

Baku, Azerbaijan

+994 50 4918219

[muhammad.ojagzada.std@bhos.edu.az](mailto:muhammad.ojagzada.std@bhos.edu.az)

**Metro Interstate Traffic Volume Prediction**

Low Level Design

Domain: Machine Learning

Creator: Muhammad Ojagzada

Date: 10.07.2022

|  |  |  |  |
| --- | --- | --- | --- |
| Date issued | Version | Description | Author |
| July 12, 2022 | 1.1 | First Draft | Muhammad Ojagzada |
| July 14, 2022 | 1.2 | Added unit test cases | Muhammad Ojagzada |

**Document Version Control**

Contents

[Introduction 3](#_Toc110433758)

[What is Low-Level Design Document? 3](#_Toc110433759)

[Scope 3](#_Toc110433760)

[Architecture 4](#_Toc110433761)

[Architecture Description 5](#_Toc110433762)

[Data Preparation 5](#_Toc110433763)

[Data Description 5](#_Toc110433764)

[Data Preprocessing 5](#_Toc110433765)

[Exploratory Data Analysis 5](#_Toc110433766)

[Feature Engineering 5](#_Toc110433767)

[Model Development 6](#_Toc110433768)

[Model implementation 6](#_Toc110433769)

[Hyper-parameter Tuning 6](#_Toc110433770)

[Model Evaluation 6](#_Toc110433771)

[Deployment 6](#_Toc110433772)

[Designing UI with Anvil 6](#_Toc110433773)

[Designing a server 6](#_Toc110433774)

[Code deployment on cloud 7](#_Toc110433775)

[Deployment Process 7](#_Toc110433776)

[Unit cases 8](#_Toc110433777)

# Introduction

## What is Low-Level Design Document?

The Low-Level Design (LLD) document aims to provide the internal logic of the program code for Metro Interstate Traffic Volume Prediction. It details the system's purpose, features, interfaces, operations, constraints, and responses to external stimuli.

The primary objective of this project is to predict whether traffic volume will be high or low on specific dates. Key factors influencing predictions include weather conditions, special days like holidays, time of day (morning, afternoon, night, etc.), temperature, day of the week, and percentage of cloud cover.

## Scope

Low-Level Design (LLD) involves a meticulous step-by-step refinement process at the component level. This process encompasses the design of data structures, necessary software architecture, source code, and performance algorithms. Data organization is initially defined during requirement analysis and further refined throughout the data design phase.

**Architecture**

**Data Preparation**

**Model**

**Development**

**Deployment**

**Deployment**

**Architecture Description**

## Data Preparation

### Data Description

The dataset is a record of credit card clients in Taiwan and includes 30,000 observations across 25 features. Each row represents an individual client, identified by a unique ID. The dataset provides various attributes of these clients, such as their credit limit, demographic information, repayment status, bill statements, and previous payments.

The **LIMIT\_BAL** feature indicates the amount of credit granted to the client in NT dollars, with values ranging from 10,000 to 1,000,000 NT dollars, and a mean of 167,484 NT dollars.

Demographic attributes include **SEX** (1 for male, 2 for female), **EDUCATION** (1 for graduate school, 2 for university, 3 for high school, 4 for others), **MARRIAGE** (1 for married, 2 for single, 3 for others), and **AGE**, which ranges from 21 to 79 years with a mean age of 35.49 years.

The repayment status features **PAY\_0** to **PAY\_6** reflect the history of payment delays, with values indicating timely payment or delay durations in months (e.g., 0 for on-time, 1 for one month delay, etc.).

Bill statement amounts for the last six months are provided by **BILL\_AMT1** to **BILL\_AMT6**, with considerable variation, including negative values and a maximum of 961,664 NT dollars. Similarly, **PAY\_AMT1** to **PAY\_AMT6** show the amounts of previous payments over the same period, ranging up to 1,684,259 NT dollars.

Finally, the **default.payment.next.month** feature indicates whether the client defaulted on their payment the following month (1 for yes, 0 for no), with a default rate of approximately 22.12%.

Overall, the dataset is rich in features that allow for detailed analysis of credit card usage, repayment behavior, and the prediction of default risks.

Data Preprocessing

In data preprocessing step, we check if there missing data, duplicate values, and datatypes of each feature. In our dataset, there was not any null and duplicate values

### Exploratory Data Analysis

In the exploratory data analysis (EDA) phase, we addressed the following:

**Data Transformation**: We replaced the columns **BILL\_AMT1** (Amount of bill statement in September 2005), **BILL\_AMT2**, etc., and **PAY\_AMT1** (Amount of previous payment in 2005), **PAY\_AMT2**, etc., with a new column called **Overdue\_pay\_mon**, which represents the month-wise due payment.

**Missing Values and Outliers**: We found that there are no missing values in the dataset. However, there are outliers. Since detecting fraud transactions is our primary focus, dealing with outliers is crucial

**EDA Tools and Techniques**: For EDA, we utilized the pandas library for initial handling and analysis. Additionally, we used **ydata profiling** to gain a comprehensive understanding of the relationships and correlations between columns.

**Handling Skewed Data**: We observed that the **Amount/Salary** column was skewed, which is typical but not ideal for our model. Initially, we applied a function transformer, but it did not yield the expected outcome. Subsequently, we used a power transform, which performed better than expected.

**Class Imbalance**: We identified a significant class imbalance in the dataset, with fraud transactions being much less frequent than non-fraud transactions, at a ratio of approximately 2:10. To address this, we applied the SMOTE (Synthetic Minority Over-sampling Technique) to balance the dataset.

Despite these efforts, the model performance remained similar before and after applying SMOTE.

### Feature Engineering

**Feature Engineering**:

**Data Transformation**: We replaced the columns **BILL\_AMT1** (Amount of bill statement in September 2005), **BILL\_AMT2**, etc., and **PAY\_AMT1** (Amount of previous payment in 2005), **PAY\_AMT2**, etc., with a new column called **Overdue\_pay\_mon**, which represents the month-wise due payment. Additionally, we removed the **ID**, **BILL\_AMT**, and **PAY\_AMT** columns as they were deemed not essential for the model.

**Handling Skewed Data**: We observed that the **Amount/Salary** column was skewed, which is typical but not ideal for our model. Initially, we applied a function transformer, but it did not yield the expected outcome. Subsequently, we used a power transform, which performed better than expected.

## Model Development

### Model Implementation

After splitting the data into training and testing sets, I created a pipeline containing a Standard Scaler and Ordinal Encoder. This pipeline was then fitted to several models, including:

1. Decision Tree
2. Random Forest
3. Logistic Regression
4. AdaBoost
5. XGBoost

I also used a stacking technique to combine different models and achieve higher accuracy and recall.

### Hyper-parameter Tuning

The best-performing model from the initial implementation was selected for further tuning. Grid Search with Cross Validation was applied to this model to find the optimal hyperparameters. These best parameters were then used to refine the model, aiming to achieve better performance.

### Model Evaluation

The final model was evaluated using the test dataset, with 20% of the dataset set aside for testing. The predicted results of the model were compared with the actual data to check the amount of error. Despite using advanced techniques and hyperparameter tuning, the model achieved an accuracy of 82% However, in recall detecting fraud specifically, the model achieved only 38% accuracy.

### Technologies Used:

* Python
* Pandas, NumPy
* Sklearn
* Matplotlib, Seaborn

The model's predictions can significantly aid financial institutions in mitigating risks associated with credit lending. Check out the project details and feel free to share your thoughts!

## Deployment

### Designing UI with Anvil

For this project, a user interface is built on Anvil. It is a web application that helps us to create applications for projects. It is a free Python-based drag-and-drop web app builder.

### Designing a server

A server should be created to run the UI application continuously. Flask server is built, and it is linked with Anvil uplink that connects Anvil UI with our server.

### Code deployment on cloud

The codes for this machine learning model should be deployed to the cloud, so that when data is entered into the application, our code runs, and a user gets the result online.

## Deployment Process

In this stage, we establish a server using Flask that runs the uplink code (server code) in parallel before developing the UI using Anvil and connecting with our code, where our model is executing, via an uplink. We will post the hole after execution or asynchronous execution. Git and GitHub are used to code in the Heroku cloud. Then, we'll configure a cron job to maintain the server and server code in operation indefinitely.

# Unit cases

|  |  |  |
| --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **Expected Result** |
| Verify whether the Application URL is  accessible to the user | 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 user is able to see input fields. | Application is accessible | User should be able to see input fields |
| Verify whether user is able to edit all input fields | Application is accessible | User should be able to edit all input fields |
| Verify whether user gets Submit button to submit the inputs | Application is accessible | User should get Submit button to submit the inputs |
| Verify whether user is presented with results on clicking submit | Application is accessible | User should be presented with results on clicking submit |
| Verify whether the results are in accordance to the selections user made | Application is accessible | The results should be in accordance to the selections user made |