Low-Level Design

Campus Placement Prediction

|  |  |
| --- | --- |
| Written By | Aditya Agrawal |
| Version | 1.0 |
| Date | 9-11-2022 |

**Document Change Control Record**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Reviews:

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Reviewer** | **Comments** |
|  |  |  |  |

### Approval Status:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Review**  **Date** | **Reviewed By** | **Approved By** | **Comments** |
|  |  |  |  |  |

**Index**

|  |  |
| --- | --- |
| **Content** | **Page No** |
| 1. Introduction | 2 |
| 1.1 What is Low-Level Design Document | 2 |
| 1.2 Scope | 2 |
| Architecture | 2 |
| 2. Architecture Description | 3 |
| 2.1 Data Description | 3 |
| 2.2 Data Gathering | 3 |
| 2.3 Raw Data Validation | 3 |
| 2.4 Data Transformation | 4 |
| 2.5 Data Preprocessing | 4 |
| 2.6 Feature Engineering | 4 |
| 2.7 Parameter Tuning | 4 |
| 2.8 Model Building | 4 |
| 2.9 Model Saving | 4 |
| 2.10 Django Setup for Data Extraction | 4 |
| 2.11 GitHub | 4 |
| 2.12 Deployment | 5 |
| 3. Unit Test Cases | 5 |

**1. Introduction**

**1.1 What is Low-Level Design Document.**

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for **‘Campus Placement Prediction’**. 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](https://en.wikipedia.org/wiki/Refinement_(computing)) 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.

**Architecture :**

**2. Architecture Description**

**2.1 Data Description**

Given is the variable name, variable type, the measurement unit, and a brief description. The concrete compressive strength is the regression problem. The order of this listing corresponds to the order of numerals along the rows of the database.

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Measurement |
| Gender | Integer | Gender of student |
| ssc\_p | Float | Ssc percentage |
| ssc\_b | Object | Ssc board |
| hsc\_p | Float | Hsc percentage |
| Hsc\_b | Object | Hsc board |
| Hsc\_s | Object | Hsc stream |
| Degree\_p | Float | Degree percentage |
| Degree\_t | Object | Graduation stream |
| Workex | Object | Are they having any work experience |
| Etest\_p | Float | Online test percentage |
| Specialisation | Object | Specialization the choosed in Mba |
| Mba\_p | Float | Mba percentage |
| Status | Object | Are they placed or not placed. This the outcome column. |

**2.2 Data Gathering**

Data source: <https://www.kaggle.com/competitions/ml-with-python-course-project/data>

Train and Test data are stored in .csv format.

**2.3 Raw Data Validation**

After data is loaded, various types of validation are required before we proceed further with any operation. Validations like checking for zero standard deviation for all the columns, checking for complete missing values in any columns, etc. These are required because The attributes which contain these are of no use. It will not play role inpredicting the status of placement.

Like if any attribute is having zero standard deviation, it means that’s all the values are the same, its mean is zero. This indicates that either the student placed or not that attribute will remain the same. Similarly, if any attribute is having full missing values, then there is no use in taking that attribute into an account for operation. It’s unnecessary increasing the chances of dimensionality curse.

**2.4 Data Transformation**

Before sending the data into the database, data transformation is required so that data are converted into such form with which it can easily insert into the database. Here, the columns which have any missing values are filled in both the train set as well as the test set with supported appropriate data types.

**2.5 Data Preprocessing**

In data preprocessing all the processes required before sending the data for model building are performed. Here, I removed all those columns who standard deviation is zero because they are not contributing anything and those columns also which are of no use in predicting the status of Placement. After that I separate label and features columns from the train and test data.

**2.6 Feature Engineering**

After preprocessing it was we saw that there are few categorical columns present in the data. For converting the categorical column, I have created the column transformer object and inside it I performed one hot encoding on categorical columns. Afterwars I save the transformer object so that it can be used for transforming the test and prediction data in same way as train data. I have converted the target column categorical values into numerical using label encoder.

To handle imbalance data we have performed oversampling using imblearn smote function.

In the same step we also performed feature scaling to bring down every feature on the same page in terms of value range.

**2.7 Parameter Tuning**

I have used sklearn pipeline library with grid search cv to peform hyper parameter tuning. I have used different algorithms in pipeline and set different parameters for their important attributes.

**2.8 Model Building**

After doing all kinds of preprocessing operations mention above and performing scaling and hyperparameter tuning, the data set is passed into 3 models, SVC, XGB and Random Forest. It was found that Radom forest classifier performs best with the highest accuracy score equals 0.89. So ‘Random forest classifier’ performed well in this problem.

**2.9 Model Saving**

Model is saved using pickle library.

**2.10 Django Setup for Data Extraction**

After saving the model, the API building process started using Flask. Web application creation was created here. Whatever the data user will enter and then that data will be extracted by the model to predict the prediction of placement status, this is performed in this stage.

**2.11 GitHub**

The whole project directory will be pushed into the GitHub repository.

**2.12 Deployment**

The cloud environment was set up and the project was deployed from GitHub into the Heroku cloud platform.

App link-<https://aditya-ineuron.herokuapp.com/>

**3. Unit Test Cases.**

|  |  |  |
| --- | --- | --- |
| **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 whether a user is able to see input fields while opening the application | 1. Application is accessible 2. The user is able to see the input fields | Users should be able to see input fields on logging in |
| Verify whether a user is able to enter the input values. | 1. Application is accessible 2. The user is able to see the input fields | The user should be able to fill the input field |
| Verify whether a user gets predict button to submit the inputs | 1. Application is accessible 2. The user is able to see the input fields | Users should get Submit button to submit the inputs |
| Verify whether a user is presented with recommended results on clicking submit | 1. Application is   accessible   1. The user is able to see the input fields. 2. The user is able to see the submit button | Users should be presented with recommended results on clicking submit |
| Verify whether a result is in accordance with the input that the user has entered | 1. Application is accessible 2. The user is able to see the input fields. 3. The user is able to see the submit button | The result should be in accordance with the input that the user has entered |