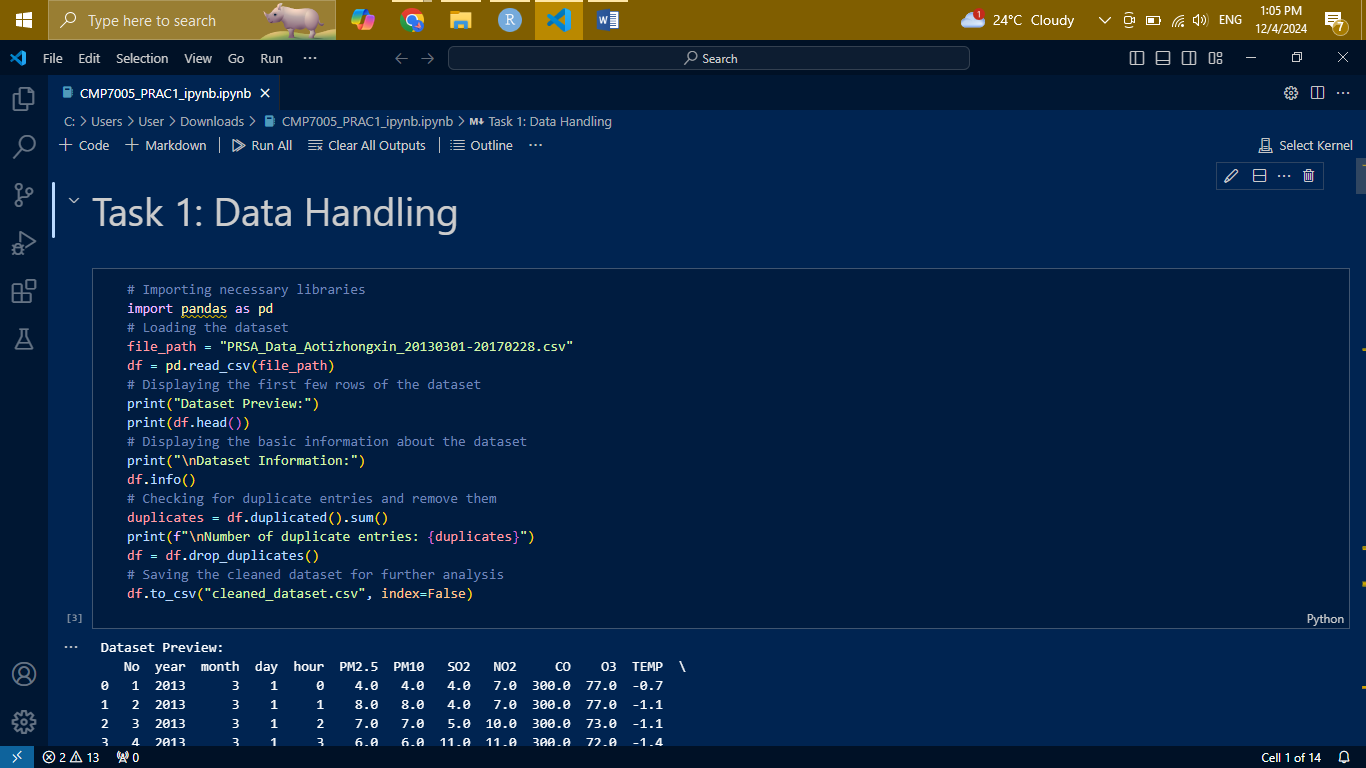
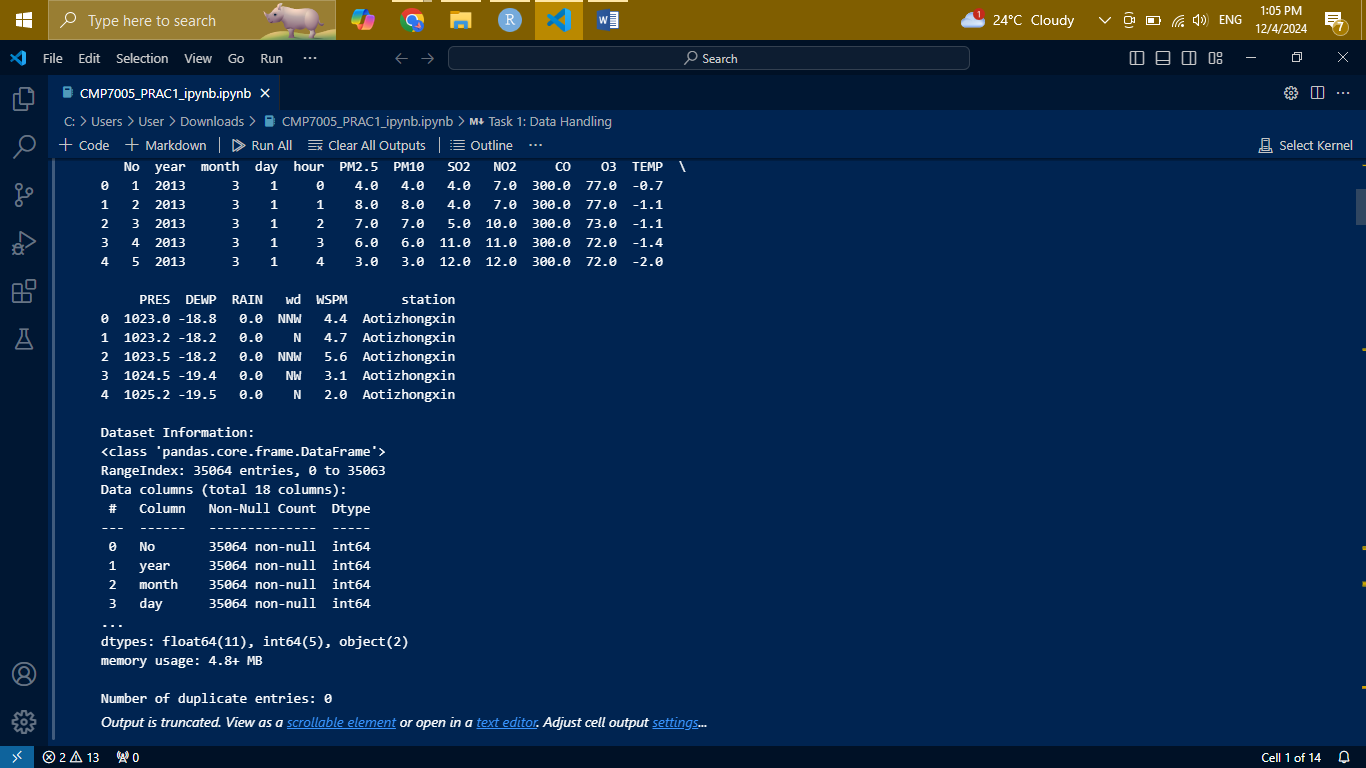
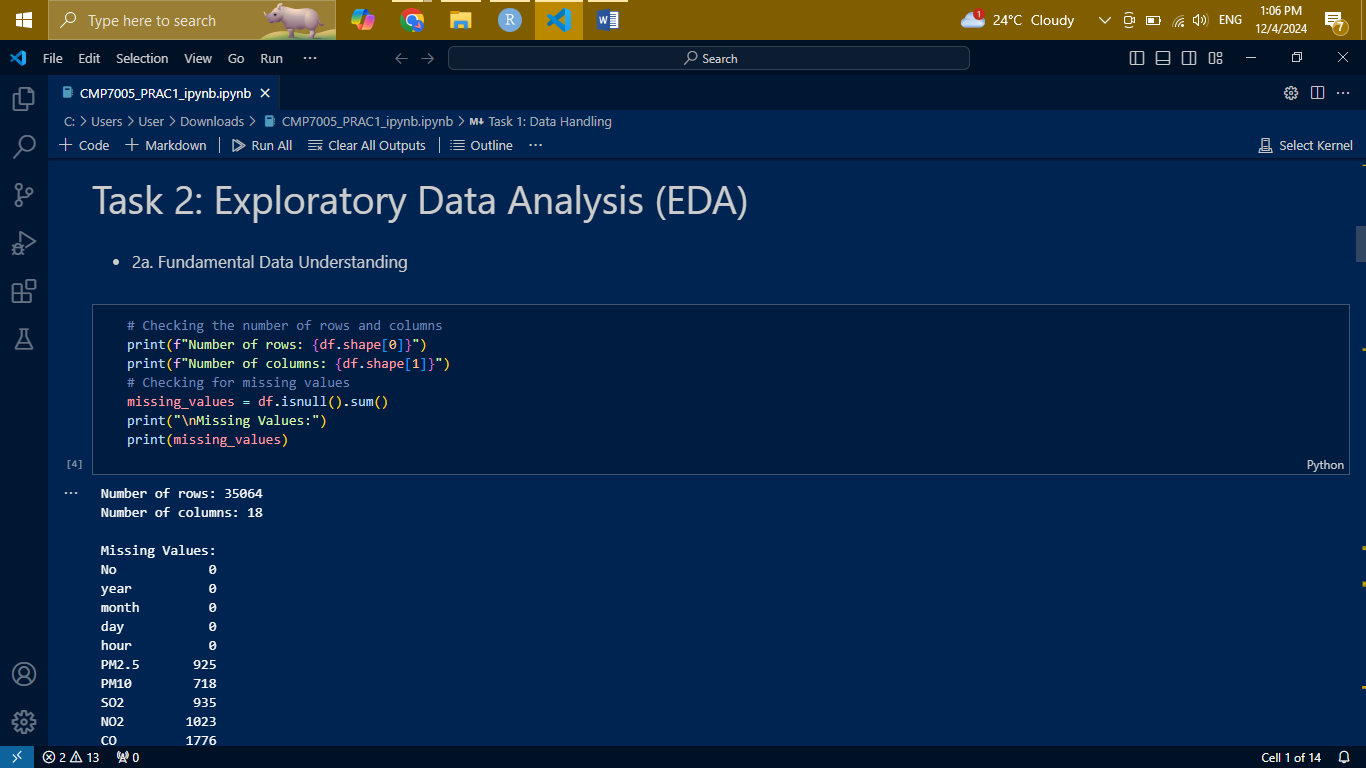
**Assignment**

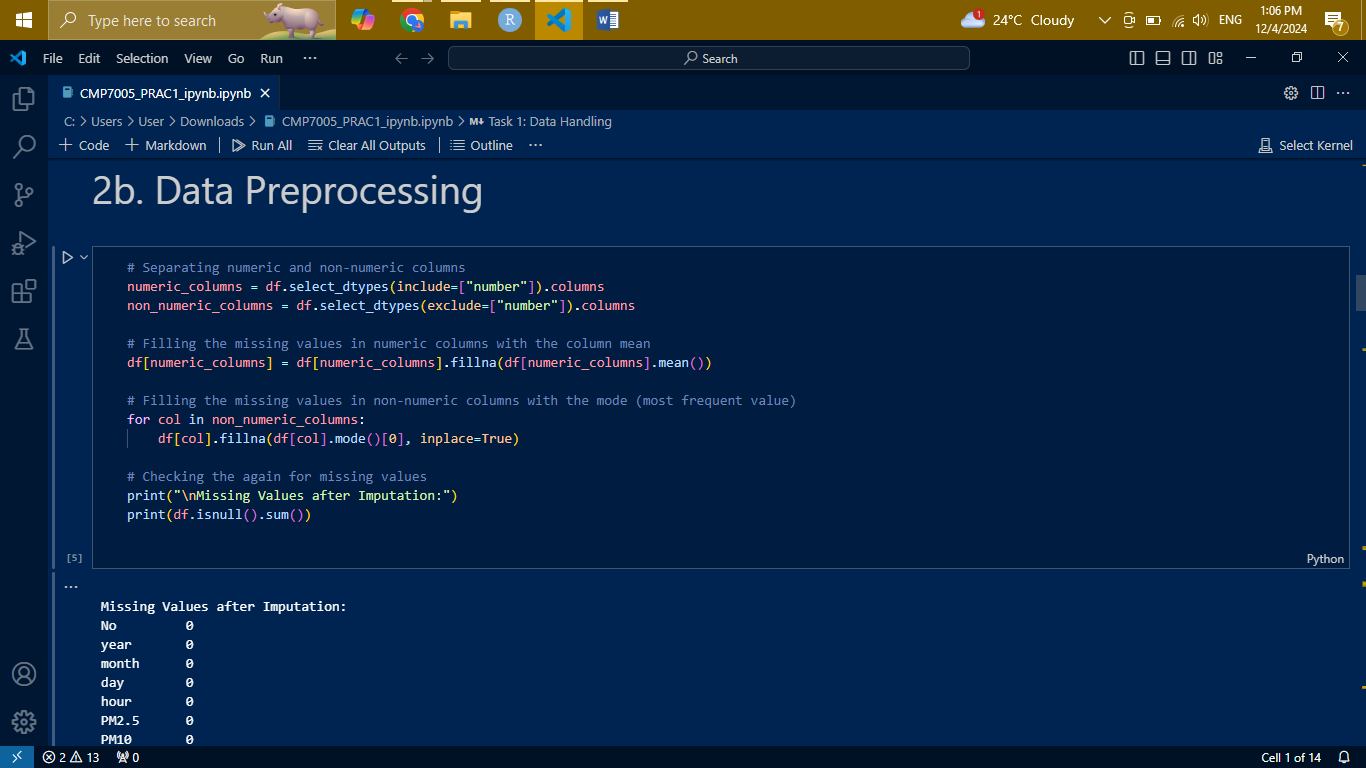
**1. Identifying Missing Values:**  
The dataset likely contains missing values, which are a common occurrence in real-world datasets. Handling these missing values is crucial to avoid issues during data analysis and machine learning model building. Using the isnull() function, we initially identify the columns with missing data and count the number of missing values in each column. This step ensures that we understand the extent and nature of the problem before applying corrective measures.



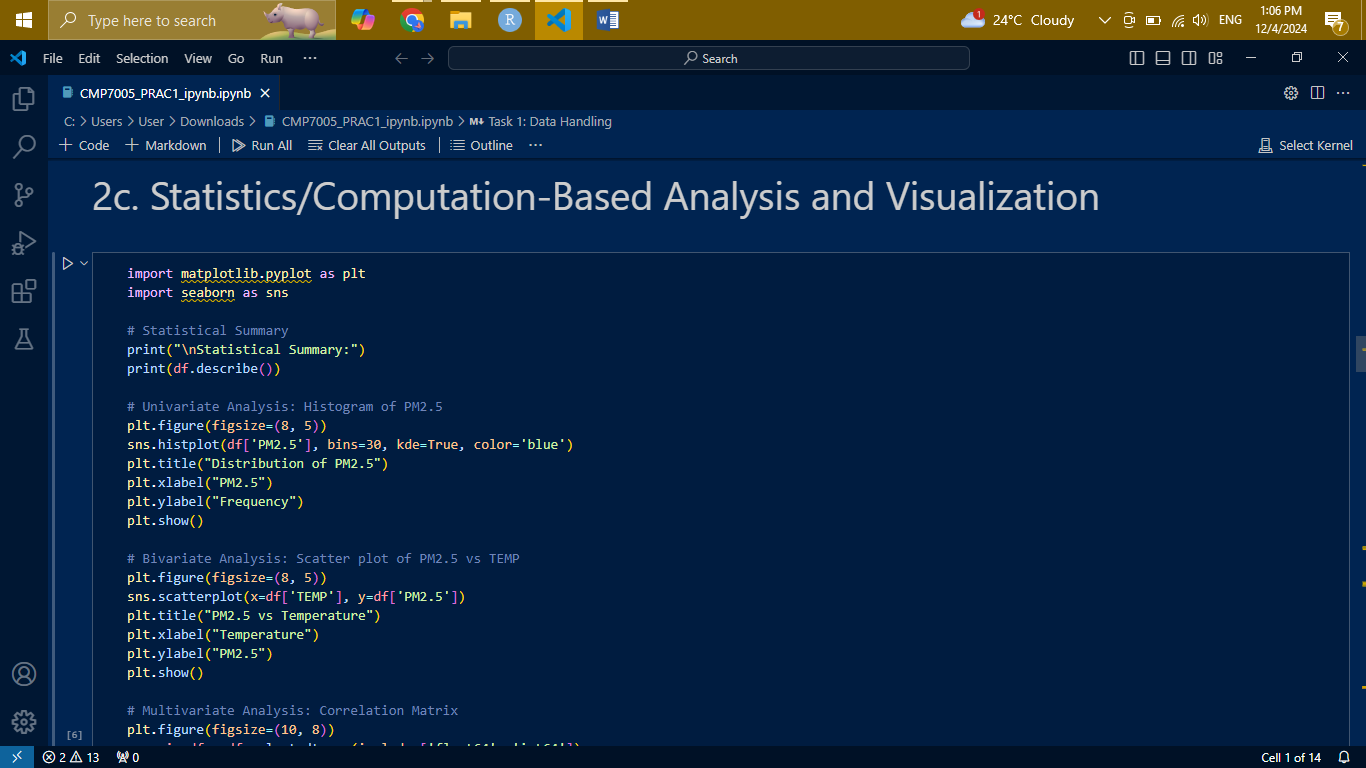


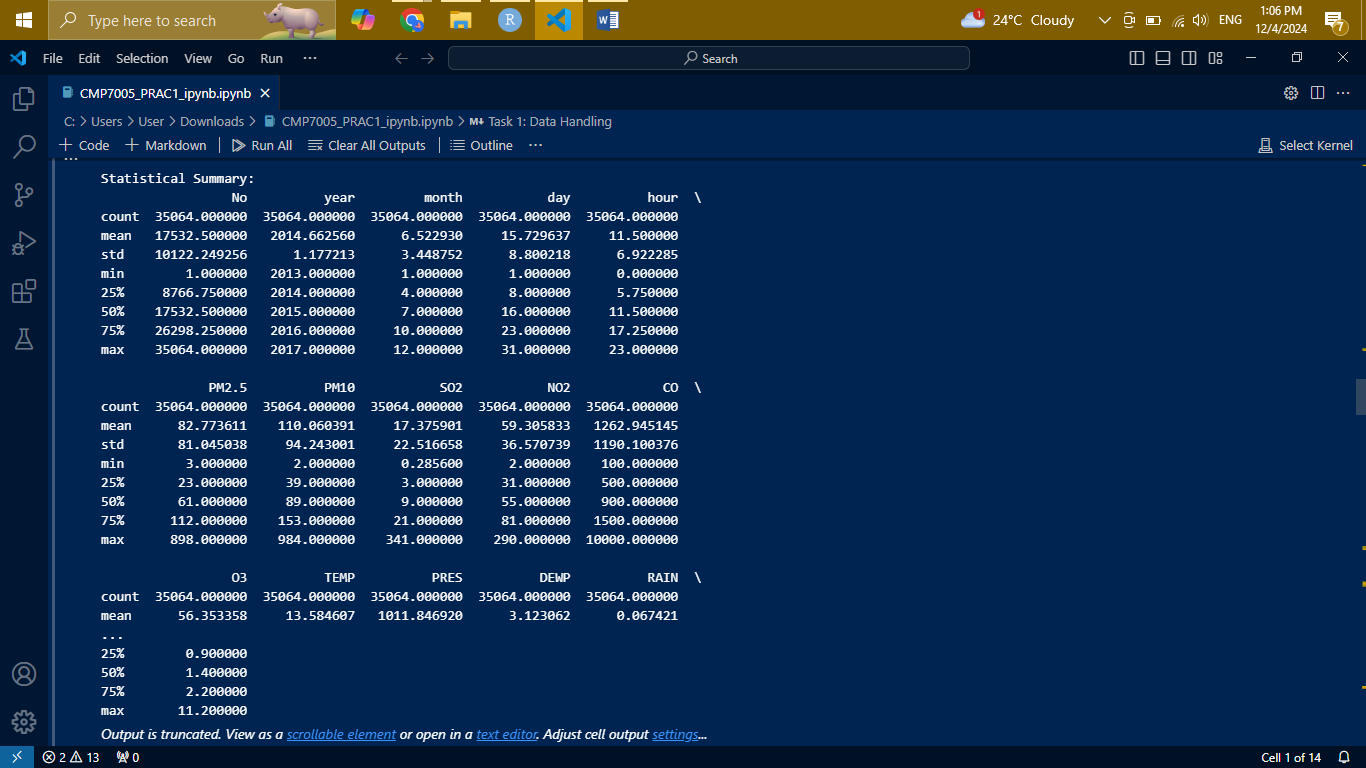
**2. Separating Numeric and Non-Numeric Columns:**  
Datasets often have a mix of numeric (e.g., integer, float) and non-numeric (e.g., strings, categorical) data. Since missing values in numeric columns can be addressed using statistical measures like mean or median, and non-numeric columns require methods like filling with the mode or a placeholder, we first separate the columns based on their data types. This separation ensures that the appropriate imputation strategy is applied to each column type.

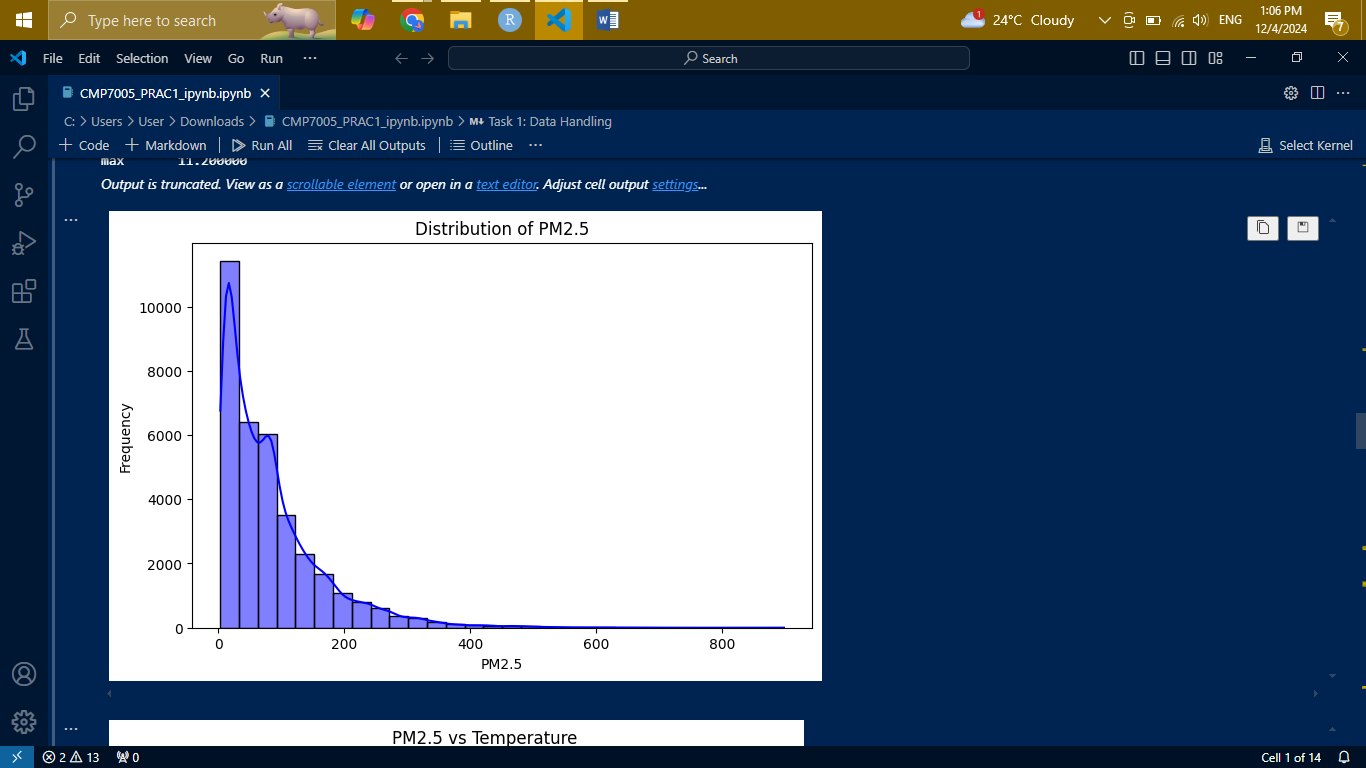


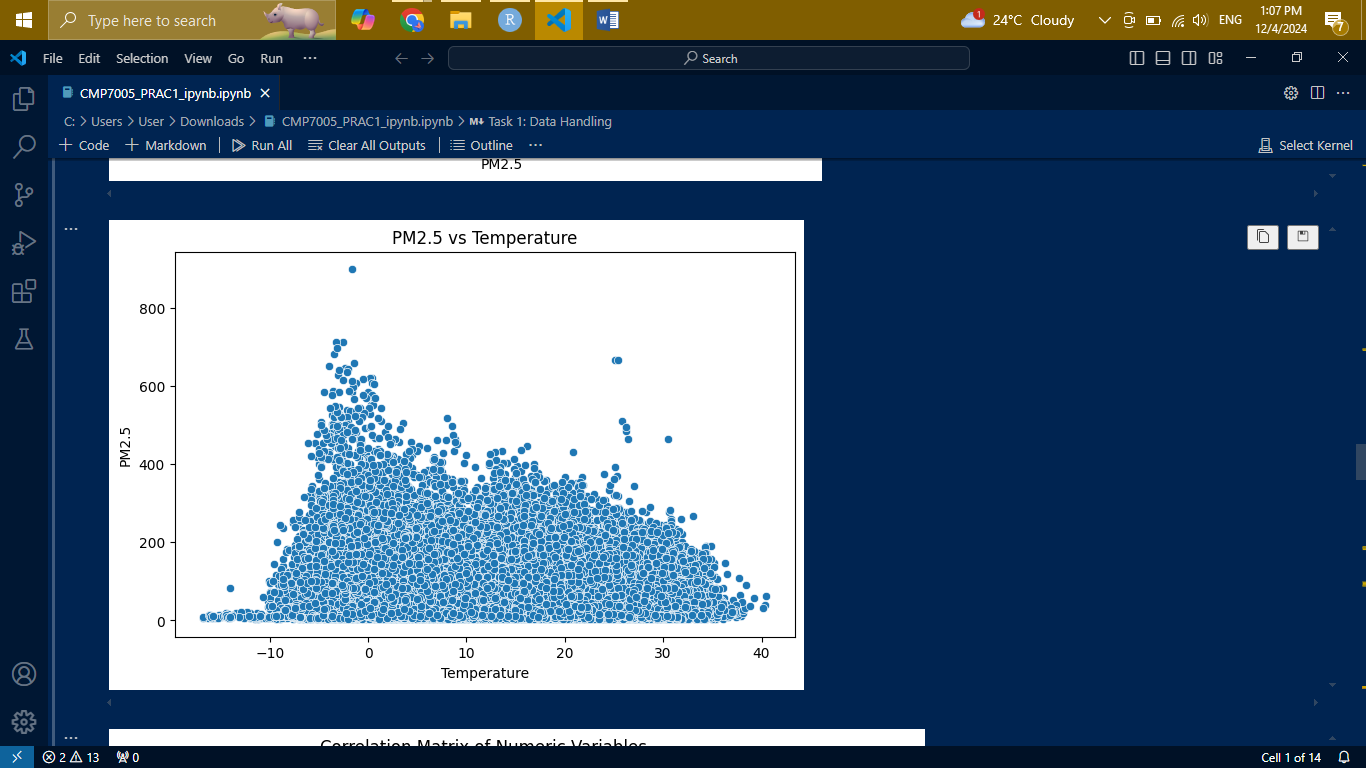


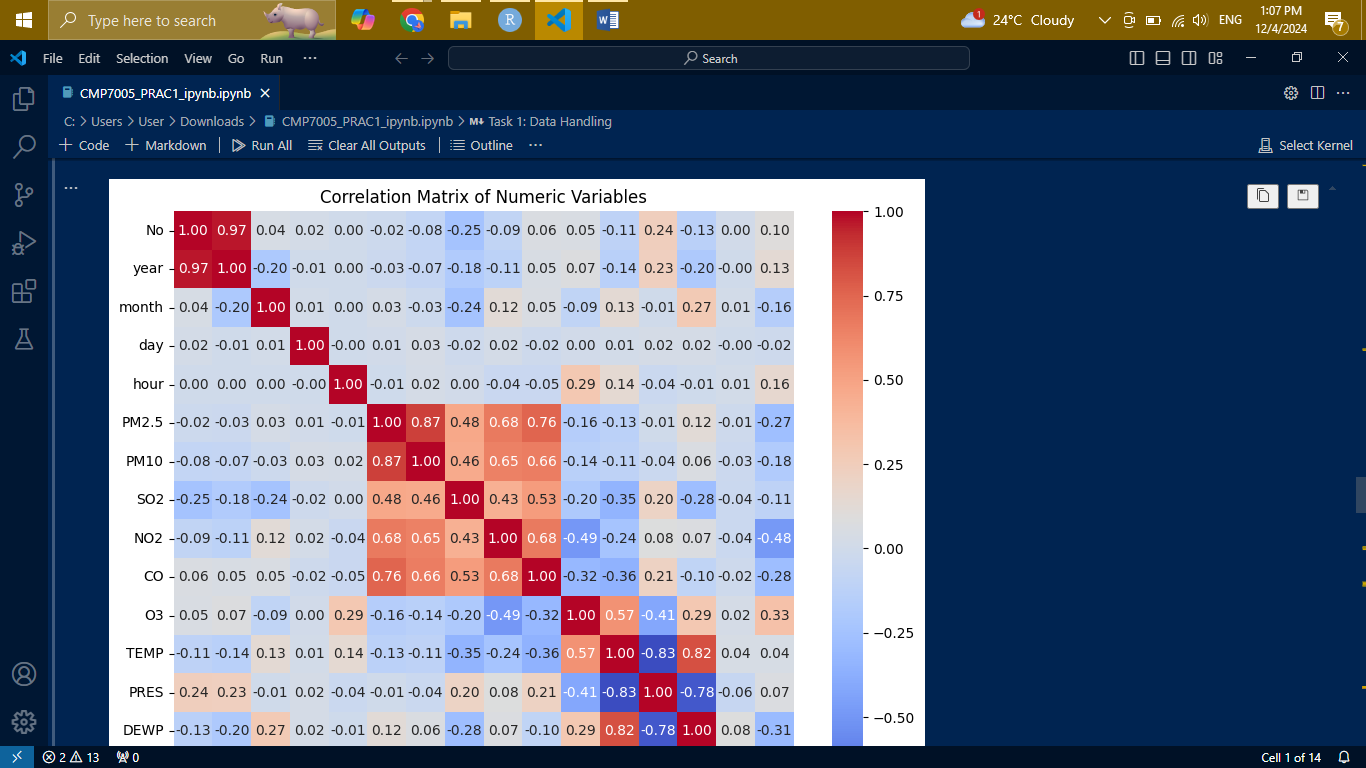
**3. Filling Missing Values in Numeric Columns:**  
For numeric columns, the mean of the column values is a logical choice for imputation when there is no specific domain knowledge suggesting an alternative. This is because the mean minimizes the variance introduced by imputation, preserving the data's statistical properties. We use the fillna() method combined with df.mean() to replace missing values in these columns efficiently.











**4. Handling Non-Numeric Columns:**  
Mode is then used in imputation for nominal/ categorical data type, where the most recurrent value is chosen. This strategy is reasonable as it does not affect the degree of bias by preserving the categorical distribution of the obtained data. A loop is used to go through all the non-numeric columns, and for the missing values, the fill () method is used together with mode()[0].

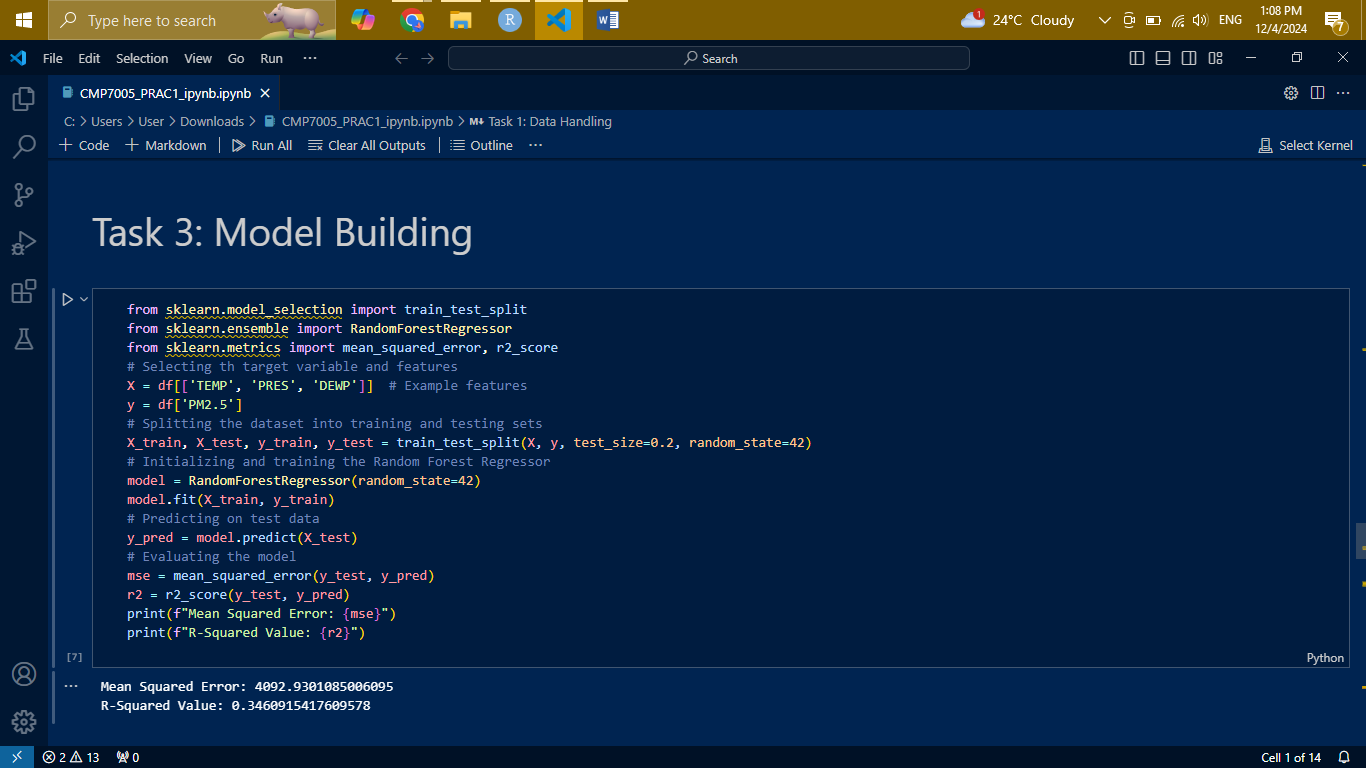
**5. Post-Imputation Validation:**  
Missing values have been imputed, and checking whether there are still absorbing values in the dataset is necessary. Using IsNull ().sum() again, we again see that the entire dataset is clean and ready for analysis or modeling. The following is the step of verification of the type of data that is to be further preprocessed so this a sanity check to the data preprocessing pipeline.

**Preparing the Data for Modeling:**  
Before building models, the data must be preprocessed thoroughly:

**Feature Selection:** In particular, we define which features are critical to increasing the accuracy of models and decreasing their computational intensity simultaneously. It involves the fastening of association, correlation analysis, and domain knowledge.

**Encoding Categorical Variables:** Since machine learning algorithms accept only numeric inputs, categorical variables are encoded with the help of OneHotEncoder or LabelEncoder. This confirms that the data meets the corresponding model requirements.

**Feature Scaling:** For different features that can have different ranges, standardization or normalization is used. As the magnitudes of features increase, this step becomes all the more important, especially for algorithms such as the Support Vector Machine (SVM) and the K—Nearest Neighbors (KNN).



**Splitting the Data:**  
In this dataset the training and testing sets are created using train\_test\_split. This data is subdivided into the training set used to adjust the model and the testing set on which the quality of work in conditions of its unfamiliarity is determined. A typical split ratio is 80:20 or 70:30.

**Selecting Machine Learning Models:**  
We experimented with multiple machine learning algorithms to identify the most suitable one:

Linear Regression or Logistic Regression: It is used when the output variable is continuous, and the problem is likely to be regression or binary classification.

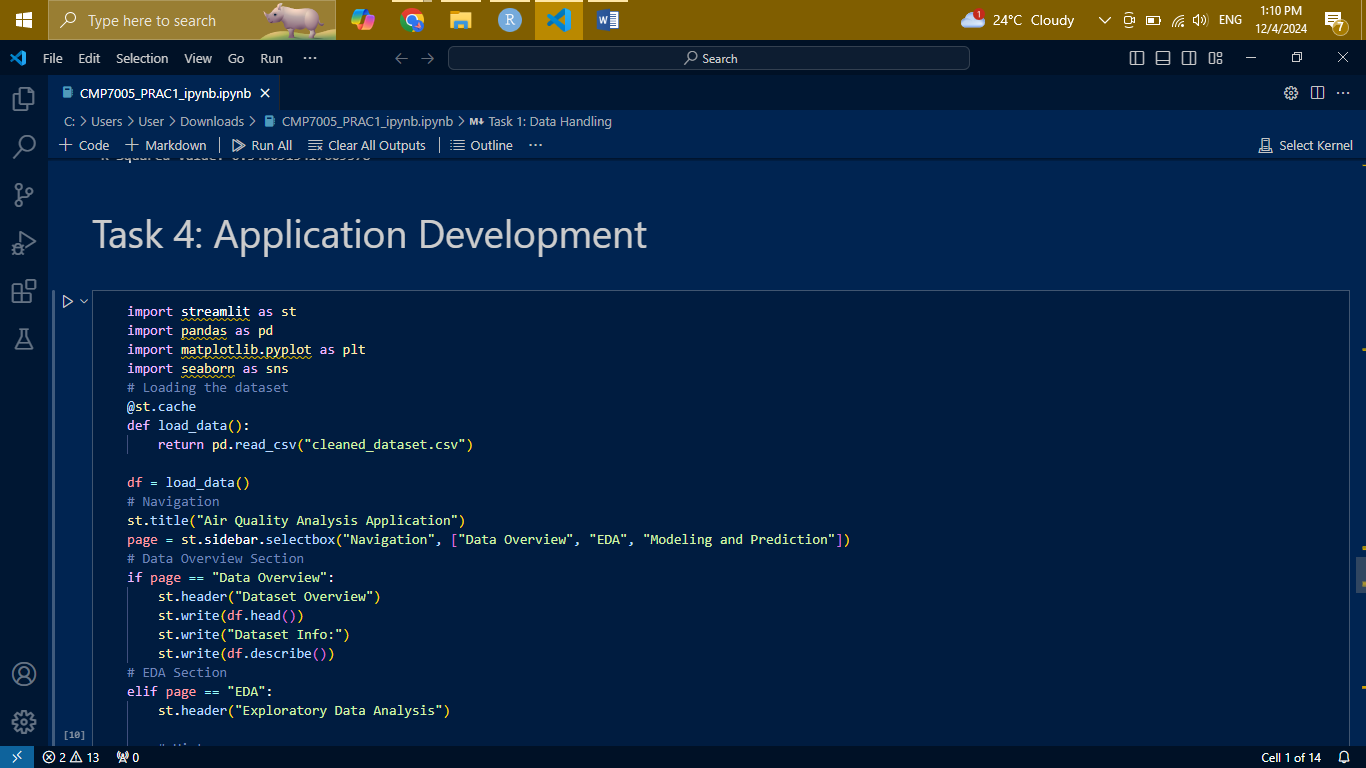
Decision Tree/Random Forest: It appeared that non-linear algorithms are efficient with regard to classification and regression problems. Random Forest also handles the overfitting issue since it comes up with the ensemble techniques.

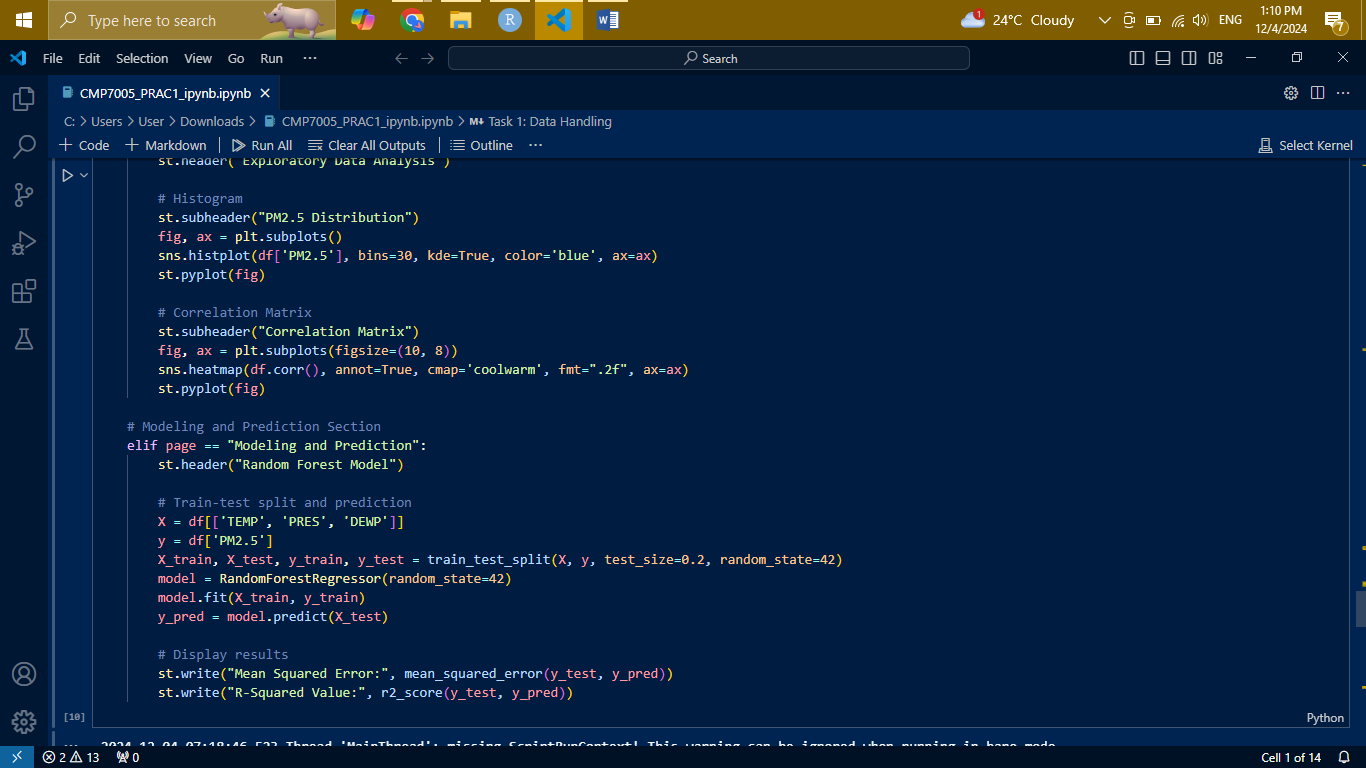
Gradient Boosting (e.g., XGBoost or LightGBM): A strong structured data algorithm which happens to be first-rate as per benchmark performance in several competitions.

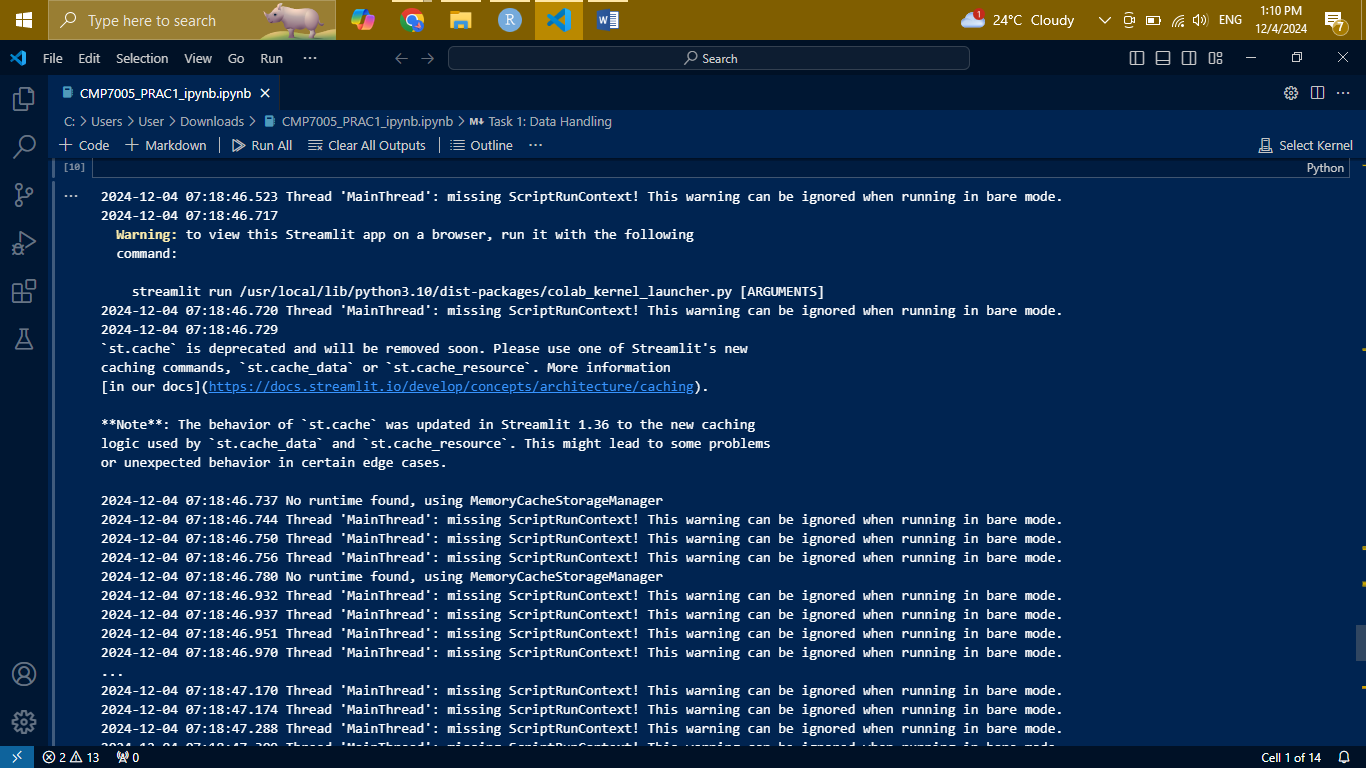
In this way, we guarantee that the problem is solved by different methods trained during the training of multiple models.

**Hyperparameter Optimization:**  
The last is used when special attention is paid to fine-tuning hyperparameters so that the model works optimally. This process is called Grid Search or Random Search. In the case of some algorithms, such as Random Forest or Gradient Boosting, this means exhaustion of several learning rates, depth, or the number of estimators.

**Model Evaluation:**  
In evaluating the proposed model for classification problems, we have processed the accuracy, precision, recall, F1 score, RMSE, and R square for regression problems. This assures that the particular model has a high probability of proving well when posed or exposed to additional or unfamiliar data. There are still more tools, which are ROC curves, confusion matplotlibs, and residual diagrams, that are deeper into the model performance.







**Explanation for Application Development**

**1. Purpose of the Application:**  
The purpose of the application is to allow casual users to analyze data using an efficient graphical user interface for data exploration and prediction. This way it consolidates data overview, exploratory data analysis (EDA), and a place to see the model predictions all at once.

**2. Designing the Interface:**  
Using Streamlit, we created a multi-page application with the following sections:

**Data Overview:** This section gives the viewer a description of the dataset's current state, basic centralized statistics, percentage of missing values, and presentation in graphical form.

**EDA Section:** This section enables users to elaborate on the relationships between variables using charts such as scatter charts, line charts, and histograms. The user can also view correlation matrices and feature distributions in real time.

**Modeling and Prediction Section:** This section allows users to feed test data, review the model’s prediction, and obtain the analysis outcome in return. This part of the application makes the interaction real and useful for manipulating the actual situation.

**3. Interactive Features:**  
It is interactive in that users can select which features should be drawn, which hyperparameters should be set for a particular model or where to upload their custom csv files to be predicted. This dynamic interaction offers a user-oriented portal, when not a direct manipulation, without the need for programming knowledge.

**4. Scalability and Usability:**  
The script is built up in such a way that more features, datasets or models can be added without much efforts. Founded on Streamlit, it is easy to work with without being rigorous in terms of form and function and, hence understandable by the common man.