LOW LEVEL DOCUMENT

CAMPUS PLACEMENT

|  |  |
| --- | --- |
| Written By | Akshat Pant |
| Document Version | 1.0 |
| Last Revised Data | 15-09-2022 |

# 1 Document Control:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 15-09-2022 | Akshat Pant |  |
|  |  |  |  |

# 2 Reviews:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Reviewer | Comments |
| 1.0 | 15-09-2022 | Akshat Pant |  |
|  |  |  |  |

# 3 Approval Status:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | Review Date | Reviewed By | Approved By | |  |
| 1.0 | 15-09-2022 | Akshat Pant | Akshat Pant |  | |

Table of Contents

[1 Document Control: 2](#_Toc89379022)

[2 Reviews: 2](#_Toc89379023)

[3 Approval Status: 2](#_Toc89379024)

[1.Introduction 5](#_Toc89379025)

[1.1 What is Low-Level design document? 5](#_Toc89379026)

[1.2. Scope 5](#_Toc89379027)

[2. Architecture 6](#_Toc89379028)

[3. Architecture Description 7](#_Toc89379029)

[3.1 Data Collection 7](#_Toc89379030)

[3.2 Data Validation 7](#_Toc89379031)

[3.3 Inserting into DB 7](#_Toc89379032)

[3.4 Retrieving Data From DB 7](#_Toc89379033)

[3.5 Data Preprocessing 7](#_Toc89379034)

[3.6 Model Selection 8](#_Toc89379035)

[3.7 Hyperparameter Tuning 8](#_Toc89379036)

[3.8 Model Saving 8](#_Toc89379037)

[3.9 Data from User 8](#_Toc89379038)

[3.10 Data Validation 8](#_Toc89379039)

[3.11 Data Preprocessing 9](#_Toc89379040)

[3.12 Model Loading 9](#_Toc89379041)

[3.13 Model.predict 9](#_Toc89379042)

[4 Test Cases 9](#_Toc89379044)

# 1.Introduction

## 1.1 What is Low-Level design document?

The goal of LLD or a low-level design document (LLD) is to give the internal logical design of the actual program code for ***Campus Placement***. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

## 1.2. Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work

# 2. Architecture

# 

# 3. Architecture Description

## 3.1 Data Collection

We have 215 Dataset row columnar data includes the 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. These is given in the comma separated value format (.csv). These data is collected from the Kaggle which contains both the test data and train data.

## 

## 3.2 Data Validation

In Data Validation part we basically check whether files inside the folder provide are having same columns or not with the data type of each column each by each file. If the file is found to matching the name of columns and dtype of the columns then we are supposed to copy the file from the folder to Training\_Batch\_Files/Good\_Data or else to Training\_Batch\_Files/Bad\_Data

## 3.3 Inserting into DB

We are going to insert the data of Good\_Data into Database that is SQLDB .For this we have used SQL Lite Database name is CampusPlacement.

## 3.4 Retrieving Data From DB

Retrieving the data and storing into a file called as campus.csv

## 3.5 Data Preprocessing

Data Preprocessing step is very necessary for model creation, We have mainly categorical data into our csv file so we need to convert that categorical data into numerical format. Data Preprocessing is getting done by the file data\_preprocessing.py

## 3.6 Model Selection

Model Selection is performed by file model\_training.py which is inside Model\_Training. For model Training we are just choosing Decision Tree Classifier because it is performing best than other models When plotting Confusion Matrix DecisionTree Classifier always had a very low False Positives and False Negetives.

## 3.7 Hyperparameter Tuning

We are doing hyperparameter tuning only for Decision Tree Classifier and because of tuning our model has a good confusion matrix.

## 

## 3.8 Model Saving

Model Saving is the final step in the training part. We are saving the model in folder Model\_for\_prediction as model.pickle and I am using module pickle for saving the ML model.

## 3.9 Data from User

On Application Starting user will be interacting with a UI which is designed using HTML/CSS.

The user enter the data into HTML form and it is further passed to our application and returns the predicted result.

## 3.10 Data Validation

It is not necessary that the csv files provided by the user will always be in required format so to check this I have included this part which is data validation.

In Data Validation part we basically check whether files inside the folder provide are having same columns or not with the data type of each column each by each file. If the file is found to matching the name of columns and dtype of the columns then we are supposed to

copy the file from the folder to Training\_Batch\_Files/Good\_Data or else to Training\_Batch\_Files/Bad\_Data.

## 3.11 Data Pre-processing

In data pre-processing I will be handling categorical columns and do feature selection along with this will do feature scaling using Standard Scaler.

## 3.12 Model Loading

Model loading is actually very simple we will be calling the model which we saved as model.pickle inside folder Model\_for\_prediction. For loading the model I am using pickle library

## 3.13 Model prediction

After loading the model everything is very simple you just have to predict the preprocessed data.

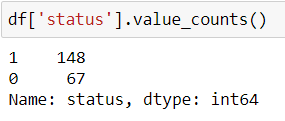
# 4 Test Cases

Test cases are given below

|  |  |  |
| --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **Expected Result** |
| Verify whether the Application URL is accessible to the user | 1. Application URL should be defined | Application URL should be accessible to the user |
| Verify whether the Application loads completely for the user when the URL is accessed | 1. Application URL is accessible 2. Application is deployed | The Application should load completely for the user when the URL is accessed |
| Verify Response time of url from backend model. | 1. Application is accessible | The latency and accessibility of application is very faster we got in heroku service. |
| Verify whether user is giving standard input. | 1. Handeled test cases at backends. | User should be able to see successfully valid results. |
| Verify whether user is able to see input fields on logging in | 1. Application is accessible 2. User is logged in to the application | User should be able to see input fields on logging in |
| Verify whether user is able to edit all input fields | 1. Application is accessible 2. User is logged in to the application | User should be able to edit all input fields |
| Verify whether user is presented with recommended results on clicking submit | 1. Application is accessible 2. User is logged in to the application | User should be presented with recommended results on clicking submit |
| Verify whether the recommended results are in accordance to the selections user made | 1. Application is accessible 2. User is logged in   to the application and database | The recommended results should be in accordance to the selections user made |

# EDA

1. Our dataset is imbalanced with a ratio of 148:67

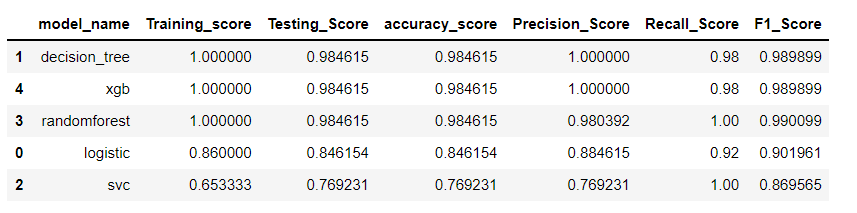


1. Since the dataset is imbalance so it is not a good idea to go with accuracy score as it will be biased towards the ‘1’ side as it is in majority

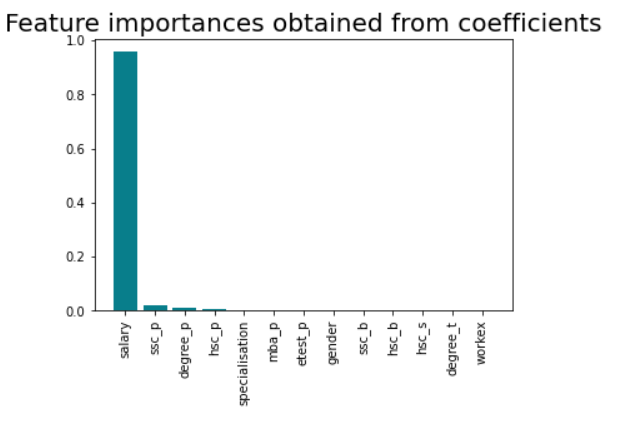
Since for us False Positives are of more concern as for us if our model predicts that Student will not get placed and in real the student is getting placed this condition will be more critical to handle as this can create more demotivation and student can loose hopes and will not prepare well enough for placements.

so for us it seems Precision will play a good role so we will be aiming to increase the Precision score.

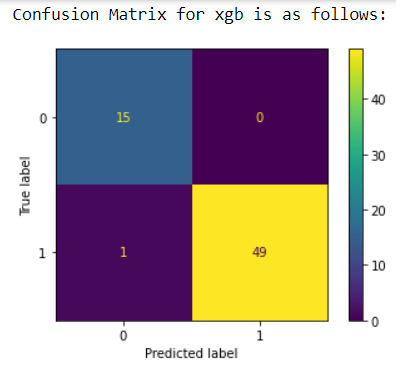
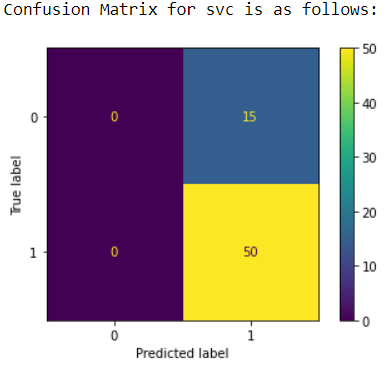
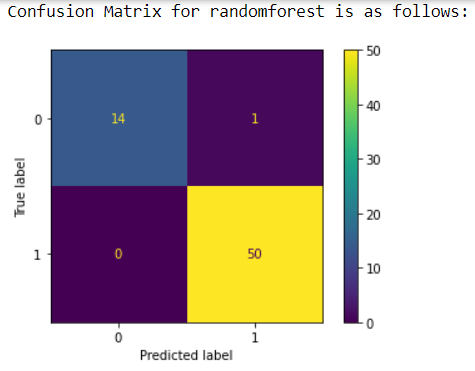
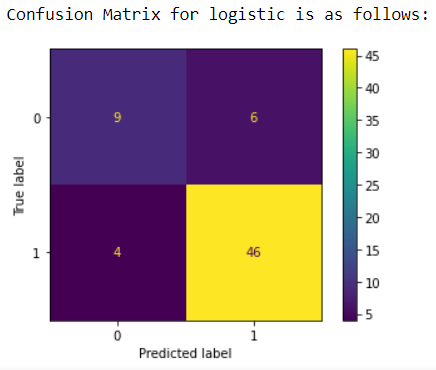
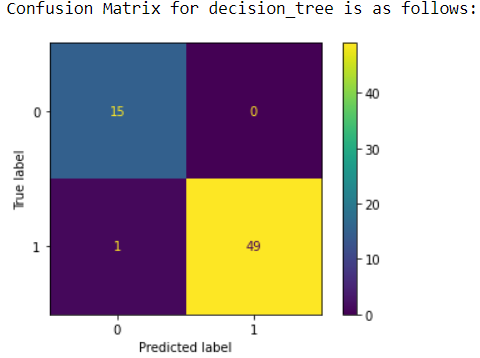
So based on different models we applied we can have a look on the Precision scores and from that we can say both Decision Tree and Random Forest performs well for us.



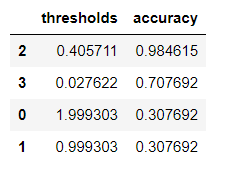
1. We also try to find out the feature importance with the help of XGboost and Random Forest and we find out that out of all features the ‘salary’ feature is the one which dominated the most.



1. The various confusion matrices for different ML algorithms are as follows:



1. The threshold value is also calculated to give the best model prediction



We can see that with threshold value of 0.405 the accuracy is maximum.

1. The ROC-AUC plot is as follows:

