Document Classification Project

Low Level Document

Written By	**Shraddha Sawant**
Document Version	**1.0**
Last Revised Date	**23 – Nov -2023**

Document Control

Change Record:

Version	Date	Author	Comments
1.0	23 – Nov -2023	Shraddha Sawant	Low Level Documentation

Reviews:

Version	Date	Reviewer	Comments

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Table of Contents

1. Introduction

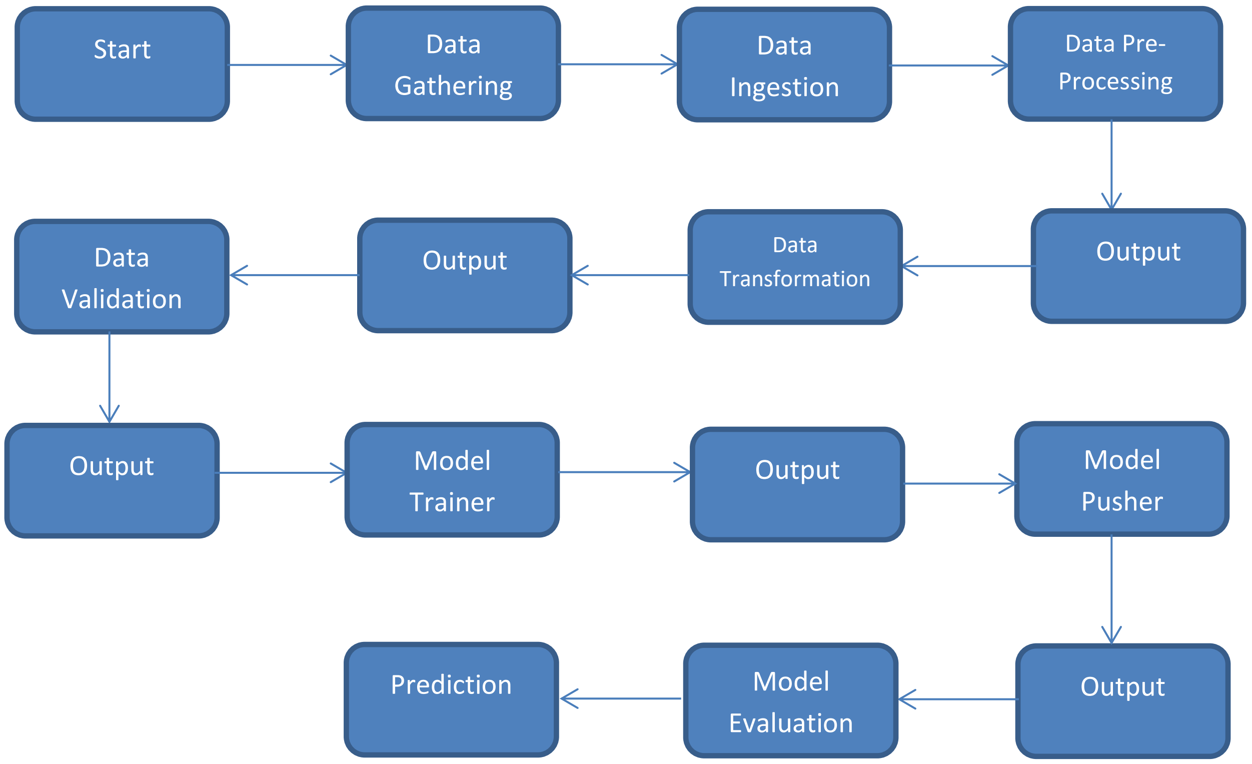
1.1 What is low level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Document Classification Process. 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 Design



3. Architecture Description

3.1 Data Description

IMDB reviews dataset is a dataset containing movie reviews. The shape of the dataset as 10,000 rows and two columns. The rows consist of reviews given my consumers. The reviews has text format and includes html tags, exclamation marks, full stops, quotation marks, etc. The second column is the tagged column 'Sentiment'. The tagging is done in two ways either positive or negative. Positive means a consumer is happy about the movie; Negative means a consumer is not happy about the movie.

3.2 Data Ingestion

Data Ingestion is the first stage of CI-CD pipeline. It includes bringing the data from either a database such as MongoDb or SQL or CSV or Json file and storing it in temporary variables. Once inside the temporary variables we can do the basic processing such as checking for duplicates, removing duplicates and checking for null values. Depending whether the data is a text data or numeric data, we perform some basic processing steps. Once the basic processing steps are done we have split the dataset into Train Set and Test Set and stored them in respective variables and CSV files to send further down in the pipeline.

3.3 Data Transformation

Data Transformation is the second stage of the CI-CD pipeline. It includes bringing the data split in data ingestion stage into temporary variables. Firstly, we split columns into their respective variables Feature Set and Target Set. The first column is given to input Feature set and the second column is given to Target Variable. In this stage, we have applied LableEncoder to our Target Column. Secondly we gave have performed feature engineering step to the input column. We have applied CountVectorizer to convert text data into 0s and 1s and then converted them into an array matrix. Once the above steps are performed we have converted both the input feature and output feature into csv files to send them further down the pipeline.

3.4 Model Trainer

In this stage or pipeline, we input data from the previous stage of Data Transformation and store it into temporary variables. We apply a model on the train input set and train output set.

We have decided on the best model to apply from our analysis stage performed using Jupyter Notebook. In this stage we push the model object into a directory to further use it in the pipeline.

3.5 Model Pusher

In this stage of the Pipeline we, push the model object and the encoder object into saved model directory so as It use both the objects on a new data that we get in next week or upcoming timeframe.

3.6 Model Evaluation

In this stage, we evaluate two models obtained, one of the models is the one which has been already trained in the pipeline and the other model is trained later when new data is obtained and then saved accordingly in the directories.

END