LOW LEVEL DESIGN

WHEAT KERNEL CLASSIFICATION

Submitted By	Arslan Ahmed Butt

Table of Contents

1. Introduction:

a. 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 Food Recommendation 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.

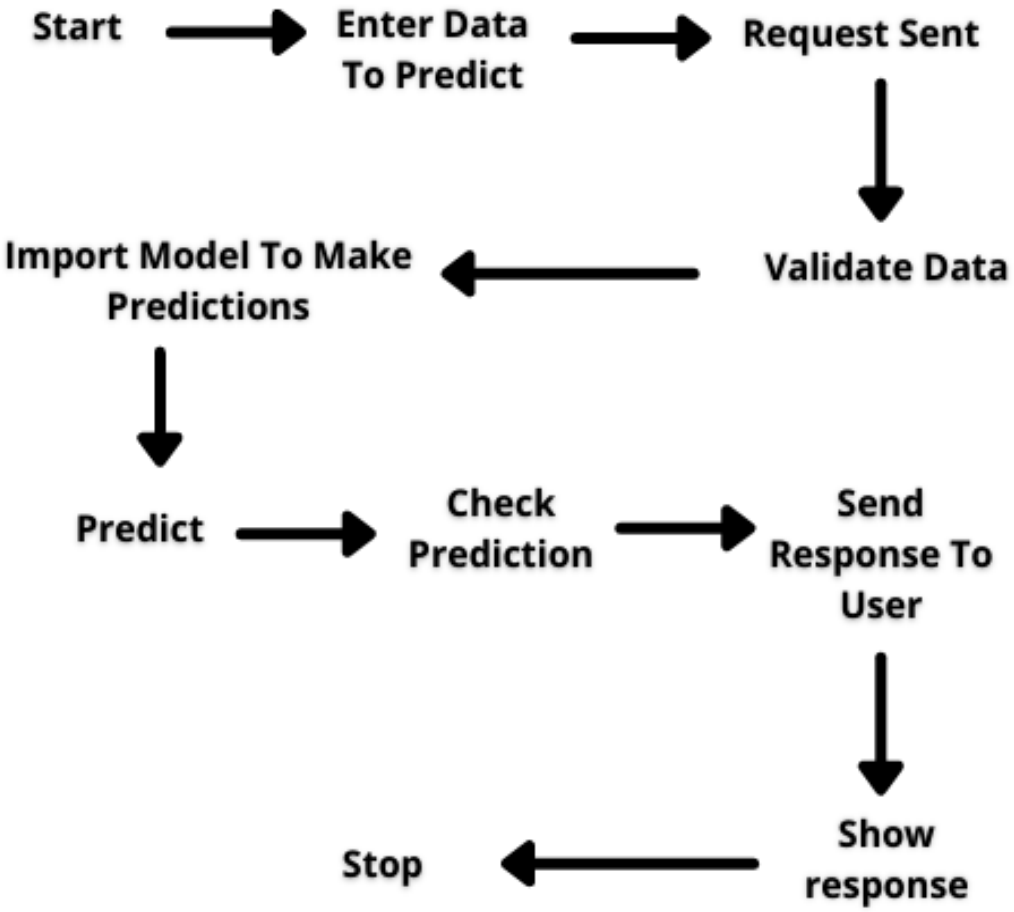
b. Scope

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

2. Project Description:

For deploying the model. I created a web app in **Django** to make a user-friendly interface for users to make predictions. A single page is created which asks the user to input data for predicting the type of kernel. After the user sends a request to make predictions. Data sent by user is validated, parsed and passed to model predict the type. The model file is imported once when the server starts running after that the user starts making predictions.

3. Architecture:



4. Architecture Description:

a. Data Description:

Measurements of geometrical properties of kernels belonging to three different varieties of wheat. A soft X-ray technique and GRAINS package were used to construct all seven, real-valued attributes. The examined group comprised kernels belonging to three different varieties of wheat: Kama, Rosa and Canadian, 70 elements each, randomly selected for the experiment.

b. Web App:

In order to get data from a user to make predictions, I created a web app in Django and a user-friendly interface which gets data from user and sends data to backend to make predictions on trained model.

c. Backend:

The backend is also written in Django. When user makes a request to make predictions, request is sent to backend function which validates, parses and transforms the data to ready it for predicting the type.

d. Predict:

After the data is ready and parsed to make predictions. The trained model which is saved in a pickle file is imported to start making prediction.

e. Response:

After the model makes the predictions, a valid response showing the type of kernel is sent back to the user which is being displayed on the web page.

5. Model Training:

a. Dataset:

There are 210 records in our dataset and there are 3 types of classes to predict Kama, rosa and Canadian classified as 0,1 and 2 respectively.

b. Model Building:

Since it is a classification problem. There are two algorithms which work best with classification problem KNN and random forest classifier. First I divided the dataset into 80% and 20% for training and testing respectively then I trained KNN model on my dataset and found the best value of k at which the model gives the best accuracy. The KNN model gives the 90% accuracy using test set at k = 4 which was the best accuracy using KNN. Next, I used random forest classifier and used random search cv on random forest to find the best parameters to find the best model. The best accuracy I could get using random forest classifier was 95% and saved the model into a pickle file for making further predictions.

6. Deployment:

I used Heroku for deploying my web app.

The web app can be accessed through the following link.
Wheat Kernel

7. Unit Test Cases:

Text 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	1. Application is accessible	User should be able to see input fields
Verify whether user is able to edit all input fields	1. Application is accessible	User should be able to edit all input fields
Verify whether user gets Submit button to submit the inputs	1. Application is accessible	User should get Submit button to submit the inputs
Verify whether user is presented with recommended results on clicking submit	1. Application is accessible	User should be presented with recommended results on clicking submit
Verify whether the recommended results are in accordance to the selections user made	1. Application is accessible	The recommended results should be in accordance to the selections user made