LOW LEVEL DESIGN

SCANIA TRUCK FAILURE PREDICTION

Document Control

|  |  |  |  |
| --- | --- | --- | --- |
| Date Issue | Version | Description | Author |
| 29/12/2021 | 1 | Initial LLD – V 1.0 | Sethu Sai Medamallela |
| 24/01/2022 | 1.1 | Initial LLD – V 1.1 | Sethu Sai Medamallela |

Content

1. Introduction
   1. What is Low Level Design Document
   2. Scope
2. Architecture
3. Architecture Description
   1. Data Description
   2. Data Transformation
   3. Feature engineering
   4. Feature Selection
   5. Testing for Classification algorithm
   6. Selecting model with best accuracy
   7. Model training
   8. Model deployment

Introduction

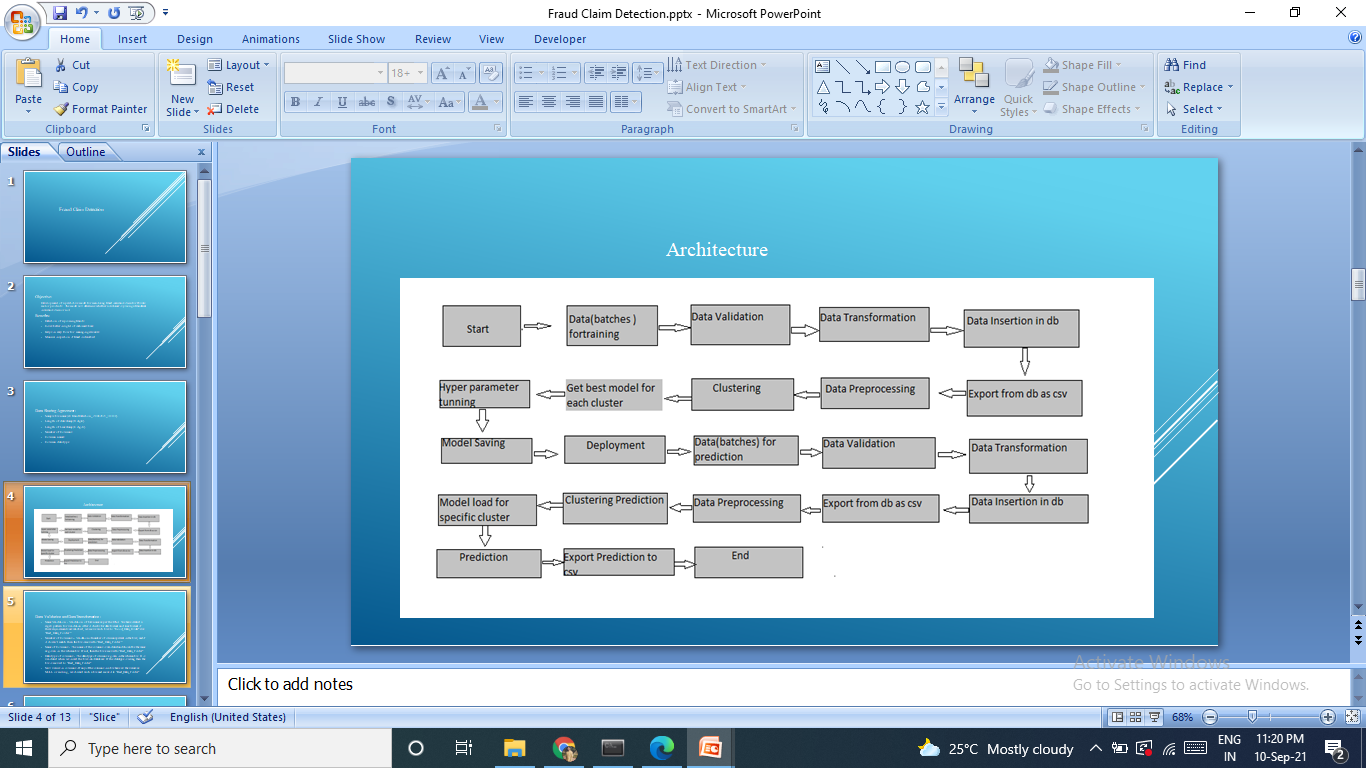
What is Low Level Design Document?

The goal of LLD or Low Level Design Document (LLDD) is to give the internal logic design of the actual program code for Predict Bank Credit Risk. LLD describes the class diagram with the methods and relations between classes and program specs. It describes the modules so that programmer can directly code from the document.

Scope

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

Architecture



Architecture

Data Description –

The scania truck failure dataset which is available in UCI Machine Learning Repository consists of 60000 rows and 171 columns, with label columns to be encrypted. The dataset consists of two files one for training and other and for testing. \For our convenience, we have split the train dataset into multiple batch of dataset each consisting of 3000 rows each.

Data Validation-

Before going to architecture, we are assuming that the data is generated from the sensors and Big Data pipeline is created which will store the sensor data in S3 buckets. Now the data is stored in S3 buckets, we are validating the filename from schema files, regex pattern, validate columns length and validating missing values in columns.

Data Transformation-

In the data transformation pipeline, we are adding quotes to string values for the columns.

Database Operation –

For the database operations, we have chosen MongoDB. In the database operations, we are inserting the good data as a collection in MongoDB. We are choosing M20 cluster in MongoDB Atlas with AWS as cloud provider. After the good data is inserted, we are exporting the good data as a collection in the form of csv and then uploading it to csv file back to S3 buckets.

Data Preprocessing

In the preprocessing pipeline, we are replacing the invalid values with nan values, encoding the targets column, imputing the missing values, removing unnecessary columns, applying standard scalar and PCA transformation for dimensionality reduction

Since we are following a customized machine learning approach, with KMeans model, we are creating the clusters of data, performing train test split and pass that cluster data for training.

Model Training-

In the previous stage, we had created clusters of data; now for that data we are training both Random Forest model and XGBoost model with Grid Search CV for hyper parameter tuning. These trained models are evaluated on the basis of AUC score and accuracy score depending on the cluster data. The trained models are saved to S3 buckets.

Now that models are trained and have their respective scores, we are using MLFlow for tracking the experiments, models, parameters and metrics. We have written the code in such a way that for a particular model, model parameters, model score and model itself are tracked in one go.

Loading of the production model-

Now that all the trained models are kept in track using MLFlow, we need to put our best models in production or staging depending on the condition. We are putting the best model of the particular cluster in production and other in staging for every cluster.

Containerization-

The entire system is containerized using Dockers with AWS and MLFlow credentials.

Deployment –

The entire solution is now containerized and we have implemented GitHub Actions as a CI-CD pipeline, which will deploy our system to AWS ECR and ECS and an endpoint is given to us for using the application.

Further improvements and Conclusion

For the improvements we can use Kubernetes for container orchestration and management, with that we can also use tools like Prometheus, Grafana for Kubernetes management. We can adapt to micro service architecture for the entire pipelines mentioned can be individually containerized and monitored using Kubernetes We also include more DevOps tools like Jenkins more automation. Most importantly retraining approach is important considering model performance from time to time.