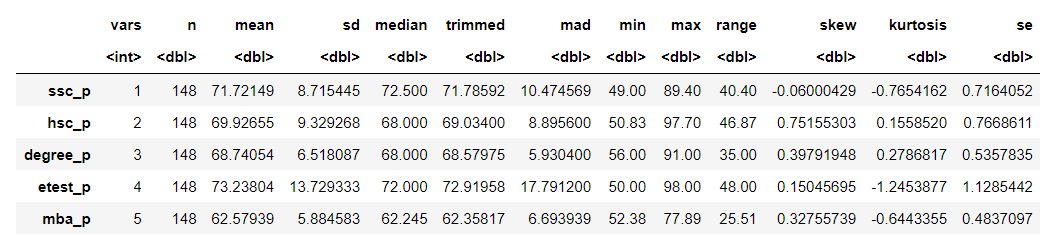
**Table 3.5:** Information about the dataset

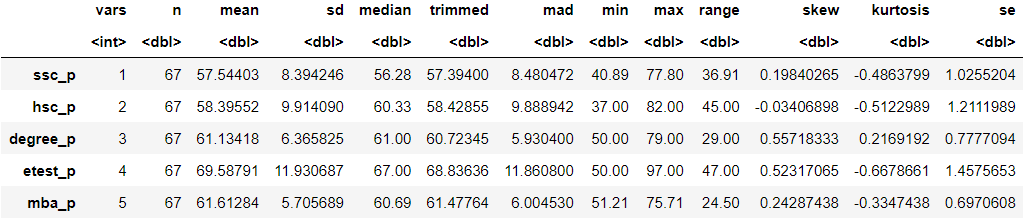
The number of records, minimum value, maximum value, standard deviation, mean, 25% of max value, 50%(median) of the max value, 75% of the max values on the min-max range values of the dataset are represented in Table 3.6.

**Table 3.6:** Description of the Dataset

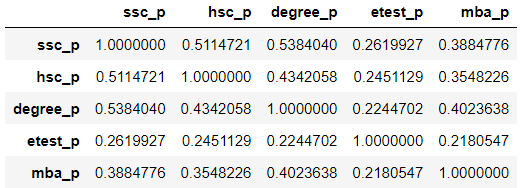


**Table 3.7:** Description of Placed Class Numerical Data Points



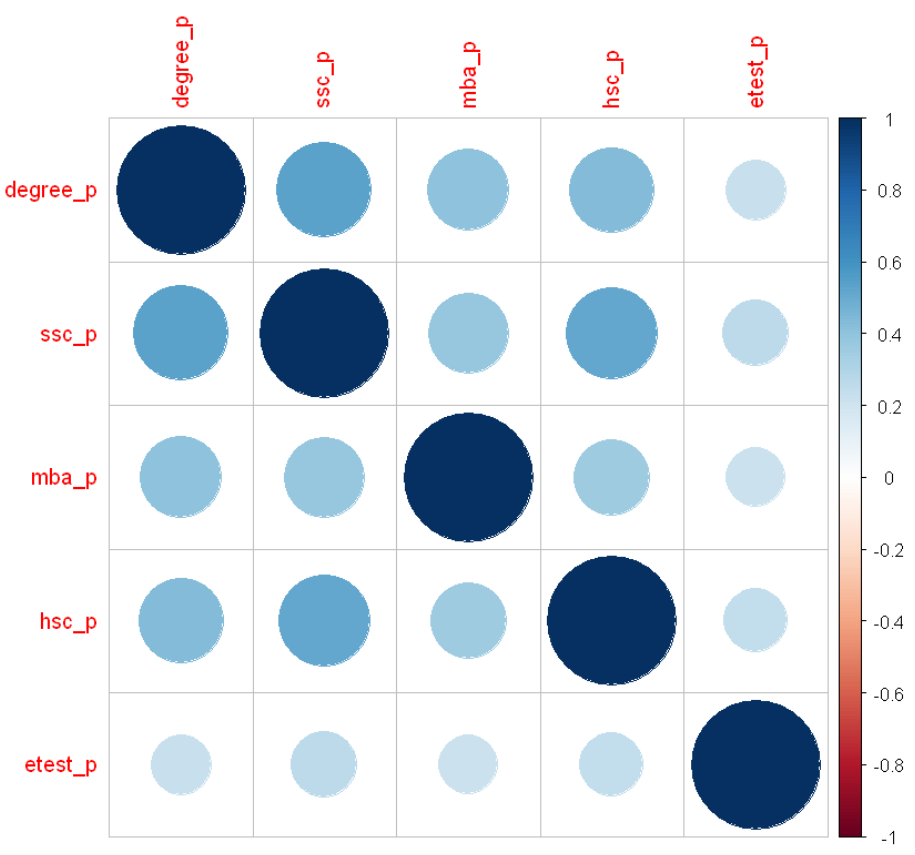
**Table 3.8:** Description of Not Placed Class Numerical Data Points

**Table 3.9:** Correlation Table of Data Points



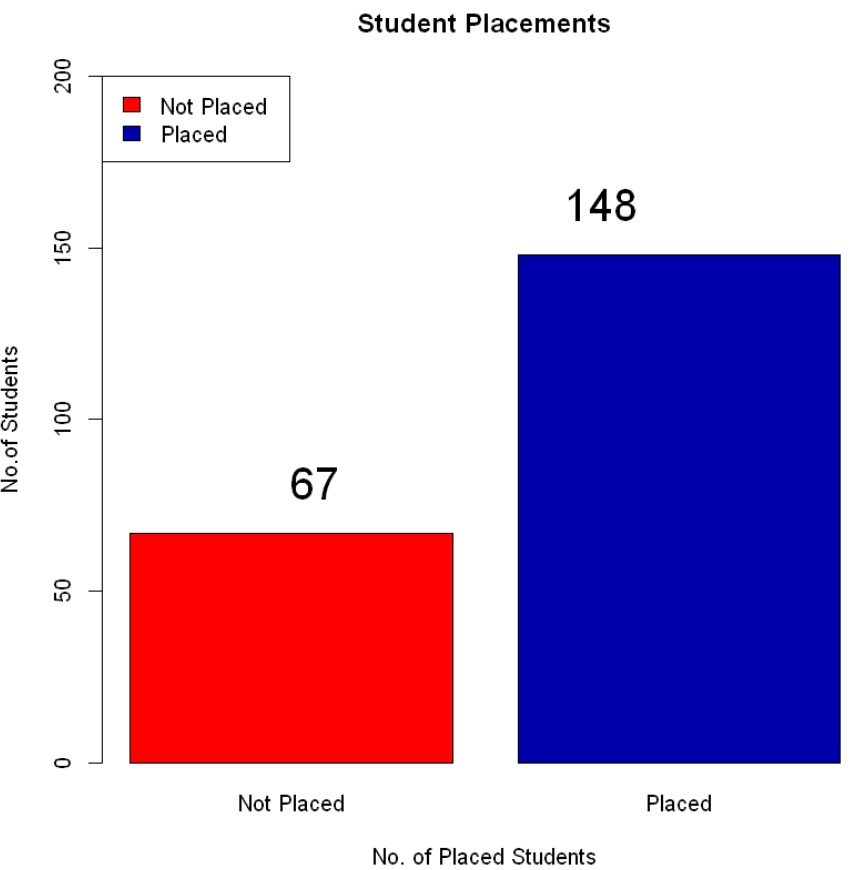
***3.1.2 Understanding dataset via Graphs***

Correlation is a statistical term portraying how much two variables move in coordination. It can also be said that how much two variables are dependent on each other. If the two variables move similarly, those variables have a positive correlation [[7]](http://www.sthda.com/english/wiki/correlation-matrix-a-quick-start-guide-to-analyze-format-and-visualize-a-correlation-matrix-using-r-software). If they move in inverse ways, they have a negative correlation. The correlation [[8]](https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html) among the features of the data points in the dataset has been shown in figure 3.2



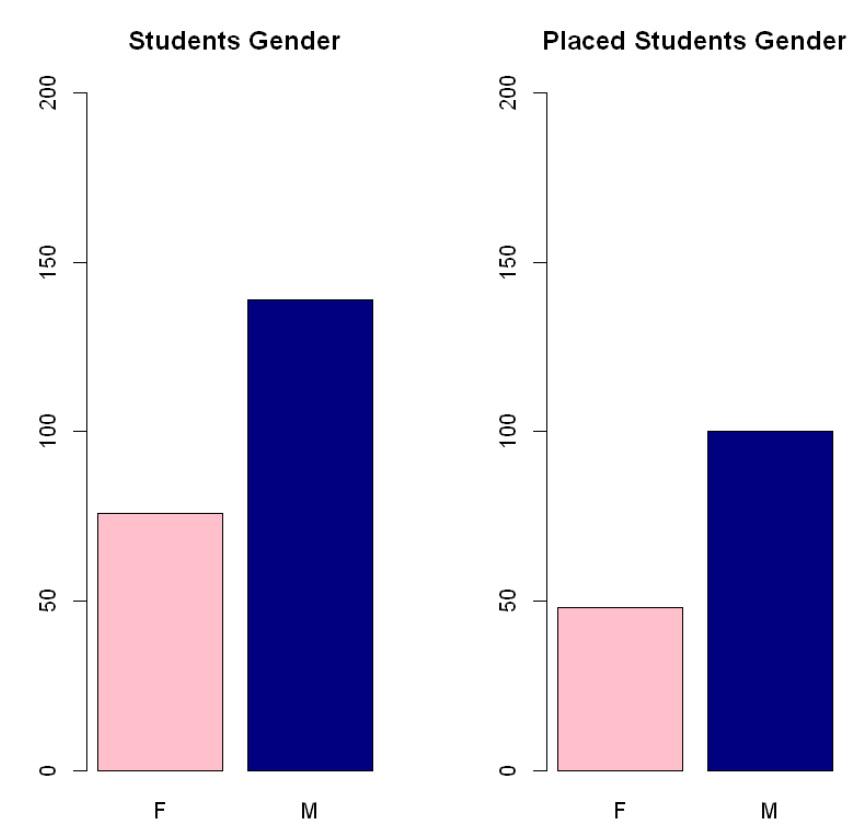
***Fig:3.2:*** *Correlation among the Data Points*

The Placement labels in the feature are called status. It is shown in the graph for the number of students Placed and Not Placed count in figure 3.3.

****

***Fig:3.3:*** *Bar Graph of the prediction class*

If we plot a gender distribution in the dataset vs. gender distribution of Placed data points, there is very little difference in placement recruitment based on gender. That graph is shown in figure 3.4.

****

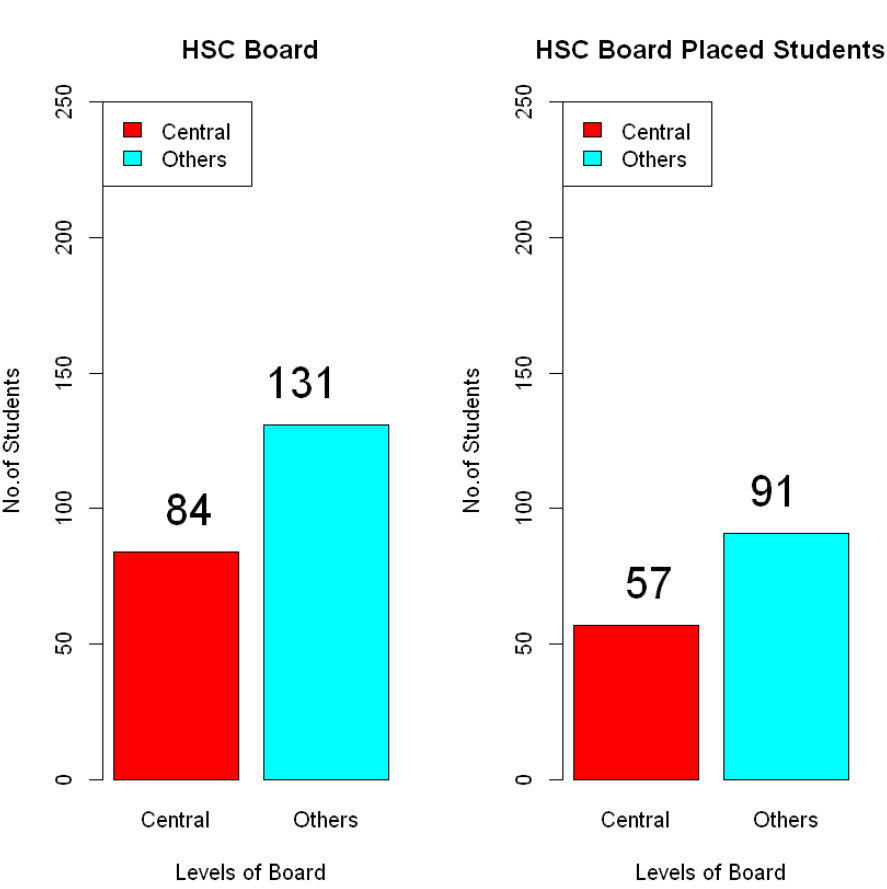
***Fig:3.4:*** *Bar Graph of gender class*

If we plot SSC board distribution in dataset vs. SSC board distribution of Placed data points, placement student board of secondary education doesn't play a significant role. That graph is shown in figure 3.5.

****

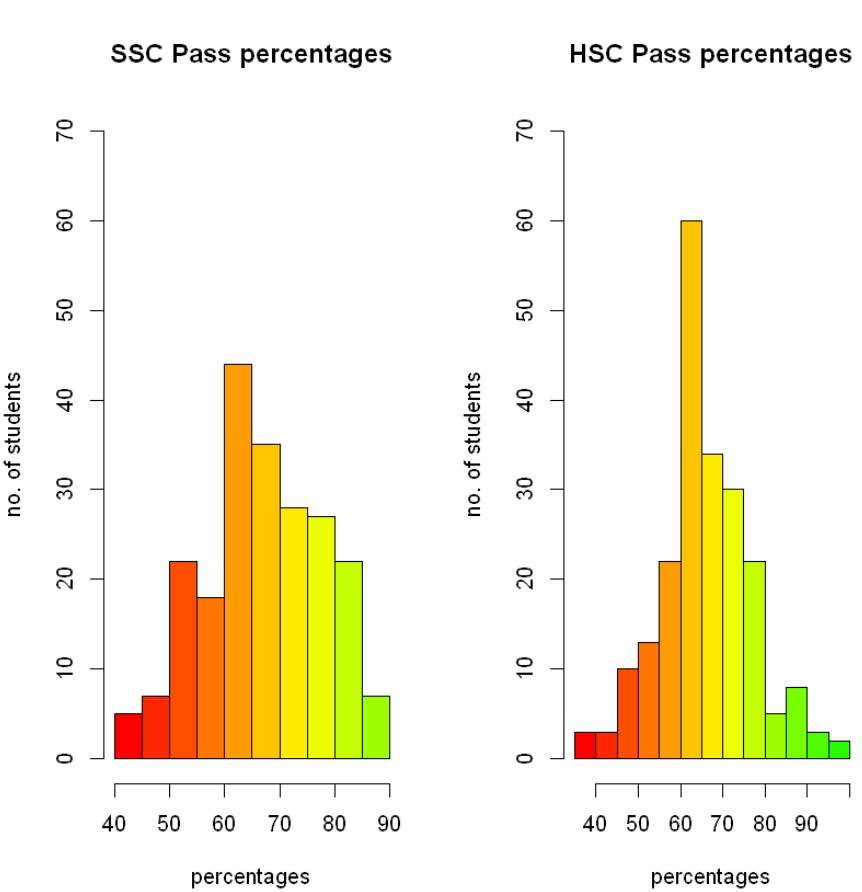
***Fig:3.5:*** *Bar Graph of SSC Board class*

If we plot HSC board distribution in dataset vs. HSC board distribution of Placed data points, the comparison b/w SSC board and HSC board there is a massive switch from central to others in HSC board. That graph is shown in figure 3.6.



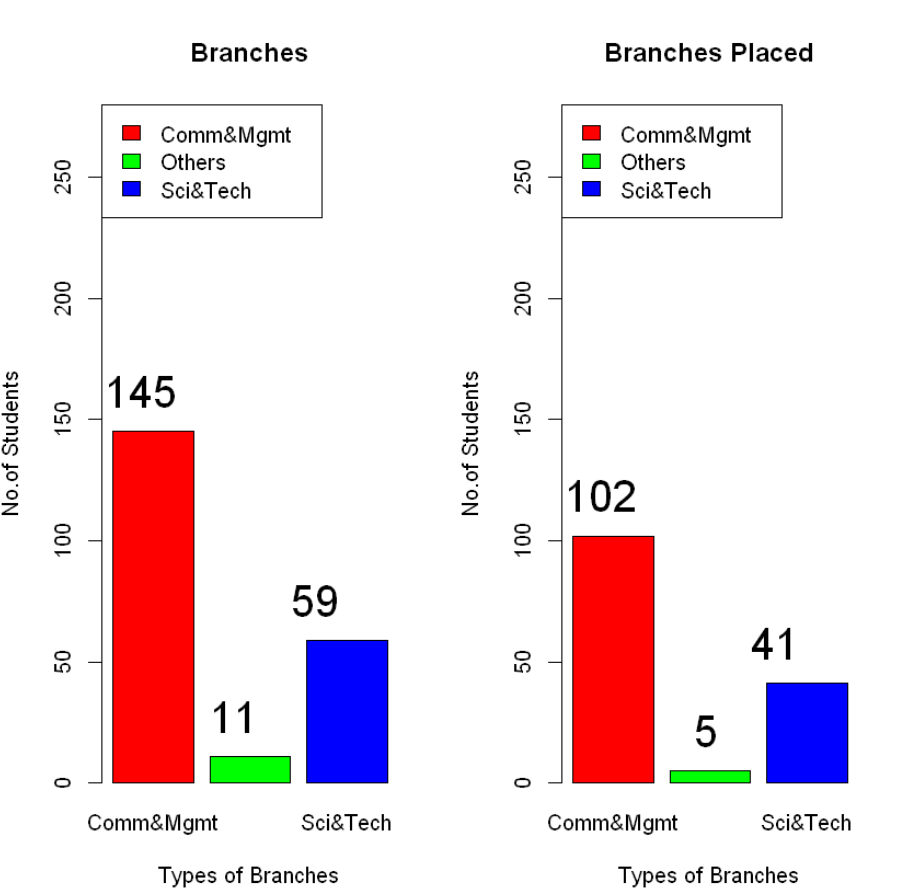
***Fig:3.6:*** *Bar Graph of HSC Board class*

Suppose we plot a histogram of SSC board pass percentages and HSC board pass percentages data points. That graph is shown in figure 3.7.

****

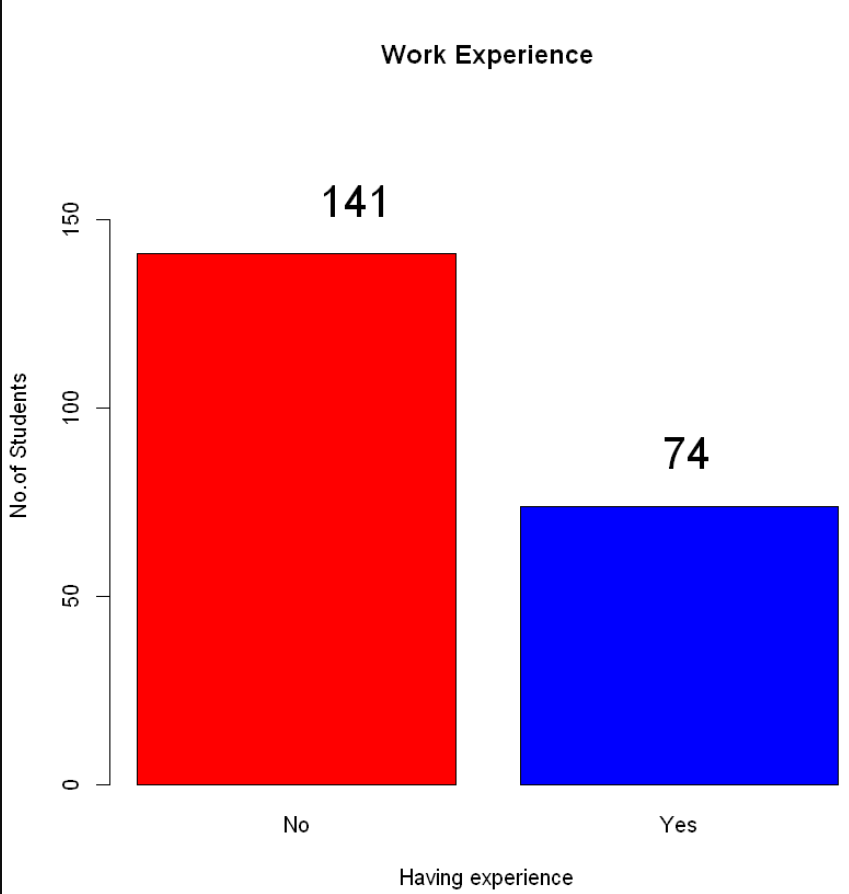
***Fig:3.7:*** *Histograms of SSC and HSC Board pass percentages*

If we plot Branches in degree data distribution Vs. Branches in the degree of Placed data points. That graph is shown in figure 3.8.

****

***Fig:3.8:*** *Bar Graph of Branches in degree*

If we plot Work experience and Work experience of placed Vs. Work experience of Not placed data points. That graphs are shown in figure 3.9 and figure 3.10.

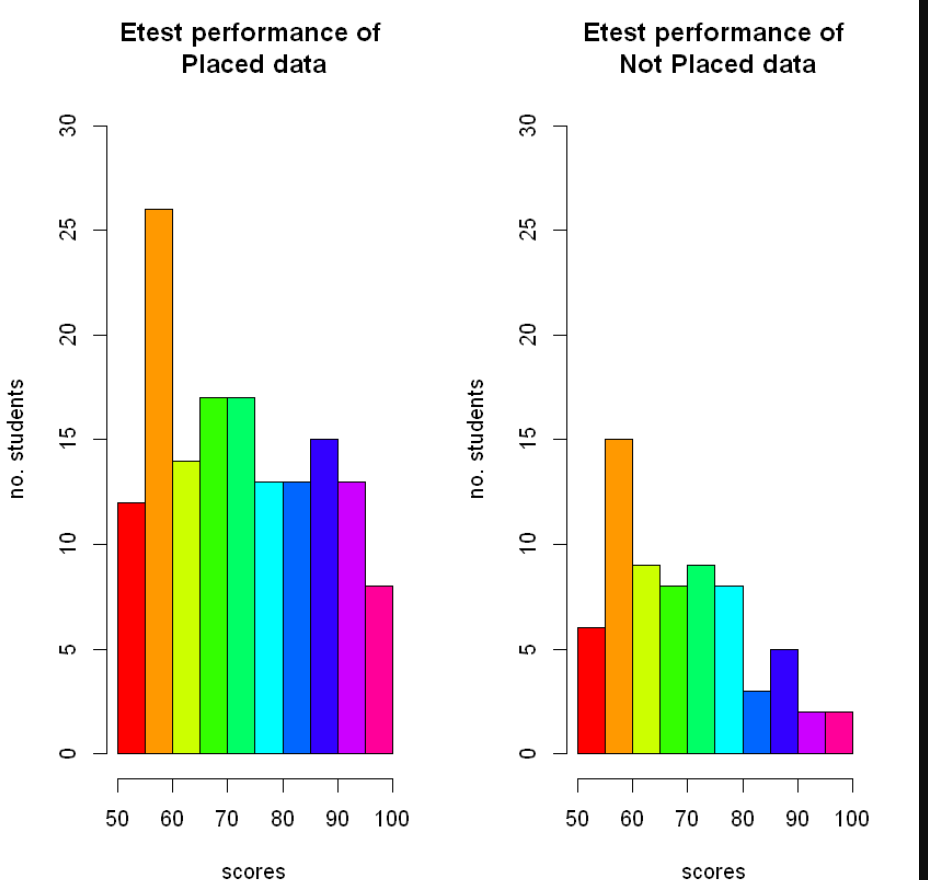


***Fig:3.9:*** *Bar Graph of Work experience*

****

***Fig:3.10:*** *Bar Graph of Work experience Placed vs. Not Placed*

Suppose we plot the Employment test of placed Vs. Employment test of Not placed data points. That graph is shown in figure 3.11.



***Fig:3.11:*** *Bar Graph of Etest Placed vs. Not Placed*

All the graphs plotted for data analysis can conclude that all attributes present in the dataset are independent and have significantly less positive correlation among the attributes. The Naive Bayes algorithm is perfectly suitable for this kind of dataset because the NB algorithm assumes that all attributes are independent and equal. Here all attributes are combined to give meaningful results from the model. However, there might be some variations in the testing performance of the model due to the imbalance of data points of class labels in the dataset.

**3.2 Data Preprocessing**

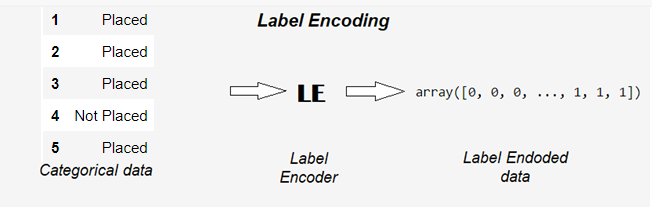
This phase is vitalfor building the model because, in this phase, we build a data structure that is compatible with the model, and we make random shuffles with the data to make the model learn the efficient way. Here we select the data in terms of x and y, where x is the input of the model and y is the model's output.

*F(x) is the ML model, x is the input, and y is the model's output.*

After selecting x and y, we observe that our prediction attribute status y and some other attributes in input x that need to be trained are in the chr data type. This data type is also known as categorical data. To make the machine understand chr data type, it needs to be converted to factor datatype. Naive Bayes and some machine learning algorithms only understand Factor datatype in R. Other algorithms like Gaussian NB and K-NN don't accept Factor datatype in R. To deal with this chr data. We need to convert chr to factor with functions in R. However, to deal with this categorical data, we need methods for encoding the data that the model can understand. There are different methods for dealing with categorical data like dropping columns, label encoding, and one-hot encoding, but we need to choose the method that suits the requirement. For this problem statement, we have a binary classification classifying placement status. Here, we choose label encoding to deal with categorical data.

***3.2.1 Label Encoding***

Label Encoding refers to changing the labels into a numeric structure that changes over them into a machine-understandable structure. ML Algorithms would then be able to choose how those labels should be worked in a superior manner. It is a significant preprocessing step for the structured dataset in supervised learning. So basically converts Placed and Not Placed class labels to 0 and 1 (or) 1 and 2 as an integer data type for labeling, which becomes machine-readable data. If there is more than one category, it labels the new category incrementally from the previous highest label for the conversion of character datatype using R. There is a function called "LabelEncoder" [[6]](https://www.rdocumentation.org/packages/superml/versions/0.5.3/topics/LabelEncoder) under the superml library. Preprocessing the data using a label encoder is implemented, as shown in figure 3.5.



***Fig:3.12:*** *Label encoding process of categorical data*

After label encoding the categorical data, we need to split the data for the training and testing of the model. We have an integrated R function called sample for dividing the dataset, which randomly divides the dataset with train and test ratios. This train ratio, test ratio ranges from 0 to 1. This function takes three variables: the number of parts, the dataset, and the ratio of the dataset. It returns a list of indices that represents 1 and 2. Here 1 is train ratio data and 2 tests ratio data.

**3.3 Naive Bayes Algorithm**

Classification is one of the types of ML problems where we use ML techniques to categorize data into a given number of classes. Classification can be performed on structured or unstructured data. The main goal of a classification problem is to identify the category/class to which new data falls. Naive Bayes Classifiers are a family of classification algorithms known as probabilistic classifiers. It is a classification and prediction algorithm based on Bayes Theorem. Bayes Theorem is the fundamental of Naive Bayes assumption is that each feature is a:

* Independent (the features are assumed to be **independent.)**
* Equal (None of the attributes is irrelevant and assumed to be contributing **equally**to the outcome.)

- Probability of occurrence of event A given that event B is true - Posterior Probability

- Probabilities of the occurrence of event A-Class prior Probability

- Probabilities of the occurrence of event B –Evidence

- Probability of the occurrence of event B given that event A is true - Likelihood

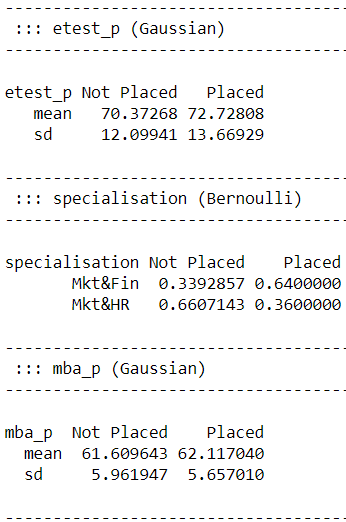
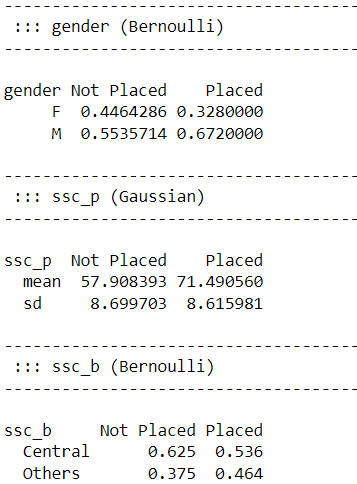
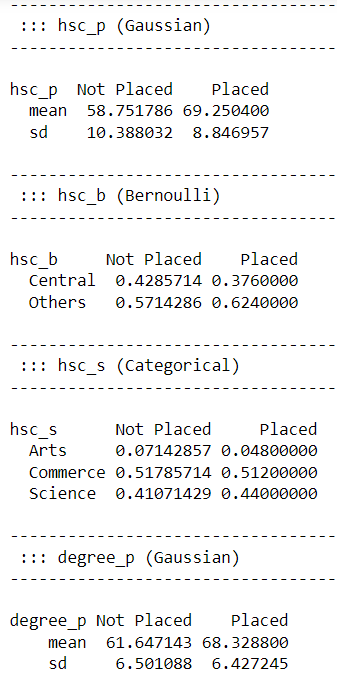
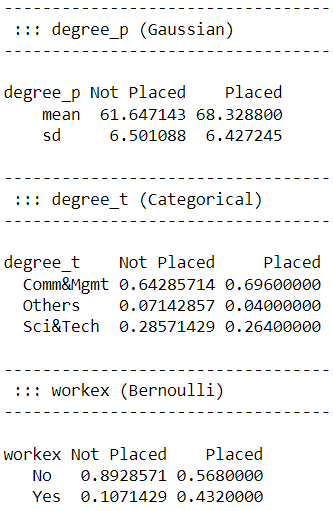
The Naive Bayes algorithm is based on the Bayes theorem, which assumes all attributes are independent between every pair of attributes. Naive Bayes classifiers work well in many real-world situations, such as document classification and spam filtering. This algorithm requires a small amount of training data to estimate the necessary parameters. Naive Bayes classifiers are extremely fast compared to more sophisticated methods.

***Algorithmic steps for Naive Bayes classification***

Step 1: Convert the data set into a frequency tables

Step 2: Create a Likelihood table by finding the probabilities values of each attribute

Step 3: Now, use the Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of the prediction.

******

***Fig:3.13:*** *Naive Bayes frequency Look-up Table*

Building a model needs a methodology to achieve good accuracy for the problem. We followed the Bayes Theorem methodology [[1]](https://en.wikipedia.org/wiki/Bayes%27_theorem) in this study. Naive Bayes is a supervised learning algorithm based on the Bayes theorem, which assumes independence between every pair of features.

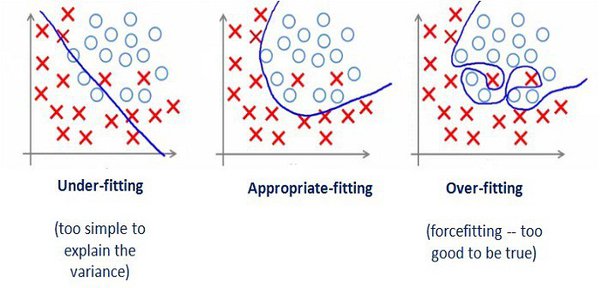
**3.4 Understanding the model algorithm**

Before developing the model, we need to understand the model's nature. Naive Bayes classifier ML algorithm has hyperparameters that can be tuned to achieve maximum performance. The hyperparameters [[3]](https://cran.r-project.org/web/packages/naivebayes/naivebayes.pdf) used in this model are

* + usekernel : parameter allows us to use a kernel density estimate for continuous variables versus a Gaussian density estimate,
  + adjust : allows us to adjust the bandwidth of the kernel density (more significant numbers mean a more flexible density estimate),
  + fL : allows us to incorporate the Laplace smoother.

**3.5 Building and testing the model**

After understanding the model, we need to train and test the model to achieve better performance results. Model complexity is one of the factors that affect performance. The complexity of the model decides the overfitting, underfitting, and appropriate fitting decisions made by the model. Overfitting occurs when your model learns the training data too well and incorporates details and noise specific to your dataset. A model is overfitting when it performs great on your training/validation set but poorly on the test set (or new real-world data). Underfitting occurs when your model over-generalizes and fails to incorporate relevant variations in your data that would give your model more predictive power. A model is underfitting when it performs poorly on training and test sets. Best Fit /Appropriate fit occurs when your model actualizes and learns the patterns of data to incorporate relevant variations in your data that would give your model more predictive power. A model is Best /Appropriate fitting when the model performs excellently in both training and testing datasets. Overfitting, Underfitting and Appropriate fitting are explained in figure 3.7.



***Fig:3.14:*** *Model complexity factors*

We need a model to appropriate fit where it can go wrong in some values in prediction but gets the good majority of it. The hyperparameter tuning can help avoid overfitting and boost the performance from underfitting, which lands on the best fit model.

Hyperparametric tuning to the Naive Bayes model may avoid overfitting. Still, by the nature of the Naive Bayes algorithm, it believes all parameters are independent and equal. It excepts that no. datapoints of all categories of prediction attribute should be equal or with the minimum difference in the count. There is a vast difference in the number of data points of all categories in our dataset in the prediction attribute. So, model testing performance varies when training data contains inconsistent data of two categories of data points. To deploy the best performing NB model, we need to use the Random sampling method, test model performance iteratively, and save the model when model performance reaches the best criteria of performance metrics.

***3.5.1 Random Sampling Experiment***

Random sampling is a part of the sampling technique in which each sample has an equal probability of being chosen. A sample chosen randomly is meant to be an unbiased representation of the total population.

To achieve the best-performing model, we need to choose the primary performance metric to test and evaluate the model. In our case, I have chosen log loss as the primary metric to test and evaluate the model. By randomly splitting data, we get data samples that may or may not get balanced data. Still, we repeatedly split the data randomly to the point where training data achieve the best examples and reasonable distribution of data of prediction attribute, which is tested and evaluated with primary performance metric log loss. We need to set the performance barrier to halt the repetition, which can be concluded from preliminary model testing. Whenever the model reaches this performance barrier, we stop iterations and test the model with other metrics for model evaluation.

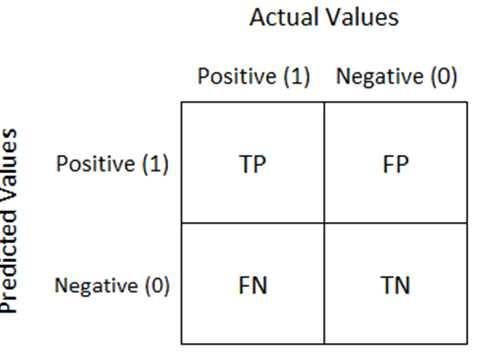
**3.6 Performance metrics**

Many learning algorithms have been proposed to date. It is generally expected that it is essential to evaluate the efficiency of an algorithm. Individuals even ended up making their metrics that suit the application. Performance metrics are often used to test and evaluate the model performance based on the requirement. For this project, the problem type is classification. In this review, we see the absolute most common measurements in a classification setting of a problem. Possible Performance metrics for classification problems are [[4]](https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007)

* + - Confusion Matrix
    - F1score (balances the precision and recall)
    - Log loss
    - Accuracy score
    - ROC curve
    - ROC-AUC score (Area Under Curve in ROC Graph)

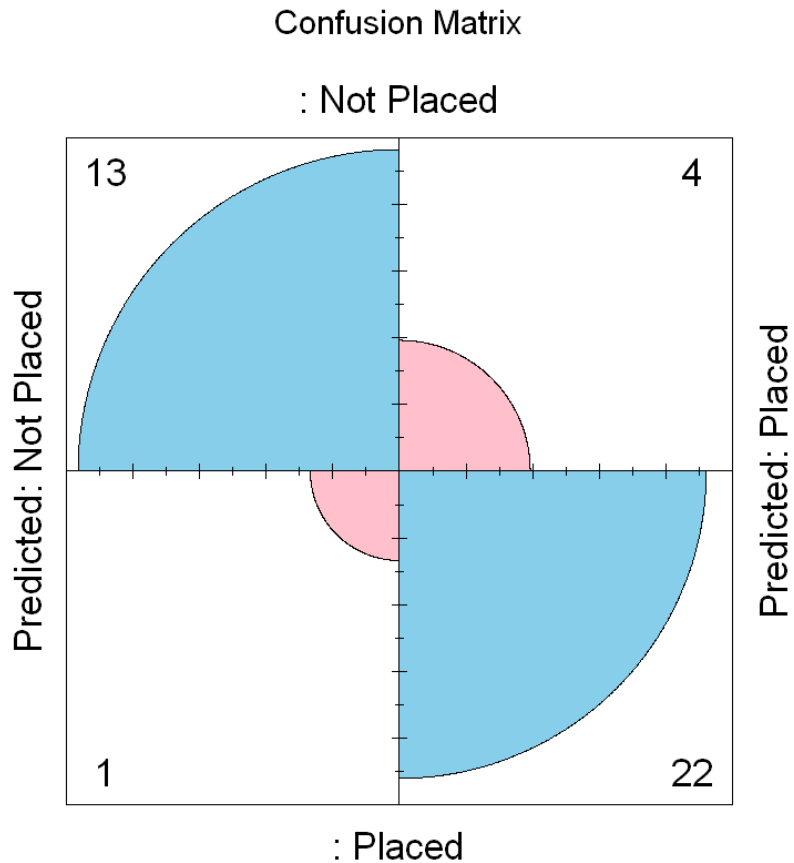
***3.6.1 Confusion Matrix***

A confusion matrix is a technique used to perform a classification algorithm. A confusion matrix is used to evaluate the accuracy of classification. By definition, a confusion matrix C is such that Ci,j equals the number of observations known to be in group i and predicted to be in group j. Thus in binary classification, the count of true negatives is C0,0, false negatives are C1,0, true positives are C1,1, and false positives are C0,1 [[9]](https://www.delftstack.com/howto/r/visualize-confusion-matrix-in-r/). The tp, fp, tn, fn are shown in figure 3.10



***Fig:3.15:*** *Structure of confusion matrix for binary classification*

R has a function called fourfoldplot in caret library, which is used for the color-separated plot of the confusion matrix. With hyperparameters from bestmodel, the Naive Bayes model gives the confusion matrix result when tested with testing data, plotted using a fourfoldplot as shown in figure 3.11.



***Fig:3.16:*** *Confusion matrix for testing data*

The Confusion Matrix has four parameters, as seen in figure 3.10. They are:

• tp: true positive

• fp: false positive

• fn: false negative

• tn: true negative

The confusion matrix structure is a 2 x 2 matrix for binary classification because of tp, fp, tn, fn. Performance measurements such as Accuracy, Sensitivity, Specificity, Precision, F1-Score, Negative Predictive Value, False Positive Rate, False Discovery Rate, and False Negative Rate are calculated using figure 3.10. Calculation formulas are given in Table 3.9.

**Table 3.10:** Performance Measurements

|  |  |  |
| --- | --- | --- |
| Sno. | Performance measure | Formula |
| 1 | Accuracy |  |
| 2 | Sensitivity |  |
| 3 | Specificity |  |
| 4 | Precision |  |
| 5 | F1-Score |  |
| 6 | Negative Predictive Value |  |
| 7 | Negative Positive Rate |  |
| 8 | False Discovery Rate |  |
| 9 | False Negative Rate |  |

* + The F1 score can be explained as a weighted average of precision and recall [[4]](https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007).
  + It is calculated with tp,tn,fp, and fn obtained from the confusion matrix
  + **F1 score reaches its best at value 1 and worst score at 0.**

***3.6.2 Log loss***

Log loss is also known as cross-entropy loss function. The loss function used in logistic regression and extensions such as neural networks is the negative log-likelihood of a logistic model that returns actual output predictions probabilities for its training data. The log loss is only defined for two or more labels. For a single sample with true label  and a probability estimate  the log loss is defined as [[4]](https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007.)[[11]](file:///D:\COLLEGE\SEM%205\PT-1-Machine%20Learning\Sklearn.metrics.log_loss.%20scikit.%20(n.d.).%20Retrieved%20November%205,%202021,%20from%20https:\scikit-learn.org\stable\modules\generated\sklearn.metrics.log_loss.html):

By calculating log loss with an NB model that has been developed with hyperparameters and random sampling data that gives a value of 0.168

***3.6.3 Accuracy score***

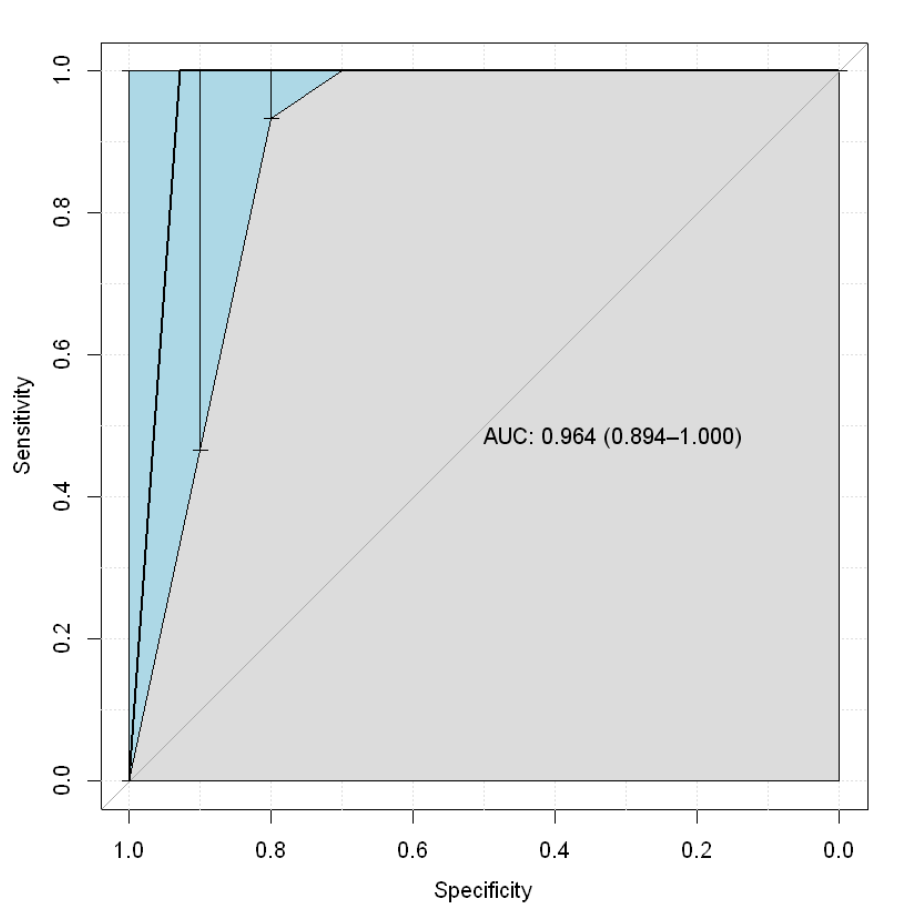
Accuracy score is the measurement that tests the model's performance by an average of output prediction vs. actual output [[4]](https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007.).

* + It ranges from 0 to 1, where 0 is worst and 1 is Best.
  + It is often multiplied by 100 to get the accuracy percentage

The accuracy score of the NB model with hyperparameters is **97.72%.**

***3.6.4 ROC AUC curve***

ROC curve is plotted against Sensitivity Vs. Specificity. Greater the area under the curve(AUC) better the model. The roc curve of the RF model [[10]](https://rviews.rstudio.com/2019/03/01/some-r-packages-for-roc-curves/) is shown in figure 3.12.



***Fig:3.17:*** *ROC curve of NB model*

***3.6.5 ROC AUC score***

Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

* + It is used to find Area Under Curve(AUC) in the ROC graph.
  + It ranges from 0 to 1, where 0 is the worst and 1 is the Best

ROC AUC score of the NB model with hyperparameters is **0.** **96.**

Chapter-4

**RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS**

Characterization of the student placement data points utilized in our review is collected from the universities placement cell, and acquired data points of 215 were obtained. Out of 14 features of datasets, 12 of them are used to train the model, and one feature is used for prediction. One attribute called salary has been removed. A model has been created and tested performance using Naive Bayes classifier (NB) [[3]](https://cran.r-project.org/web/packages/naivebayes/naivebayes.pdf) machine learning techniques for classification.

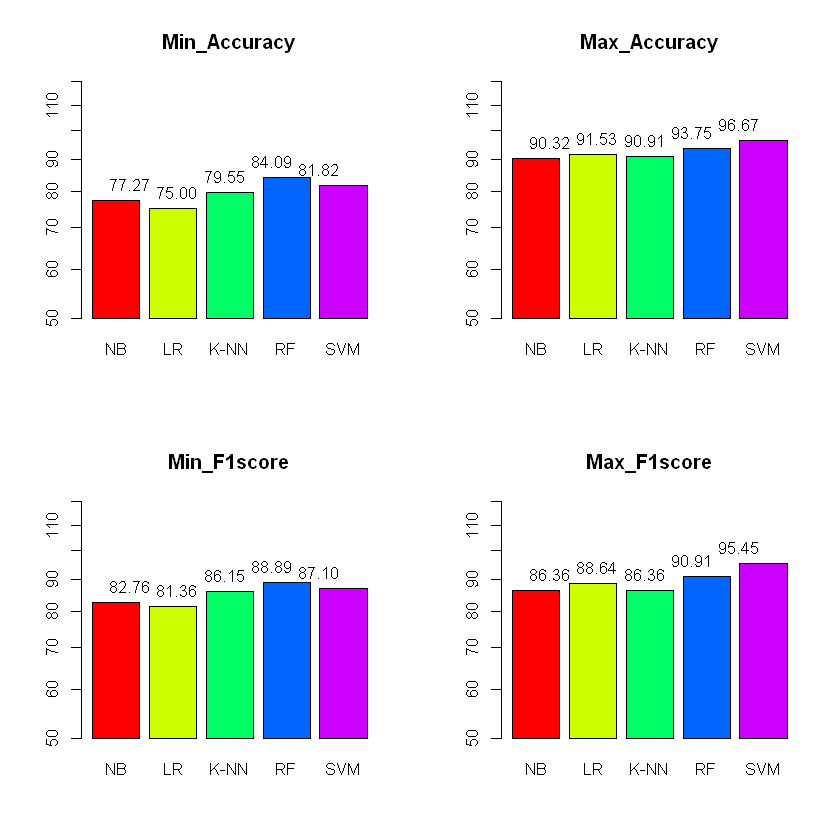
**4.1 Analyzing performance**

To test the model's performance and efficiency of the dataset the model, we used performance metrics for the Naive Bayes model, which is tuned with initial hyperparameters. To evaluate the dataset, we used other ML algorithms like Logistic Regression, K-NN, Random Forest, and Support Vector Machine to implement performance comparison with the NB model. In this model comparison, we concluded that the impact of imbalance data is present in all algorithms by testing with performance metrics. For evaluation of the model, we used k-fold cross-validation. The results of model performances are shown in table 4.1.

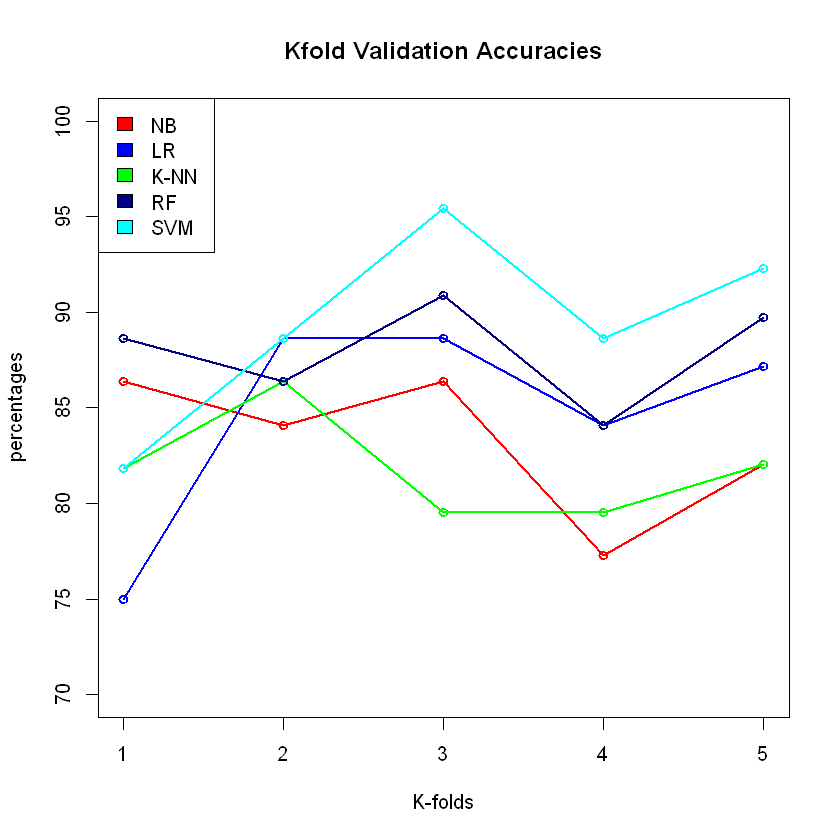
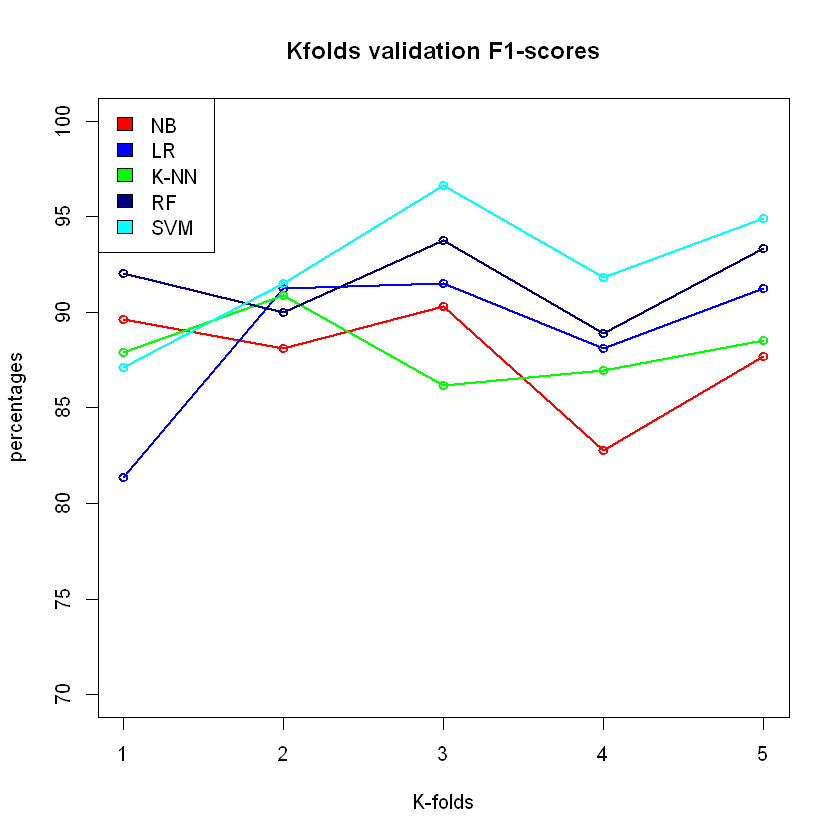
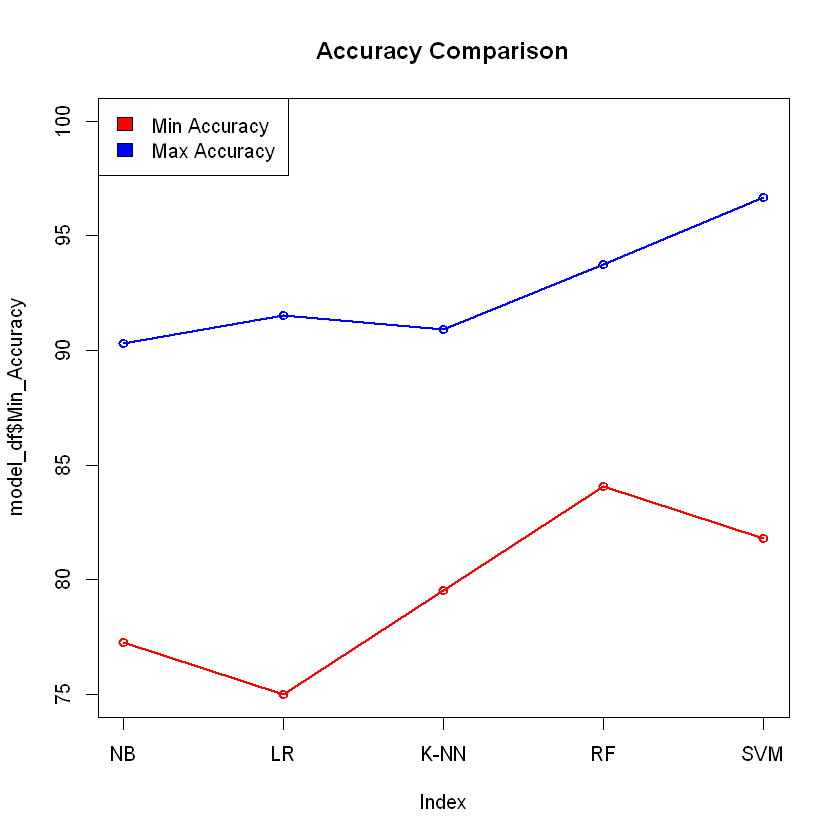
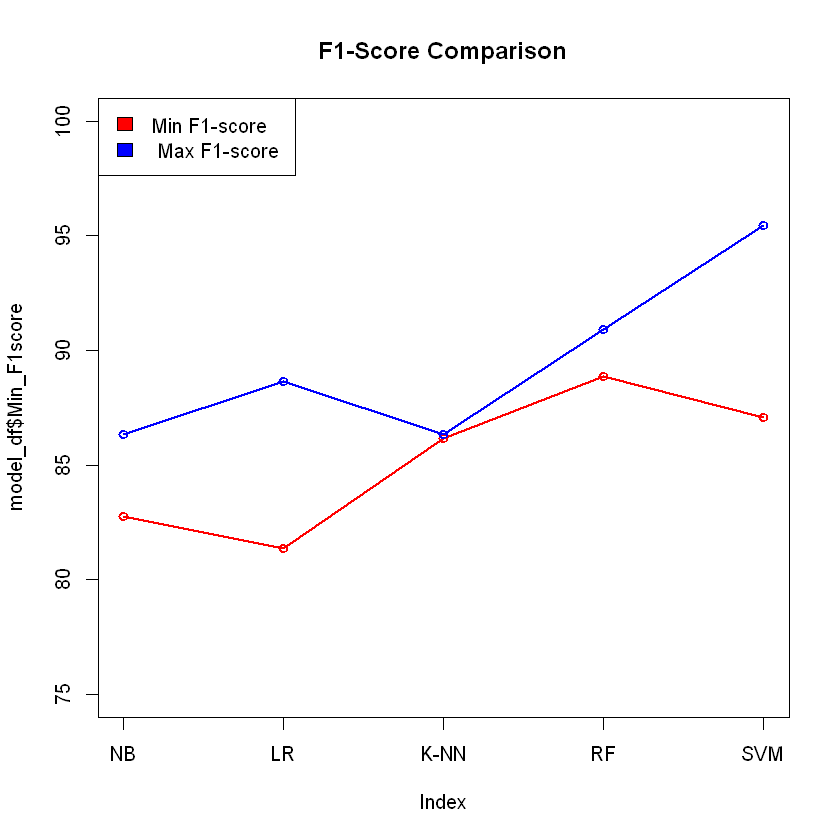
**Table 4.1:** Different Algorithms Model performances

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Min\_Accuracy | Max\_Accuracy | Min\_F1score | Max\_F1score | KFold\_Accuracy | KFold\_F1score |
| Naive Bayes | 77.27273 | 90.32258 | 82.75862 | 86.36364 | 83.22844 | 87.71825 |
| Logistic Regression | 75.00000 | 91.52542 | 81.35593 | 88.63636 | 84.70862 | 88.69462 |
| K-NN | 79.54545 | 90.90909 | 86.15385 | 86.36364 | 81.86480 | 88.08457 |
| Random Forest | 84.09091 | 93.75000 | 88.88889 | 90.90909 | 87.94872 | 91.60714 |
| SVM | 81.81818 | 96.66667 | 87.09677 | 95.45455 | 89.37063 | 92.40148 |

We can plot some graphs from the performance table to better understand different algorithms' performances on the dataset. The graphs are shown below figures.



***Fig:4.1:*** *Bar graphs for performance of model*

***Fig:4.2:*** *Line graphs for performance of models*

***Fig:4.3:*** *Line graphs for K-fold performance of models*

From the above graphs, we can conclude that the Random forest and Support vector machine is the best-performing Algorithms for this dataset, even though there is an inconsistent performance in testing. This inconsistent model performance is due to imbalanced data, which is also proved by k-fold cross-validation. In figure 4.3 best performing fold is 3, and the worst-performing fold is 1, which is the fold used for testing the remaining folds used for training. If we use 1,2,4,5 among 5 folds for training, we get the best performance from every model. In my work, I wanted to experiment with a Naive Bayes algorithm. To achieve the best-performing model, we need to do a random sampling experiment to get the best performing data samples. In this random sampling experiment, we split training and testing samples randomly and tested the performance as mentioned in random sampling experiment chapter 3.5.1. After testing the model with a random split sample and evaluating the model with Log Loss, we reach a model accuracy of **97.72%** with a minimum LogLoss of 1.12.

**4.2 Final results**

After tuning the model with hyperparameters and training data with random sampling, the final result parameters used to build the model are mentioned in table 4.1.

**Table 4.2:** Parameters of Naive Bayes model

|  |  |  |
| --- | --- | --- |
| **Validation metric** | **Train size** | **Test size** |
| LogLoss | 0.8 | 0.2 |

For this classification problem, the performance metrics calculated for the model are Confusion Matrix, f1\_score, logloss, accuracy score, and roc\_auc score. These performance calculations are presented in table 4.2.

**Table 4.3:** Results of Naive Bayes model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Confusion matrix** | **F1score** | **Log loss** | **Roc Auc score** | **Accuracy** |
| |  |  | | --- | --- | | 13 | 1 | | 0 | 13 | | 0.98360 | 0.12246 | 0.96428 | 97.72% |

**SUMMARY AND CONCLUSIONS**

**5.1 Summary of the Project**

In this study, a qualitative methodology has been used to reach our aim. A student placement prediction solution is needed when there is a requirement to know whether a student gets placement or not. Depending on the probability of placement, we can improve a student's academic performance to increase placement probability in dream companies. For this problem, the solution is built on considering student academic attributes like pass percentages, board of education, the branch of degree, employment test results, and more. The main attribute that plays a significant role in prediction is employment test results among all of these attributes. Many students can't get placed in their dream companies due to a lack of communication and basic technical skills. If we could know the student's commonly lacking academic areas, we could introduce new programs to enhance their skills. However, if we could narrow it down to knowing students who may not get placed, we can keep more focus on those students to enhance their skills by giving training on lacking skillset. ML algorithms make a perfect solution for our problem of student placement prediction.

Naive Bayes classifier is one of the ML algorithms that can solve this problem. NB algorithm assumes that each feature in the dataset is independent and equal. This algorithm is based on the Bayes theorem, which calculates posterior probability. The Naive Bayes algorithm creates a frequency table known as a look-up table for categorical data and a mean and standard deviation for numerical data. During the development of the NB algorithm, a look-up table is created for every feature. While calculating the posterior probability of the given input data, each feature's likelihood, prior probability, and evidence are calculated according to input categories. Based on the calculated values, the highest probability of the prediction class category prediction of the calculated probabilities is given as output.

To solve this classification problem, we need to build a dataset of student academic performance with labels of getting placed and not placed used to train the NB algorithm for placement prediction. Dataset can be prepared by collecting past academic data of students from universities and selecting important attributes for the dataset. These extracted features are stored in comma-separated value(.csv) format. These self-prepared or computerized datasets may not always be perfect, and they consist of missing values and fault symbols. We need to understand the dataset and clean it to make it easier for machines to handle these errors. This process is called data preprocessing.

Data preprocessing can be done using integrated R dplyr libraries in a jupyter notebook. After understanding and cleaning data, there is a need to understand the relations between the features and how these relations affect our aim. for understanding relations, a dataset can be visualized using built-in plots like a bar, hist, plot, or we can import library ggplot module and its techniques [[2]](https://scikit-learn.org/stable/visualizations.html.) inside jupyter notebook. We can understand data clearly and build a model according to our requirements with these visualizations. After data preprocessing and understanding, we need to study ML algorithms to build the model.

The naive Bayes algorithm has some tuning parameters called hyperparameters. These hyperparameters can be changed as per the requirement to achieve better performance. Some parameters are assigned once according to the requirement in these hyperparameters and left unchanged for the rest of the testing. These hyperparameters help the model best fit rather than underfitting if there are fewer amounts of data and overfitting for more than enough data. With these parameters, we can build the model for Student placement prediction. But there is a need to validate the performance according to the requirements.

When there is an imbalance in the dataset, we can use resampling techniques to balance data, but when data points in the dataset are low and imbalanced, we need to do random sampling to train data. Random sampling is a technique where we randomly shuffle data and split them into training and testing sets. These training and testing sets are used to build a model and validate the performance. We repeat random sampling until we get acceptable performance from the model.

Performance metrics of classification problems help us validate the model's performance. The performance metrics used are confusion matrix, f1\_score,logloss, accuracy\_score, roc\_auc\_curve, roc\_auc\_score. With these, we can validate our model the requirements of the problem.

At the end of this study, we achieved an accuracy of **97.72%** with the dataset of student academic attributes. By looking at the results when that kind of success rate was obtained, it is possible to say that the study achieved success. We need to know R, R dependent libraries, and ML techniques to understand this algorithm. This Naive Bayes model can be modified and installed into personal computers for placement prediction. This binary classification model can further be remodeled and used for other classification problems or improved model we more data training.

**5.2 Conclusion of the Project**

The Naïve Bayes model is validated with different performance metrics in which we took LogLoss as our primary metric to test performance. The loss function used in logistic regression and extensions such as neural networks is the negative log-likelihood of a logistic model that returns actual output predictions probabilities for its training data. After developing the model with better hyperparameters, we should validate it with other performance metrics. We can test model accuracy against training data with a performance metric accuracy score. Visualizing performance makes us understand the loose ends of the algorithm.

The Student Placement Prediction model for university student evaluation using the Naive Bayes ML algorithm has been successfully developed with 97.72% accuracy. This NB model can be used to predict the placement of the student.

**REFERENCES**

[1] Wikimedia Foundation. (2022, March 22). *Bayes' theorem*. Wikipedia. Retrieved April 11, 2022, from <https://en.wikipedia.org/wiki/Bayes%27_theorem>

[2] *Random sampling*. Corporate Finance Institute. (2021, July 29). Retrieved April 11, 2022, from <https://corporatefinanceinstitute.com/resources/knowledge/other/random-sampling/>

[3] Comprehensive R Archive Network (CRAN). (n.d.). *Package naivebayes*. CRAN. Retrieved April 11, 2022, from <https://cran.r-project.org/web/packages/naivebayes/>

[4] Vidiyala, R. (2020, July 26). Performance metrics for classification machine learning problems. Medium. Retrieved November 2, 2021, from <https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007>

[5] Narkhede, S. (2021, June 15). Understanding confusion matrix. Medium. Retrieved November 2, 2021, from <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>

[6] *LabelEncoder: Label encoder*. RDocumentation. (n.d.). Retrieved April 14, 2022, from <https://www.rdocumentation.org/packages/superml/versions/0.5.3/topics/LabelEncoder> [7] *Correlation matrix : A quick start guide to analyze, format, and visualize a correlation matrix using R software*. STHDA. (n.d.). Retrieved April 15, 2022, from <http://www.sthda.com/english/wiki/correlation-matrix-a-quick-start-guide-to-analyze-format-and-visualize-a-correlation-matrix-using-r-software>

[8] Taiyun Wei, V. S. (2021, November 18). An introduction to corrplot package. Retrieved April 15, 2022, from

<https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html>

[9] Khintibidze, L. (2021, May 16). *Visualize confusion matrix using caret package in R*. Delft Stack. Retrieved April 15, 2022, from

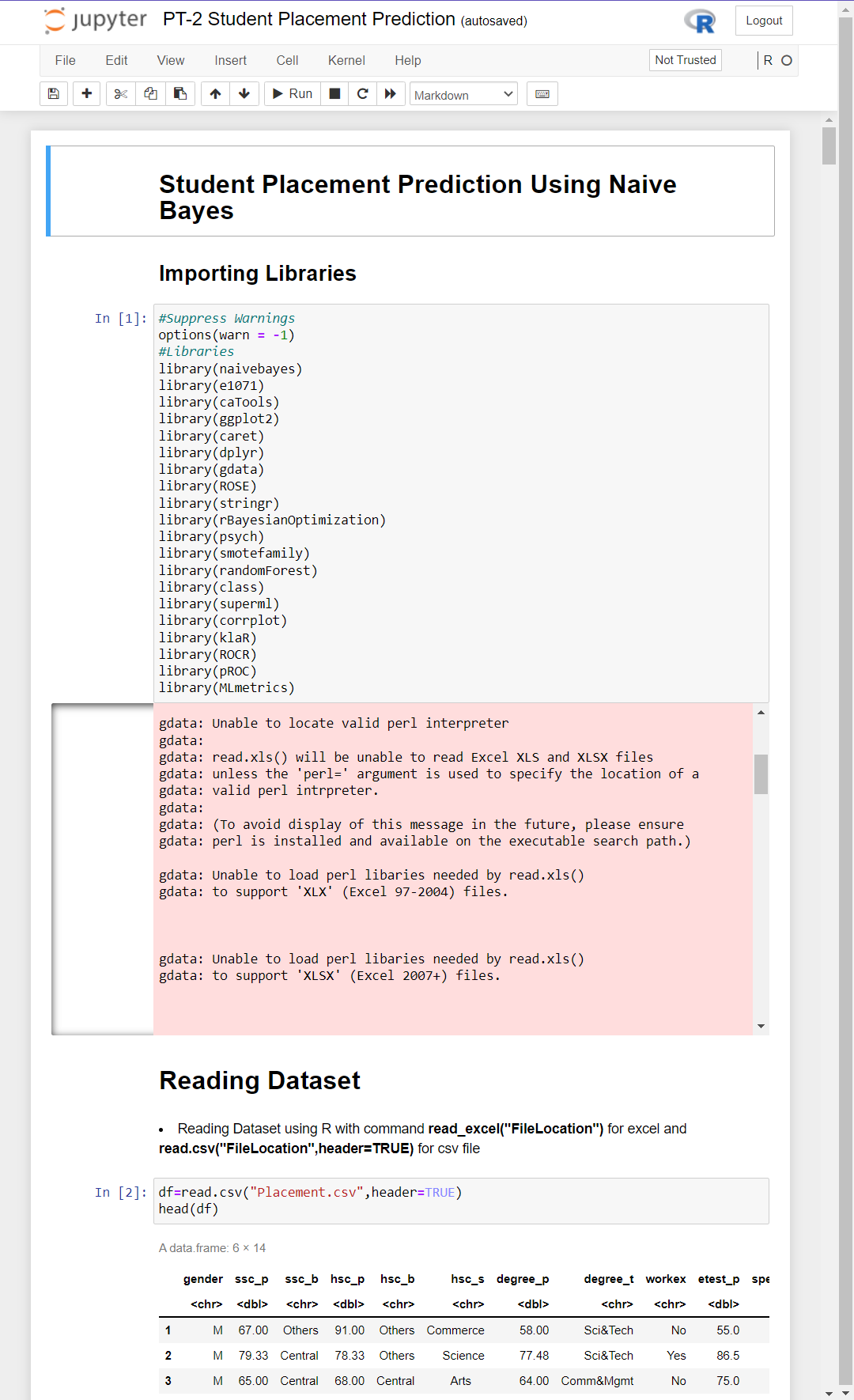
<https://www.delftstack.com/howto/r/visualize-confusion-matrix-in-r/>

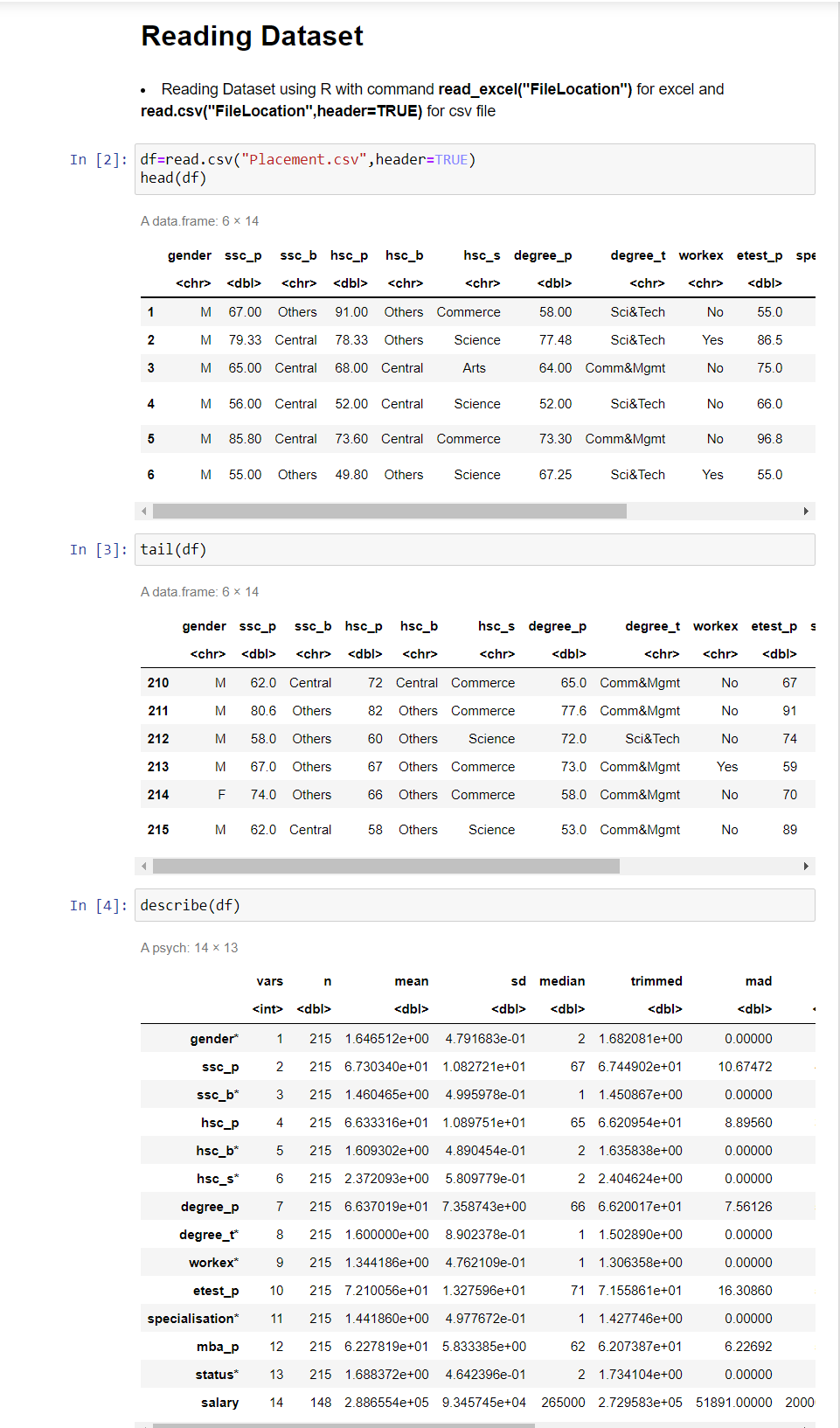
[10] Views, R. (2019, March 1). *Some R packages for ROC curves*. · R Views. Retrieved April 15, 2022, from <https://rviews.rstudio.com/2019/03/01/some-r-packages-for-roc-curves/>

[11] *Logloss: Log loss / cross-entropy loss*. RDocumentation. (n.d.). Retrieved April 15, 2022, from <https://www.rdocumentation.org/packages/MLmetrics/versions/1.1.1/topics/LogLoss>

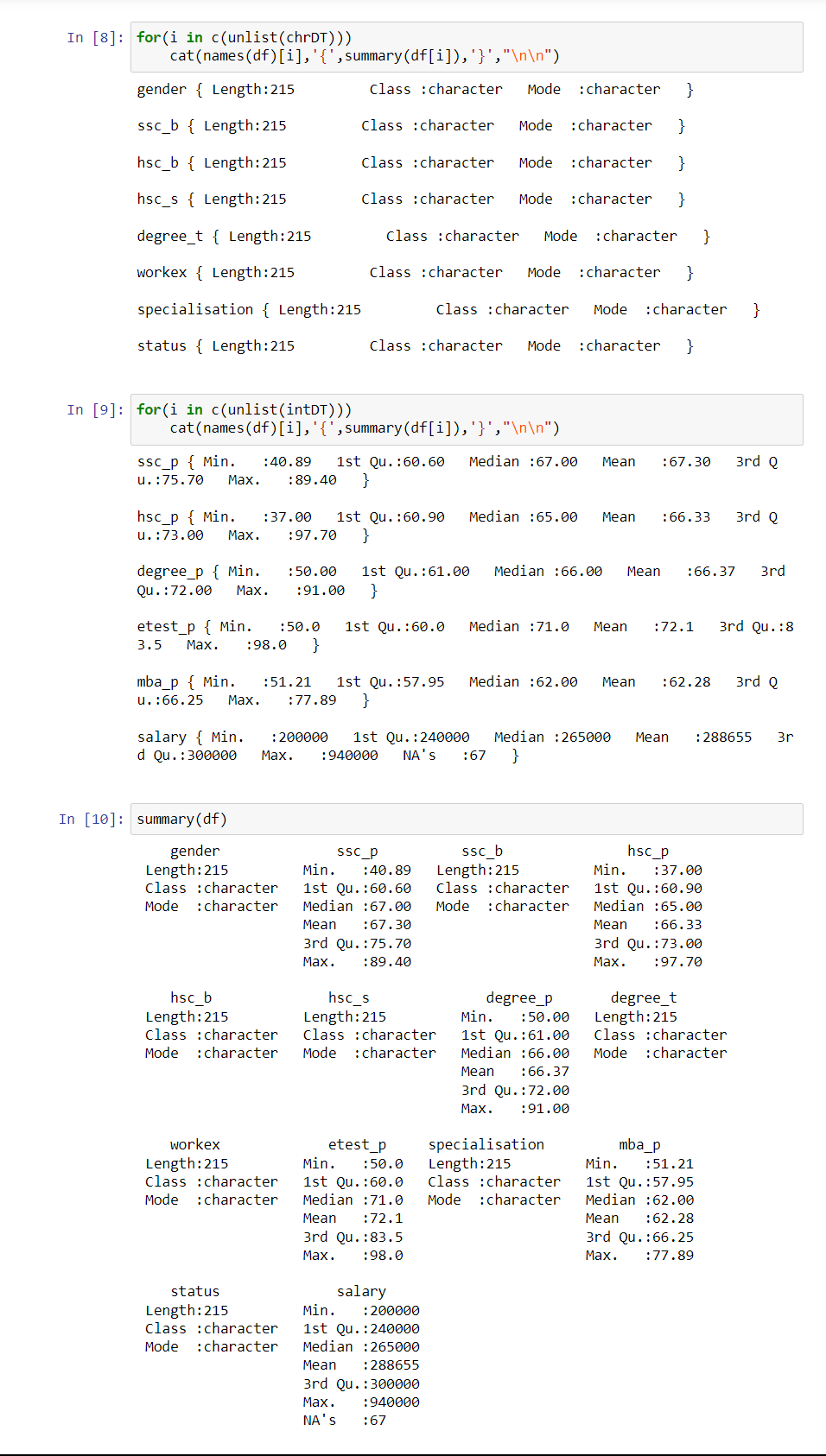
**APPENDIX**

**A. SCREENSHOTS**

****

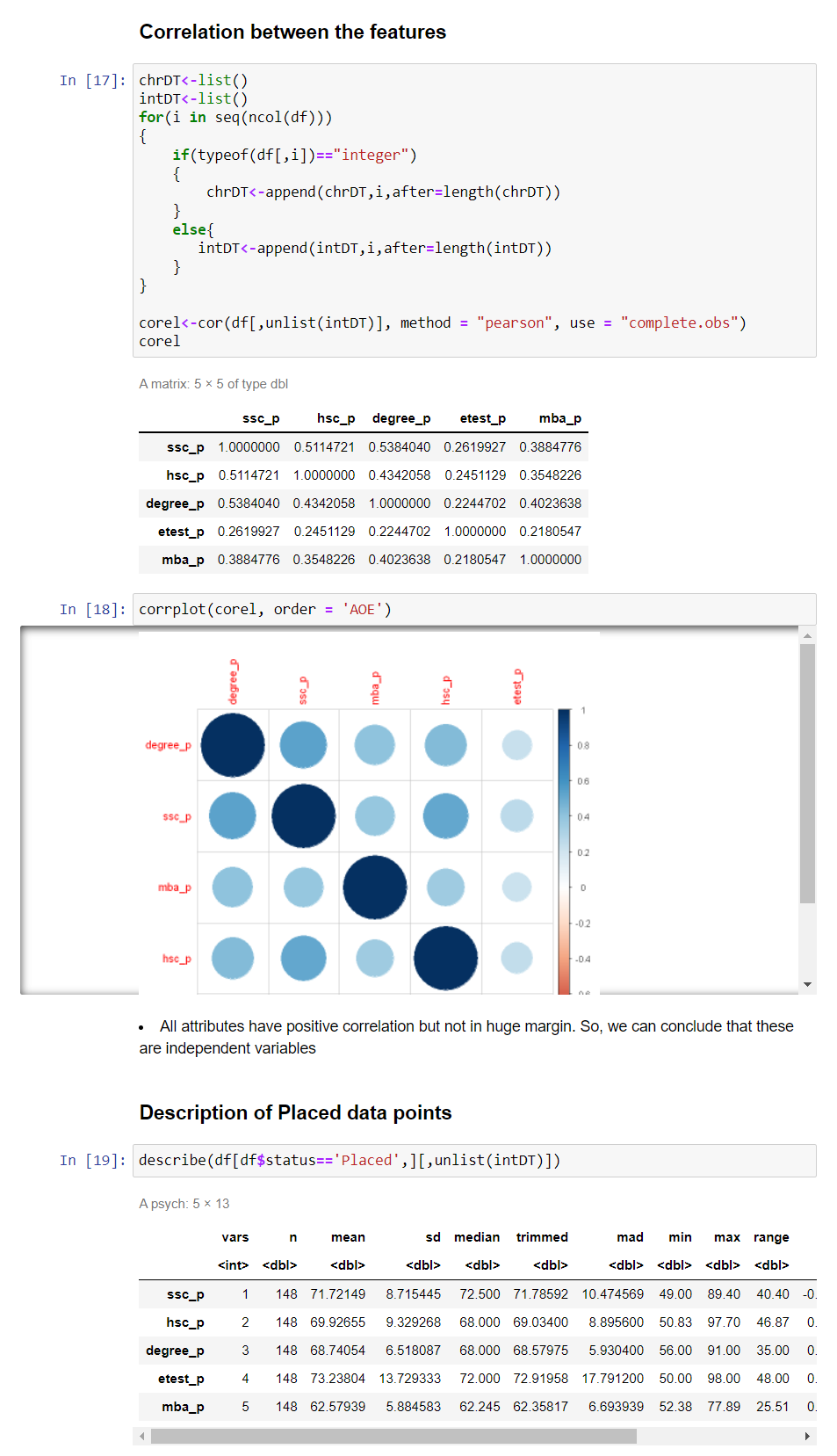
****

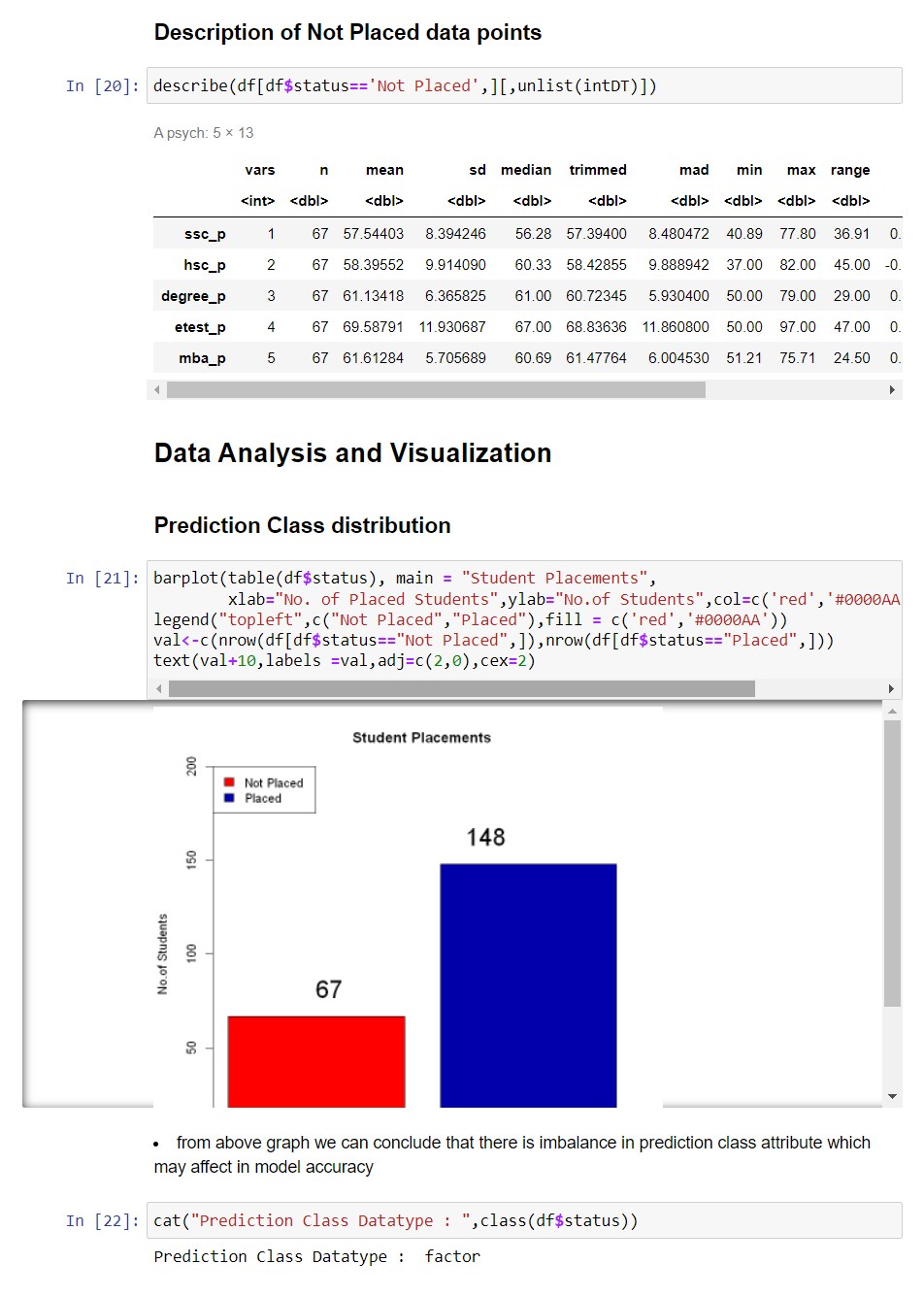
****

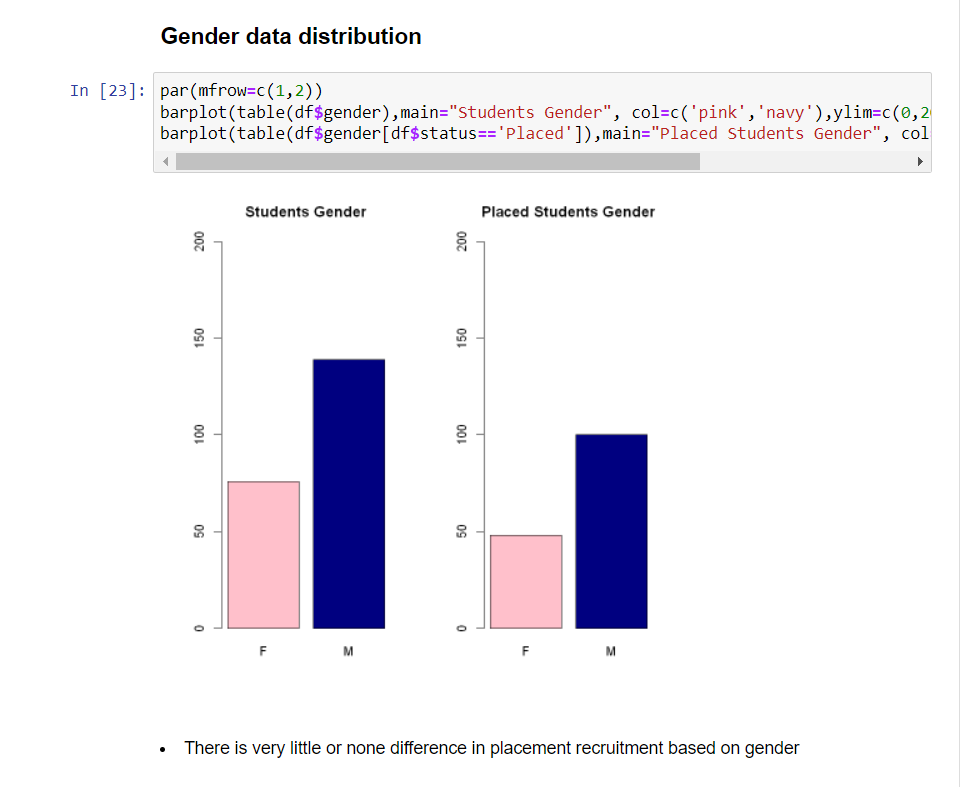
****

****

****

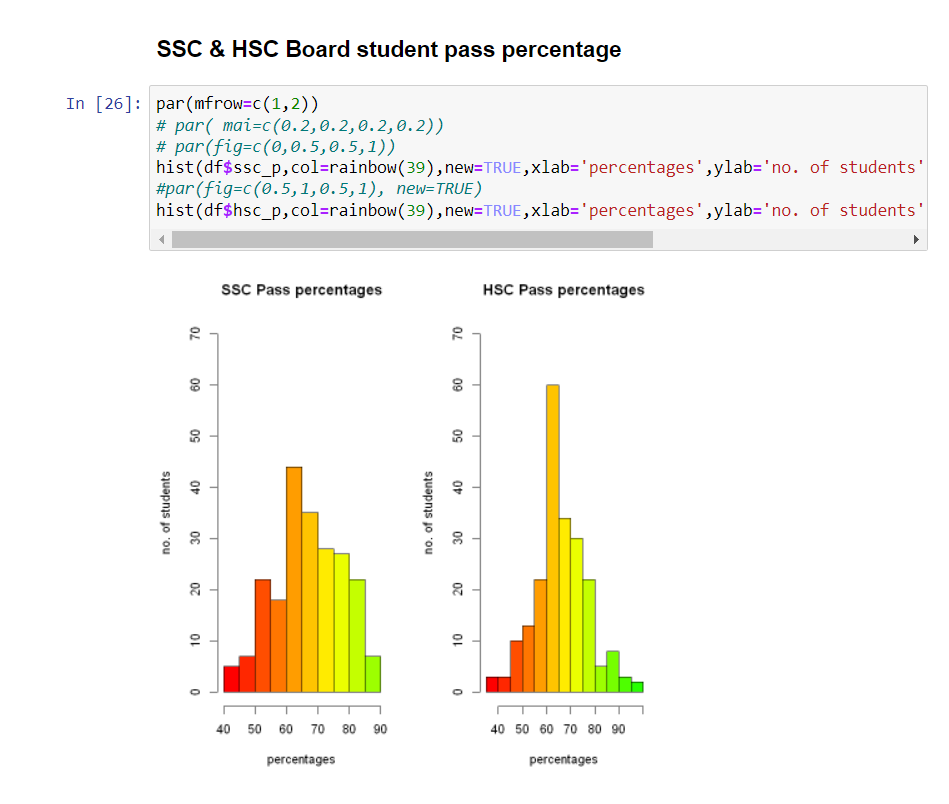
****

****

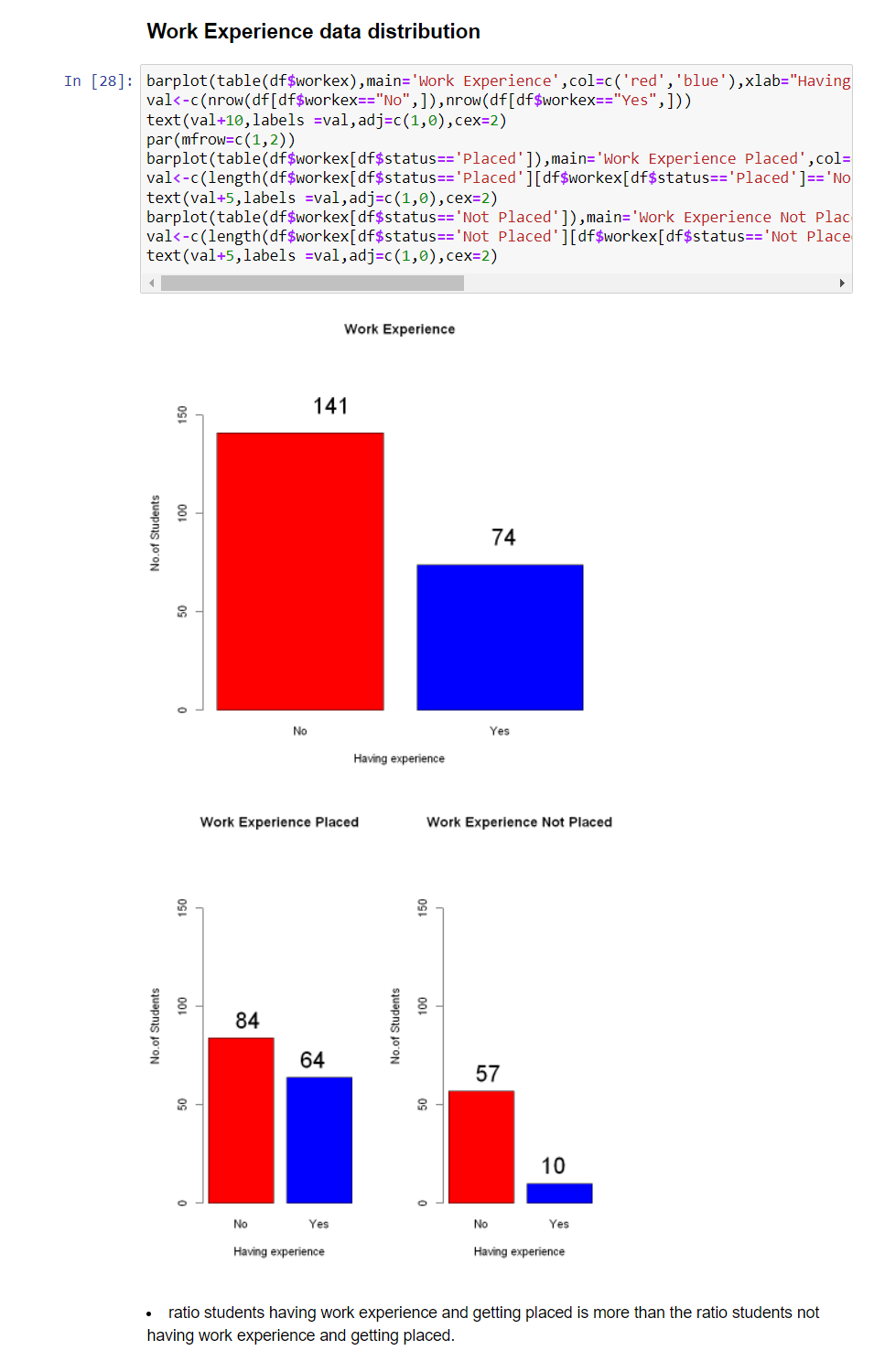
****

****

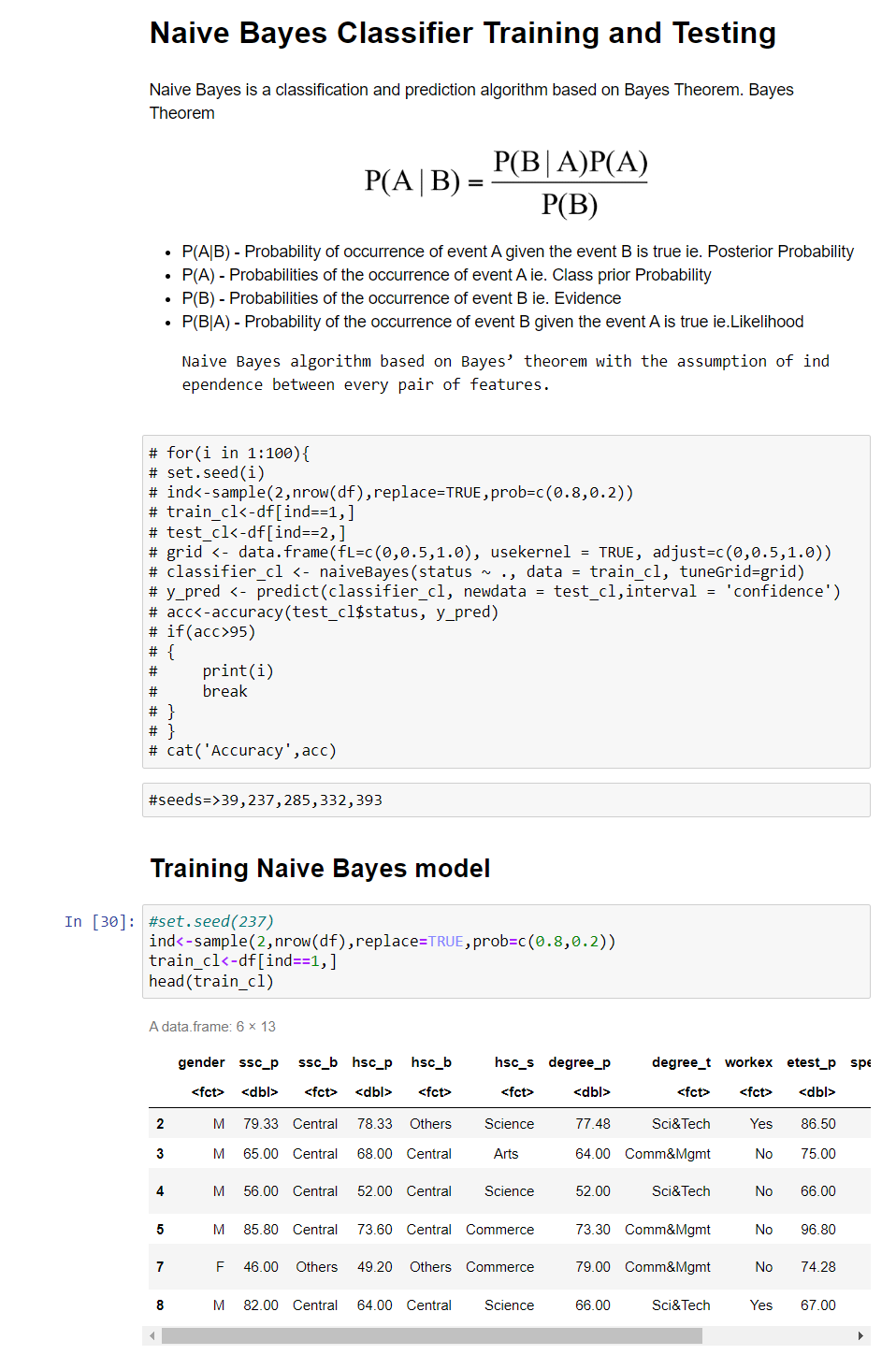
****

****

****

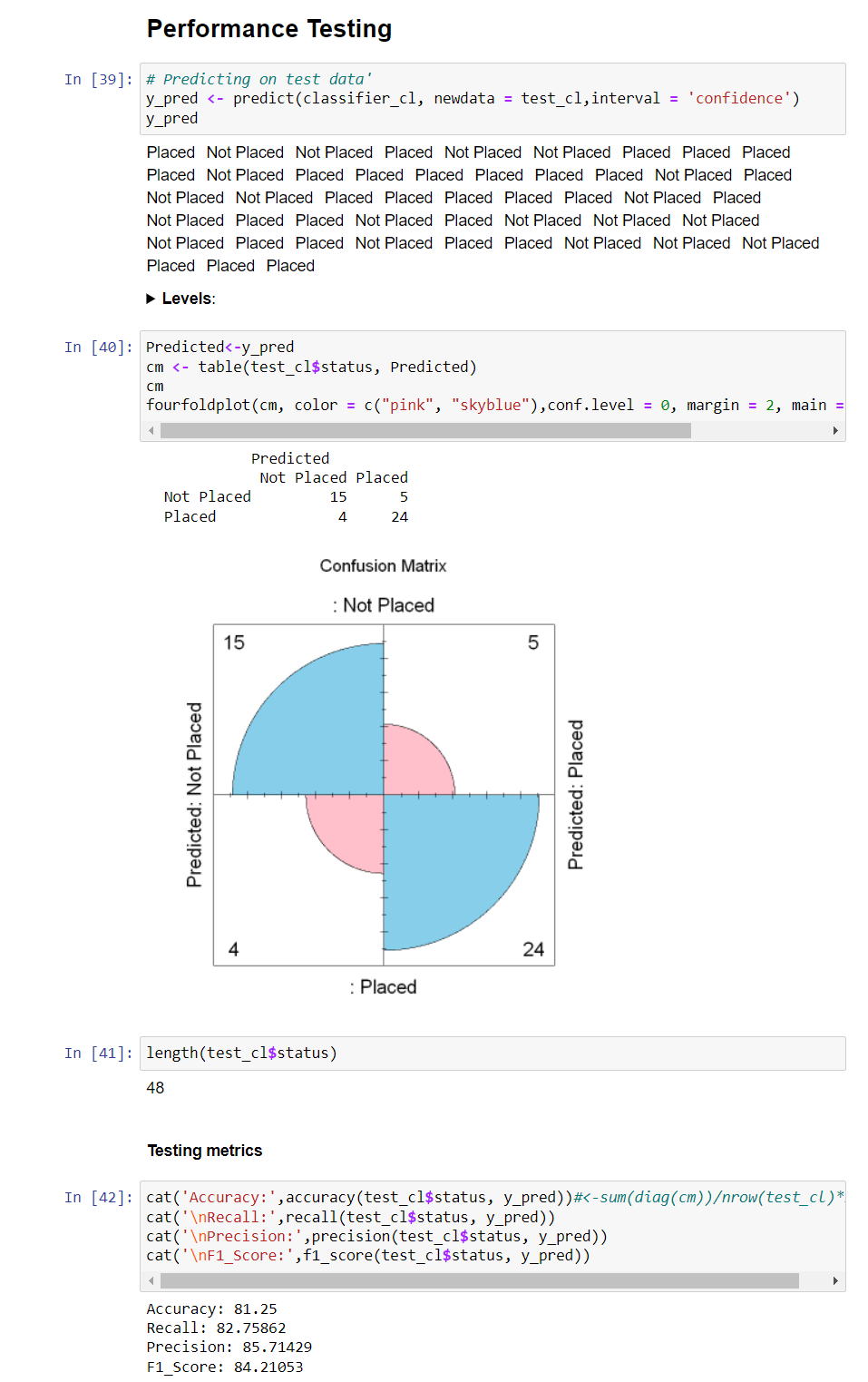
****

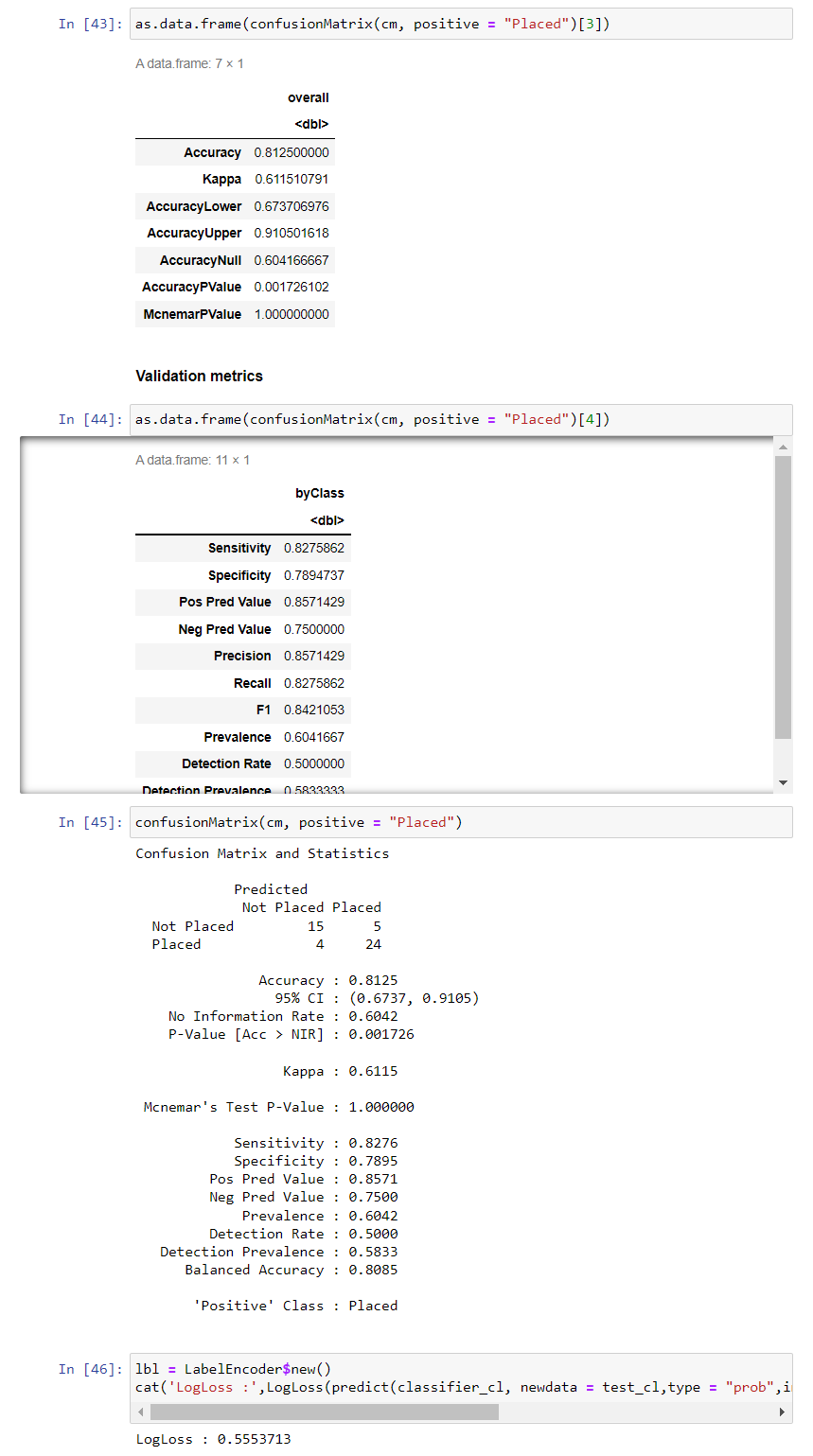
****

****

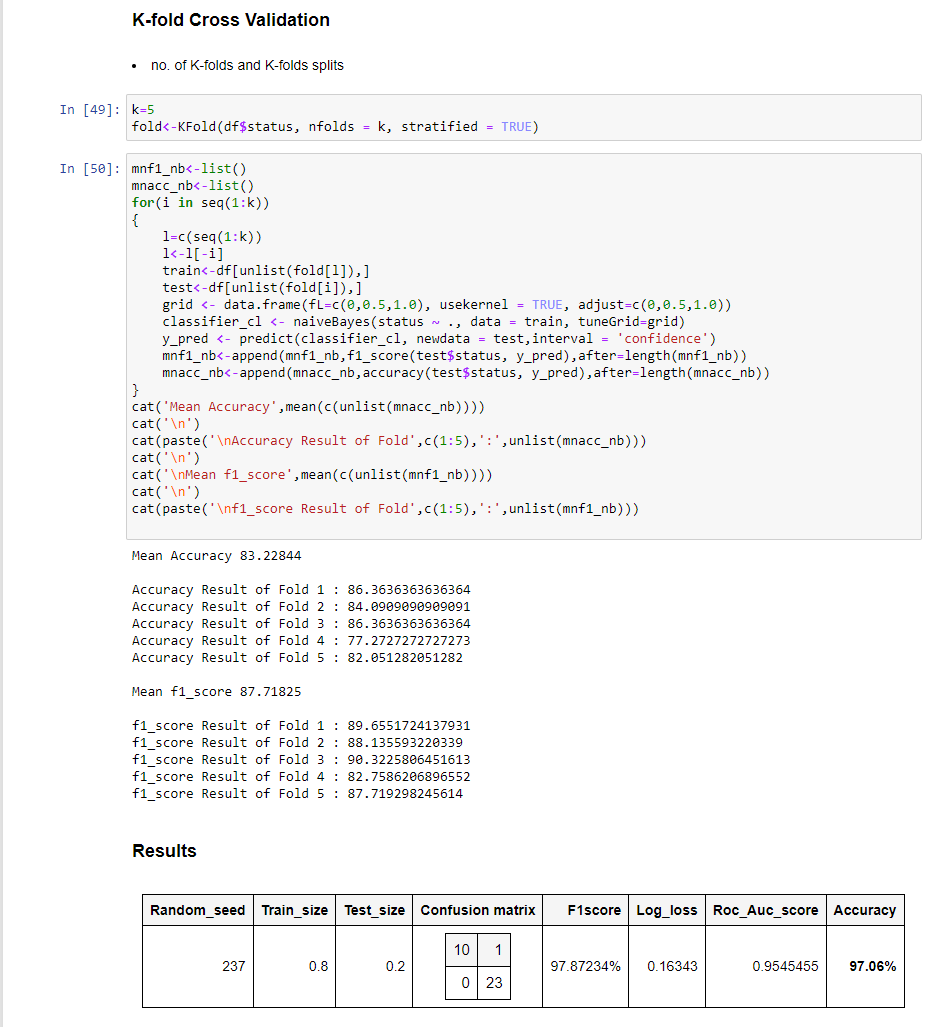
****

****

****

****

****

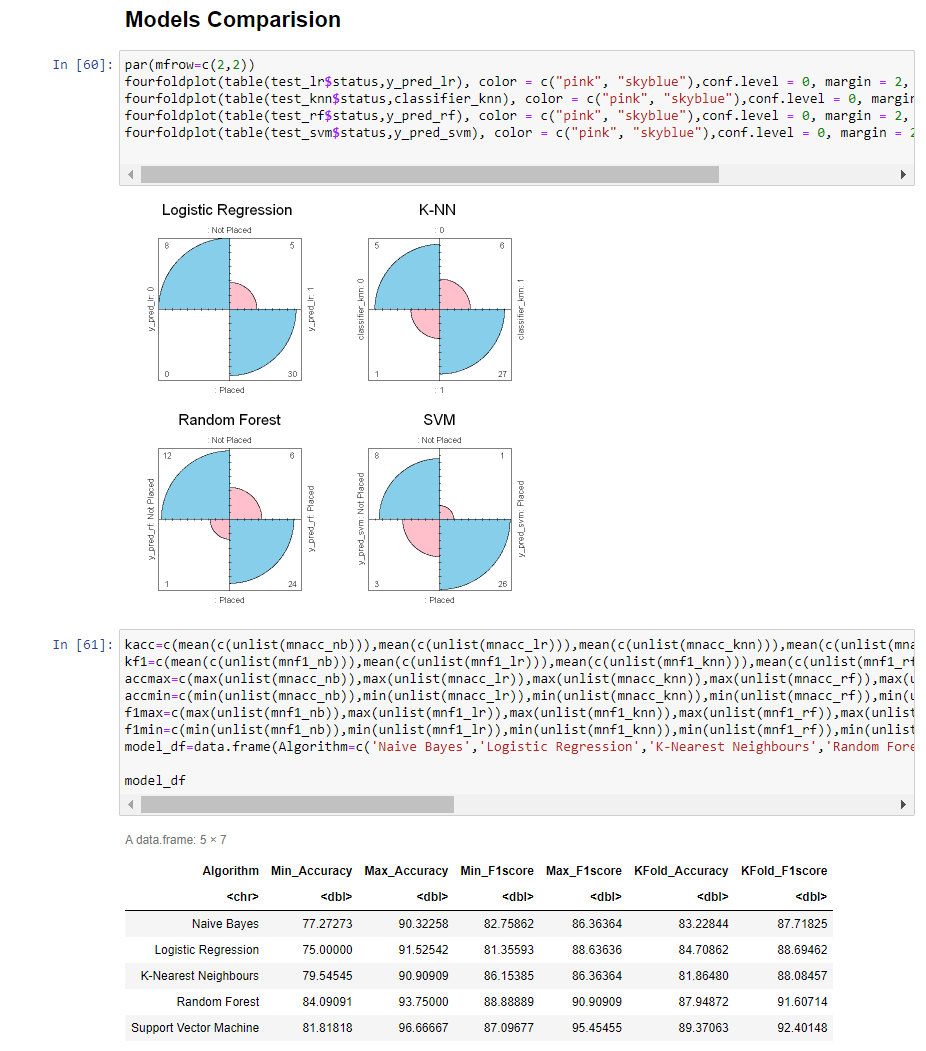
****

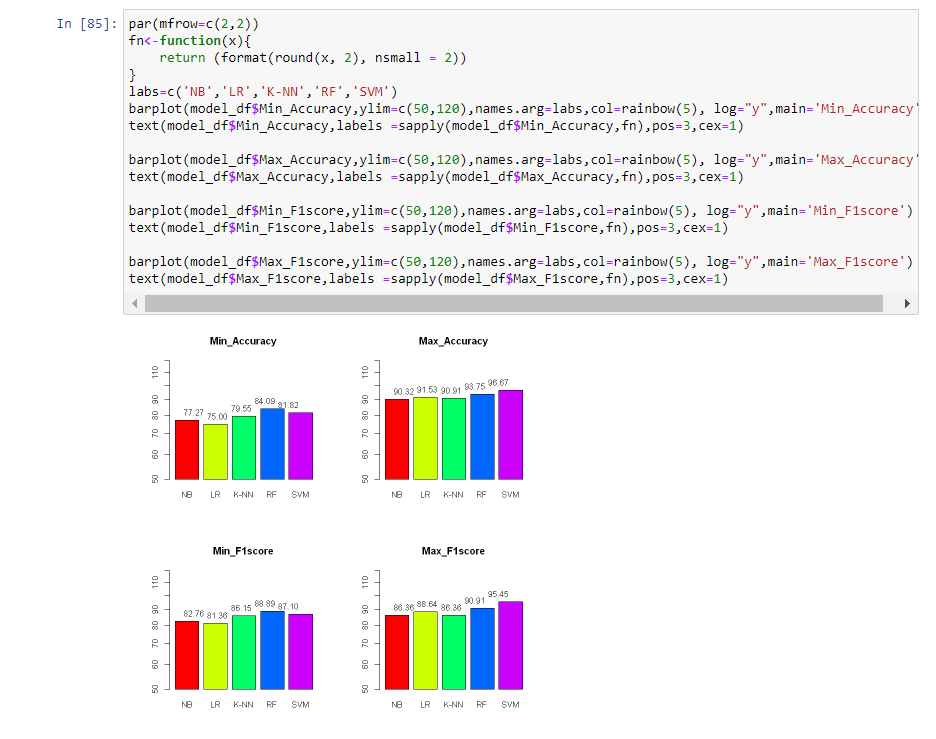
****

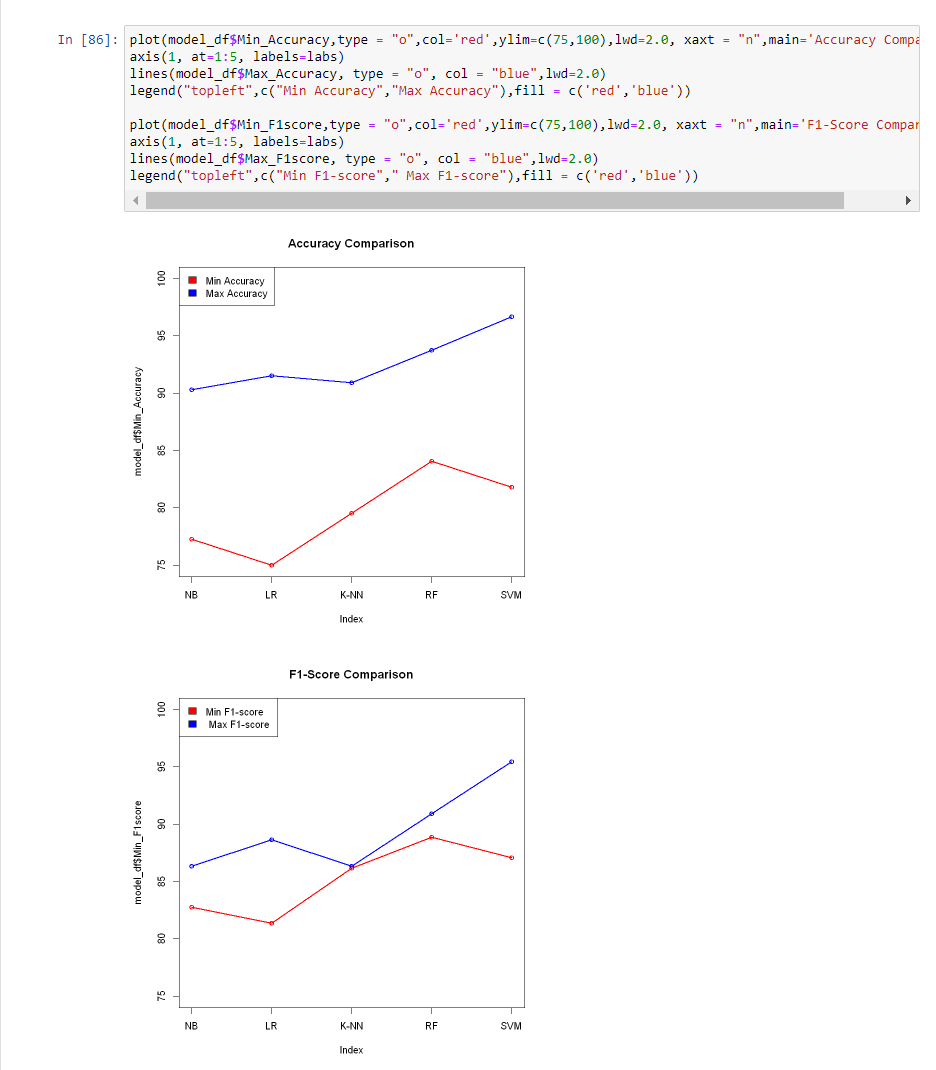
****

****

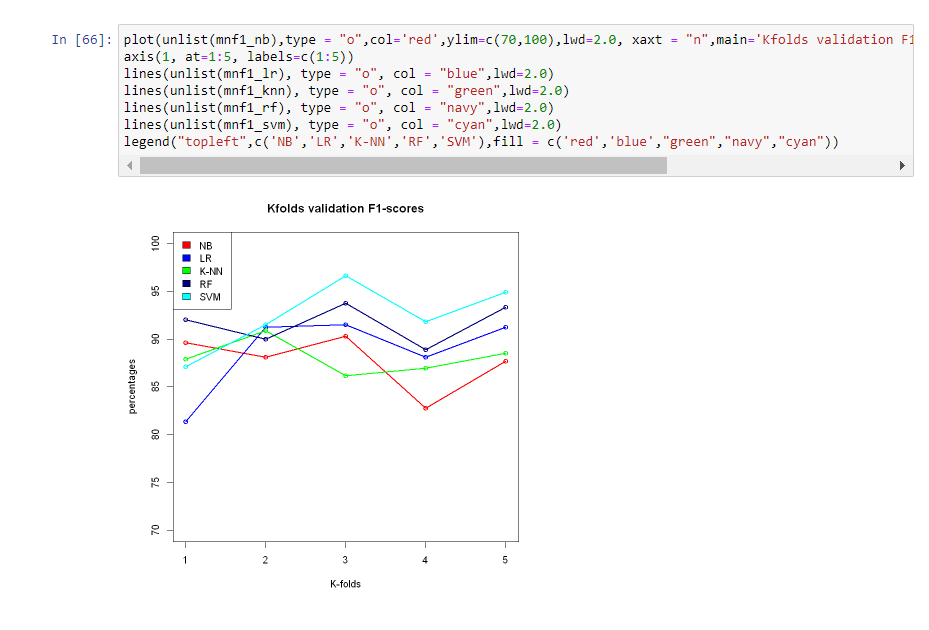
****

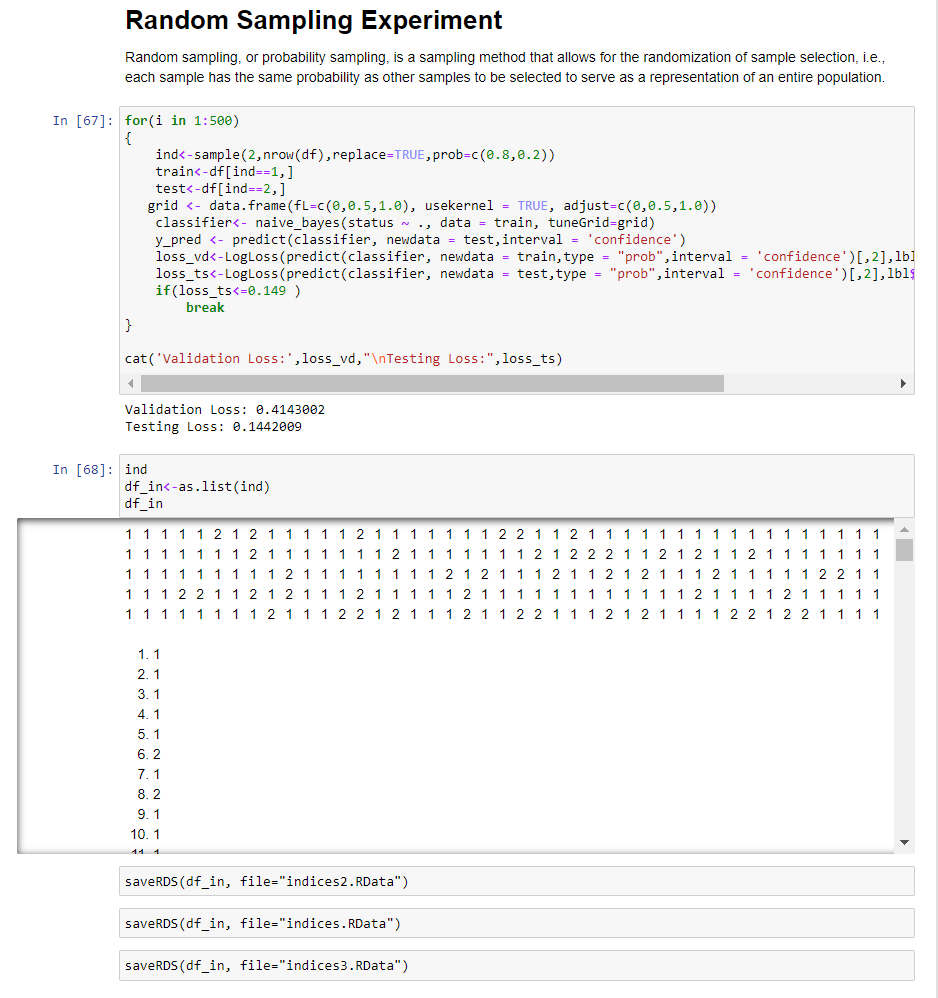
****

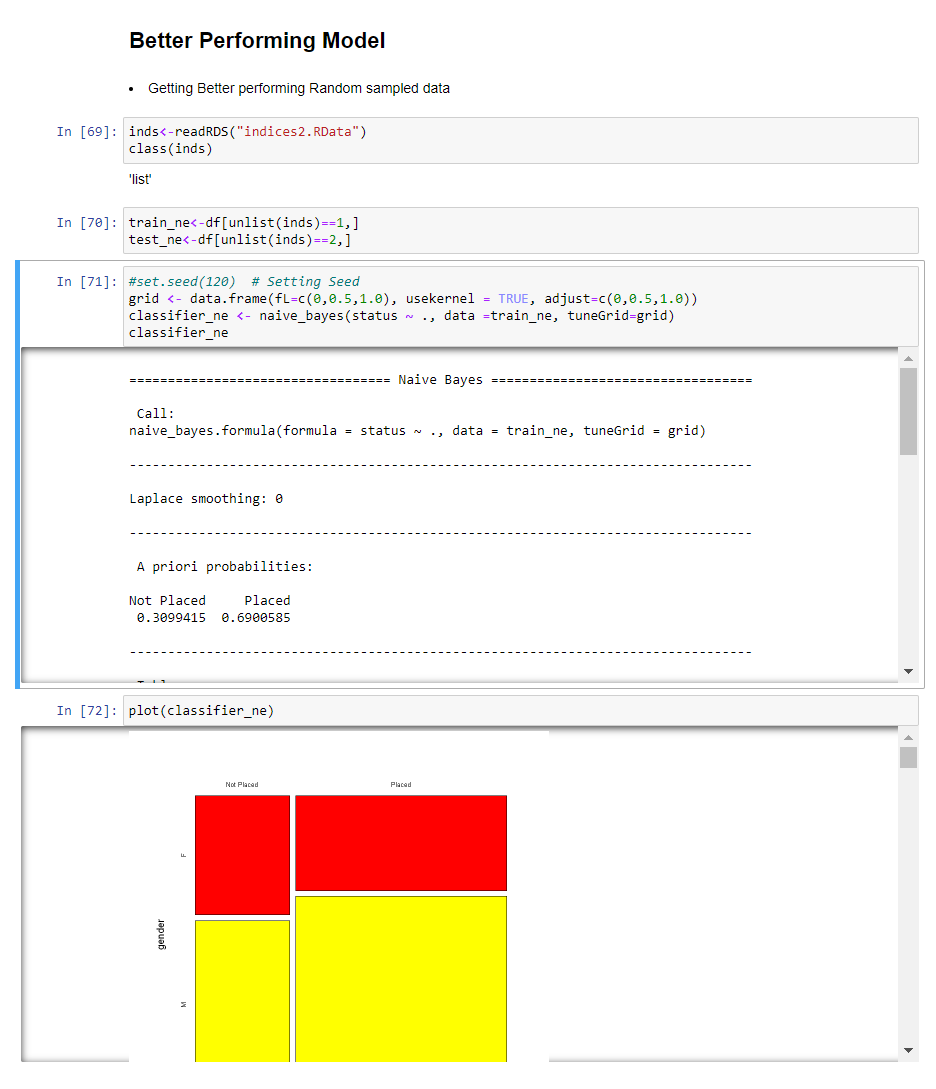
****

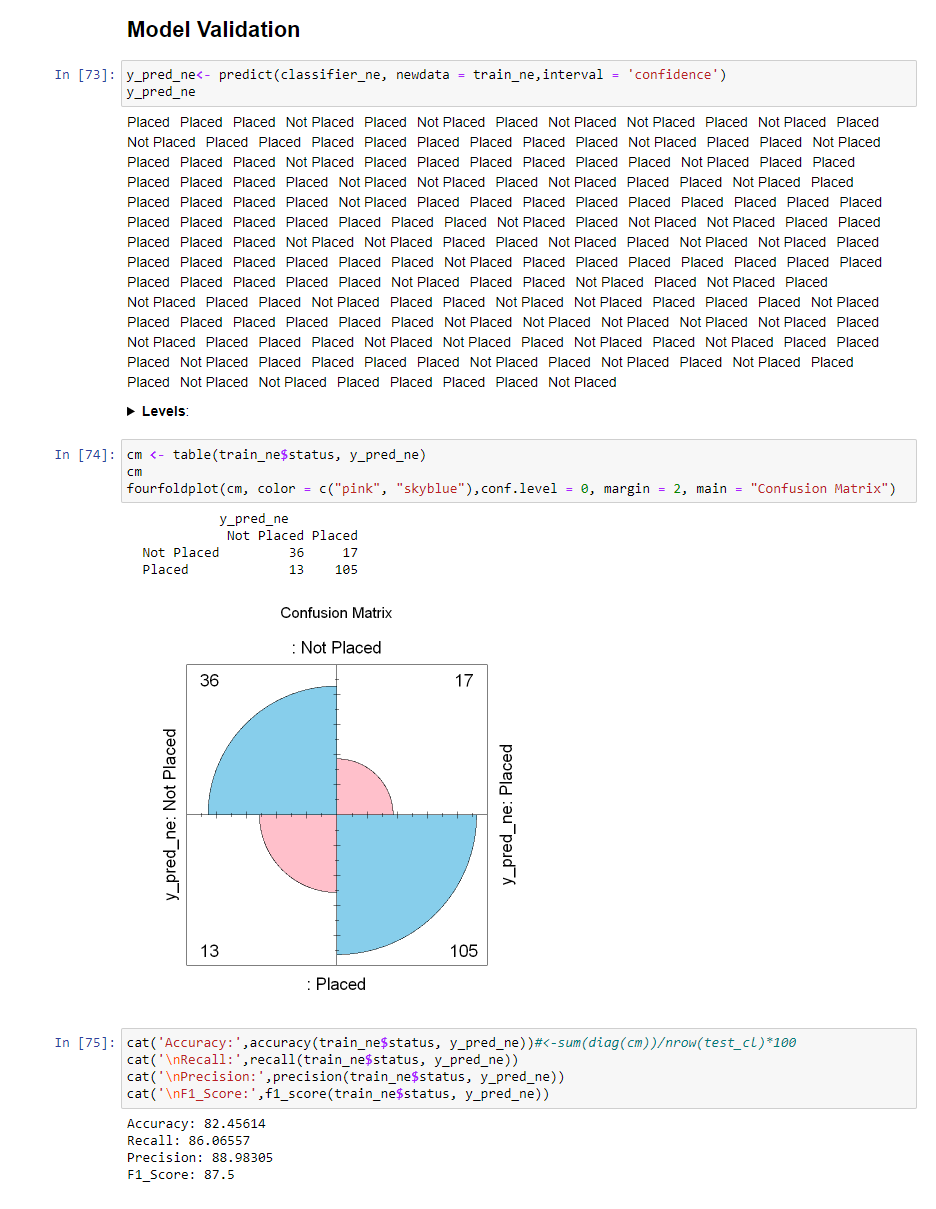
****

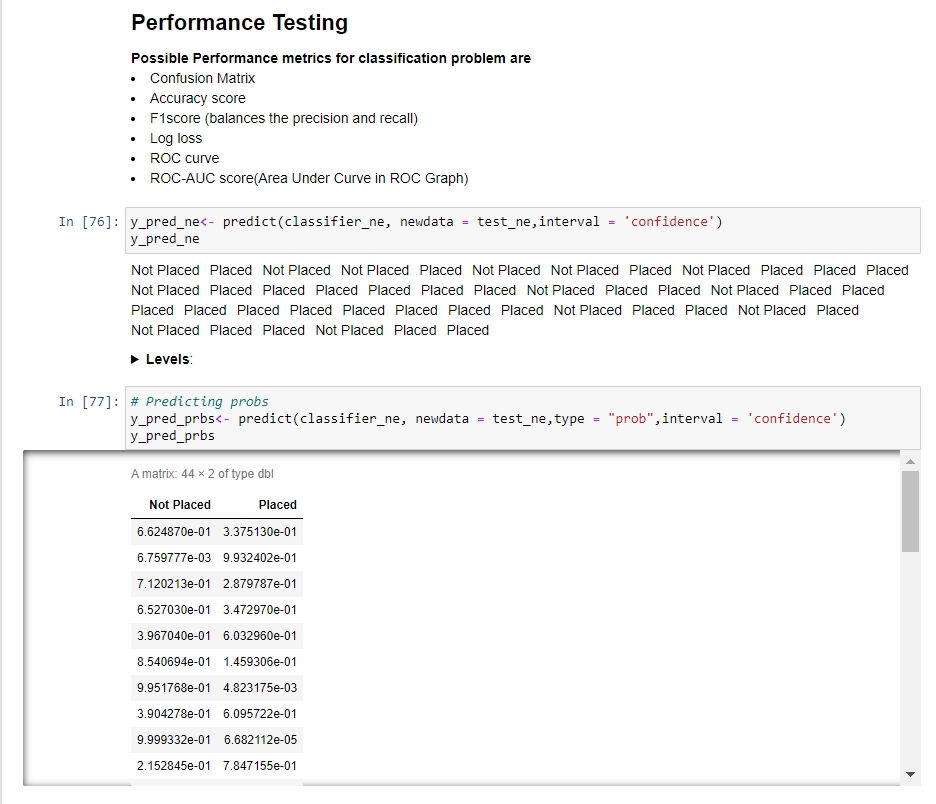
****

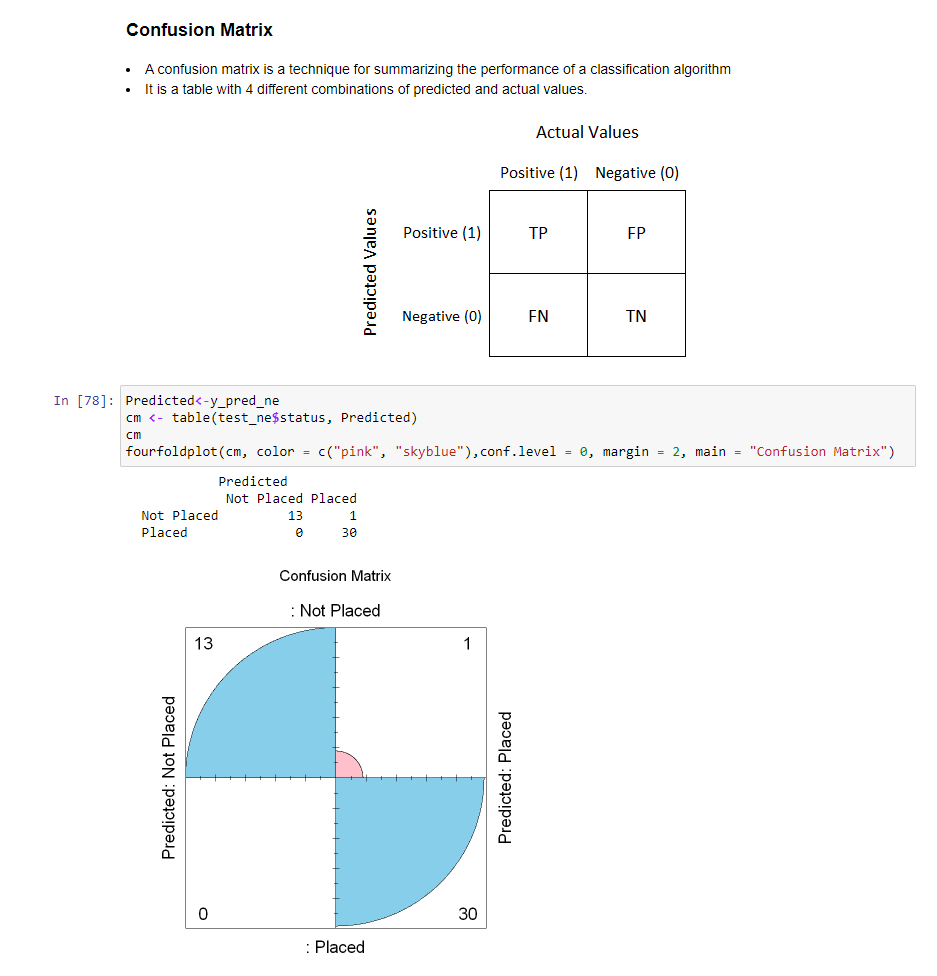
****

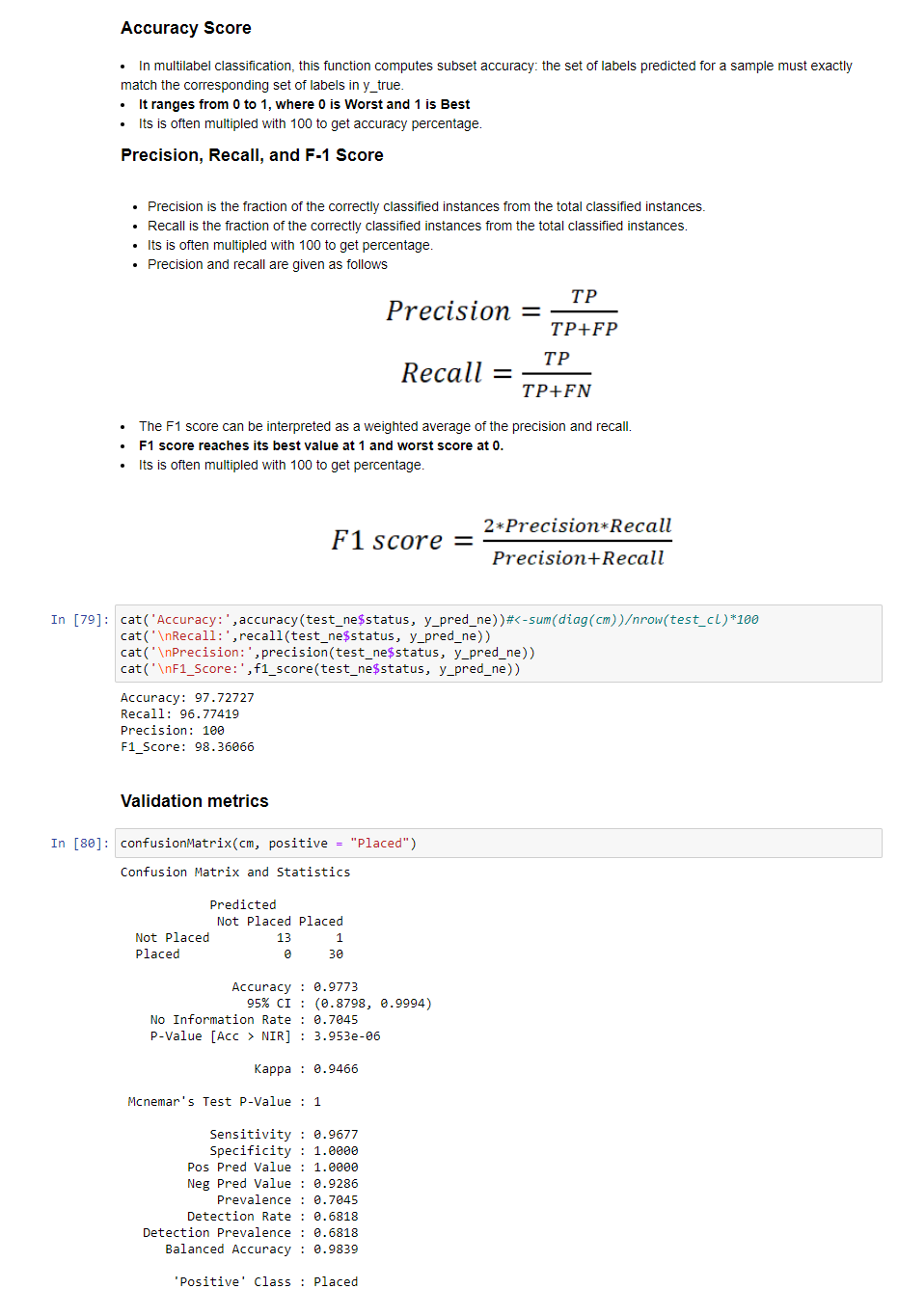
****

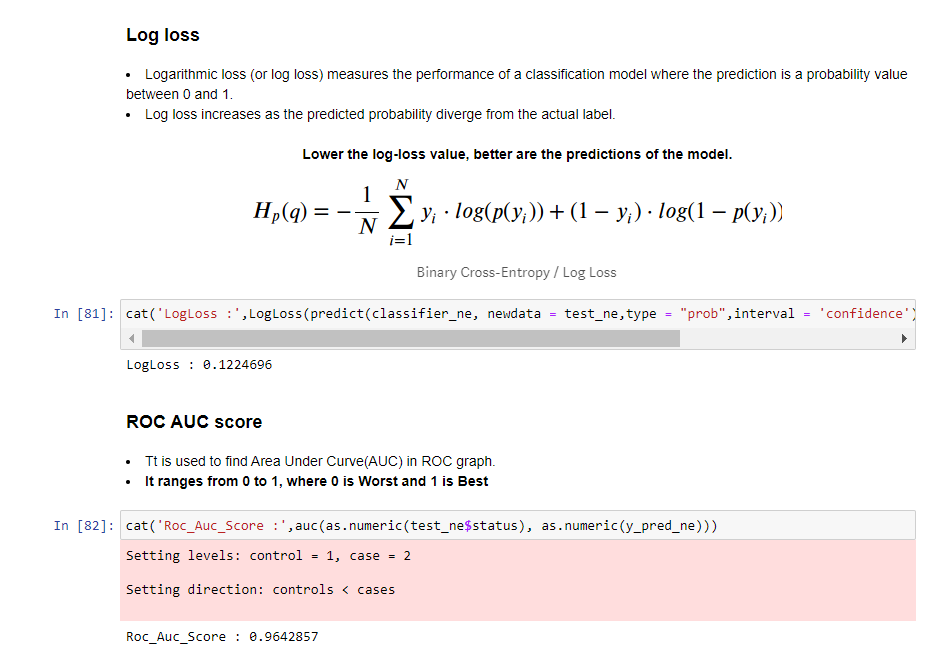
****

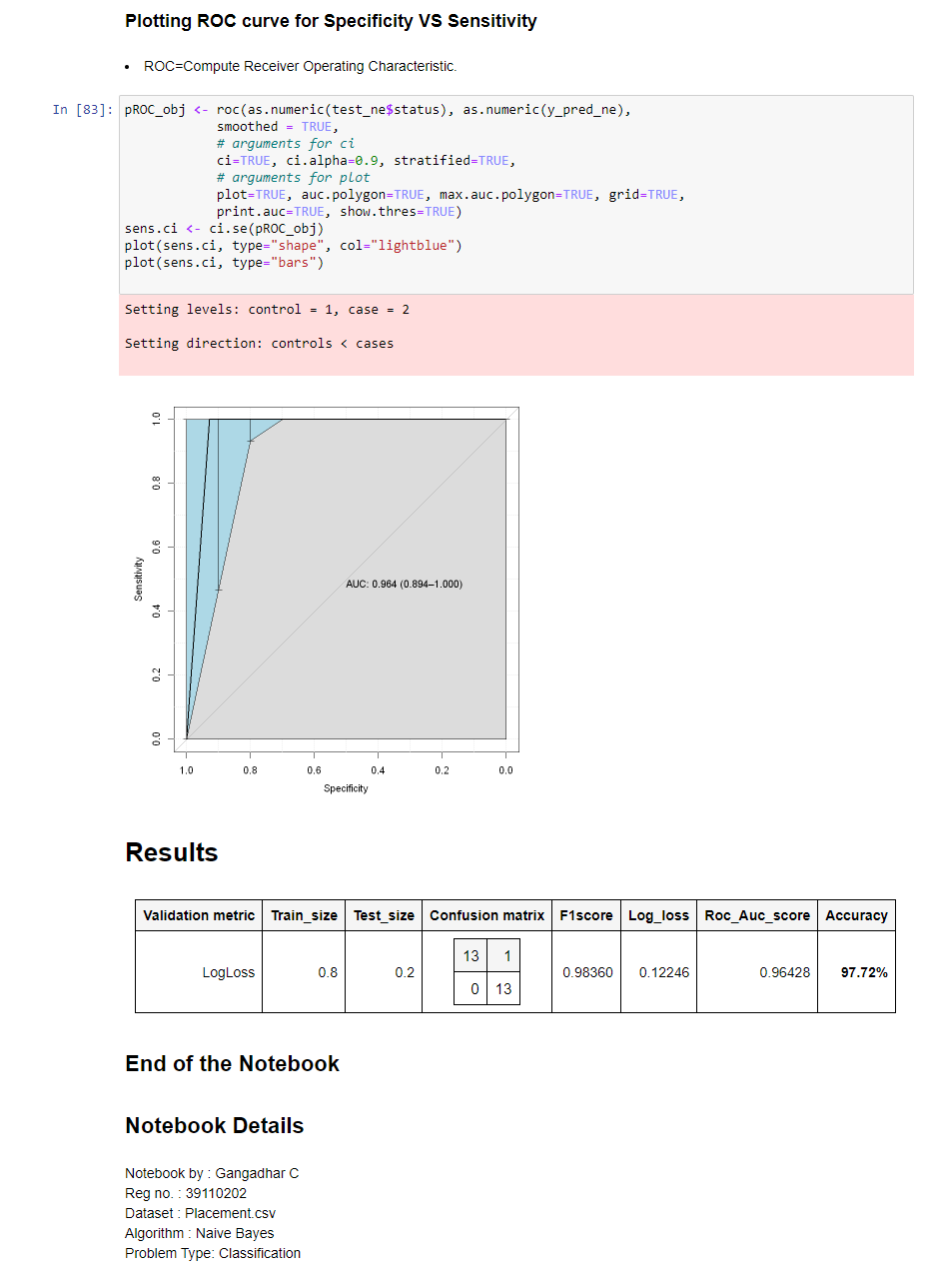
****

****

****

****

****

****

**B. SOURCE CODE**

|  |
| --- |
| #Suppress Warnings options(warn = -1) #Libraries library(naivebayes) library(e1071) library(caTools) library(ggplot2) library(caret) library(dplyr) library(gdata) library(ROSE) library(stringr) library(rBayesianOptimization) library(psych) library(smotefamily) library(randomForest) library(class) library(superml) library(corrplot) library(klaR) library(ROCR) library(pROC) library(MLmetrics)  df=read.csv("Placement.csv",header=TRUE) head(df)  tail(df)  describe(df)  str(df)  chrDT<-list() intDT<-list() for(i in seq(ncol(df))) {     if(typeof(df[,i])=="character")     {         chrDT<-append(chrDT,i,after=length(chrDT))     }     else{        intDT<-append(intDT,i,after=length(intDT))     } } cat("Character Datatype columns :\n") for(i in names(df)[unlist(chrDT)]) cat('> ',i,"\n")  cat("\nInteger Datatype columns :\n") for(i in names(df)[unlist(intDT)]) cat('> ',i,"\n")  for(i in c(unlist(chrDT)))     cat(names(df)[i],'{',summary(df[i]),'}',"\n\n")  for(i in c(unlist(intDT)))     cat(names(df)[i],'{',summary(df[i]),'}',"\n\n")  summary(df)  str(df)  for(i in names(df)) {   if(typeof(df[[i]])=="character")   {     df[[i]]=as.factor(df[[i]])   } } str(df)  df=df[,-ncol(df)] #deleting salary attribute   head(df)  summary(df)  cat("Size of DataFrame :",dim(df)[1],'x',dim(df)[2],"\n") cat("No. of NULL values in Dataset : ",sum(is.na(df)))  head(data.frame(df$status))  chrDT<-list() intDT<-list() for(i in seq(ncol(df))) {     if(typeof(df[,i])=="integer")     {         chrDT<-append(chrDT,i,after=length(chrDT))     }     else{        intDT<-append(intDT,i,after=length(intDT))     } }  corel<-cor(df[,unlist(intDT)], method = "pearson", use = "complete.obs") corel  corrplot(corel, order = 'AOE')  describe(df[df$status=='Placed',][,unlist(intDT)])  describe(df[df$status=='Not Placed',][,unlist(intDT)])  barplot(table(df$status), main = "Student Placements",         xlab="No. of Placed Students",ylab="No.of Students",col=c('red','#0000AA'),ylim=c(0,200)) legend("topleft",c("Not Placed","Placed"),fill = c('red','#0000AA')) val<-c(nrow(df[df$status=="Not Placed",]),nrow(df[df$status=="Placed",])) text(val+10,labels =val,adj=c(2,0),cex=2)  cat("Prediction Class Datatype : ",class(df$status))  par(mfrow=c(1,2)) barplot(table(df$gender),main="Students Gender", col=c('pink','navy'),ylim=c(0,200)) barplot(table(df$gender[df$status=='Placed']),main="Placed Students Gender", col=c('pink','navy'),ylim=c(0,200))  par(mfrow=c(1,2))  barplot(table(df$ssc\_b), main = "SSC Board",         xlab="Levels of Board", ylab="No.of Students",col =rainbow(2),ylim=c(0,200)) legend("topleft",c("Central","Others"),fill = rainbow(2)) val<-c(nrow(df[df$ssc\_b=="Central",]),nrow(df[df$ssc\_b=="Others",])) text(val+10,labels =val,adj=c(1,0),cex=2)  barplot(table(df$ssc\_b[df$status =='Placed']), main = "SSC Board Placed Students",         xlab="Levels of Board", ylab="No.of Students",col =rainbow(2),ylim=c(0,200)) legend("topleft",c("Central","Others"),fill = rainbow(2)) val<-c(   nrow(df[df$status =='Placed',][as.data.frame(df$ssc\_b[df$status =='Placed'])=='Central',]),   nrow(df[df$status =='Placed',][as.data.frame(df$ssc\_b[df$status =='Placed'])=='Others',]) ) text(val+10,labels =val,adj=c(1,0),cex=2)    par(mfrow=c(1,2))  x<-barplot(table(df$hsc\_b), main = "HSC Board",         xlab="Levels of Board", ylab="No.of Students",col =rainbow(2),ylim=c(0,250)) legend("topleft",c("Central","Others"),fill = rainbow(2)) val<-c(nrow(df[df$hsc\_b=="Central",]),nrow(df[df$hsc\_b=="Others",])) text(val+10,labels =val,adj=c(1,0),cex=2)  barplot(table(df$hsc\_b[df$status =='Placed']), main = "HSC Board Placed Students",         xlab="Levels of Board", ylab="No.of Students",col =rainbow(2),ylim=c(0,250)) legend("topleft",c("Central","Others"),fill = rainbow(2)) val<-c(   nrow(df[df$status =='Placed',][as.data.frame(df$hsc\_b[df$status =='Placed'])=='Central',]),   nrow(df[df$status =='Placed',][as.data.frame(df$hsc\_b[df$status =='Placed'])=='Others',]) ) text(val+10,labels =val,adj=c(1,0),cex=2)   par(mfrow=c(1,2)) # par( mai=c(0.2,0.2,0.2,0.2)) # par(fig=c(0,0.5,0.5,1)) hist(df$ssc\_p,col=rainbow(39),new=TRUE,xlab='percentages',ylab='no. of students',main='SSC Pass percentages',ylim=c(0,70)) #par(fig=c(0.5,1,0.5,1), new=TRUE) hist(df$hsc\_p,col=rainbow(39),new=TRUE,xlab='percentages',ylab='no. of students',main='HSC Pass percentages',ylim=c(0,70))  par(mfrow=c(1,2))#, mar=c(4,4,4,1), oma=c(0.5,0.5,0.5,0))  barplot(table(df$degree\_t), main = "Branches",      xlab="Types of Branches", ylab="No.of Students",col =rainbow(3),ylim=c(0,280)) legend("topleft",c("Comm&Mgmt","Others","Sci&Tech"),fill = rainbow(3)) val<-c(nrow(df[df$degree\_t=="Comm&Mgmt",]),nrow(df[df$degree\_t=="Others",]),nrow(df[df$degree\_t=="Sci&Tech",])) text(val+10,labels =val,adj=c(1,0),cex=2)  barplot(table(df$degree\_t[df$status=='Placed']), main = "Branches Placed",      xlab="Types of Branches", ylab="No.of Students",col =rainbow(3),ylim=c(0,280)) legend("topleft",c("Comm&Mgmt","Others","Sci&Tech"),fill = rainbow(3)) val<-c(nrow(df[df$status=='Placed',][as.data.frame(df$degree\_t[df$status =='Placed'])=="Comm&Mgmt",]),        nrow(df[df$status=='Placed',][as.data.frame(df$degree\_t[df$status =='Placed'])=="Others",]),        nrow(df[df$status=='Placed',][as.data.frame(df$degree\_t[df$status =='Placed'])=="Sci&Tech",])       ) text(val+10,labels =val,adj=c(1,0),cex=2)   barplot(table(df$workex),main='Work Experience',col=c('red','blue'),xlab="Having experience", ylab="No.of Students",ylim=c(0,180)) val<-c(nrow(df[df$workex=="No",]),nrow(df[df$workex=="Yes",])) text(val+10,labels =val,adj=c(1,0),cex=2) par(mfrow=c(1,2)) barplot(table(df$workex[df$status=='Placed']),main='Work Experience Placed',col=c('red','blue'),xlab="Having experience", ylab="No.of Students",ylim=c(0,180)) val<-c(length(df$workex[df$status=='Placed'][df$workex[df$status=='Placed']=='No']),length(df$workex[df$status=='Placed'][df$workex[df$status=='Placed']=='Yes'])) text(val+5,labels =val,adj=c(1,0),cex=2) barplot(table(df$workex[df$status=='Not Placed']),main='Work Experience Not Placed',col=c('red','blue'),xlab="Having experience", ylab="No.of Students",ylim=c(0,180)) val<-c(length(df$workex[df$status=='Not Placed'][df$workex[df$status=='Not Placed']=='No']),length(df$workex[df$status=='Not Placed'][df$workex[df$status=='Not Placed']=='Yes'])) text(val+5,labels =val,adj=c(1,0),cex=2)  par(mfrow=c(1,2))  hist(df$etest\_p[df$status=='Placed'],col=rainbow(10),new=TRUE,ylim=c(0,30),main='Etest performance of \nPlaced data',xlab='scores',ylab='no. students') hist(df$etest\_p[df$status=='Not Placed'],col=rainbow(10),new=TRUE,ylim=c(0,30),main='Etest performance of\n Not Placed data',xlab='scores',ylab='no. students')  # for(i in 1:100){ # set.seed(i)   # ind<-sample(2,nrow(df),replace=TRUE,prob=c(0.8,0.2)) # train\_cl<-df[ind==1,] # test\_cl<-df[ind==2,] # grid <- data.frame(fL=c(0,0.5,1.0), usekernel = TRUE, adjust=c(0,0.5,1.0)) # classifier\_cl <- naiveBayes(status ~ ., data = train\_cl, tuneGrid=grid) # y\_pred <- predict(classifier\_cl, newdata = test\_cl,interval = 'confidence') # acc<-accuracy(test\_cl$status, y\_pred) # if(acc>95) # { #     print(i) #     break # } # } # cat('Accuracy',acc)#seeds=>39,237,285,332,393 #set.seed(237)   ind<-sample(2,nrow(df),replace=TRUE,prob=c(0.8,0.2)) train\_cl<-df[ind==1,] head(train\_cl)  test\_cl<-df[ind==2,] head(test\_cl)  grid <- data.frame(fL=c(0,0.5,1.0), usekernel = TRUE, adjust=c(0,0.5,1.0)) classifier\_cl <- naive\_bayes(status ~ ., data = train\_cl, tuneGrid=grid) classifier\_cl[4]  plot(classifier\_cl) #plot(classifier\_cl, prob = "conditional") y\_pred\_vd <- predict(classifier\_cl, newdata = train\_cl,interval = 'confidence') y\_pred\_vd  accuracy<-function(y\_act,y\_preds) {     cm <- table(y\_act,y\_preds)     return(sum(diag(cm))/length(y\_act)\*100) } recall<-function(y\_act,y\_preds) {     cm <- table(y\_act,y\_preds)     return((cm[2,2]/(cm[2,2]+cm[1,2]))\*100) } precision<-function(y\_act,y\_preds) {     cm <- table(y\_act,y\_preds)     return((cm[2,2]/(cm[2,2]+cm[2,1]))\*100) } f1\_score<-function(y\_act,y\_preds) {     return((2\*precision(y\_act,y\_preds)\*recall(y\_act,y\_preds))/(precision(y\_act,y\_preds)+recall(y\_act,y\_preds))) }  cm <- table(train\_cl$status, y\_pred\_vd) cm  cat('Accuracy:',accuracy(train\_cl$status, y\_pred\_vd))#<-sum(diag(cm))/nrow(test\_cl)\*100 cat('\nRecall:',recall(train\_cl$status, y\_pred\_vd)) cat('\nPrecision:',precision(train\_cl$status, y\_pred\_vd)) cat('\nF1\_Score:',f1\_score(train\_cl$status, y\_pred\_vd))  lbl = LabelEncoder$new() cat('LogLoss :',LogLoss(predict(classifier\_cl, newdata = train\_cl,type = "prob",interval = 'confidence')[,2],lbl$fit\_transform(train\_cl$status)))  # Predicting on test data' y\_pred <- predict(classifier\_cl, newdata = test\_cl,interval = 'confidence') y\_pred  Predicted<-y\_pred cm <- table(test\_cl$status, Predicted) cm fourfoldplot(cm, color = c("pink", "skyblue"),conf.level = 0, margin = 2, main = "Confusion Matrix")  length(test\_cl$status)  cat('Accuracy:',accuracy(test\_cl$status, y\_pred))#<-sum(diag(cm))/nrow(test\_cl)\*100 cat('\nRecall:',recall(test\_cl$status, y\_pred)) cat('\nPrecision:',precision(test\_cl$status, y\_pred)) cat('\nF1\_Score:',f1\_score(test\_cl$status, y\_pred))  as.data.frame(confusionMatrix(cm, positive = "Placed")[3])  as.data.frame(confusionMatrix(cm, positive = "Placed")[4])  confusionMatrix(cm, positive = "Placed")  lbl = LabelEncoder$new() cat('LogLoss :',LogLoss(predict(classifier\_cl, newdata = test\_cl,type = "prob",interval = 'confidence')[,2],lbl$fit\_transform(test\_cl$status)))  cat('Roc\_Auc\_Score :',auc(as.numeric(test\_cl$status), as.numeric(y\_pred)))   pROC\_obj <- roc(as.numeric(test\_cl$status), as.numeric(y\_pred),             smoothed = TRUE,             # arguments for ci             ci=TRUE, ci.alpha=0.9, stratified=FALSE,             # arguments for plot             plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,             print.auc=TRUE, show.thres=TRUE) sens.ci <- ci.se(pROC\_obj) plot(sens.ci, type="shape", col="lightblue") plot(sens.ci, type="bars")  k=5 fold<-KFold(df$status, nfolds = k, stratified = TRUE)  mnf1\_nb<-list() mnacc\_nb<-list() for(i in seq(1:k)) {     l=c(seq(1:k))     l<-l[-i]     train<-df[unlist(fold[l]),]     test<-df[unlist(fold[i]),]     grid <- data.frame(fL=c(0,0.5,1.0), usekernel = TRUE, adjust=c(0,0.5,1.0))     classifier\_cl <- naiveBayes(status ~ ., data = train, tuneGrid=grid)     y\_pred <- predict(classifier\_cl, newdata = test,interval = 'confidence')     mnf1\_nb<-append(mnf1\_nb,f1\_score(test$status, y\_pred),after=length(mnf1\_nb))     mnacc\_nb<-append(mnacc\_nb,accuracy(test$status, y\_pred),after=length(mnacc\_nb)) } cat('Mean Accuracy',mean(c(unlist(mnacc\_nb)))) cat('\n') cat(paste('\nAccuracy Result of Fold',c(1:5),':',unlist(mnacc\_nb))) cat('\n') cat('\nMean f1\_score',mean(c(unlist(mnf1\_nb)))) cat('\n') cat(paste('\nf1\_score Result of Fold',c(1:5),':',unlist(mnf1\_nb)))   ind\_lr<-sample(2,nrow(df),replace=TRUE,prob=c(0.8,0.2)) train\_lr<-df[ind\_lr==1,] test\_lr<-df[ind\_lr==2,] classifier\_lr <-glm(status ~ .,data = train\_lr,family = "binomial") y\_pred\_lr <- predict(classifier\_lr, newdata = test\_lr,type = "response") y\_pred\_lr <- ifelse(y\_pred\_lr >0.5, 1, 0) acc<-accuracy(test\_lr$status, y\_pred\_lr) cat('Accuracy:',acc)  mnf1\_lr<-list() mnacc\_lr<-list() for(i in seq(1:k)) {     l=c(seq(1:k))     l<-l[-i]     train<-df[unlist(fold[l]),]     test<-df[unlist(fold[i]),]     classifier\_lr <-glm(status ~ .,data = train,family = "binomial")     y <- predict(classifier\_lr, newdata = test,interval = 'confidence')     y <- ifelse(y>0.5, 1, 0)     mnf1\_lr<-append(mnf1\_lr,f1\_score(test$status, y),after=length(mnf1\_lr))          mnacc\_lr<-append(mnacc\_lr,accuracy(test$status, y),after=length(mnacc\_lr)) } cat('Mean Accuracy',mean(c(unlist(mnacc\_lr)))) cat('\n') cat(paste('\nAccuracy Result of Fold',c(1:5),':',unlist(mnacc\_lr))) cat('\n') cat('\nMean f1\_score',mean(c(unlist(mnf1\_lr)))) cat('\n') cat(paste('\nf1\_score Result of Fold',c(1:5),':',unlist(mnf1\_lr)))   lbl = LabelEncoder$new() df\_knn=df for(i in chrDT){ df\_knn[,i]<-lbl$fit\_transform(df\_knn[,i]) } head(df\_knn)  ind\_knn<-sample(2,nrow(df\_knn),replace=TRUE,prob=c(0.8,0.2)) train\_knn<-df\_knn[ind\_knn==1,] test\_knn<-df\_knn[ind\_knn==2,] train\_scale <- scale(train\_knn[,1:ncol(df\_knn)-1]) test\_scale <- scale(test\_knn[,1:ncol(df\_knn)-1]) classifier\_knn <-knn(train=train\_scale,test=test\_scale,cl=train\_knn$status,k=floor(sqrt(nrow(df\_knn)))) acc<-accuracy(test\_knn$status, classifier\_knn) cat("Validation Accuracy:",acc)  mnf1\_knn<-list() mnacc\_knn<-list() for(i in seq(1:k)) {     l=c(seq(1:k))     l<-l[-i]     train\_knn<-df\_knn[unlist(fold[l]),]     test\_knn<-df\_knn[unlist(fold[i]),]     train\_scale <- scale(train\_knn[,1:ncol(df\_knn)-1])     test\_scale <- scale(test\_knn[,1:ncol(df\_knn)-1])     classifier\_knn <-knn(train=train\_scale,test=test\_scale,cl=train\_knn$status,k=floor(sqrt(nrow(df\_knn))))     mnf1\_knn<-append(mnf1\_knn,f1\_score(test\_knn$status, classifier\_knn),after=length(mnf1\_knn))     mnacc\_knn<-append(mnacc\_knn,accuracy(test\_knn$status, classifier\_knn),after=length(mnacc\_knn)) } cat('Mean Accuracy',mean(c(unlist(mnacc\_knn)))) cat('\n') cat(paste('\nAccuracy Result of Fold',c(1:5),':',unlist(mnacc\_knn))) cat('\n') cat('\nMean f1\_score',mean(c(unlist(mnf1\_knn)))) cat('\n') cat(paste('\nf1\_score Result of Fold',c(1:5),':',unlist(mnf1\_knn)))   ind\_rf<-sample(2,nrow(df),replace=TRUE,prob=c(0.8,0.2)) train\_rf<-df[ind\_rf==1,] test\_rf<-df[ind\_rf==2,] classifier\_rf <-randomForest(status ~ ., data=train\_rf, importance=TRUE,proximity=TRUE) y\_pred\_rf <- predict(classifier\_rf, newdata = test\_rf,interval = 'confidence') acc<-accuracy(test\_rf$status, y\_pred\_rf) acc  mnf1\_rf<-list() mnacc\_rf<-list() for(i in seq(1:k)) {     l=c(seq(1:k))     l<-l[-i]     train<-df[unlist(fold[l]),]     test<-df[unlist(fold[i]),]     classifier\_rf <-randomForest(status ~ ., data=train, importance=TRUE,proximity=TRUE)     y <- predict(classifier\_rf, newdata = test,interval = 'confidence')     mnf1\_rf<-append(mnf1\_rf,f1\_score(test$status, y),after=length(mnf1\_rf))     mnacc\_rf<-append(mnacc\_rf,accuracy(test$status, y),after=length(mnacc\_rf)) } cat('Mean Accuracy',mean(c(unlist(mnacc\_rf)))) cat('\n') cat(paste('\nAccuracy Result of Fold',c(1:5),':',unlist(mnacc\_rf))) cat('\n') cat('\nMean f1\_score',mean(c(unlist(mnf1\_rf)))) cat('\n') cat(paste('\nf1\_score Result of Fold',c(1:5),':',unlist(mnf1\_rf)))   ind\_svm<-sample(2,nrow(df),replace=TRUE,prob=c(0.8,0.2)) train\_svm<-df[ind\_svm==1,] test\_svm<-df[ind\_svm==2,] classifier\_svm = svm(status ~ ., data=train\_svm,type = 'C-classification',kernel = 'linear') y\_pred\_svm <- predict(classifier\_svm, newdata = test\_svm,interval = 'confidence') acc<-accuracy(test\_svm$status, y\_pred\_svm) acc  k=5 #fold<-KFold(df$status, nfolds = k, stratified = TRUE) mnf1\_svm<-list() mnacc\_svm<-list() for(i in seq(1:k)) {     l=c(seq(1:k))     l<-l[-i]     train<-df[unlist(fold[l]),]     test<-df[unlist(fold[i]),]     classifier\_svm = svm(status ~ ., data=train\_svm,type = 'C-classification',kernel = 'linear')     y <- predict(classifier\_svm, newdata = test,interval = 'confidence')     mnf1\_svm<-append(mnf1\_svm,f1\_score(test$status, y),after=length(mnf1\_svm))     mnacc\_svm<-append(mnacc\_svm,accuracy(test$status, y),after=length(mnacc\_svm)) } cat('Mean Accuracy',mean(c(unlist(mnacc\_svm)))) cat('\n') cat(paste('\nAccuracy Result of Fold',c(1:k),':',unlist(mnacc\_svm))) cat('\n') cat('\nMean f1\_score',mean(c(unlist(mnf1\_svm)))) cat('\n') cat(paste('\nf1\_score Result of Fold',c(1:k),':',unlist(mnf1\_svm)))   par(mfrow=c(2,2)) fourfoldplot(table(test\_lr$status,y\_pred\_lr), color = c("pink", "skyblue"),conf.level = 0, margin = 2, main = "Logistic Regression ") fourfoldplot(table(test\_knn$status,classifier\_knn), color = c("pink", "skyblue"),conf.level = 0, margin = 2, main = "K-NN ") fourfoldplot(table(test\_rf$status,y\_pred\_rf), color = c("pink", "skyblue"),conf.level = 0, margin = 2, main = "Random Forest") fourfoldplot(table(test\_svm$status,y\_pred\_svm), color = c("pink", "skyblue"),conf.level = 0, margin = 2, main = "SVM")   kacc=c(mean(c(unlist(mnacc\_nb))),mean(c(unlist(mnacc\_lr))),mean(c(unlist(mnacc\_knn))),mean(c(unlist(mnacc\_rf))),mean(c(unlist(mnacc\_svm)))) kf1=c(mean(c(unlist(mnf1\_nb))),mean(c(unlist(mnf1\_lr))),mean(c(unlist(mnf1\_knn))),mean(c(unlist(mnf1\_rf))),mean(c(unlist(mnf1\_svm)))) accmax=c(max(unlist(mnacc\_nb)),max(unlist(mnacc\_lr)),max(unlist(mnacc\_knn)),max(unlist(mnacc\_rf)),max(unlist(mnacc\_svm))) accmin=c(min(unlist(mnacc\_nb)),min(unlist(mnacc\_lr)),min(unlist(mnacc\_knn)),min(unlist(mnacc\_rf)),min(unlist(mnacc\_svm))) f1max=c(max(unlist(mnf1\_nb)),max(unlist(mnf1\_lr)),max(unlist(mnf1\_knn)),max(unlist(mnf1\_rf)),max(unlist(mnf1\_svm))) f1min=c(min(unlist(mnf1\_nb)),min(unlist(mnf1\_lr)),min(unlist(mnf1\_knn)),min(unlist(mnf1\_rf)),min(unlist(mnf1\_svm))) model\_df=data.frame(Algorithm=c('Naive Bayes','Logistic Regression','K-Nearest Neighbours','Random Forest',"Support Vector Machine"),Min\_Accuracy=accmin,Max\_Accuracy=f1max,Min\_F1score=f1min,Max\_F1score=accmax,KFold\_Accuracy=kacc,KFold\_F1score=kf1)  model\_df  par(mfrow=c(2,2)) fn<-function(x){     return (format(round(x, 2), nsmall = 2)) } labs=c('NB','LR','K-NN','RF','SVM') barplot(model\_df$Min\_Accuracy,ylim=c(50,120),names.arg=labs,col=rainbow(5), log="y",main='Min\_Accuracy') text(model\_df$Min\_Accuracy,labels =sapply(model\_df$Min\_Accuracy,fn),pos=3,cex=1)  barplot(model\_df$Max\_Accuracy,ylim=c(50,120),names.arg=labs,col=rainbow(5), log="y",main='Max\_Accuracy') text(model\_df$Max\_Accuracy,labels =sapply(model\_df$Max\_Accuracy,fn),pos=3,cex=1)  barplot(model\_df$Min\_F1score,ylim=c(50,120),names.arg=labs,col=rainbow(5), log="y",main='Min\_F1score') text(model\_df$Min\_F1score,labels =sapply(model\_df$Min\_F1score,fn),pos=3,cex=1)  barplot(model\_df$Max\_F1score,ylim=c(50,120),names.arg=labs,col=rainbow(5), log="y",main='Max\_F1score') text(model\_df$Max\_F1score,labels =sapply(model\_df$Max\_F1score,fn),pos=3,cex=1)  plot(model\_df$Min\_Accuracy,type = "o",col='red',ylim=c(75,100),lwd=2.0, xaxt = "n",main='Accuracy Comparison') axis(1, at=1:5, labels=labs) lines(model\_df$Max\_Accuracy, type = "o", col = "blue",lwd=2.0) legend("topleft",c("Min Accuracy","Max Accuracy"),fill = c('red','blue'))  plot(model\_df$Min\_F1score,type = "o",col='red',ylim=c(75,100),lwd=2.0, xaxt = "n",main='F1-Score Comparison') axis(1, at=1:5, labels=labs) lines(model\_df$Max\_F1score, type = "o", col = "blue",lwd=2.0) legend("topleft",c("Min F1-score"," Max F1-score"),fill = c('red','blue'))  plot(model\_df$KFold\_Accuracy,type = "o",col='red',ylim=c(80,95),lwd=2.0, xaxt = "n",main='K-Fold Accuracy and F1-Score Comparison') axis(1, at=1:5, labels=labs) lines(model\_df$KFold\_F1score, type = "o", col = "blue",lwd=2.0) legend("topleft",c("Accuracy","F1-score"),fill = c('red','blue'))  plot(unlist(mnacc\_nb),type = "o",col='red',ylim=c(70,100),lwd=2.0, xaxt = "n",main='Kfold Validation Accuracies',xlab='K-folds',ylab='percentages') axis(1, at=1:5, labels=c(1:5)) lines(unlist(mnacc\_lr), type = "o", col = "blue",lwd=2.0) lines(unlist(mnacc\_knn), type = "o", col = "green",lwd=2.0) lines(unlist(mnacc\_rf), type = "o", col = "navy",lwd=2.0) lines(unlist(mnacc\_svm), type = "o", col = "cyan",lwd=2.0) legend("topleft",c('NB','LR','K-NN','RF','SVM'),fill = c('red','blue',"green","navy","cyan"))   plot(unlist(mnf1\_nb),type = "o",col='red',ylim=c(70,100),lwd=2.0, xaxt = "n",main='Kfolds validation F1-scores',xlab='K-folds',ylab='percentages') axis(1, at=1:5, labels=c(1:5)) lines(unlist(mnf1\_lr), type = "o", col = "blue",lwd=2.0) lines(unlist(mnf1\_knn), type = "o", col = "green",lwd=2.0) lines(unlist(mnf1\_rf), type = "o", col = "navy",lwd=2.0) lines(unlist(mnf1\_svm), type = "o", col = "cyan",lwd=2.0) legend("topleft",c('NB','LR','K-NN','RF','SVM'),fill = c('red','blue',"green","navy","cyan"))  for(i in 1:500) {     ind<-sample(2,nrow(df),replace=TRUE,prob=c(0.8,0.2))     train<-df[ind==1,]     test<-df[ind==2,]    grid <- data.frame(fL=c(0,0.5,1.0), usekernel = TRUE, adjust=c(0,0.5,1.0))     classifier<- naive\_bayes(status ~ ., data = train, tuneGrid=grid)     y\_pred <- predict(classifier, newdata = test,interval = 'confidence')     loss\_vd<-LogLoss(predict(classifier, newdata = train,type = "prob",interval = 'confidence')[,2],lbl$fit\_transform(train$status))     loss\_ts<-LogLoss(predict(classifier, newdata = test,type = "prob",interval = 'confidence')[,2],lbl$fit\_transform(test$status))     if(loss\_ts<=0.149 )         break }  cat('Validation Loss:',loss\_vd,"\nTesting Loss:",loss\_ts)  ind df\_in<-as.list(ind) df\_in saveRDS(df\_in, file="indices2.RData")  #saveRDS(df\_in, file="indices.RData")  #saveRDS(df\_in, file="indices3.RData") inds<-readRDS("indices2.RData")  class(inds)  train\_ne<-df[unlist(inds)==1,] test\_ne<-df[unlist(inds)==2,]  #set.seed(120)  # Setting Seed grid <- data.frame(fL=c(0,0.5,1.0), usekernel = TRUE, adjust=c(0,0.5,1.0)) classifier\_ne <- naive\_bayes(status ~ ., data =train\_ne, tuneGrid=grid) classifier\_ne  plot(classifier\_ne)  y\_pred\_ne<- predict(classifier\_ne, newdata = train\_ne,interval = 'confidence') y\_pred\_ne  cm <- table(train\_ne$status, y\_pred\_ne) cm fourfoldplot(cm, color = c("pink", "skyblue"),conf.level = 0, margin = 2, main = "Confusion Matrix")  cat('Accuracy:',accuracy(train\_ne$status, y\_pred\_ne))#<-sum(diag(cm))/nrow(test\_cl)\*100 cat('\nRecall:',recall(train\_ne$status, y\_pred\_ne)) cat('\nPrecision:',precision(train\_ne$status, y\_pred\_ne)) cat('\nF1\_Score:',f1\_score(train\_ne$status, y\_pred\_ne))  y\_pred\_ne<- predict(classifier\_ne, newdata = test\_ne,interval = 'confidence') y\_pred\_ne  # Predicting probs y\_pred\_prbs<- predict(classifier\_ne, newdata = test\_ne,type = "prob",interval = 'confidence') y\_pred\_prbs  Predicted<-y\_pred\_ne cm <- table(test\_ne$status, Predicted) cm fourfoldplot(cm, color = c("pink", "skyblue"),conf.level = 0, margin = 2, main = "Confusion Matrix")  cat('Accuracy:',accuracy(test\_ne$status, y\_pred\_ne))#<-sum(diag(cm))/nrow(test\_cl)\*100 cat('\nRecall:',recall(test\_ne$status, y\_pred\_ne)) cat('\nPrecision:',precision(test\_ne$status, y\_pred\_ne)) cat('\nF1\_Score:',f1\_score(test\_ne$status, y\_pred\_ne))  confusionMatrix(cm, positive = "Placed")  cat('LogLoss :',LogLoss(predict(classifier\_ne, newdata = test\_ne,type = "prob",interval = 'confidence')[,2],lbl$fit\_transform(test\_ne$status)))  cat('Roc\_Auc\_Score :',auc(as.numeric(test\_ne$status), as.numeric(y\_pred\_ne)))  pROC\_obj <- roc(as.numeric(test\_ne$status), as.numeric(y\_pred\_ne),             smoothed = TRUE,             # arguments for ci             ci=TRUE, ci.alpha=0.9, stratified=TRUE,             # arguments for plot             plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,             print.auc=TRUE, show.thres=TRUE) sens.ci <- ci.se(pROC\_obj) plot(sens.ci, type="shape", col="lightblue") plot(sens.ci, type="bars") |