Low Level Design

FLIGHT PRICE PREDICTION

Document Control

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 0.1 | 17/03/2024 | Prince Francis | Low Level Design Document |
|  |  |  |  |
|  |  |  |  |

CONTENTS

Document Version…………………………………………………………………………………………………….2

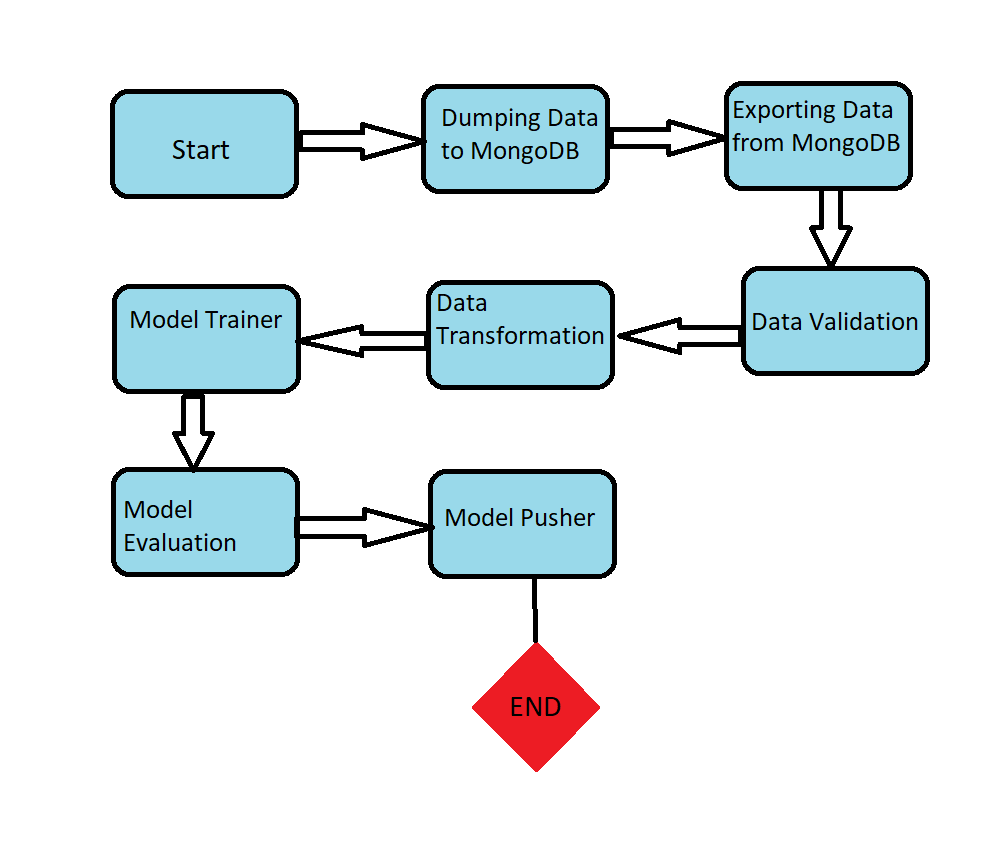
1. Introduction……………………………………………………………………………………………………….4
   1. Why this Low-Level Design Document?..........................................................4
   2. Scope…………………………………………...................................................................4
2. Architecture………..…………………………………………………………………………………………….5
   1. Data Description…..……………………………………………………………………………………..5
   2. Data Dumping …………………………………………………………………………………………….5
   3. Data Ingestion …..………………………………………………………………………………………..6
   4. Data Validation ………..………………………………………………………………………………….6
   5. Data Transformation …..……………………………………………………………………………….6
   6. Model Trainer……..……………………………………………………………………………………….6
   7. Model Evaluation…………………………………………………………………………………………6
   8. Model Pusher………………………………………………………………………………………………6
   9. Deployment………………………………………………………………………………………………...7
3. Introduction
   1. Why this Low-Level Design Document?

The goal of LLD or low-level design document(LLDD) is to give the internal logical design of the actual program code for Flight Price Prediction system. 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. 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.

1. Architecture



* 1. Data Description
* Data\_Train.xlsx – Contains training dataset with 10683 rows and 11 columns
* Test\_set.xlsx – Contains 2671 rows and 10 columns excluding the output column.
* Sample\_submission.xlsx – This contains the output variable column being predicted. It is provided as a reference.
  1. Data Dumping

Data is first dumped into the MongoDB. Both the Train and Test set are being loaded to separate collection under the same databases. For this the data is being first converted to JSON file format.

* 1. Data Ingestion

The dumped data is then converted back to dataframe using Pandas. Thereafter the training data is further split to train and test data and saved to concerned artifact directory.

* 1. Data Validation

In data validation we can check the data for outliers. We can check if the data is meaningful or not. If there are missing values in the data. Splitted data i.e Train and Test set is then checked for data drift against the base dataset.

* 1. Data Transformation

Dataset is then preprocessed or feature engineering is done on top of it. Rows containing null values are being removed. After that normalization is performed over the dataset. PCA transformation is used for dimentionality reduction. Finally the objects of normalization and PCA are serialized and stored to concerned artifact directory.

* 1. Model Trainer

Model is then trained over the top of the best model been chosen by us. Different regression models can be chosen to train the dataset. Evaluation metric used is r2\_score. We have to check if the model is overfitting or underfitting first.

If either of the two are not true then we can choose the best performing model.

Furthermore, we can use hyperparameter tuning technique like GridSearchCV or RandomSearchCV for improving the model accuracy. We can store the model object in the concerned artifact directory

* 1. Model Evaluation

This part of the architecture is used to check if the current model is better than the previous model. Since the coding lines inorder to solve the logic would be greater we can also create a separate helper function file for this.

* 1. Model Pusher

The objects of normalization, PCA and model can be only pushed into the saved\_models directory only if the performance of the current model is better that the previous model

* 1. Deployment

Finally we can push the code to Github. Then, deploy it on AWS.