**Low Level Design(LLD)**

Prediction of Heating and Cooling Load of a Building

|  |  |
| --- | --- |
| Done by | Hariprasath G |
| Document Version | 0.2 |
| Last Revised Date | 06-08-2023 |

**Document Control**

**Change Record:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 0.1 | 02-08-2023 | Hariprasath G | Introduction and System Architecture |
| 0.2 | 06-08-2023 | Hariprasath G | Architecture Description and Test Report |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Reviews:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Reviewer** | **Comments** |
|  |  |  |  |

**Approval Status:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Review Date** | **Reviewed by** | **Approved by** | **Comments** |
|  |  |  |  |  |

|  |  |
| --- | --- |
| **Contents** |  |
| 1. Introduction……………………………………………….. | 4 |
| 1.1 What is Low-Level Design Document?......................... | 4 |
| 1.2 Scope…………………………………………………. | 4 |
| 1. System Architecture………………………………………. | 5 |
| 1. Architecture Description………………………………….. | 6 |
| 3.1 Data Description……………………………………… | 6 |
| 3.2 Data Validation……………………………………….. | 7 |
| 3.3 Data Transformation…………………………………. | 8 |
| 3.3.1 Handling Duplicates……………………………. | 8 |
| 3.3.2 Feature Validation……………………………… | 9 |
| 3.3.3 Dimensionality Reduction……………………… | 10 |
| 3.4 Model Trainer………………………………………… | 11 |
| 3.4.1 Trainer\_y1……………………………………… | 11 |
| 3.4.2 Trainer\_y2……………………………………… | 11 |
| 3.5 Utlility………………………………………………... | 12 |
| 3.5.1 Create Directory………………………………... | 12 |
| 3.5.2 Save and Load………………………………….. | 12 |
| 3.6 Model Loader………………………………………… | 13 |
| 3.7.1 Loader\_y1………………………………………. | 13 |
| 3.7.2 Loader\_y2………………………………………. | 13 |
| 3.8 Application……………………………………………. | 14 |
| 1. Test Report………………………………………………... | 15 |
|  |  |

1. **Introduction**
   1. **What is Low-Level Design Document?**

Low-level design refers to the detailed and specific design specifications for individual components or modules within a system. It involves defining the internal workings, interfaces, algorithms, data structures, error handling, security measures, performance optimizations, and other implementation details required for the development and integration of the system components.

* 1. **Scope**

The scope of a low-level design document, encompasses the detailed design specifications and considerations for individual components or modules within a system. It involves defining the internal workings, interfaces, and interactions of each component, as well as addressing specific design aspects like data structures, algorithms, error handling, security measures, and performance optimization. The document outlines the granular details of the system's design, including the model architecture, data preprocessing steps, feature engineering techniques, hyperparameter tuning, model training and evaluation procedures, deployment architecture, monitoring mechanisms, and maintenance guidelines. It serves as a comprehensive guide for developers to implement and integrate the components effectively, ensuring a cohesive and well-designed system that meets the project's requirements.

1. **System Architecture**

****



Start

Data Validation

Handling Duplicates

Validation of Datatypes

Dimensionality Reduction



Model Loader on Y1

Front-End

Data From User

Prediction

End

Application

Model Training on Y1

Model Training on Y2

Model Loader on Y2

1. **Architecture Description**
   1. **Data Description**

This study looked into assessing the heating load and cooling load requirements of buildings (that is, energy efficiency) as a function of building parameters.

We perform energy analysis using 12 different building shapes simulated in Ecotect. The buildings differ with respect to the glazing area, the glazing area distribution, and the orientation, amongst other parameters. We simulate various settings as functions of the afore-mentioned characteristics to obtain 768 building shapes. The dataset comprises 768 samples and 8 features, aiming to predict two real valued responses. It can also be used as a multi-class classification problem if the response is rounded to the nearest integer.

The dataset contains eight attributes (or features, denoted by X1...X8) and two responses (or outcomes, denoted by y1 and y2). The aim is to use the eight features to predict each of the two responses.

Specifically:

X1 Relative Compactness

X2 Surface Area

X3 Wall Area

X4 Roof Area

X5 Overall Height

X6 Orientation

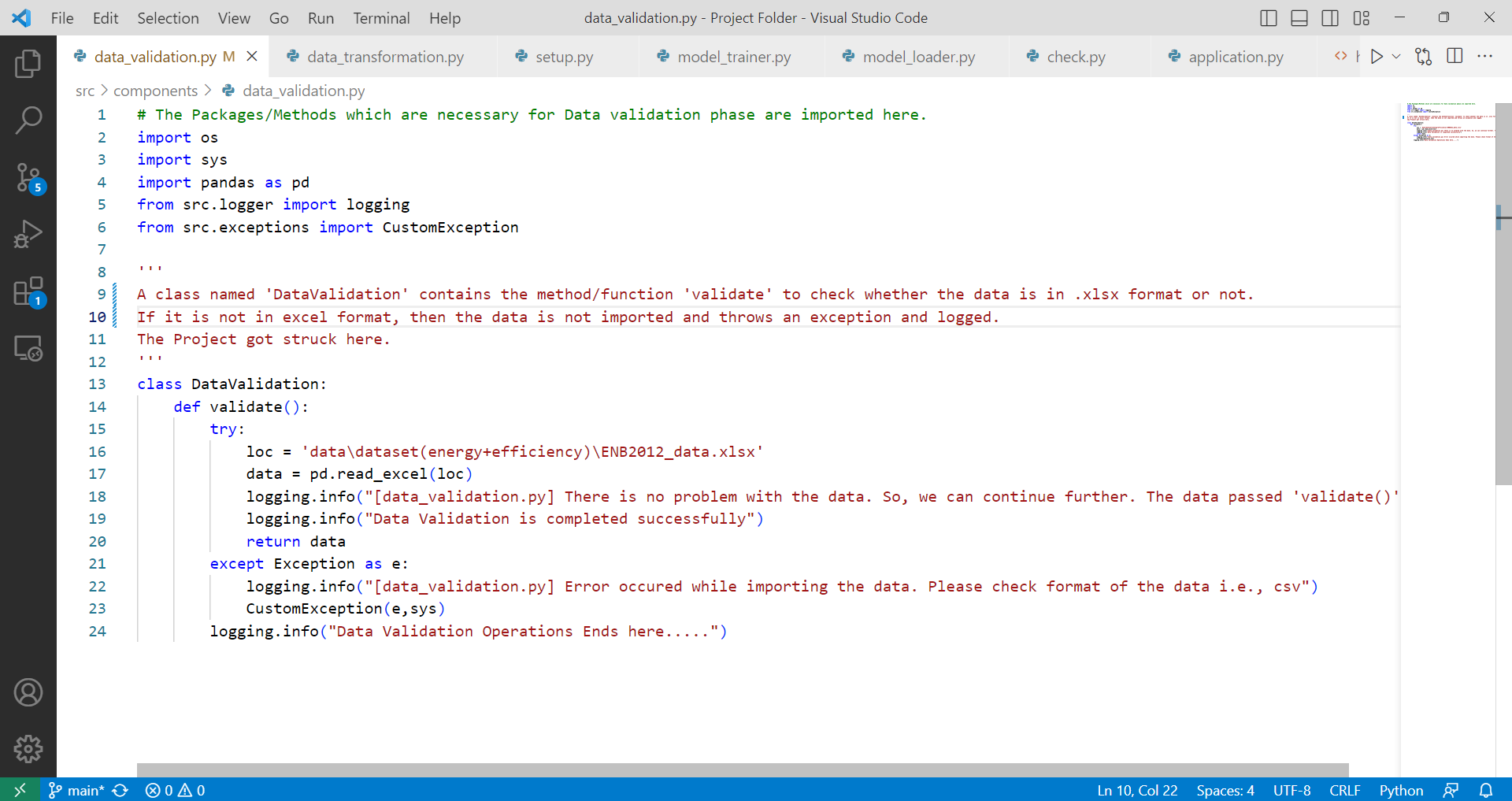
X7 Glazing Area

X8 Glazing Area Distribution

Y1 Heating Load

Y2 Cooling Load

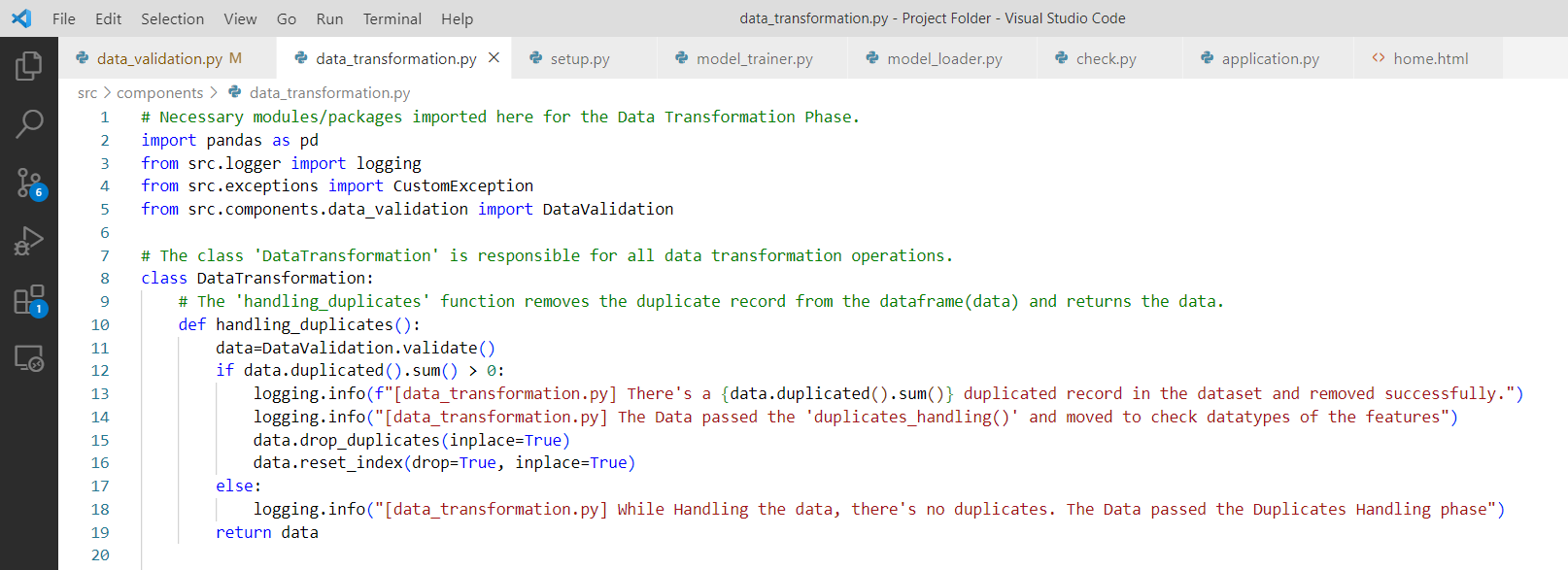
* 1. **Data Validation**

****

The class ‘DataValidation’ from ‘data\_validation.py’ file of folder located in ‘src/components/’ is resposible for all operations of data validation. The Necessary packages like logger, exceptions, os and pandas were imported. The class ‘DataValidation’ contains only one function i.e., validate(). In try block, the desired location/path of the dataset is assigned to ‘loc’, list of features names were assigned to ‘column\_names’ and the data is read with the help of ‘read\_excel’ method of ‘pandas’ package, then assigned to ‘data’. The message inside the logging function is recorded only if all operations mentioned above in the try block were successfully completed. Or else the except block is triggered and raises an exception in the terminal and log will be recorded as ‘Error Occured’.

* 1. **Data Transformation**

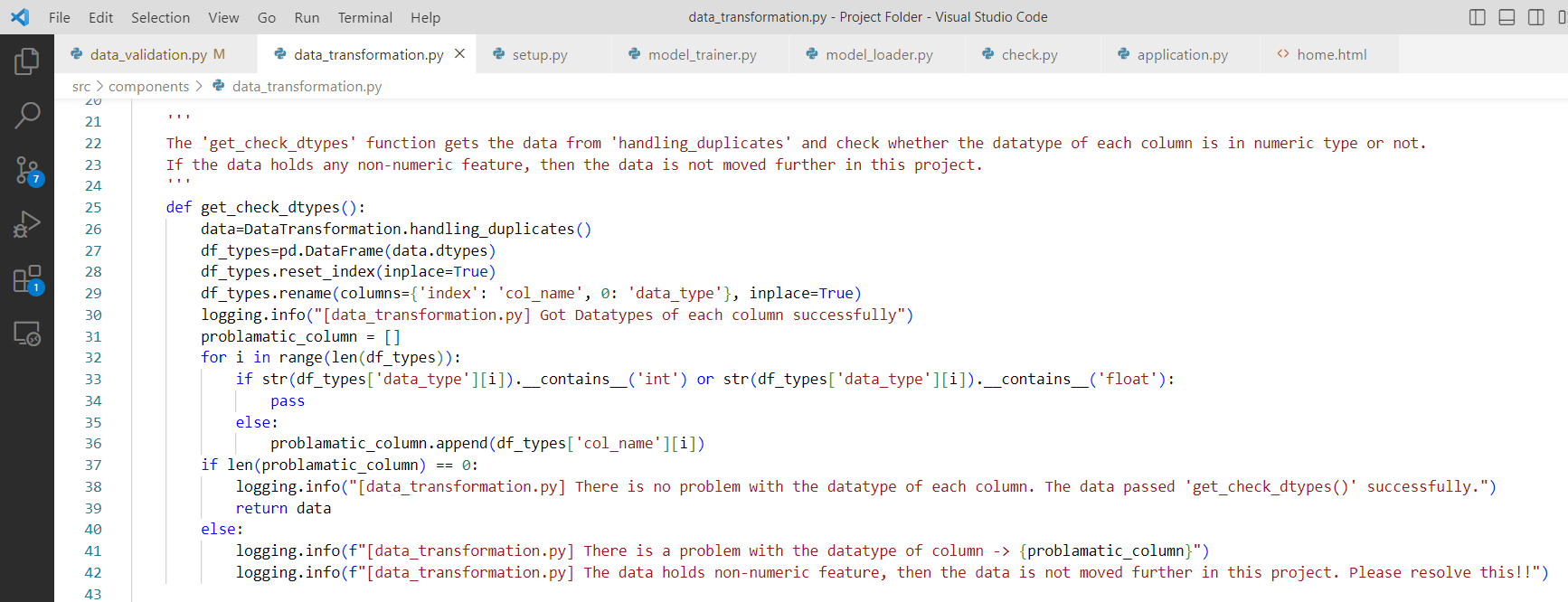
**3.3.1 Handling Duplicates**

****

The class ‘DataTransformation’ from data\_transformation.py file of folder ‘src/components/’ is responsible for all operations related to data transformation like handling duplicates, checking data type of the features and dimensionality reduction. The necessary packages like sklearn, pickle, uitls, etc., were imported.

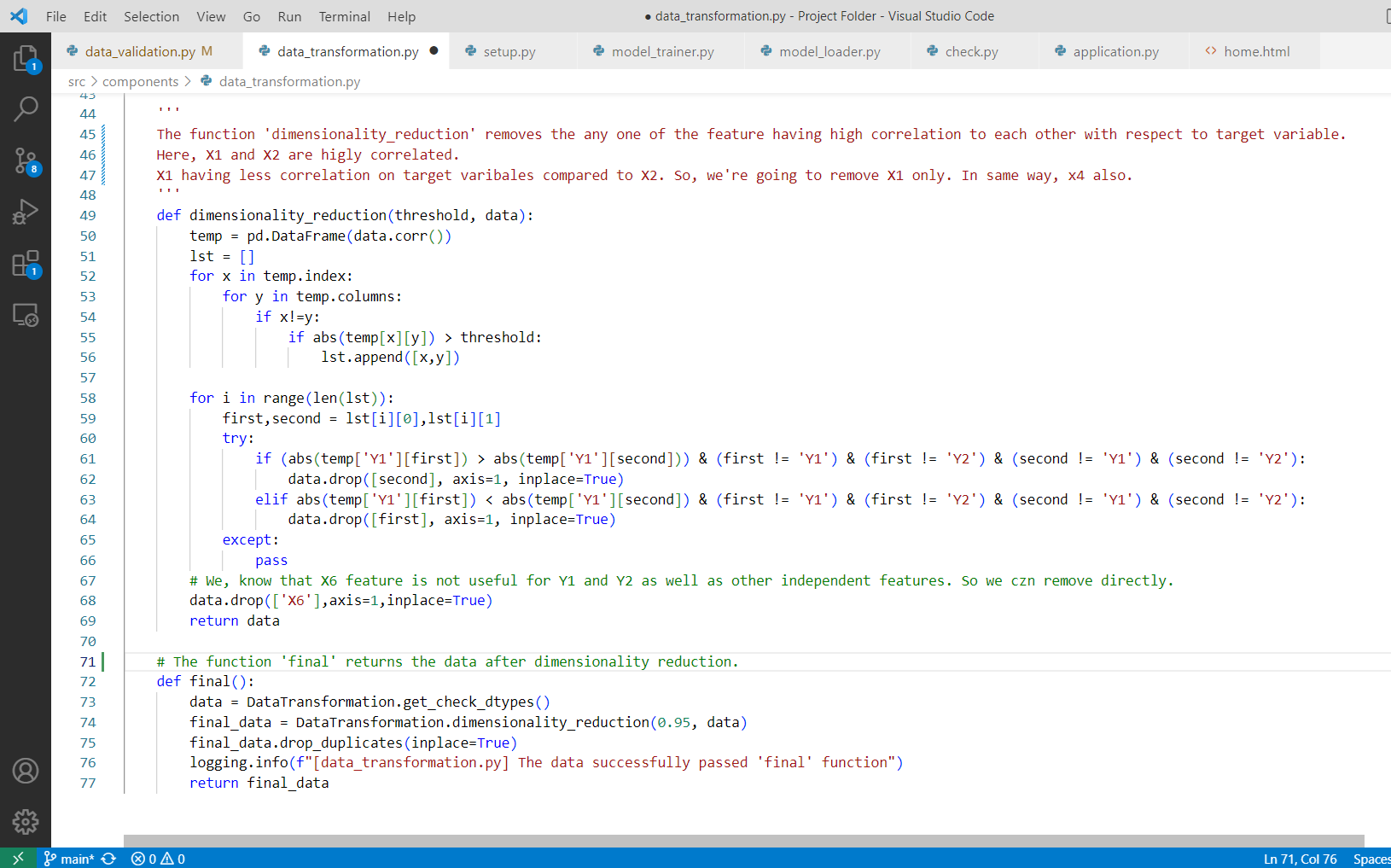
The dataset returned from ‘validate’ function of ‘data\_validation.py’ is called here and assigned to the variable ‘data’. The ‘if block’ of function ‘handling\_duplicates’ checks for the duplicate counts in the dataset and removes if the count is greater than zero with the help of ‘drop\_duplicates’ method in pandas DataFrame. The message is logged after removal of duplicates from the dataset. If no duplicates found ‘else’ part is triggered and the message is logged successfully. Then, returns the data.

**3.3.2 Feature Validation**



The ‘get\_check\_dtypes’ function checks the datatype of all features in dataset. The data returned from the function ‘handling\_duplicates’ of ‘data\_transformation.py’ is called here and assigned to the variable ‘data’. The datatypes of data is recorded in DataFrame and First column is renamed as ‘col\_name’ and Second column as ‘data\_type’. We’ve created a logic to check the datatypes of all features in the data of int/float. If no record found with other datatypes, the data is returned and waits for function call. Then, the message of operation completion is logged using ‘logging’ function.

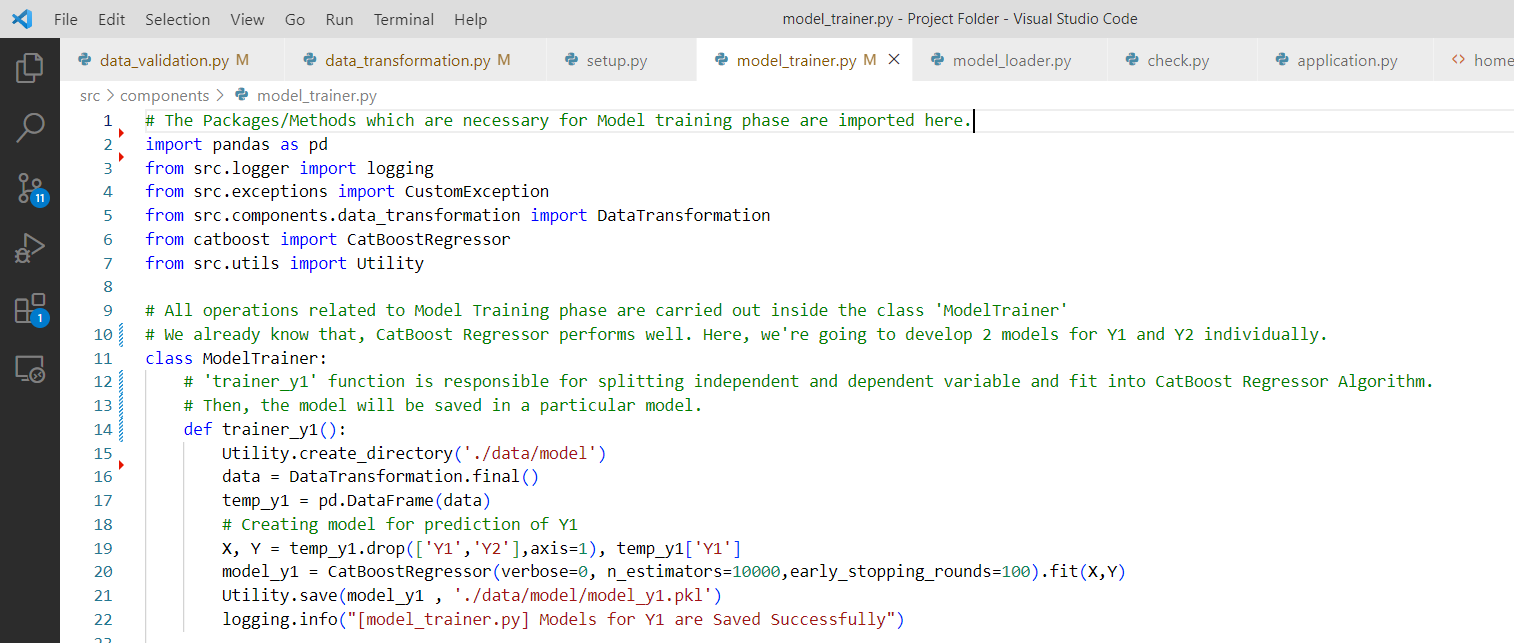
**3.3.3 Dimensionality Reduction**

****

The ‘dimensionality\_reduction’ function removes any one of the feature having high correlation with respect to other feature. The data returned from ‘get\_check\_dtypes’ function is called and assigned to the variable ‘data’. We have to set a threshold manually with the float value nearer to 1.0(maximum value). Here, we set a threshold of 0.95 manually. So that, any one of the feature having high correlation with other feature is removed. The data is saved in desired folder as ‘cleaned\_data’ because the data preprocessing phase ends here. Then, operation completion message will be logged. Finally, the function ‘final’ return the data.

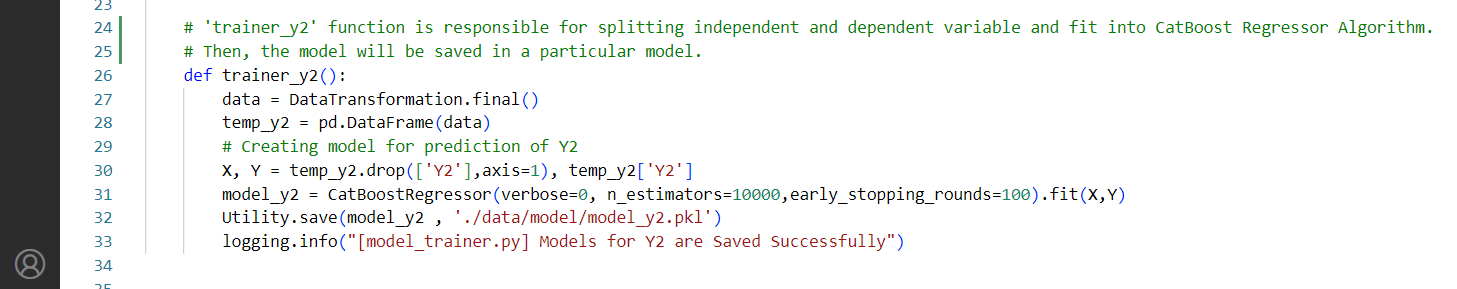
**3.4 Model Trainer**

**3.4.1 Trainer\_y1**

****

The function ‘trainer\_y1’ is responsible for developing a model which predicts Heating Load(Y1) of a building. First, it gets data from ‘final’ funcion of ‘data\_transformation.py’. Independent and dependent variable were fit into CatBoost Regressor and the model is saved in desired location(.data/model) named as model\_y1.pkl. The successful model creation message is logged.

**3.4.2 Trainer\_y2**

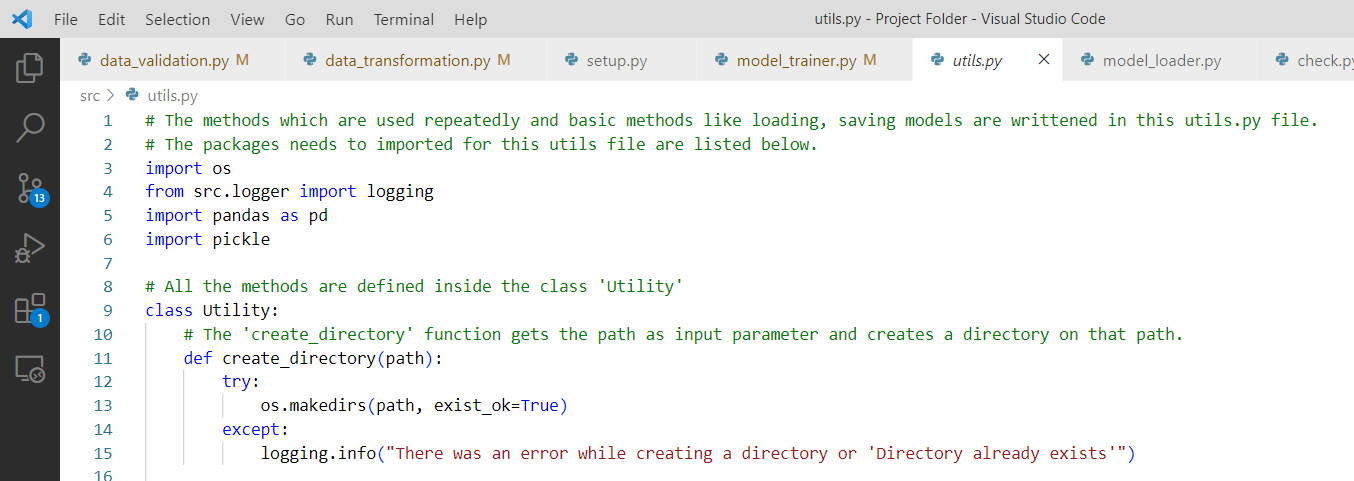
****

The function ‘trainer\_y2’ is responsible for developing a model which predicts Cooling Load(Y2) of a Building. First, it gets data from ‘final’ funcion of ‘data\_transformation.py’. Independent and dependent variable were fit into CatBoost Regressor and the model is saved in desired location(.data/model) named as model\_y2.pkl. The successful model creation message is logged.

**3.5 Utility**

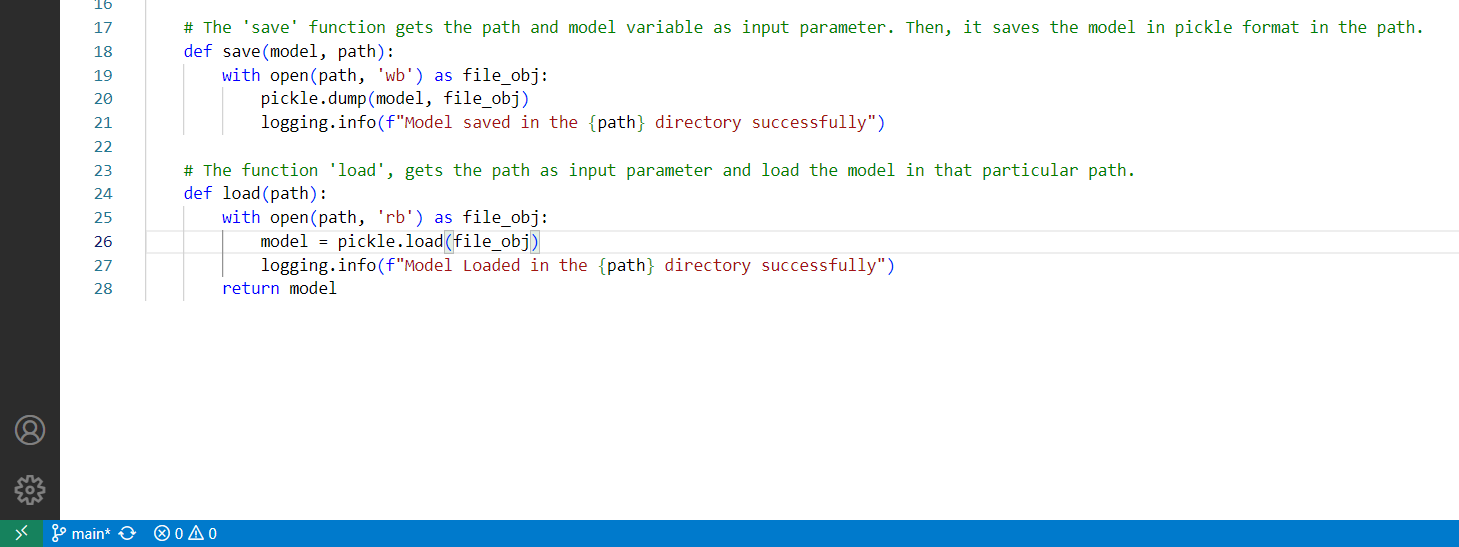
The module ‘utils’ contains a class ‘Utility’. The functions defined in the class ‘Utility’ are reusable functions from different modules which includes ‘create\_directory’, ‘load’ and ‘save’.

**3.5.1 Create Directory**

****

The ‘create\_directory’ function needs one input parameter i.e., the path or location where we need to make a directory. In try block, ‘makedirs’ method of ‘os’ module is used to create a directory on desired path/location. Else, the message of error occurance is logged.

**3.5.2 Save and Load**

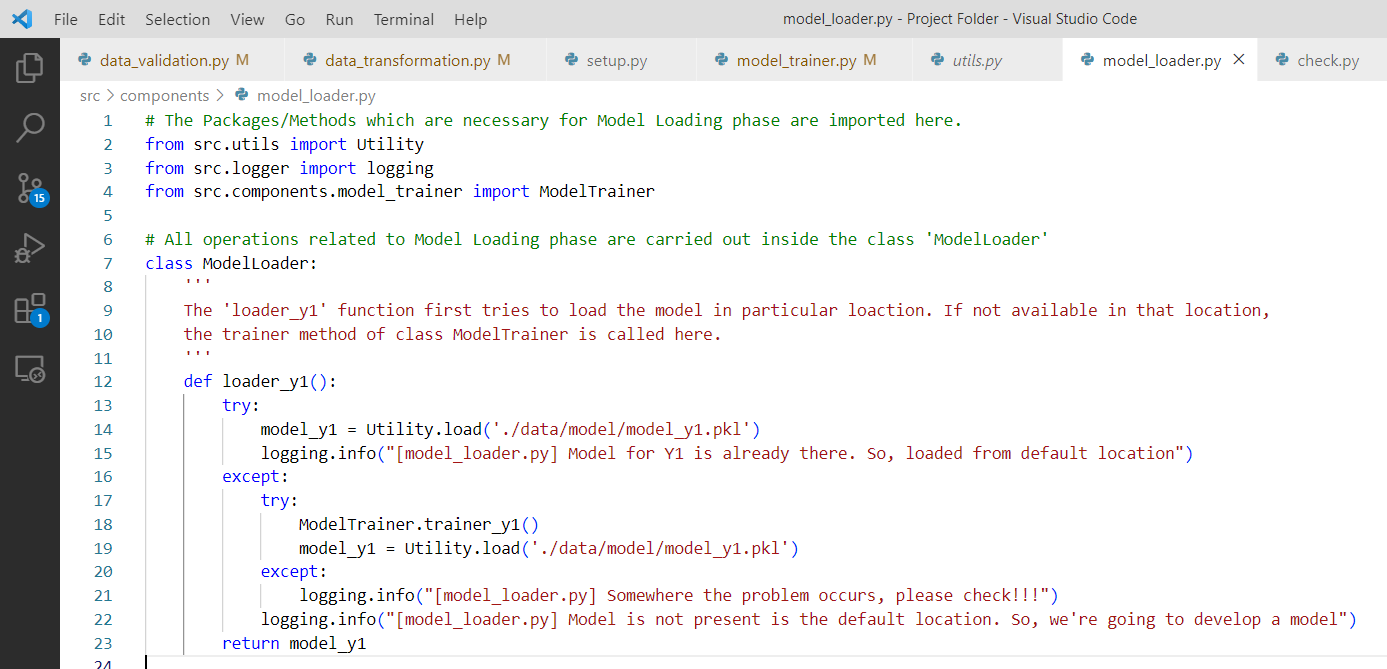
****

The ‘save’ function takes two input parameters of trained model and path. By using ‘pickle’ module the model is dumped in a ‘write-binary’ mode of desired location/path.

The ‘load’ function takes one input as parameter of desired location or path. It brings back the dumped pickle file in a ‘read-binary’ mode of desired location.

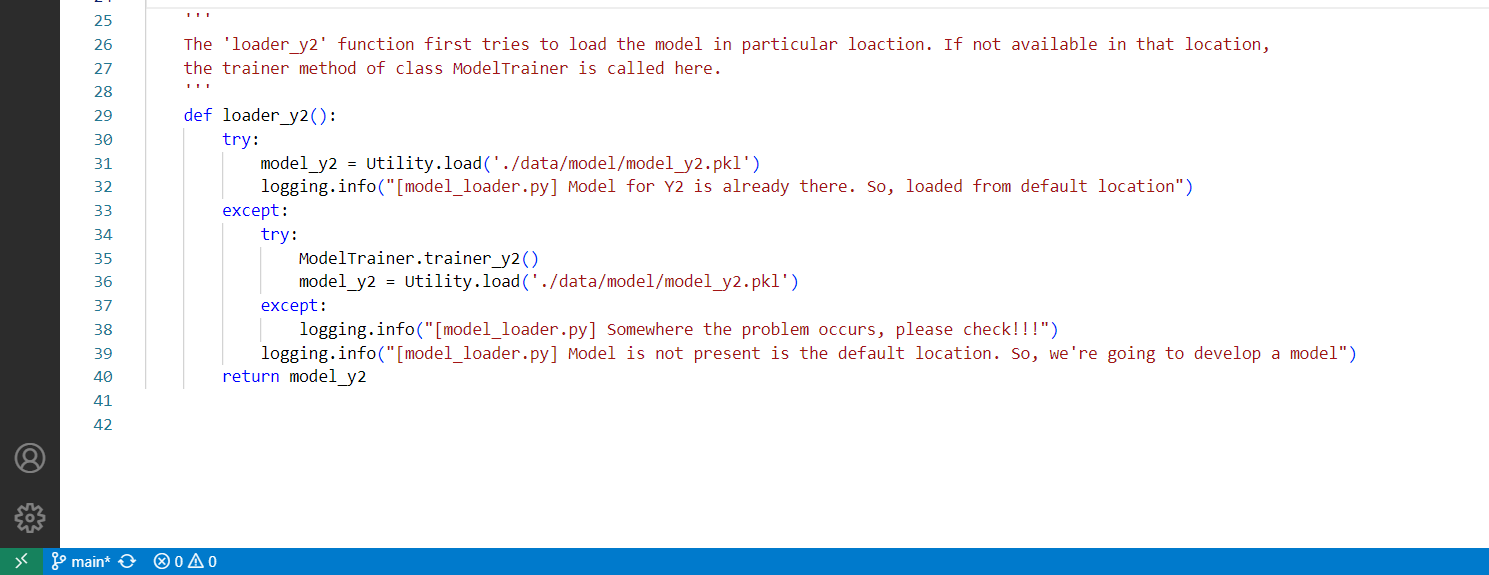
**3.6 Model Loader**

**3.6.1 ModelLoader\_y1**

****

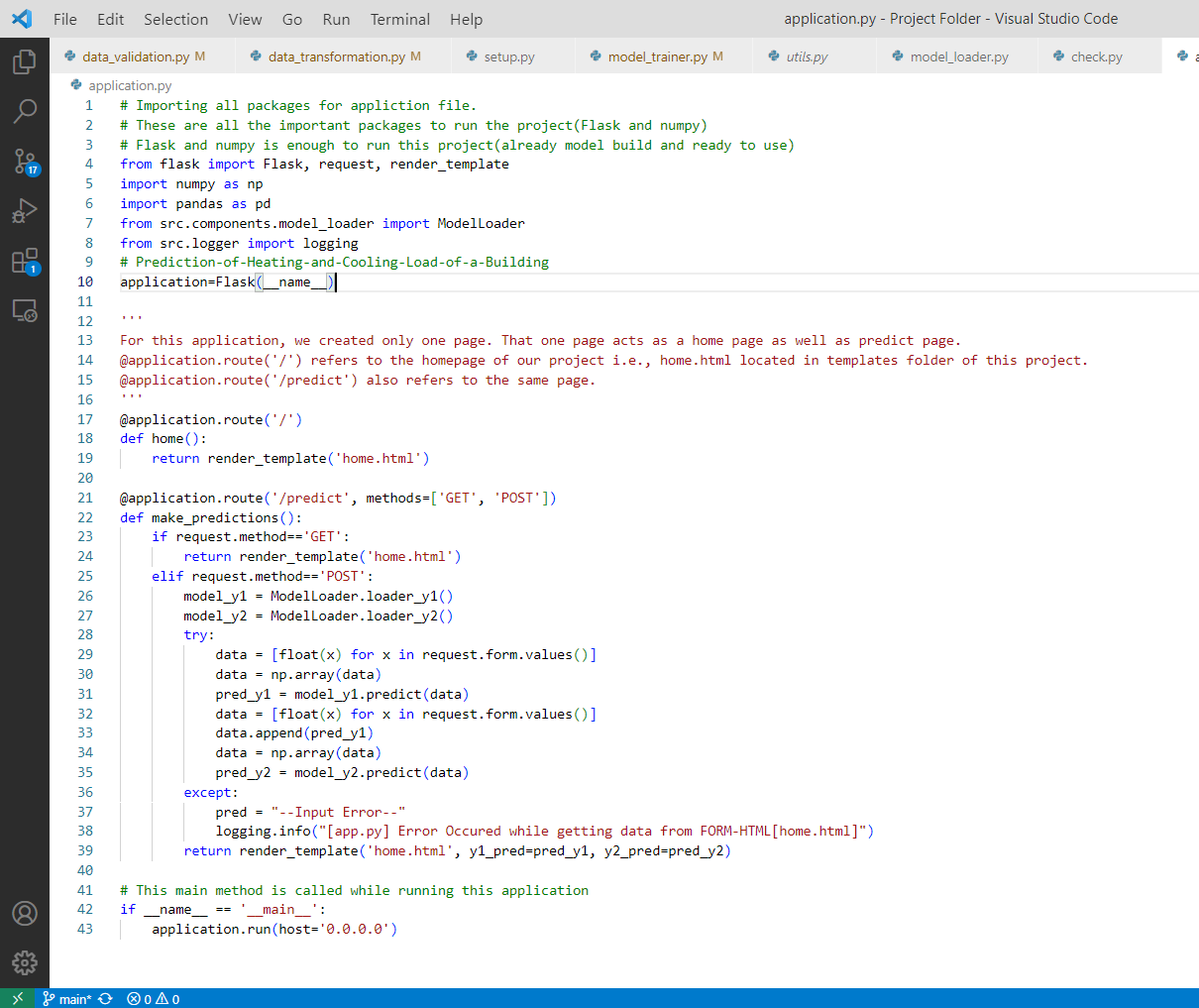
The necessary packages for ‘ModelLoader’ class will be imported. The ‘loader\_y1’ function get ‘model\_y1’ from the desired location. If there in no ‘model\_y1’ in the desired folder, the ‘trainer\_y1’ function in ‘ModelTrainer’ class is called to generate the desired model from the dataset. Then, loader is called again and the ‘model\_y1.pkl’ is assigned to the variable ‘model\_y1’ and returned.

**3.6.2 ModelLoader\_y2**

****

The necessary packages for ‘ModelLoader’ class will be imported. The ‘loader\_y2’ function get ‘model\_y2’ from the desired location. If there in no ‘model\_y1’ in the desired folder, the ‘trainer\_y2’ function in ‘ModelTrainer’ class is called to generate the desired model from the dataset. Then, loader is called again and the ‘model\_y2.pkl’ is assigned to the variable ‘model\_y2’ and returned.

**3.7 Application**

****

The necessary packages like flask, model\_loader, etc., were imported. Then, inherting all attributes and methods in the class Flask, is assigned to an object ‘application’. The front-end design is developed and stored in a ‘templates’ folder. The home page of the project and prediction page using same template(home.html). After getting all inputs from the textbox and button click action, the prediction is shown is the new page which was rendered in the function ‘make\_prediction’. The ‘model\_loader’ is called and assigned to the variable ‘model’. The inputs from the front-end is received via ‘request’ method. Then, those data were converted into numpy array and passed to the predict method of ‘model’. Then it is returned to the ‘home.html’ page with the predicted result with the help of ‘render\_template’ method.

**4. Test Report**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **Expected Result** | **Actual Result** |
| To check the working condition of URL | URL of the project and web browser | It should work in every platform. | Working perfectly in all opeating system like Android, ios and mac. |
| To check the working condition on all web browser. | URL of the project and multiple web browsers | It should work in all web browser. | Tested in browser like chrome, safari and Edge. |
| To check the behaviour of the application with wrong input | URL of the project and web browser | It should return input error | Returns as Input Error |
| To check the output prediction | URL, web browser and desired inputs | It should return a output prediction in the web page | Returns prediction in the web page |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |