1. Choose a project from an internship portal and try to write a HLD and LLD based on the sample given in your portal for a respective project .

For writing HLD and LLD I am choosing Forest Cover Prediction project.

HLD ---Page No 2 – 9

LLD --- Page No 10 – 16

**HLD**

# Abstract

Forest land is highly required for developing ecosystem management. Any changes that occur in ecosystem should be carefully noticed to avoid further loss. This model is helpful in noticing the changes occurred due to heavy floods or any other calamities which affected the forest land.

1. **Introduction**

# Why High Level Design?

The main purpose of HLD is to add the necessary details to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level. Document includes the following aspects:

✔ Architecture Design and Definition

✔ User Interface Implementation

✔ Hardware/Software Interface

✔ Performance Requirements

✔ Non – functional attribute definitions

✔ Security, Reliability, Maintainability, Portability, Reusability, Application Compatibility, Resource Utilization, Serviceability

# Scope

Documentation provides the system structure, such as DB architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non– technical terms which should be understandable to the administrator.

# Definitions

|  |  |
| --- | --- |
| **Term** | **Description** |
| IDE | Integrated Development Editor |
| MLA | Machine Learning Algorithm |
| EDA | Exploratory Data Analysis |
| ETL | Extract, Transform, Load |

1. **General Description**

# Problem Statement

To create machine learning model to detect anomalies in credit repayment transaction and predict whether the account will be defaulted or not.

# Proposed Solution

The proposed model will be capable of studying forest data based on various parameters like elevation,slope,soil type,aspect,wilderness area. Based on this we can predict the cover type of any given forest.

# Further Improvements

Further we can enhance our model with additional information like average temperature ,precipitation.This extra information can help us in predicting the forest cover type with much more accuracy.

# Requirements

Model performance depends on various factors like dataset we use to train it, prediction algorithm, hyper tuning parameters

✔ Input Dataset – Data should be clean without any outliers, missing values, garbage values

✔ Outliers – can greatly impact the model performance

✔ Missing/Garbage values – Data collected often have missing and garbage values which has to cleaned before training the model

✔ Choice of algorithm – Depending on the problem statement appropriate algorithm is used to train

# Tools

Python programming language is our primary tool to build the model with few open-source libraries for various subtasks

# Pandas

Python package providing fast, flexible and expressive data structure designed to make working with relational or labelled data both easy and intuitive

# 3.2 Numpy

Python library used for working with arrays with domain specific functionalities in Linear algebra, Fourier transform and Matrices.

# Scikit-Learn

High level framework designed for supervised and unsupervised machines learning algorithms. It provides efficient tools for statistical and scientific models for illustrations.

# Streamlit cloud

It is a container based cloud Software as a Service (SaaS) used to deploy, manage and scale modern applications. It provides elegant, flexible solutions for developers to publish the application.

# Plot Visualization

Seaborn and Matplotlib libraries used to create 2D graphs and plots to visualize the data

# Streamlit

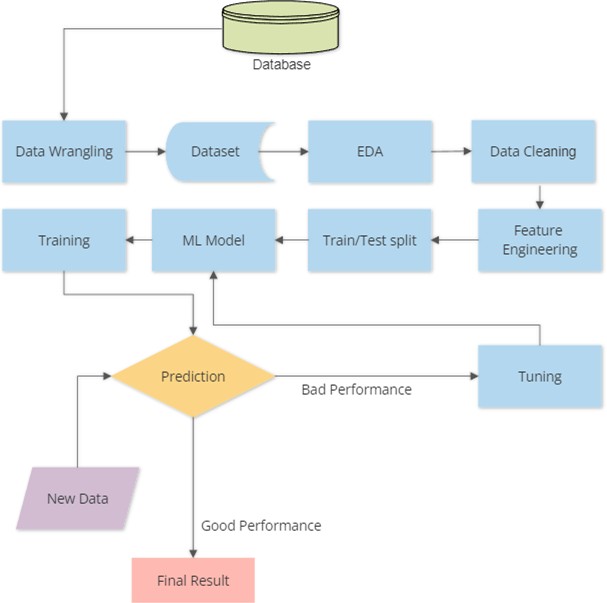
# We have used streamlit library for creating API for the model.Using this API user can do prediction for the data which they will provide.

# Github

Cloud based online software development platform used for storing, tracking, and collaborating on software projects. It enables developers to upload their own code files and to collaborate with fellow developers on open-source projects

# Design Architecture

* 1. **Model Training and Evaluation**

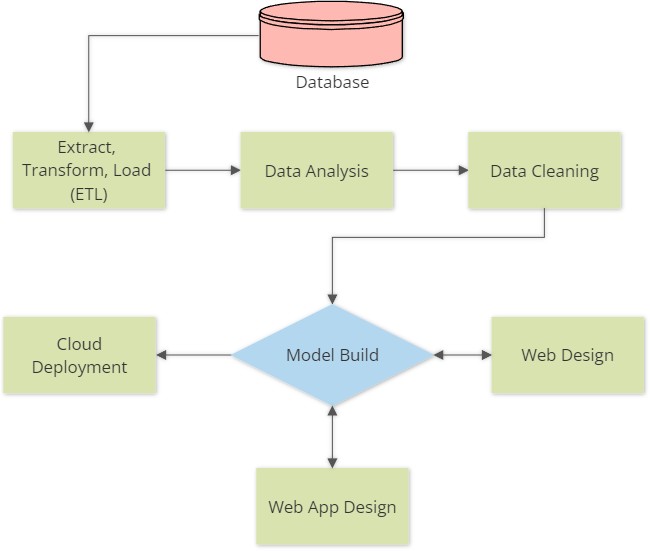


**Data Wrangling** – Process of transforming and mapping the raw data into desirable format with the intent of making it more valuable dataset

**EDA** – Analysis of data to summarize their main characteristics often using statistical graphs and visualization methods

**Feature Engineering** – addition and construction of additional features to improve model performance and accuracy

# Deployment Process



**ETL** – Three phase process where data is collected from multiple sources, transformed into desired format and loaded into multiple output containers

**Web App** – Front end app interface using Streamlit library

# Model Management

* 1. **Event Log**

Model should log every event from start to end for debugging purpose. We can spot the error in logs and identify which part of the code need to be addressed and corrected. Logging is essential to see the internal execution flow and can be used to improve the architecture design for faster execution.

# Error Handling

Error Handling makes a model robust, reliable ensuring the execution wont stop abruptly. Even in the worst case scenario the execution will terminate in between intimating the user the reason for termination. We should imagine all possible worst case scenarios and able to code the handlers to terminate safely or jump a step ahead in the execution after intimating the user.

# Performance

As we discussed before model performance entirely depends on training dataset, quality of the data, authenticity of the data, algorithm used and hyper-parameter tuning. But can assure the performance will be good assuming the above criteria are all satisfied.

# Reusability

All codes are written in modular fashion so that for future improvements codes can be reused or modified without affecting other modules. Improvements are necessary for better performance, faster execution and updated dataset training for the model.

# Compatibility

No model is good if it not compatible. Model should be build in such a way it can perform in all scenarios provided with required input details. For this reason we need to use widely prevalent version of framework or library supporting maximum systems to run the model.

# Portability

Models build in containers have high portability. Portable containers will make sure the model executes and performs well even if the system doesn't have necessary libraries pre-installed.

# Deployment

Deploying the model in cloud platform makes it readily available for action. We can run and extract the results any time. Cloud platform will provide the necessary hardware and software support for the execution.

# Conclusion

We have build multiple classification models, trained and tested with same datasets, and finally performed parameter tuning for decently performing model to achieve best prediction and f1\_score.

**LLD**

1. **Introduction**

# Why Low-Level Design?

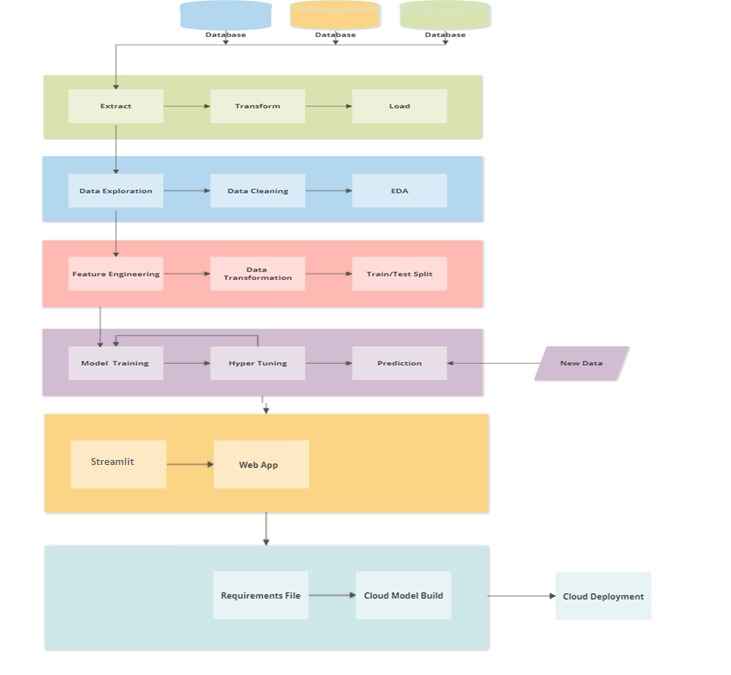
Low-Level design is a component-level design process with detailed description and logic for every module. It is also called micro level design designed by administrators and developers converting high level solution into detailed solution.

# Scope

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code. Low-level design is created based on the high level design. LLD describes the class diagrams with the methods and relations between classes and program specs.

# Architecture

* 1. **Architecture Design**



# Extract, Transform and Load

Here the data is retrieved from multiple databases and then transformation is applied to extract the required data in useable format. Before this process the data will be a clutter and will not be viable for data analysis.

Later the transformed data is sent to different entities based on the request.

# Data Exploration

Data received will be in tabular format but it may contain data in unstructured manner like string data type might be used for columns having numbers. So our first step is to explore all rows and columns for an eagle eye overview.

# Data Cleaning

After being explored we should have an idea about the quality of the data to step into data cleaning process. This step is very crucial for the model performance as it may affect the model prediction if wrong data is used to train. Many columns may have missing values or garbage values which can be handled by replacing it with mean or median in-case of numerical data or we can drop the corresponding rows after thorough examination making sure that it won’t affect the prediction.

# Data Transformation

Data should be normally distributed for maximum model performance, so we transform the data using various techniques like log transformation, polynomial transformation and then train the model for premium performance.

# EDA (Exploratory Data Analysis)

It is an approach of analysing the data to summarize the main characteristics using statistical graphics and data visualization methods. It shows what the data can tell us beyond the formal modelling and thereby contrasts traditional hypothesis.In this project we have taken help of pandas profiling library for doing EDA.

# 2.7 Feature Engineering

It is a technique that leverages data to create new variables that aren't in the training set. It can produce new features for both supervised and unsupervised learning with the goal of simplifying and speeding up data transformation while enhancing model accuracy.

# 2.8 Train/Test Split

We now split the dataset into two set, one for training with 80% data and the remaining 20% data is used for testing the model for accuracy. You can split the dataset in any ratio with the majority for training and minority for testing respectively. We use the random state value to get the same training and test dataset for training all models repeatedly after hyper tuning.

# 2.9 Model Building

Performance of the model not only depends on the training data but also with choice of algorithm. Depending on the problem statement the algorithm is chosen and trained with tuning to yield best results. In our case we chose to build the model using multiple classification algorithms like Logistic Regression, Decision Tree, SVM Classifier, Random Forest and Naive Bayes. We train all models with same training dataset and predict the accuracy with same test data.

# 2.10 Hyper Tuning

Tuning is performed to improve predicting accuracy based on problem statement. Some problem concentrates more false negative, in that case we tune our model for better recall accuracy. In case of true positive we tune for better precision accuracy. We compare all models and perform tuning for best to improve recall accuracy because our model is concerned more on False Negative.

After doing model building and hyper tuning we decide which model is the best model for doing prediction and then we do prediction using the decided model.

# 2.11 Prediction

After the training and tuning we feed test data to predict the result. We compare the predicted result with actual result to calculate the accuracy. If the model performance is not up-to the mark we will fine-tune the parameters till we achieve the best accuracy.

# 2.12 Streamlit API

We need a front-end interface to receive input from the user. We use Streamlit library for making our webapp.In the webapp we take the input in form of csv file.We do this because there are too many columns in the dataset and it is very time consuming to take input for all the columns.

# 2.13 Requirements file

This file helps to build the cloud container with necessary packages to run the application. Without this we cannot deploy as it may not know the environment for the app to run.

# 2.14 Cloud deployment

Once we provide project files and requirement file the cloud container starts to build the environment, installing necessary packages and runs the application as Web App.

# 3. Unit Test Case

|  |  |
| --- | --- |
|  | |
| **Test Case Description** | **Expected Result** |
| Verify whether Application URL is accessible to the user | URL should be accessible to the user |
| Verify whether the Application loads completely for the user when the URL is accessed | The Application should load completely for the user when the URL is accessed |
| Verify whether user is able to see the input field. | User should be able to see input fields |
| Verify whether user is presented with the result after the input file has been submitted. | User should be presented with prediction results |
| Verify whether the recommended results are in accordance to the selections user made | The recommended results should be in accordance to the selections user made |