**Name: Harshad Rajan Depolkar PRN:2201208003**

**Note: I have made some changes to the given JSON file, specifically in the “feature handling code” and the “hyperparameters code”, as per the assignment requirements**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Purpose**

The purpose of this document is to provide a brief overview of the assignment. I am excited to tackle this challenging problem, where I have streamlined and automated basic data science concepts and machine learning techniques.

**Scope**

The scope of this assignment is broad and versatile. We are generalizing and automating tasks, which is beneficial for businesses with limited knowledge in data science. They can track their business outcomes more effectively. This aligns with what “**Dendirities.ai”** is doing—building an application similar to PowerBI, where you only need to select and input your data. The application then analyzes the data and provides insights. In my view, this assignment is a part of this process.

**Overview**

For this assignment, I needed to automate the entire data science workflow, from dataset extraction to model implementation using GridSearchCV. I aimed to make it dynamic, although there were some concepts I was unfamiliar with. I appreciate this opportunity to learn and apply new skills.

I have written the code based on my logic and understanding. If I missed any part or overlooked something, please let me know. There were a few points I wasn’t aware of, so I kept those sections as they were. I understand the concepts but couldn’t fully code them.

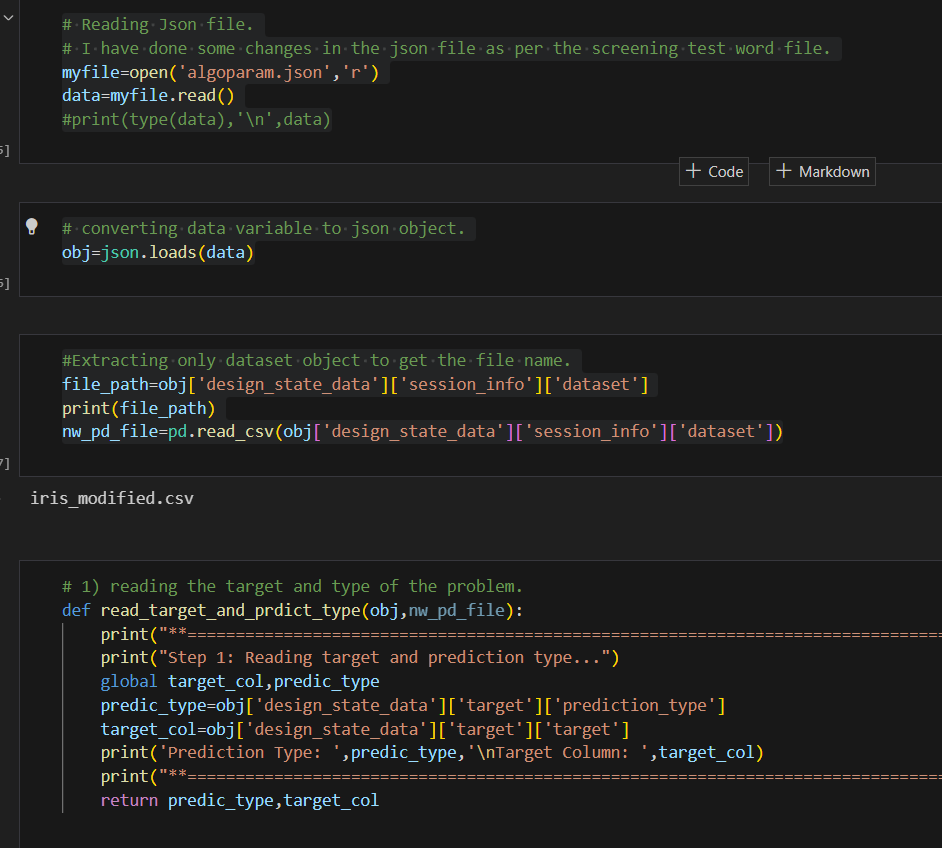
Overall, I believe I have covered about 85% of the assignment. I will now break down each step briefly so you can understand my approach.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

I have used only two models for demonstration, and they are working correctly:

* **Regression:** RandomForestRegressor
* **Classification:** RandomForestClassifier

1. **Read the target and type of regression to be run.**



**Explanation of the Code:**

 **Reading the JSON File**

**Steps:**

1. Open the file in read mode.
2. Read the content into a string variable called data.
3. Convert this string to a JSON object using json.loads().

 **Extracting File Path**

**Steps:**

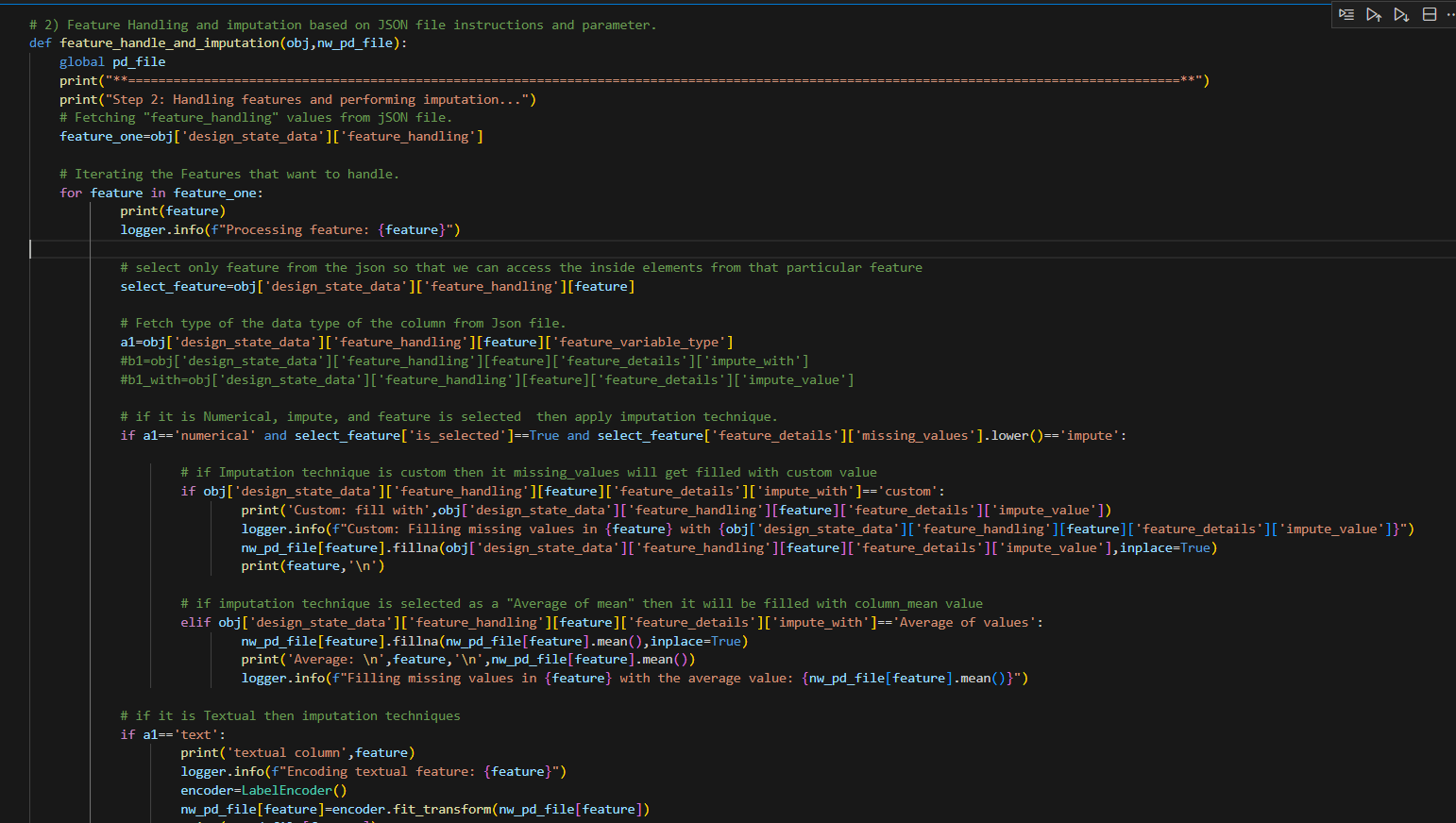
1. Extract the file path from the JSON object.
2. Print the file path for verification.
3. Read the dataset into a DataFrame nw\_pd\_file using pd.read\_csv().

 **Reading Target and Prediction Type**

* **Purpose:** To read the target variable and the type of problem (regression or classification) from the JSON object.
* **Key Actions:**
  + Define a function read\_target\_and\_predict\_type() that takes the JSON object and DataFrame as input.
  + Extract the prediction type (e.g., "regression" or "classification") and target column name from the JSON object.
  + Print these values for verification.
  + Return the prediction type and target column.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2) Read the features (which are column names in the csv) and figure out what missing imputation needs to be applied and apply that to the columns loaded in a dataframe.**

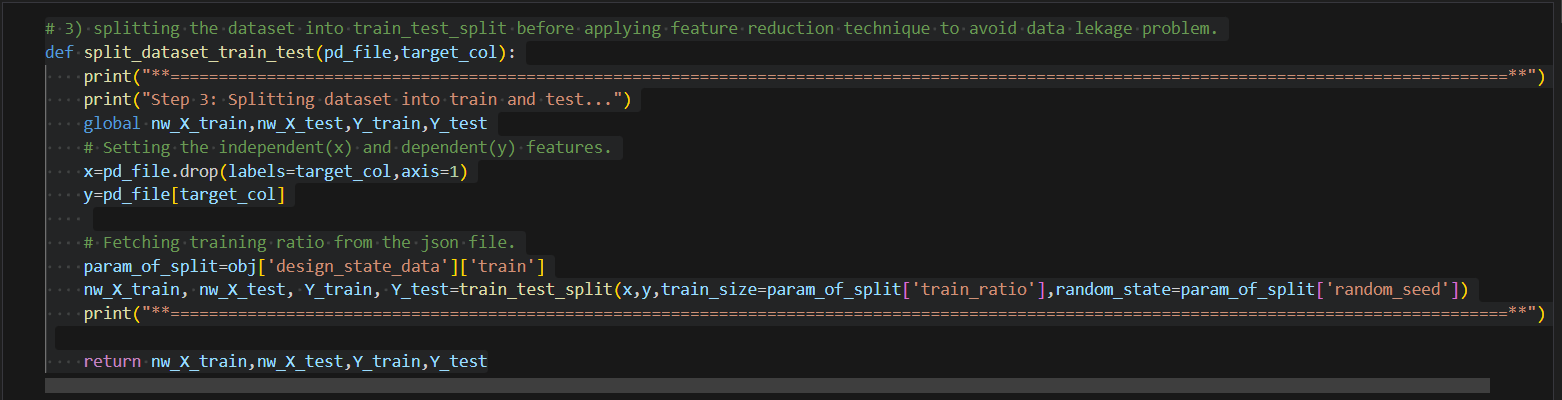
****

The function feature\_handle\_and\_imputation() handles feature processing and imputation based on instructions provided in a JSON file. Here’s a brief explanation of each part:

1. **Reading Feature Handling Instructions**
   * **Purpose:** To read the feature handling instructions from the JSON file.
   * **Key Actions:** Extract the feature\_handling section from the JSON object.
2. **Iterating Through Features**
   * **Purpose:** To process each feature specified in the JSON file.
   * **Key Actions:** Loop through each feature and check its properties for further actions.
3. **Imputation for Numerical Features**
   * **Purpose:** To handle missing values in numerical features.
   * **Key Actions:**
     + Check if the feature is numerical, selected, and marked for imputation.
     + If the imputation method is "custom," fill missing values with a specified custom value.
     + If the imputation method is "Average of values," fill missing values with the column’s mean.
4. **Handling Textual Features**
   * **Purpose:** To encode textual features.
   * **Key Actions:** Use LabelEncoder to transform text columns into numerical values.
5. **Updating the DataFrame**
   * **Purpose:** To update the original DataFrame with the processed features.
   * **Key Actions:** Store the updated DataFrame in a global variable pd\_file.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**3) Compute feature reduction based on input. See the screenshot below where there can be No Reduction, Corr with Target, Tree-based, PCA. Please make sure you write code so that all options can work. If we rerun your code with a different Json it should work if we switch No Reduction to say PCA.**

****

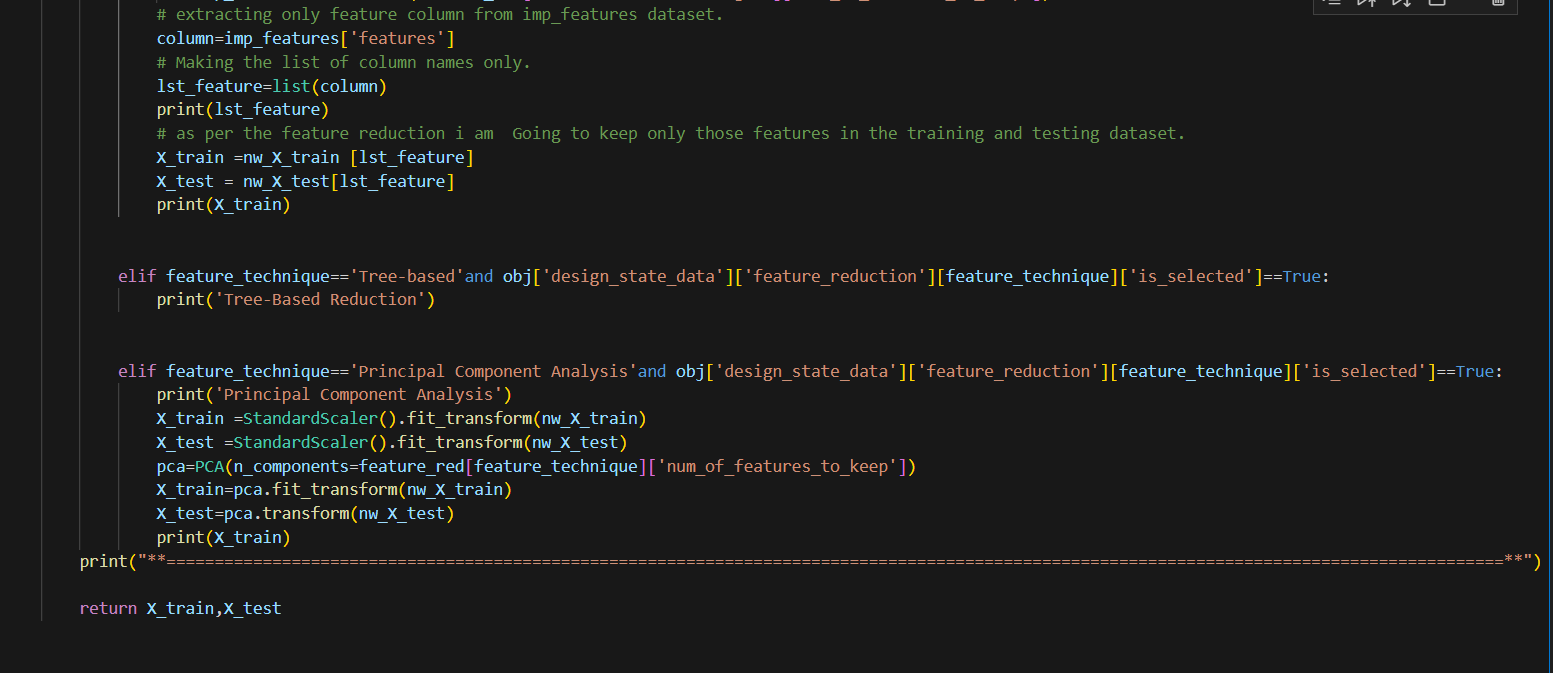
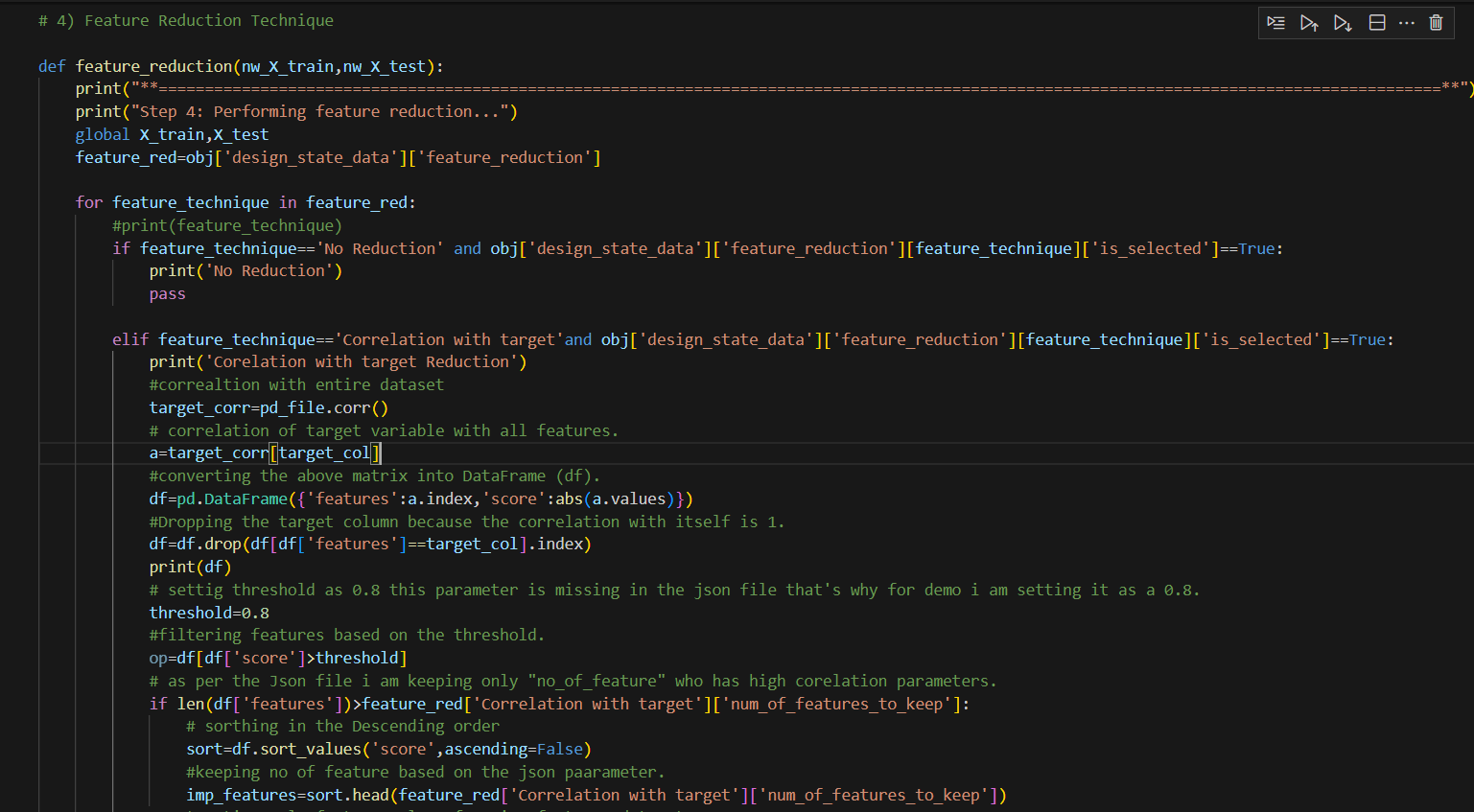
The function split\_dataset\_train\_test() is responsible for splitting the dataset into training and testing sets to prevent data leakage before applying any feature reduction techniques.

Here’s a brief explanation:

1. **Setting Independent and Dependent Variables**
   * **Purpose:** To separate the dataset into features (x) and the target variable (y).
   * **Key Actions:**
     + **x:** All columns except the target column are considered as features.
     + **y:** The target column that needs to be predicted.
2. **Splitting the Dataset**
   * **Purpose:** To divide the dataset into training and testing sets based on a specified ratio.
   * **Key Actions:**
     + Extract the training ratio and random seed from the JSON object.
     + Use train\_test\_split() to split the features and target into training (nw\_X\_train, Y\_train) and testing sets (nw\_X\_test, Y\_test).
3. **Avoiding Data Leakage**
   * **Purpose:** Ensures that feature reduction techniques are applied only to the training set, preventing information from the test set from influencing the model during training.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**4) Parse the Json and make the model objects (using sklearn) that can handle what is required in the “prediction\_type” specified in the JSON (See #1 where “prediction\_type” is specified). Keep in mind not to pick models that don’t apply for the prediction\_type specified**

****

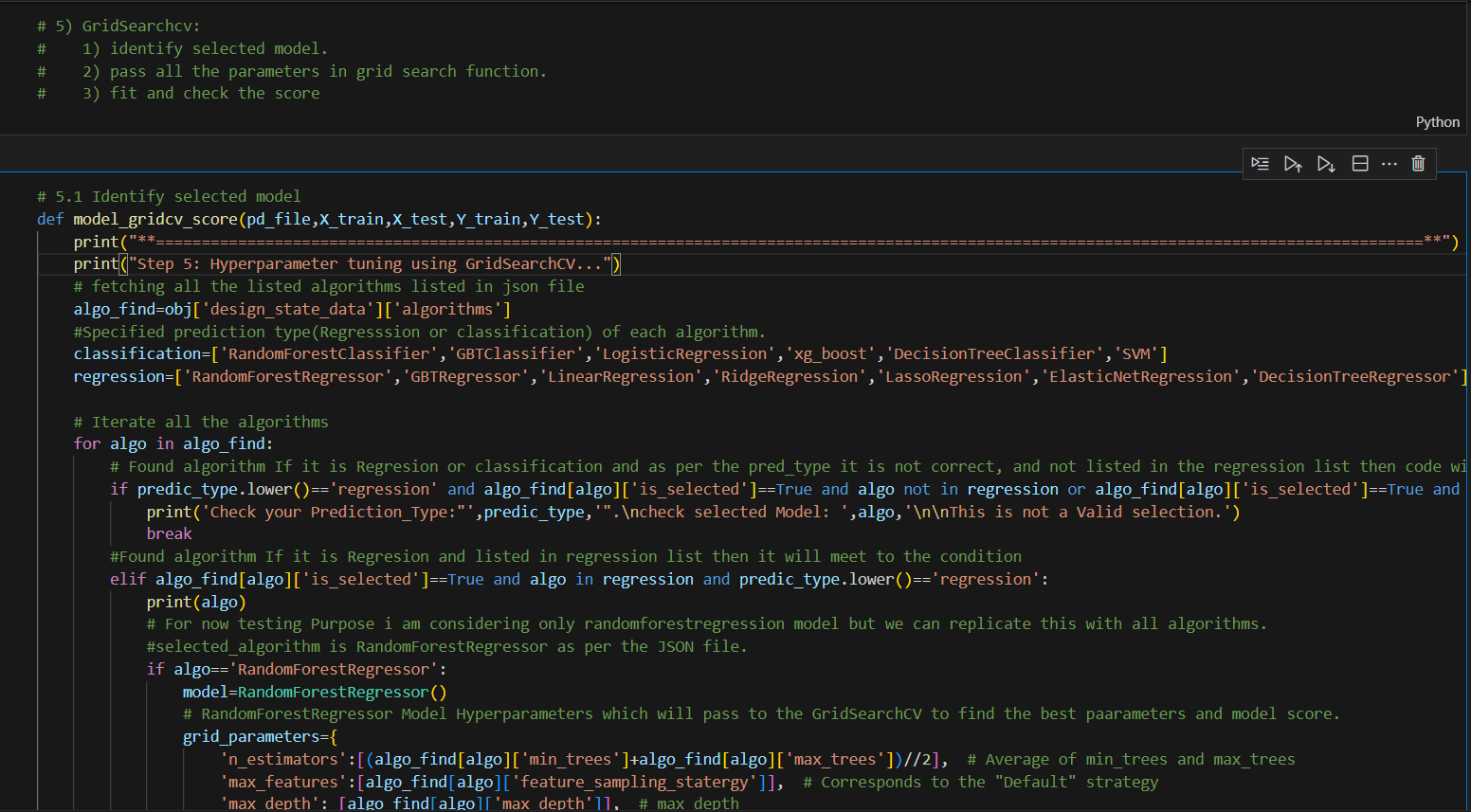
The function feature\_reduction() performs feature reduction on the training and testing datasets using different techniques specified in a JSON file. Feature reduction helps in simplifying the model, improving performance, and avoiding overfitting by reducing the number of features.

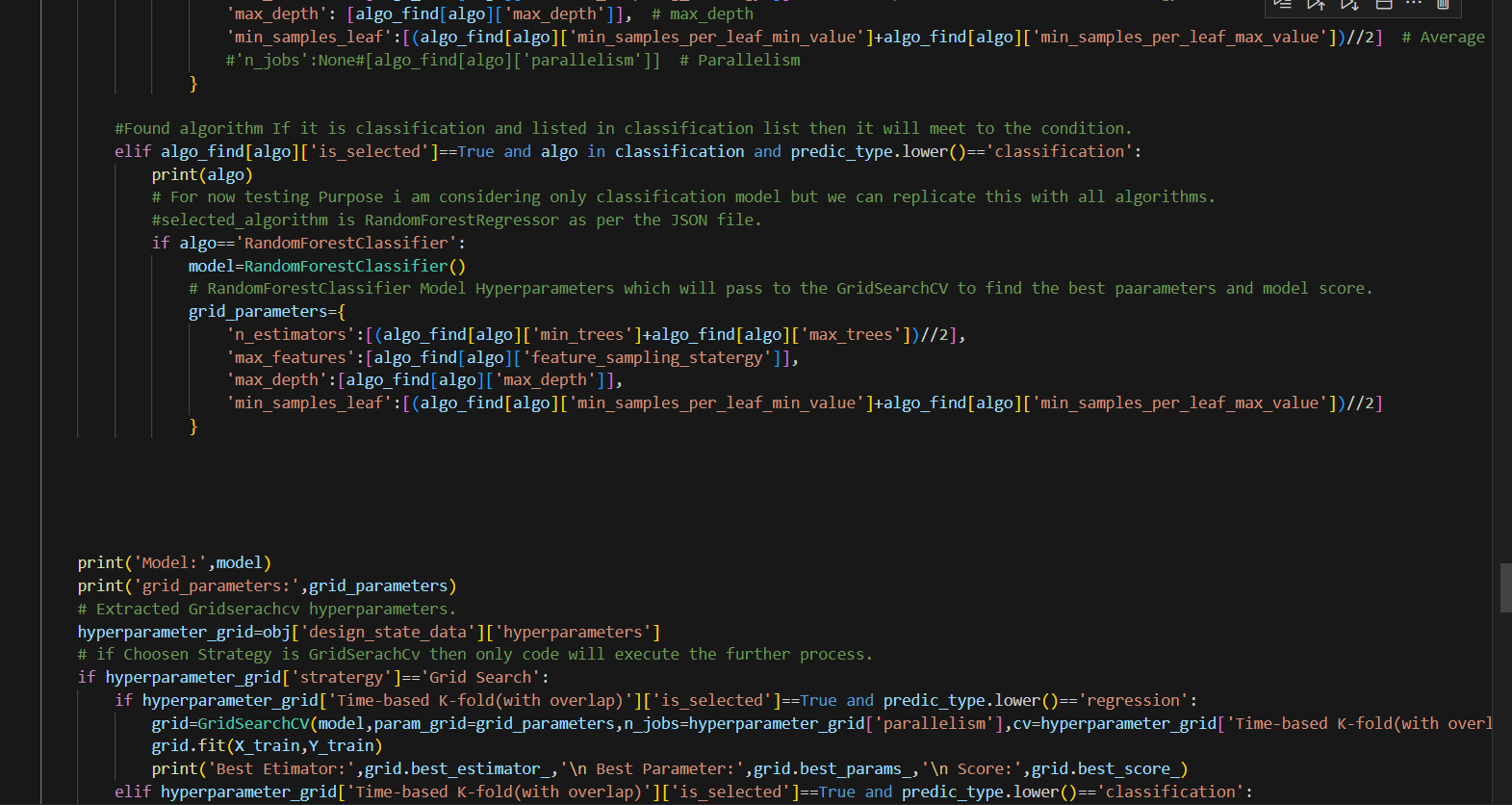
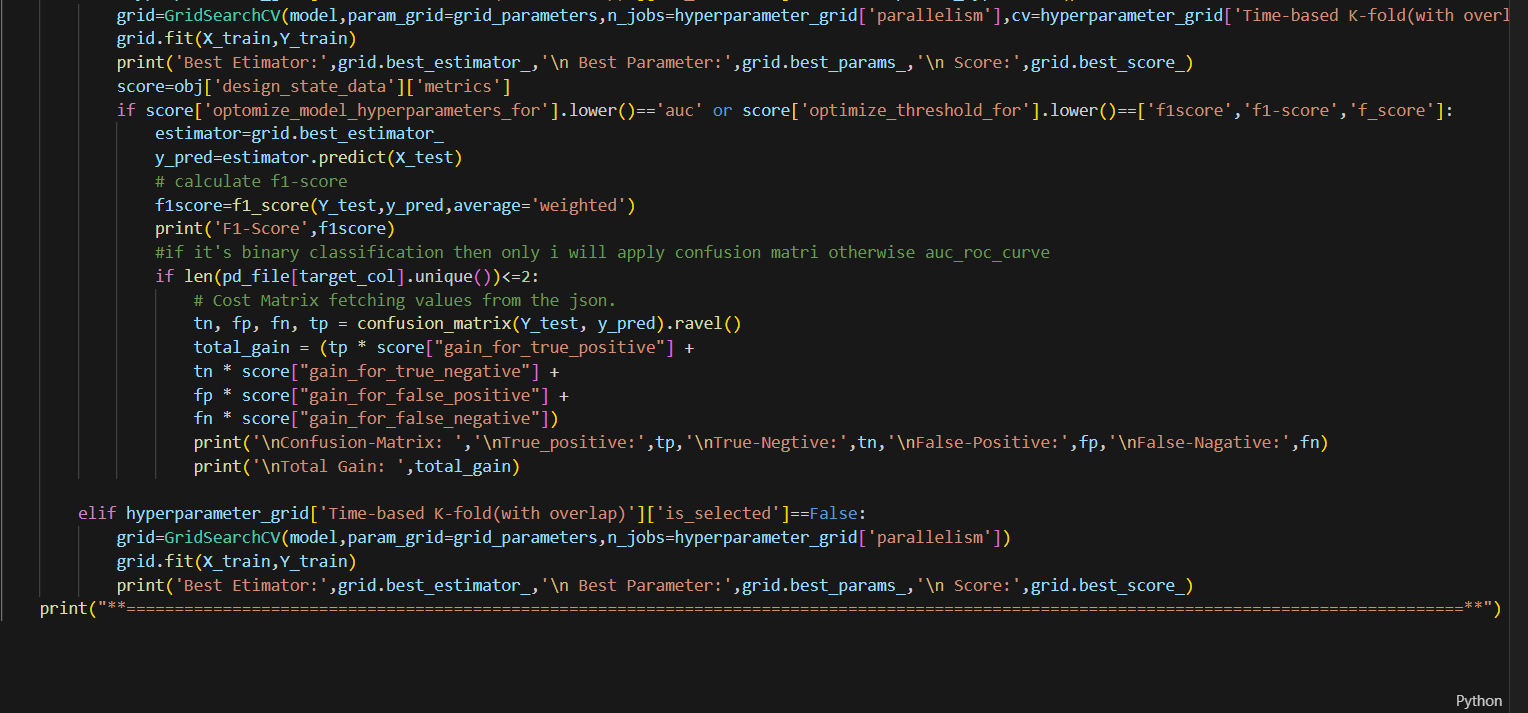
Here’s a brief explanation:

1. **Initial Setup**
   * **Purpose:** Initialize the reduction process by fetching the feature reduction techniques from the JSON object.
   * **Key Actions:**
     + The JSON file is accessed to retrieve the techniques specified under "feature\_reduction".
2. **No Reduction**
   * **Purpose:** Skip feature reduction if "No Reduction" is selected in the JSON file.
   * **Key Actions:**
     + If "No Reduction" is selected, the function simply passes without making any changes.
3. **Correlation with Target**
   * **Purpose:** Reduce features based on their correlation with the target variable.
   * **Key Actions:**
     + **Correlation Matrix:** Calculate the correlation of all features with the target variable.
     + **Filtering Features:** Identify features with a correlation score above a specified threshold (set at 0.8 for demonstration).
     + **Selecting Features:** Keep only the top features based on the number specified in the JSON file.
     + **Update Datasets:** The training and testing datasets are updated to include only the selected features.
4. **Tree-Based Reduction**
   * **Purpose:** Placeholder for tree-based feature reduction (not fully implemented).
   * **Key Actions:**
     + This section is intended for applying tree-based reduction techniques, like feature importance from tree-based models, but is currently not implemented.
5. **Principal Component Analysis (PCA)**
   * **Purpose:** Reduce dimensionality using PCA, which transforms the data into principal components.
   * **Key Actions:**
     + **Scaling:** Standardize the features before applying PCA.
     + **PCA Transformation:** Reduce the dataset to a specified number of principal components as mentioned in the JSON file.
     + **Update Datasets:** The training and testing datasets are transformed to include only the principal components.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**5) Run the fit and predict on each model – keep in mind that you need to do hyper parameter tuning i.e., use GridSearchCV**

****



 **Initial Setup**

* **Purpose:** The function begins by printing a message indicating that hyperparameter tuning is starting.
* **Key Actions:**
  + Extract the list of algorithms and their configurations from the JSON file (algo\_find).
  + Define two lists, classification and regression, which contain the names of classification and regression algorithms, respectively.

 **Algorithm Selection**

* **Purpose:** To ensure that the correct algorithm is chosen based on the prediction type (regression or classification).
* **Key Actions:**
  + The function iterates over all algorithms specified in the JSON file.
  + For each algorithm, it checks if the algorithm is selected (is\_selected is True) and if it matches the prediction type (predic\_type).
  + If there is a mismatch between the prediction type and the selected algorithm, the function prints an error message and terminates.

 **Model Setup for Regression**

* **Purpose:** Configure the model and its hyperparameters if the algorithm is a regression model.
* **Key Actions:**
  + The function checks if the selected algorithm is in the regression list.
  + If it is, the corresponding model (e.g., RandomForestRegressor) is instantiated.
  + A dictionary grid\_parameters is created, containing hyperparameters for GridSearchCV, which are derived from the JSON file (e.g., n\_estimators, max\_features, etc.).

 **Model Setup for Classification**

* **Purpose:** Configure the model and its hyperparameters if the algorithm is a classification model.
* **Key Actions:**
  + The function checks if the selected algorithm is in the classification list.
  + If it is, the corresponding model (e.g., RandomForestClassifier) is instantiated.
  + Similar to regression, a dictionary grid\_parameters is created with hyperparameters for GridSearchCV.

 **Grid Search for Hyperparameter Tuning**

* **Purpose:** Perform hyperparameter tuning using GridSearchCV based on the selected model and its hyperparameters.
* **Key Actions:**
  + The function checks if the "Grid Search" strategy is selected in the JSON configuration.
  + If "Time-based K-fold (with overlap)" is selected, it configures GridSearchCV with the specified number of folds (num\_of\_folds) and parallelism (n\_jobs) and fits the model to the training data (X\_train, Y\_train).
  + After fitting, it prints the best estimator, best parameters, and the best score from GridSearchCV.

 **Evaluation and Scoring**

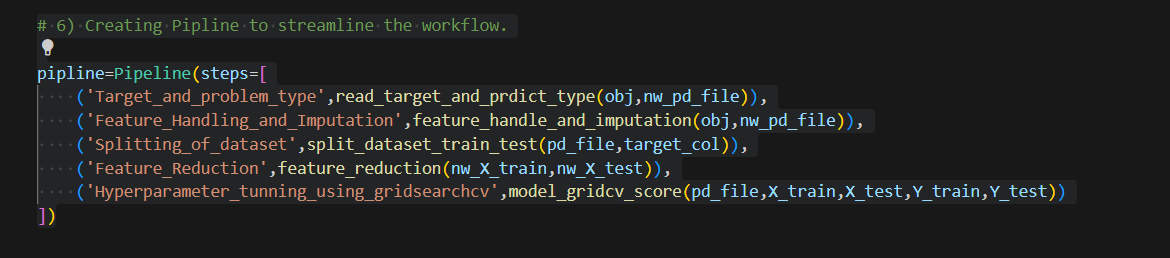
* **Purpose:** Evaluate the model's performance based on specific metrics.
* **Key Actions:**
  + If the selected scoring metric is AUC or F1-Score, the function uses the best estimator from GridSearchCV to predict on the test data (X\_test).
  + For binary classification problems, the confusion matrix is computed, and the total gain is calculated based on a cost matrix from the JSON configuration.

 **Handling Other Scenarios**

* **Purpose:** Handle cases where "Time-based K-fold (with overlap)" is not selected.
* **Key Actions:**
  + If "Time-based K-fold (with overlap)" is not selected, the function performs GridSearchCV with the default cross-validation settings and prints the best estimator, best parameters, and best score.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**6) Created Pipeline to streamline the workflow**.



In this step, I have constructed a machine learning pipeline using the Pipeline class from scikit-learn. A pipeline allows you to streamline the workflow by chaining together multiple processing steps in a sequence. Each step in the pipeline corresponds to a specific function or method that processes the data, and the output of one step is passed as input to the next.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_