**Low Level Design(LLD)**

Prediction of LC50 value using Quantitative structure–activity

relationship models (QSAR models)

|  |  |
| --- | --- |
| Done by | Hariprasath G |
| Document Version | 0.2 |
| Last Revised Date | 07-07-2023 |

**Document Control**

**Change Record:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 0.1 | 01-07-2023 | Hariprasath G | Introduction and System Architecture |
| 0.2 | 05-07-2023 | Hariprasath G | Architecture Description and Unit Test Cases |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Reviews:**

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

**Approval Status:**

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

|  |  |
| --- | --- |
| **Contents** |  |
| 1. Introduction…………………………………………………... | 4 |
| 1.1 What is Low-Level Design Document?........................... | 4 |
| 1.2 Scope…………………………………………………... | 4 |
| 1. System Architecture………………………………………... | 5 |
| 1. Architecture Description…………………………………… | 5 |
| 3.1 Data Description……………………………………….. | 6 |
| 3.2 Data Validation………………………………………… | 6 |
| 3.3 Data Transformation…………………………………… |  |
| 3.3.1 Handling Duplicates……………………………... |  |
| 3.3.2 Handling Duplicates Level2……………………... |  |
| 3.3.3 Feature Validation……………………………….. |  |
| 3.3.4 |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

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

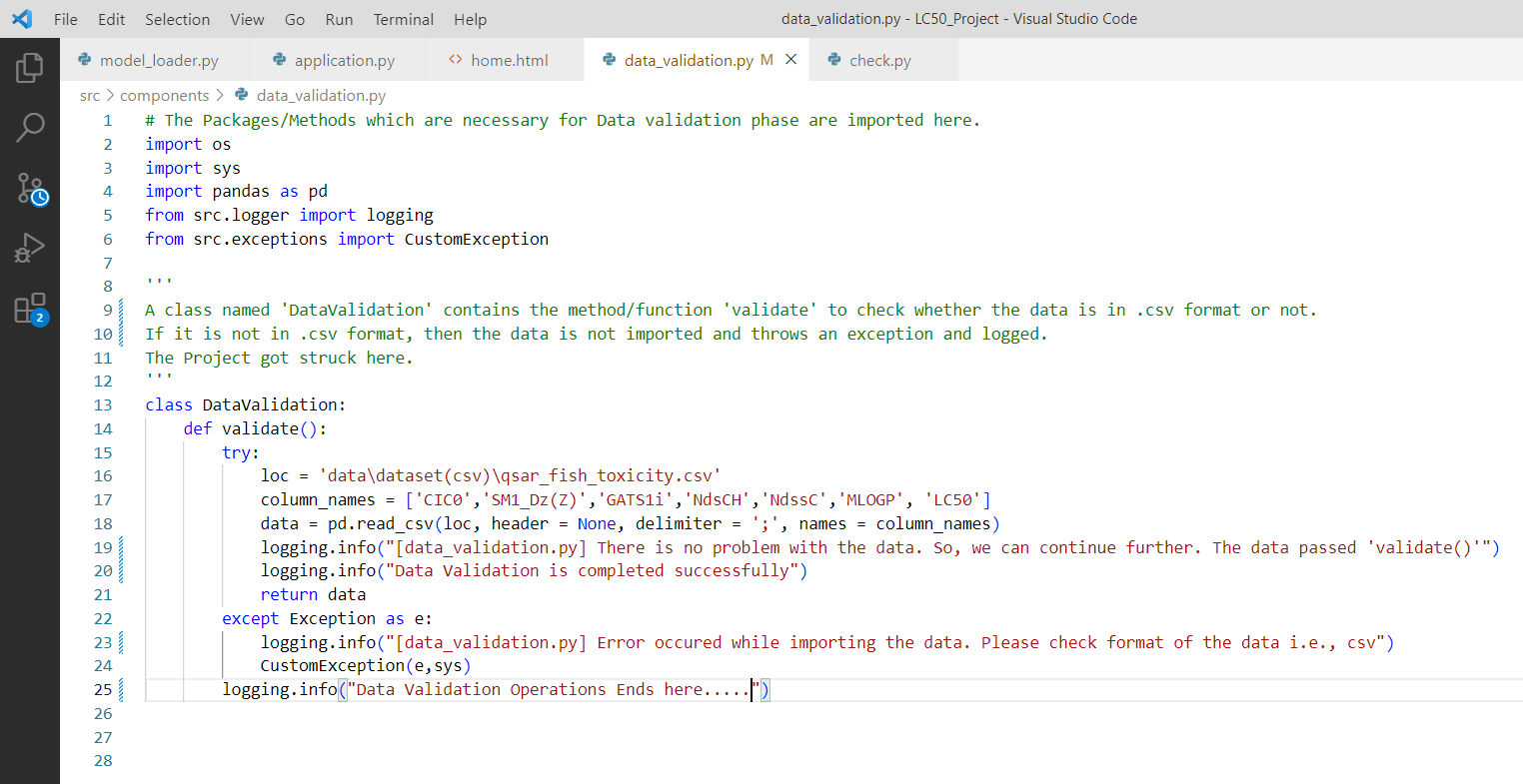
Data Validation

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

Data set containing values for 6 attributes (molecular descriptors) of 908 chemicals used to predict quantitative acute aquatic toxicity towards the fish Pimephales promelas (fathead minnow).

This dataset was used to develop quantitative regression QSAR models to predict acute aquatic toxicity towards the fish Pimephales promelas (fathead minnow) on a set of 908 chemicals. LC50 data, which is the concentration that causes death in 50% of test fish over a test duration of 96 hours, was used as model response. The model comprised 6 molecular descriptors: MLOGP (molecular properties), CIC0 (information indices), GATS1i (2D autocorrelations), NdssC (atom-type counts), NdsCH ((atom-type counts), SM1\_Dz(Z) (2D matrix-based descriptors). Details can be found in the quoted reference: M. Cassotti, D. Ballabio, R. Todeschini, V. Consonni. A similarity-based QSAR model for predicting acute toxicity towards the fathead minnow (Pimephales promelas), SAR and QSAR in Environmental Research (2015), 26, 217-243; doi: 10.1080/1062936X.2015.1018938

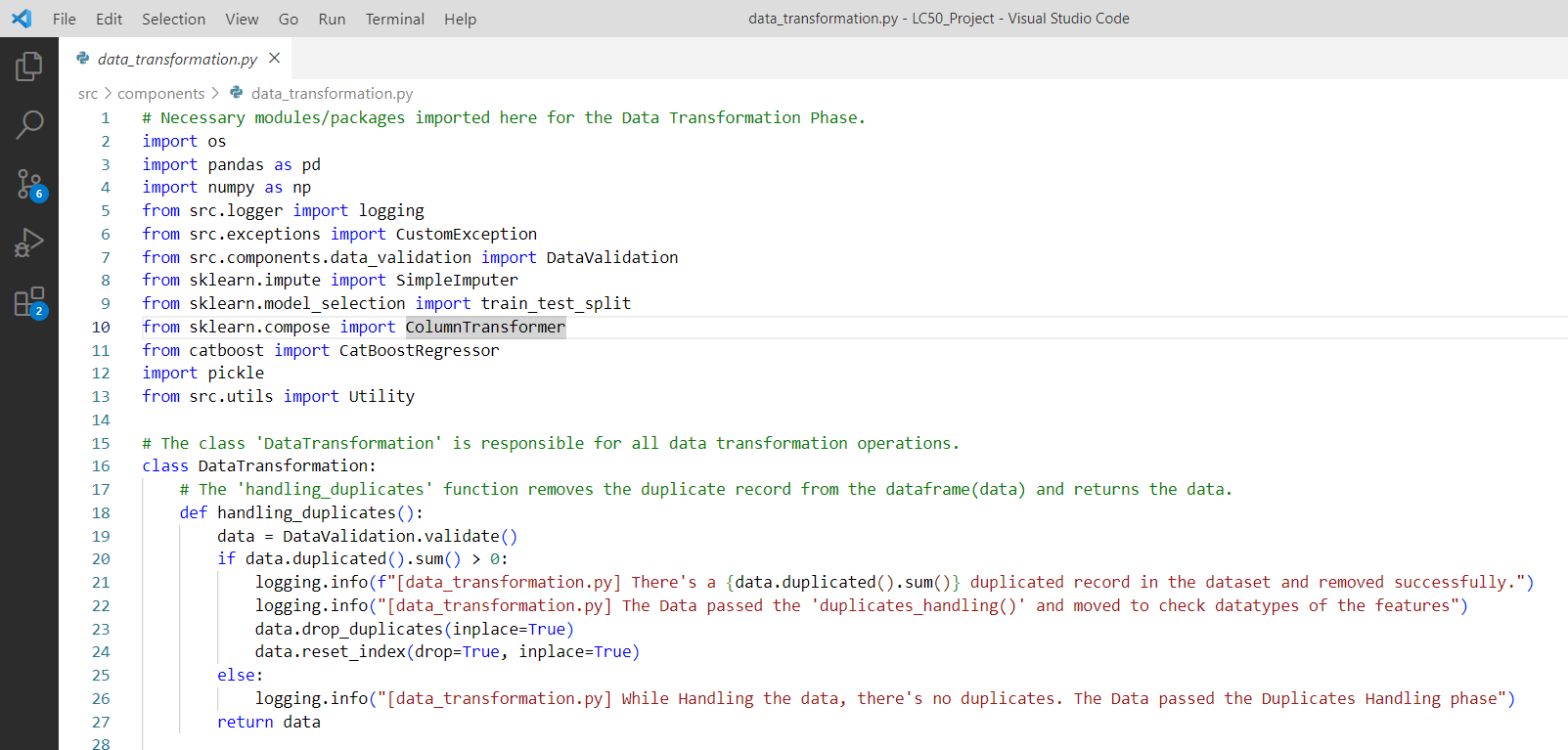
* 1. **Data Validation**

****

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\_csv’ 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**

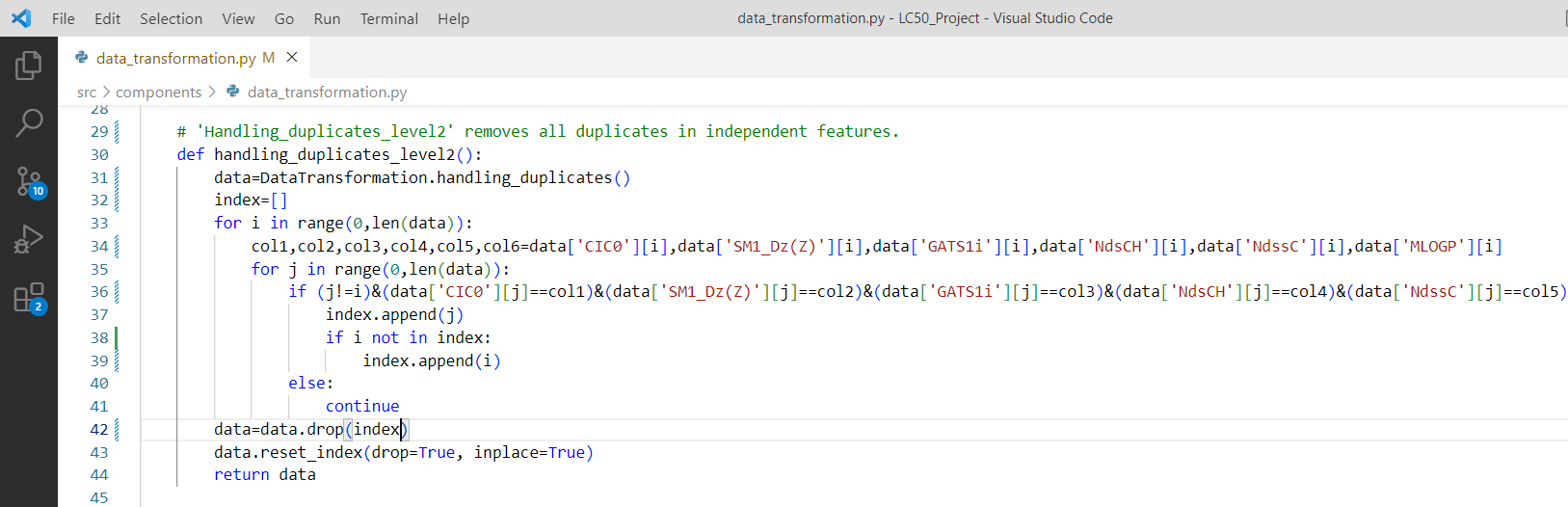
**3.3.1 Handling Duplicates**

****

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, handling missing values, handling outliers, dimensionality reduction and splitting dataset. The necessary packages like sklearn, pickle, uitls, etc., were imported.

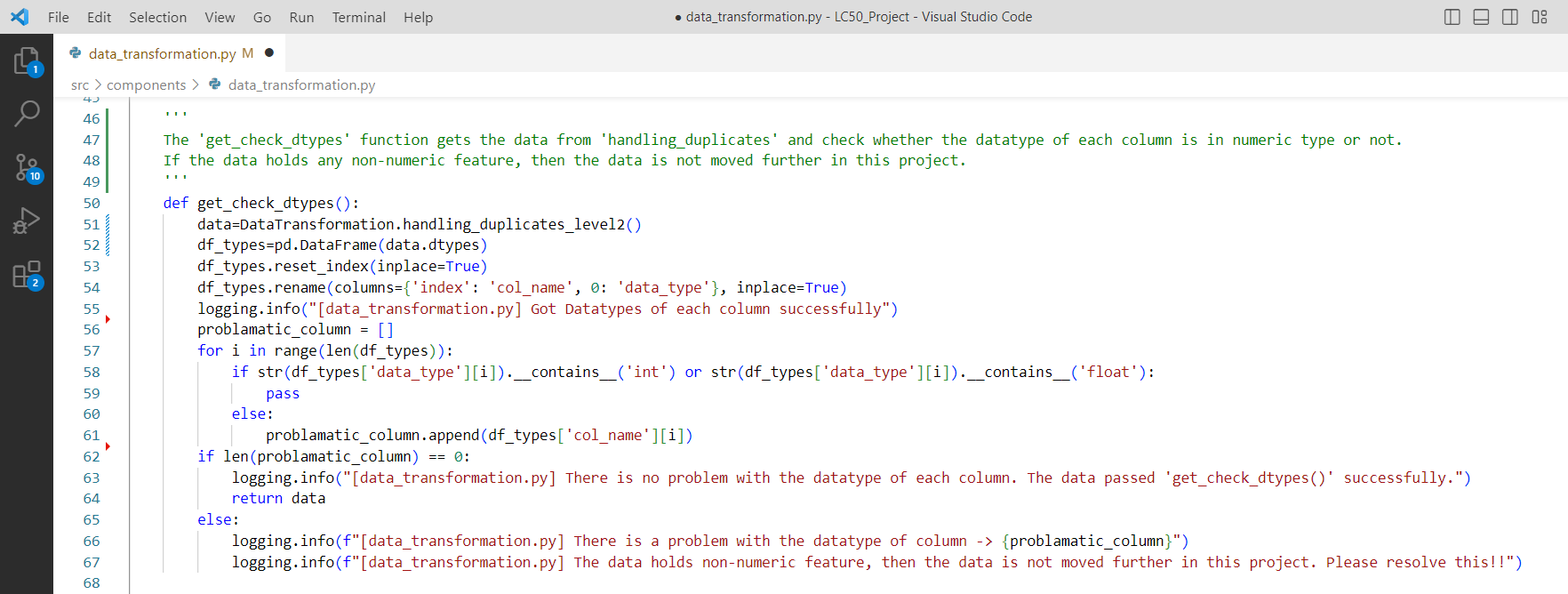
The dataset returned from ‘validate’ function of ‘data\_validation.py’ is called here and assigned to the variable ‘data’. The ‘if block’ of function ‘handling\_duplicates’ checks for the duplicate counts in the dataset and removes if the count is greater than zero with the help of ‘drop\_duplicates’ 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 Handling Duplicates Level2**

****

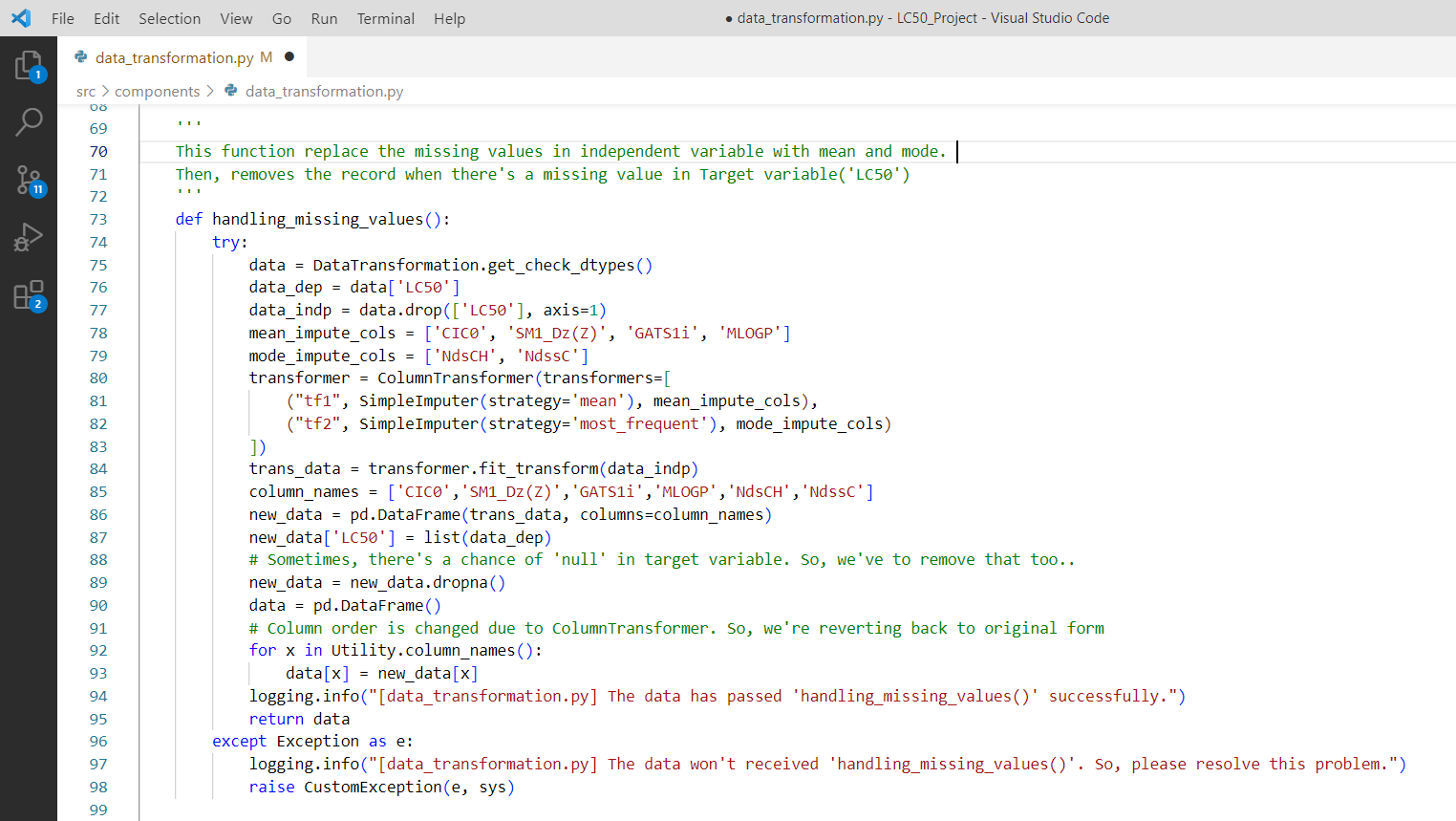
The ‘handling\_duplicates\_level2’ function removes all records of duplicates in independent feature. The data returned from the function ‘handling\_duplicates’ of ‘data\_transformation.py’ is called here and assigned to the variable ‘data’. Here, we’ve created an algorithm to remove all duplicated independent records as well as original record because we’re not sure which record is original. ‘for’ loop run for number of times which is equal to the length of data. Every single record is compared with all other record without considering target variable and index value of current record as well as the index of found record were updated in the list(index). The index values in the dataset is reseted using ‘reset\_index’ method of pandas. Finally, return the data.

**3.3.3 Feature Validation**



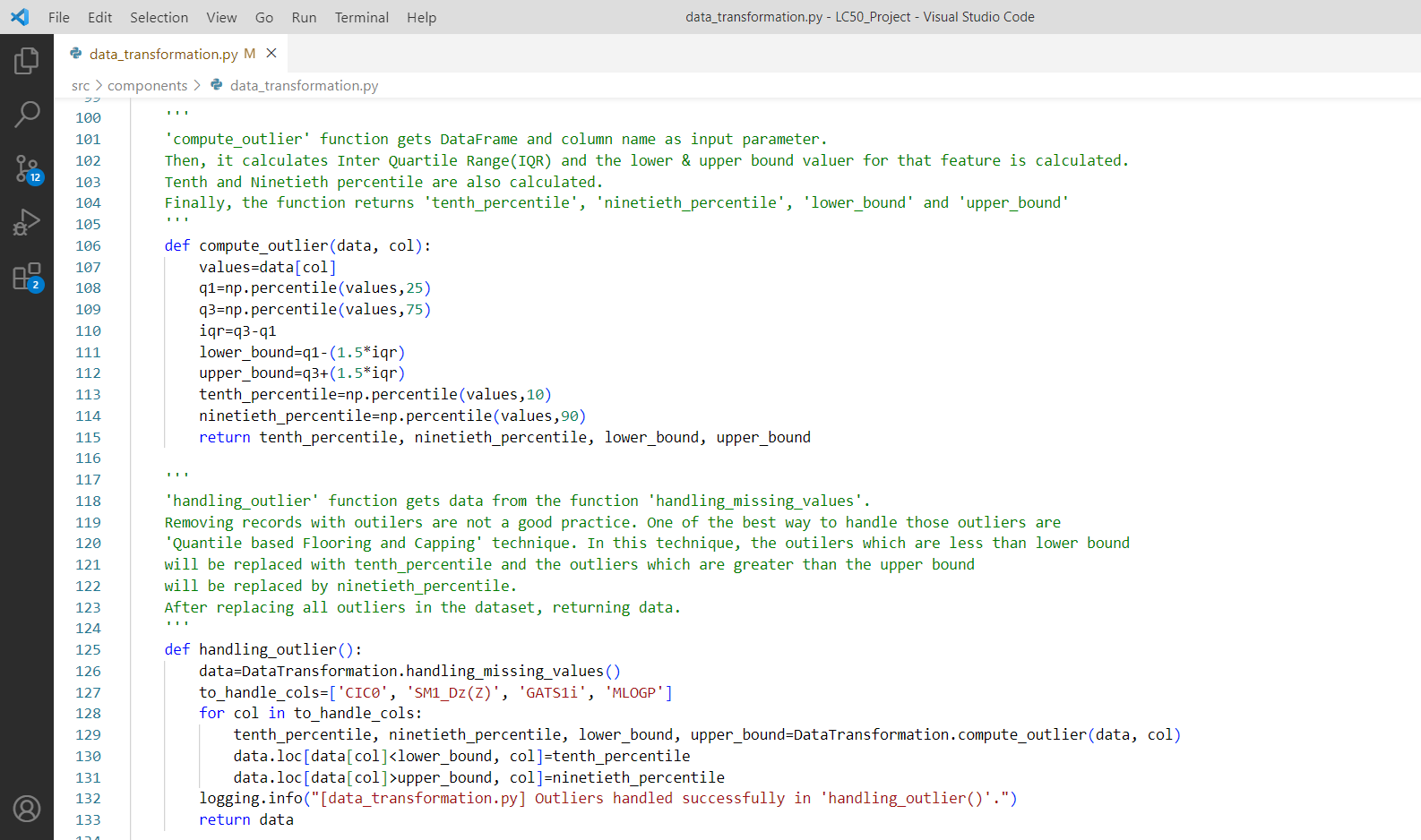
The ‘get\_check\_dtypes’ function checks the datatype of all features in dataset. The data returned from the function ‘handling\_duplicates\_level2’ of ‘data\_transformation.py’ is called here and assigned to the variable ‘data’. The datatypes of data is recorded in DataFrame and First column is renamed 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.

**3.3.4 Handling Missing Values**

****

The ‘handling\_missing\_values’ function replaces all missing values with the help of ‘ColumnTransformer’ and ‘SimpleImputer’ in sklearn package. In try block, the data returned from the ‘get\_check\_dtypes’ function is called and assigned to the variable ‘data’. There are 4 continuous feature and 2 categorical feature in the independent variable. The missing values of the features with continuous data are replaced with mean and the missing values of features with categorical data are replaced with mode(most\_frequent). Then the dataset is arranged in the original form and returned. If error occurs in the try block, then ‘except’ block is trigger and raises an exception and the messages were logged.

**3.3.5 Handling Outliers**



The ‘handling\_outlier’ function replaces all outliers in the features of continuous data. The data returned from ‘handling\_missing\_values’ function are called and assigned to the variable ‘data’. The continuous features were defined in the variable ‘to\_handle\_cols’. The ‘for’ loop iterates through the variable ‘to\_handle\_cols’. The ‘compute\_outlier’ function takes input parameter as data and column name of ‘to\_handle\_cols’ in for loop. So that the function ‘compute\_outlier’ calculates the 25th percentile, 75th percentile, Inter Quantile Range, lower bound, upper bound, 10th percentile and 90th percentile. Then, returns 10th percentile, 90th percentile, lower bound and upper bound of the iput column. Then the datapoint which is greater than the upper bound value is replaced with 90th percentile and the datapoint which is lesser than the lower bound value is replaced with 10th percentile. Then, operation completion message is logged. Finally, return the data.