UNIVERISTY OF SCIENCE - VIETNAM NATIONAL UNIVERSITY OF HCM

Faculty of Information Technology

APPLIED MATHEMATICS AND STATISTICS FOR IMFORMATION TECHNOLOGY

21CLC07

Report

PROJECT 03 LINEAR REGRESSION

LECTURER

Ngô Đình Hy
Nguyễn Văn Quang Huy
Trần Hà Sơn
Nguyễn Đình Thúc

MEMBER

Full Name : Phạm Hồng Gia Bảo
MSSV : 21127014

2022-2023

Table of Contents

Table of Contents 2

I. PERSONAL INFORMATION 3

II. ACHEIVEMENT 3

III. GENERAL TO PROBLEM 3

1. Linear Regression 3
2. K-fold Cross Validation 4
   a. Cross-validation 4
   b. K-Fold Cross-Validation 5

IV. LIBRARY IMPORTED & FUNCTION USED 6

1. Library Pandas 6
2. Library Numpy 6
3. Library Sklearn 6
4. Library Matplotlib 7
5. Library Seaborn 7

V. DATA PREPARATION 7

VI. SOLUTION TO THE PROBLEM 8

1. Use all first 11 features provided by the problem 8
   a. Implementation 8
   b. Result 8
   c. Reflection & Comment 9

2. Analyzing the influence of conscientiousness, agreeableness, extraversion, nueroticism, openess_to_experience 9
   a. Implementation 9
   b. Result 11
   c. Reflection & Comment 12

3. Analyzing the influence of English, Logical, Quant 12
   a. Implementation 12
   b. Result 14
   c. Reflection & Comment 14

4. Build our own models and search for the best model for optimal results 15
   a. Implementation 15
   b. Result 17
   d. Reflection & Comment 19

VII. REFERENCE 20

I. PERSONAL INFORMATION

Student's Name	Phạm Hồng Gia Bảo
Student's ID	21127014
Class	21CLC07
Email	phgbao21@clc.fitus.edu.vn

II. ACHEIVEMENT

No.	Task Requirements	Completed
1	Build a salary prediction model for engineers using linear regression model	**100%**
	1a: Use all first 11 features provided by the problem: **Gender, 10percentage, 12percentage, CollegeTier, Degree, collegeGPA, CollegeCityTier, English, Logical, Quant, Domain**	**100%**
	1b: Analyzing the influence of **conscientiousness, agreeableness, extraversion, nueroticism, openess_to_experience** based on the scores of AMCAT tests.	**100%**
	1c: Analyzing the influence of **English, Logical, Quant** on the salary of engineers based on the scores of AMCAT tests.	**100%**
	1d: Build our own models and search for the best model for optimal results.	**100%**
2	Report on the results, evaluation, and comments on the constructed models.	**100%**

III. GENERAL TO PROBLEM

1. Linear Regression

- **Linear regression is a statistical modeling** technique **used to analyze the relationship between a dependent variable and one or more independent variables**. It assumes a linear relationship between the independent variables (also called predictors, features, or input variables) and the dependent variable (also called the target variable or output variable).
- The goal of linear regression is to find the best-fitting line that minimizes the difference between the predicted values and the actual values of the dependent variable. This line is represented by a linear equation of the form:

$$Y = b0 + b1X1 + b2X2 + ... + bn*Xn$$

where Y is the dependent variable, b0 is the intercept (the value of Y when all independent variables are zero), b1, b2, ..., bn are the coefficients (representing the impact of each independent variable), X1, X2, ..., Xn are the independent variables.

- MAE stands for Mean Absolute Error. It is a commonly used metric for evaluating the performance of regression models. MAE measures the average absolute difference between the predicted values and the actual values of the dependent variable. The formula for calculating MAE is as follows:

$$MAE = (1/n) * \Sigma|Y_i - \hat{Y}_i|$$

Where:

  - Y_i represents the actual values of the dependent variable.
  - $\hat{Y}_i$ represents the predicted values of the dependent variable.
  - n is the total number of observations.

- **Types of Linear Regression:**

    - Simple Linear Regression: Involves one independent variable.
    - Multiple Linear Regression: Involves multiple independent variables.
- **Limitations of Linear Regression:**
    - Assumes a linear relationship, which may not hold for complex or non-linear data.
    - Sensitive to outliers and influential points.
    - Assumes independence of observations, which may be violated in time-series or spatial data.
    - Cannot handle categorical variables directly (require encoding techniques).
- **Some common applications of linear regression**
    - **Sales and Marketing Analysis**: Linear regression can be used to analyze the impact of advertising expenditure, pricing, or other marketing variables on sales. It helps businesses understand the relationship between marketing efforts and revenue generation.
    - **Financial Analysis**: Linear regression can be applied in financial analysis to study the relationship between variables such as stock prices, interest rates, or economic indicators. It can help in predicting future trends and making investment decisions.
    - **Demand Forecasting**: Linear regression can be used to predict the demand for a product or service based on historical data and other relevant factors such as price, promotions, and seasonality. This information is valuable for inventory management and production planning.

2. K-fold Cross Validation

a. Cross-validation

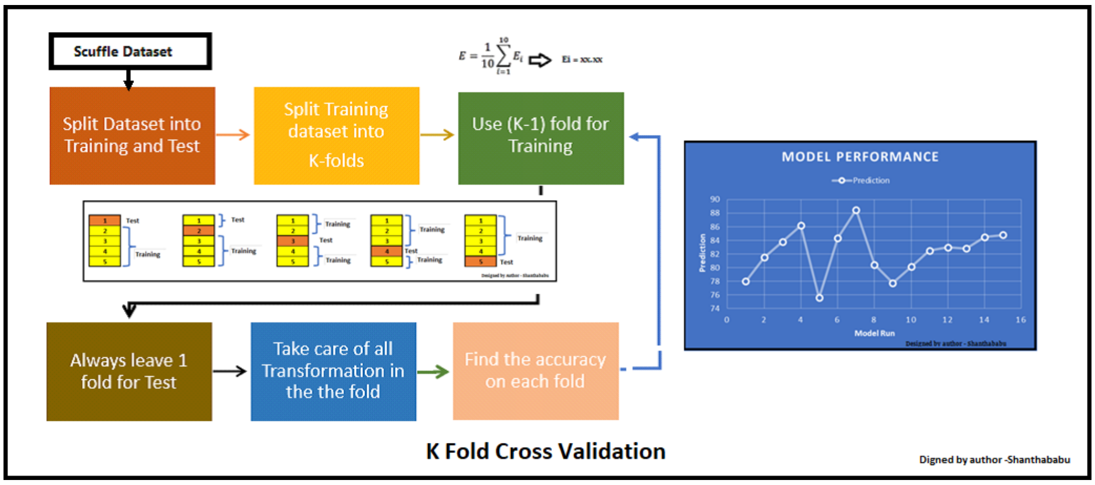
- This is a technique **used in machine learning and statistical modeling to assess the performance and generalization ability of a model**. It involves partitioning the dataset into multiple subsets, performing model training and evaluation on different combinations of these subsets, and then aggregating the results to obtain an overall assessment of the model's performance.
- The general process of Cross-Validation involves the following steps:
    - Data Partitioning: The original dataset is divided into two or more subsets. One subset, often referred to as the "training set," is used for model training, while the remaining subsets, known as "validation sets" or "test sets," are used for model evaluation.
    - Model Training and Evaluation: The model is trained on the training set and then evaluated on the validation/test set(s). The evaluation metric(s) of interest, such as accuracy, mean squared error, or F1 score, are computed to measure the model's performance.
    - Iteration: Steps 1 and 2 are repeated multiple times, typically with different partitions of the data. Each iteration involves using a different subset as the validation/test set while keeping the rest as the training set.
    - Aggregation of Results: The performance metrics obtained from each iteration are combined, often by taking their mean or median, to provide an overall assessment of the model's performance.
- Cross-validation helps address potential issues related to overfitting or underfitting a model. It provides a more reliable estimate of how well the model

will perform on unseen data by simulating the process of training and testing on different subsets of the data.

- **One commonly used cross-validation technique is k-fold cross-validation**, where the data is divided into k subsets (folds). The model is trained and evaluated k times, with each subset serving as the validation/test set once. The results are then averaged to obtain a more robust estimate of the model's performance.

b. K-Fold Cross-Validation

- **K-fold cross-validation is a method used to evaluate predictive models.** It involves dividing the dataset into k subsets or folds. The model is then trained and evaluated k times, with each fold serving as the validation set once. The performance metrics from each fold are averaged to estimate how well the model will generalize to unseen data. This technique is useful for assessing, selecting, and tuning models, providing a more reliable measure of their effectiveness.
- By performing training and testing on different subsets of the data, k-fold cross-validation helps prevent overfitting. Rather than relying on a single train-test split, the model is evaluated multiple times, leading to a more generalized model.
- To implement k-fold cross-validation, the dataset is split into three sets: training, testing, and validation. This approach tackles the challenge of handling large volumes of data.
- The testing and training datasets are used to build the model and assess hyperparameters. The model is validated multiple times based on a parameter called K, which represents the number of subsets the data will be divided into. K should be an integer.



Scuffle Dataset
$E = \frac{1}{10}\sum_{i=1}^{10} E_i$ ⇨ Ei = xx.xx
Split Dataset into Training and Test
Split Training dataset into K-folds
Use (K-1) fold for Training
Test
Training
Designed by author - Shanthababu
MODEL PERFORMANCE
Prediction
Model Run
Designed by author - Shanthababu
Always leave 1 fold for Test
Take care of all Transformation in the the fold
Find the accuracy on each fold
K Fold Cross Validation
Digned by author -Shanthababu

- **How do you use K-fold cross validation?**
  - **Split the data**: Divide your dataset into k equal-sized subsets (folds). Typically, k is chosen as 5 or 10, but you can adjust it based on your needs.
  - **Train and validate**: Iterate over the k folds. In each iteration, use k-1 folds for training the model and the remaining fold for validation. Train your model on the training folds and evaluate its performance on the validation fold.

- **Performance metrics**: Calculate the performance metric(s) of interest (e.g., accuracy, precision, recall) for each fold. These metrics quantify how well the model generalizes to unseen data.
- **Average the results**: Average the performance metrics obtained from the k folds to obtain a more robust estimate of the model's performance. This average value represents the overall performance of the model.
- **Model selection and tuning**: Based on the cross-validation results, you can compare different models or hyperparameter settings and select the one that performs the best on average across the folds.
- **Final evaluation**: After selecting the model or hyperparameters, you can retrain the model using the entire dataset and evaluate its performance on a separate test set to obtain a final performance estimation.

IV. LIBRARY IMPORTED & FUNCTION USED

1. Library Pandas

Pandas is used to read file .csv and print data as DataFrame. Pandas offers primary data structure DataFrame. A Series is a one-dimensional labeled array capable of holding data of any type, while a DataFrame is a two-dimensional labeled data structure that consists of columns of potentially different types. These data structures are designed to handle tabular data efficiently.

- **pandas.read_csv(name_of _file):** allows us to read data from a CSV file.
- **pandas.iloc[]:** a function in pandas that allows us to access data from a DataFrame based on integer-based indexing.
- **pandas.DataFrame(data):** used in pandas to create a DataFrame from existing data. The data parameter can be various types, such as a list, dictionary, NumPy array, or another DataFrame.
- **pandas.copy(deep=True):** used in pandas to create a copy of a DataFrame. By default, it performs a deep copy, meaning that all the data and the underlying objects are duplicated. This ensures that any changes made to the copied DataFrame do not affect the original DataFrame.
- **pandas.corr()**: used in pandas to compute the correlation coefficient between different pairs of features in a DataFrame. The correlation coefficient measures the strength and direction of the linear relationship between two variables.

2. Library Numpy

To perform operations on matrices, arrays, and similar structures in NumPy, you can use the append() function.

- **numpy.mean()** function is used to calculate the average or mean of an array
- **numpy.absolute()**: used to compute the absolute value or magnitude of elements in an array

3. Library Sklearn

To use linear regression, k-fold cross-validation, and calculate the mean squared error (MSE)

- **sklearn.linear_model.LinearRegression()**: this is a class used to perform linear regression and build a linear regression model. Linear regression is a statistical modeling technique used to model the relationship between a dependent variable and one or more independent variables.
- **LinearRegression.fit(X, y):** a function of the **LinearRegression** class in scikit-learn. It is used to train or fit the linear regression model using the input features X and corresponding target variable y from the training dataset.

- **LinearRegression.predict(X_test):** a function of the **LinearRegression** class in scikit-learn. It is used to make predictions based on the linear regression model that has been trained and fitted using the fit() method.
- **LinearRegression.coef_** is attribute returns an array of estimated coefficients for the linear regression model after it has been fitted. The number of coefficients will be equal to the number of features used to train the model.
- **sklearn.model_selection**: a module in scikit-learn that provides various functions and classes for model selection, data analysis, and evaluation. It includes functions and classes for tasks such as splitting datasets into train and test sets, cross-validation, and hyperparameter tuning.
- **sklearn.model_selection.cross_val_score(estimator, X, y, cv=5, scoring)** : function that performs k-fold cross-validation. It evaluates the performance of an estimator (model) on a given dataset X with corresponding target variable y using the k-fold cross-validation technique.
- **sklearn.metrics:** a module in scikit-learn that provides classes and functions for accessing various evaluation metrics and performance scores.
- **sklearn.metrics.mean_absolute_error(y, y_pred):** used to compute the mean absolute error (MAE) between the true target values y and the predicted target values y_pred.

4. Library Matplotlib

In this library, I use **matplotlib.pyplot**, is a module in the Matplotlib library to visualize data in Python. This **pyplot** module provides a simple and convenient interface to create various types of plots and visualizations.

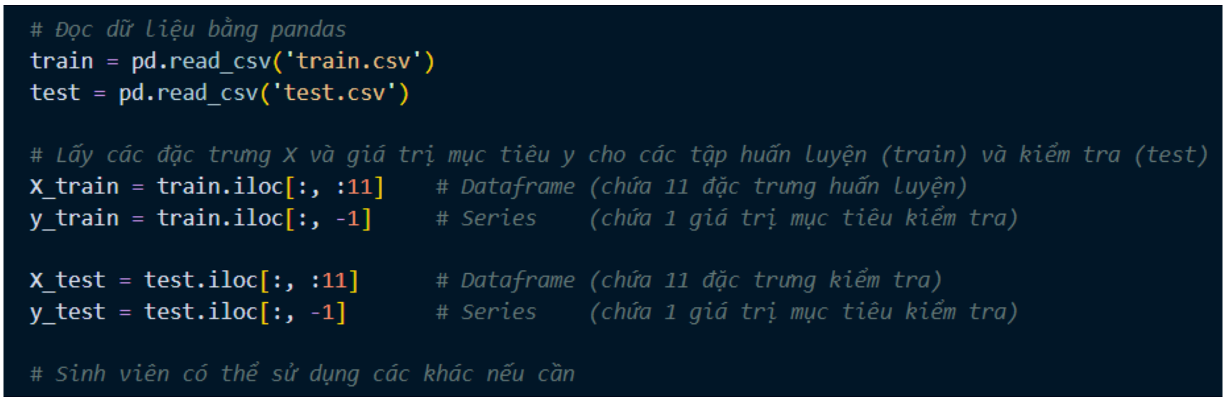
- **plt.subplots(figsize=(15, 15)):** used to create a figure with subplots, allowing you to create multiple plots within a single figure. The figsize parameter specifies the size of the figure in inches.

5. Library Seaborn

Seaborn simplifies the process of creating visually appealing plots by providing default styles and color palettes, as well as specialized functions for creating different types of plots.

- **sns.heatmap(corr_matrix,cmap="viridis", annot=True, linewidths=.5, ax = ax):** is used to generate a random 5x5 correlation matrix and creates a heatmap plot to visualize it

V. DATA PREPARATION



```
# Đọc dữ liệu bằng pandas
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')

# Lấy các đặc trưng X và giá trị mục tiêu y cho các tập huấn luyện (train) và kiểm tra (test)
X_train = train.iloc[:, :11]    # Dataframe (chứa 11 đặc trưng huấn luyện)
y_train = train.iloc[:, -1]     # Series    (chứa 1 giá trị mục tiêu kiểm tra)

X_test = test.iloc[:, :11]      # Dataframe (chứa 11 đặc trưng kiểm tra)
y_test = test.iloc[:, -1]       # Series    (chứa 1 giá trị mục tiêu kiểm tra)

# Sinh viên có thể sử dụng các khác nếu cần
```

- Firstly, we use the Pandas library to read data from the train1.csv and test1.csv files into the train and test DataFrames, respectively.

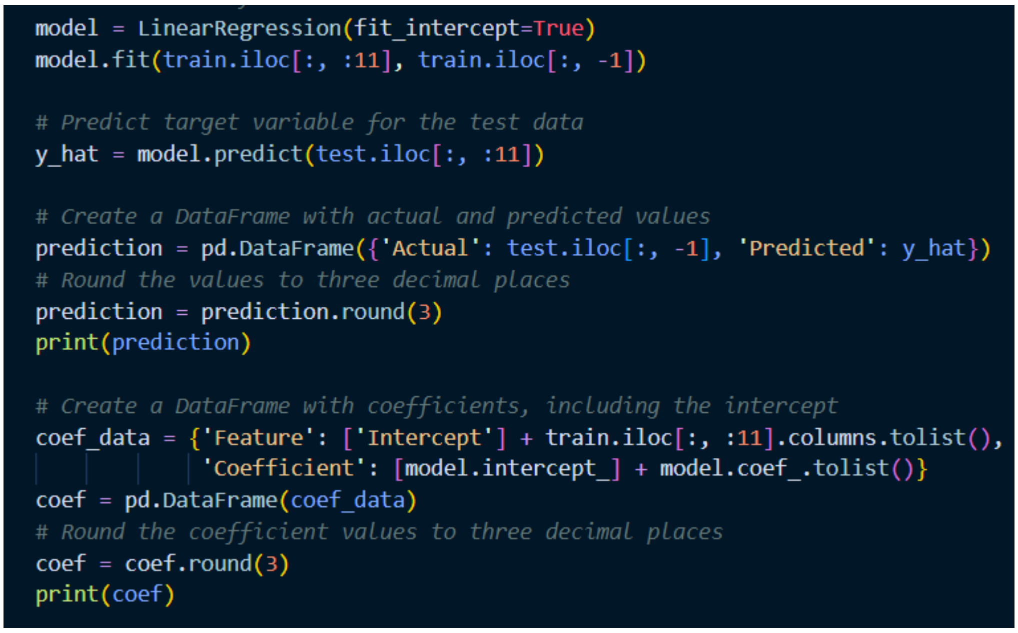
- Next, the code splits the data into features (X) and target values (y) for the training (train) and testing (test) sets.
    - X_train is a DataFrame containing 11 features for the training set.
    - y_train is a Series containing the target value for the training set.
    - X_test is a DataFrame containing 11 features for the testing set.
    - y_test is a Series containing the target value for the testing set.

VI. SOLUTION TO THE PROBLEM

1. Use all first 11 features provided by the problem

a. Implementation

- Firstly, we use **LinearRegression** class to create a linear regression model. It fits the model to the training data by providing the features **train.iloc[:, :11]** and the target values **train.iloc[:, -1]**. The trained model is then used to predict the target values for the test data, which are stored in the **y_hat** variable.
- A **DataFrame** named prediction is created to compare the actual target values **test.iloc[:, -1]** with the predicted values **y_hat**. It is printed using the print function with rounded values.
- Another **DataFrame** named **coef** is created to store the feature names (train.iloc[:, :11].columns) and their corresponding coefficients **model.coef_**. The coefficients represent the weights assigned to each feature by the linear regression model. It is also printed using the print function with rounded values.

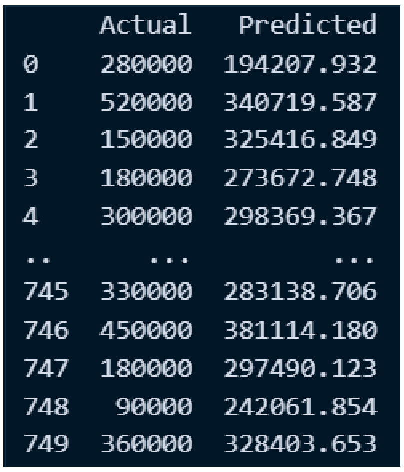


```
model = LinearRegression(fit_intercept=True)
model.fit(train.iloc[:, :11], train.iloc[:, -1])

# Predict target variable for the test data
y_hat = model.predict(test.iloc[:, :11])

# Create a DataFrame with actual and predicted values
prediction = pd.DataFrame({'Actual': test.iloc[:, -1], 'Predicted': y_hat})
# Round the values to three decimal places
prediction = prediction.round(3)
print(prediction)

# Create a DataFrame with coefficients, including the intercept
coef_data = {'Feature': ['Intercept'] + train.iloc[:, :11].columns.tolist(),
             'Coefficient': [model.intercept_] + model.coef_.tolist()}
coef = pd.DataFrame(coef_data)
# Round the coefficient values to three decimal places
coef = coef.round(3)
print(coef)
```

- Then we call a function **calculateMAE** to calculate the mean absolute error MAE between the actual target values y_test and the predicted values y_hat.

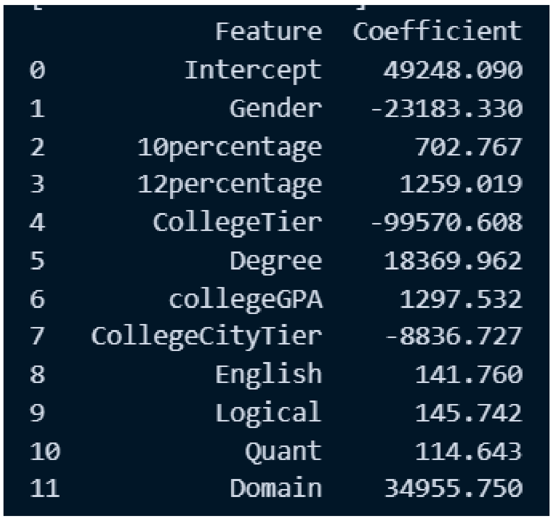
b. Result

- Firstly, we view predictions on the test data.



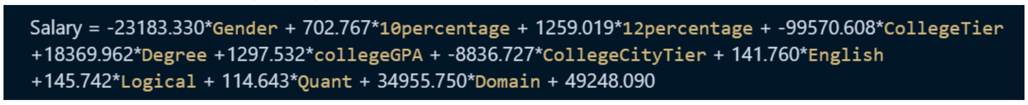
```
     Actual   Predicted
0    280000  194207.932
1    520000  340719.587
2    150000  325416.849
3    180000  273672.748
4    300000  298369.367
..      ...         ...
745  330000  283138.706
746  450000  381114.180
747  180000  297490.123
748   90000  242061.854
749  360000  328403.653
```

- We access the **coef_** attribute of the linear regression model to view the coefficients of the model. By doing this, we can obtain the formula for the linear regression model.



```
             Feature  Coefficient
0          Intercept    49248.090
1             Gender   -23183.330
2       10percentage      702.767
3       12percentage     1259.019
4        CollegeTier   -99570.608
5             Degree    18369.962
6         collegeGPA     1297.532
7    CollegeCityTier    -8836.727
8            English      141.760
9            Logical      145.742
10             Quant      114.643
11            Domain    34955.750
```

- MAE of the best model generated: **105052.53**
- Based on the coefficients obtained from the linear regression model, we can derive the following **Regression Formula**:



```
Salary = -23183.330*Gender + 702.767*10percentage + 1259.019*12percentage + -99570.608*CollegeTier
+18369.962*Degree +1297.532*collegeGPA + -8836.727*CollegeCityTier + 141.760*English
+145.742*Logical + 114.643*Quant + 34955.750*Domain + 49248.090
```

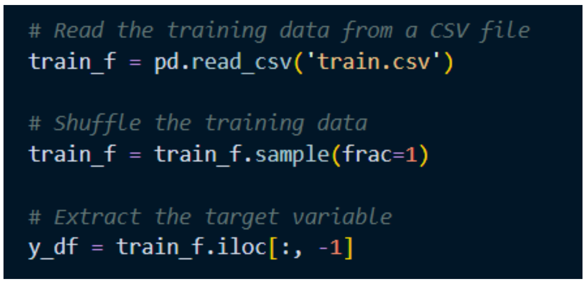
c. Reflection & Comment

- This is the first model which includes the first 11 features affecting the salary of an engineer. As we can see, the result is quite good in the test set. Hower, because this is first time, we train the dataset so we cannot view any significant differences.

2. Analyzing the influence of conscientiousness, agreeableness, extraversion, nueroticism, openess_to_experience

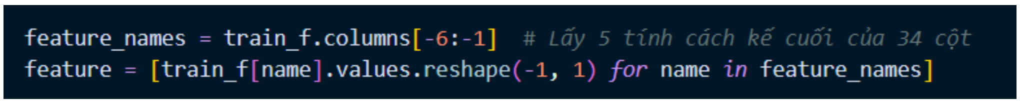
a. Implementation

- For training a linear regression model and evaluating its performance using cross-validation. Firstly, we read a CSV file named **'train1.csv'** into a pandas DataFrame, shuffles the data randomly, and selects the last column as the target variable 'y_df'. It then initializes a LinearRegression model.



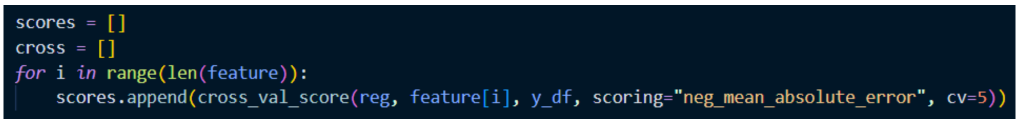
```
# Read the training data from a CSV file
train_f = pd.read_csv('train.csv')

# Shuffle the training data
train_f = train_f.sample(frac=1)

# Extract the target variable
y_df = train_f.iloc[:, -1]
```

- The next step involves selecting a subset of features from the DataFrame. Specifically, it chooses the last 5 features from a total of 34 columns. Each selected feature is reshaped into a 2D array (column vector) using .values.reshape(-1, 1).



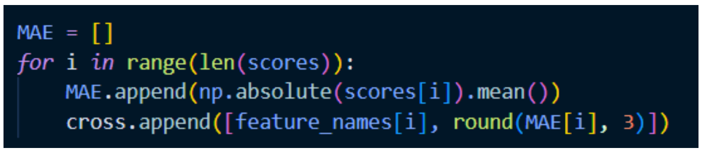
```
feature_names = train_f.columns[-6:-1]  # Lấy 5 tính cách kế cuối của 34 cột
feature = [train_f[name].values.reshape(-1, 1) for name in feature_names]
```

- Then performing the cross-validation using the cross_val_score function from scikit-learn. It calculates the negative mean absolute error (neg_mean_absolute_error) for each feature individually, using 5-fold cross-validation. The mean absolute error values are stored in the scores list.



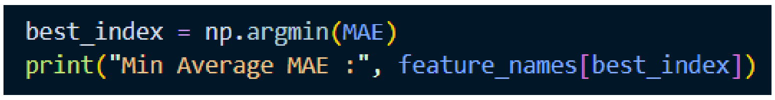
```
scores = []
cross = []
for i in range(len(feature)):
    scores.append(cross_val_score(reg, feature[i], y_df, scoring="neg_mean_absolute_error", cv=5))
```

- After that, we calculate the average mean absolute error (MAE) for each feature and stores it in the MAE list. Additionally, it keeps track of the feature names and their corresponding average MAE in the cross list.



```
MAE = []
for i in range(len(scores)):
    MAE.append(np.absolute(scores[i]).mean())
    cross.append([feature_names[i], round(MAE[i], 3)])
```

- We create a DataFrame named 'test' using the cross list, with columns labeled 'Feature' and 'Average MAE'. This DataFrame contains the feature names and their respective average MAE values.
- Finally, identifying the index of the feature with the lowest MAE using np.argmin(). It prints the feature name corresponding to the minimum average MAE.



```
best_index = np.argmin(MAE)
print("Min Average MAE :", feature_names[best_index])
```

- **After choosing the best feature, we train the best personality model with the best feature above in the whole training test.**
- We create a new set of training and testing data using the best-performing feature identified earlier **feature_names[best_index]**. The feature values are extracted from the train2 and test2 DataFrames using the selected feature name. The feature values are reshaped into a 2D array using **.values.reshape(-1, 1).** Additionally, the target variables **y_k_train and y_k_test** are extracted from the respective DataFrames.

```
X_k_train = train2[feature_names[best_index]].values.reshape(-1, 1)
y_k_train = train2.iloc[:, -1]

X_k_test = test2[feature_names[best_index]].values.reshape(-1, 1)
y_k_test = test2.iloc[:, -1]
```

- A new instance of the LinearRegression model named reg2 is created, and the model is fitted using the training data **X_k_train and y_k_train**.

```
reg2 = LinearRegression().fit(X_k_train, y_k_train)
```

- Next, predictions are made on the test data (X_k_test) using the trained model, and the predicted values are stored in **y_k_test_pred**.
- A DataFrame named k_predic is created to display the actual values from the 'test1.csv' file (y_k_test) and the corresponding predicted values (y_k_test_pred).

```
k_predic = pd.DataFrame({'Data in file': y_k_test, 'Predicted': y_k_test_pred})
print(k_predic.round(3))
```

- A DataFrame named k_coef is created to display the coefficient of the best-performing feature (feature_names[best_index]) in the trained model reg2.

```
k_coef = pd.DataFrame({'Feature': feature_names[best_index], 'Coefficient': reg2.coef_})
print(k_coef.round(3))
```

- Both k_predic and k_coef DataFrames are printed, rounding the values to three decimal places.
- Finally, we call a function calculateMAE to calculate the mean absolute error MAE between the actual target values **y_k_test** and the predicted values **y_k_test_pred**.

b. Result

- After compiling and running, we have the result:

No	Feature	Average MAE
0	conscientiousness	124282.849
1	agreeableness	123518.272
2	extraversion	123694.288
3	nueroticism	123447.339
4	openess_to_experience	123703.538

- After retraining the best feature in the whole training set:

```
     Data in file   Predicted
0          280000  316828.694
1          520000  296119.311
2          150000  297530.805
3          180000  294185.517
4          300000  290122.466
..            ...         ...
745        330000  328713.438
746        450000  303649.413
747        180000  326681.913
748         90000  322476.271
749        360000  299884.362

[750 rows x 2 columns]
       Feature  Coefficient
0  nueroticism   -16021.494
Intercept value  304647.553
```

- MAE generated for this model: **119361.917**
- Based on this model, we can have the Regression Formula:

$$\text{Salary} = -16021.494 * \text{‘}nueroticism\text{‘} + 304647.553$$

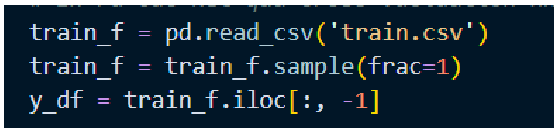
c. Reflection & Comment

- With 5 features required to test, there is no far difference between these features which is approximately 123 thousand. But we can recognize that **neuroticism** got the minimum value when compared to the rest. For the largest in Average MAE, the **conscientiousness** takes that position with 124282 thousand roughly.
- After we re-trained and test the model using the best features, we got the MAE = 119361.917 which is still good, but we cannot compare to the MAE generated from the first 11 features above. This model is quite a bit higher than the last 11 feature model which is only 105052.53 in MAE.
- Therefore, we understand that the average salary depends on various factors. Some factors may have a minimal impact, and using only one feature and discarding all other features cannot fully reflect the salary. Due to the nature of the salary model being influenced by multiple factors, building the model with as many features as possible still provides us with the best results (so far).
- Using a single feature, even if it reflects the best among all features, is not sufficient to evaluate the data. Therefore, we need to increase the number of features. That's why we need to ask ourselves if it's possible to build a model using fewer than 11 features that can provide better results. In the next part, we will try to create a model that yields better results.

3. Analyzing the influence of English, Logical, Quant

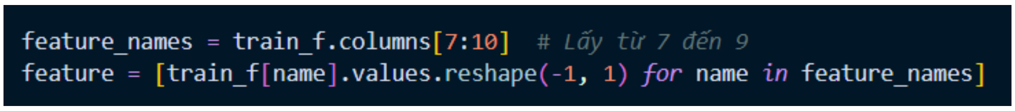
a. Implementation

- Firstly, we read a CSV file named 'train.csv' into a pandas DataFrame called **train_f**. The data in train_f is shuffled randomly using sample(frac=1), which randomizes the order of the rows. The target variable y_df is extracted from train_f by selecting the last column.



```python
train_f = pd.read_csv('train.csv')
train_f = train_f.sample(frac=1)
y_df = train_f.iloc[:, -1]
```

- An instance of the LinearRegression model is created and stored in the variable reg.
- Then, we select a subset of features from train_f by indexing columns 7 to 9 (inclusive). These feature columns are stored in feature_names as a list. For each feature in feature_names, a 2D array (column vector) representation of the feature values is created using .values.reshape(-1, 1). These feature arrays are stored in the feature list.



```python
feature_names = train_f.columns[7:10]  # Lấy từ 7 đến 9
feature = [train_f[name].values.reshape(-1, 1) for name in feature_names]
```

- We perform cross-validation using the cross_val_score function from scikit-learn. It calculates the negative mean absolute error (neg_mean_absolute_error) for each feature individually, using 5-fold cross-validation. The mean absolute error values are stored in the scores list.

```
scores = []
cross = []
for i in range(len(feature)):
    scores.append(cross_val_score(reg, feature[i], y_df, scoring="neg_mean_absolute_error", cv=5))
```

- Next, the code calculates the average mean absolute error (MAE) for each feature and stores it in the MAE list. Additionally, it keeps track of the feature names and their corresponding average MAE in the cross list.

```
# Calculate MAE and append to an array
MAE = []
for i in range(len(scores)):
    MAE.append(np.absolute(scores[i]).mean())
    cross.append([feature_names[i], round(MAE[i], 3)])
```

- A DataFrame named test is created using the cross list, with columns labeled 'Feature' and 'Average MAE'. This DataFrame contains the feature names and their respective average MAE values.

```
test = pd.DataFrame(cross, columns=['Feature', 'Average MAE'])
print(test)
```

- The test DataFrame is printed to display the feature names and their average MAE values.
- The index of the feature with the lowest MAE is identified using np.argmin(MAE). The code then prints the corresponding feature name as the feature with the minimum average MAE.

```
best_index = np.argmin(MAE)
print("Min average MAE is:", feature_names[best_index])
```

- **After that, we have the best feature and we're going to train that feature with the whole training set.**
- We first read two CSV files named **'train.csv' and 'test.csv'** into pandas DataFrames train2 and test2, respectively.
- The feature selected for prediction is extracted from train2 and test2 using feature_names[best_index]. The feature values are reshaped into a 2D array using .values.reshape(-1, 1). These arrays are assigned to X_k_train and X_k_test.

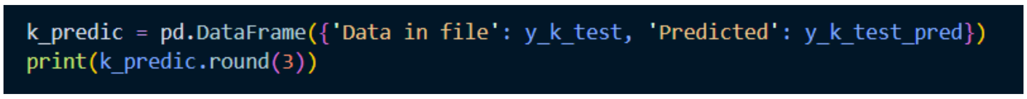
```
X_k_train = train2[feature_names[best_index]].values.reshape(-1, 1)
y_k_train = train2.iloc[:, -1]

X_k_test = test2[feature_names[best_index]].values.reshape(-1, 1)
y_k_test = test2.iloc[:, -1]
```

- The target variables are extracted from train2 and test2 using .iloc[:, -1] and assigned to y_k_train and y_k_test.
- A new instance of the LinearRegression model is created and fitted on the training data (X_k_train and y_k_train). The trained model is stored in reg2.

```
reg2 = LinearRegression().fit(X_k_train, y_k_train)

y_k_test_pred = reg2.predict(X_k_test)
```

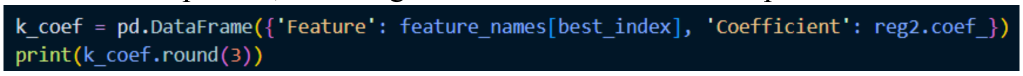
- Predictions are made on the test data (X_k_test) using the trained model, and the predicted values are stored in y_k_test_pred.

- A DataFrame named k_predic is created to display the actual values from the 'test.csv' file (y_k_test) and the corresponding predicted values (y_k_test_pred).



```
k_predic = pd.DataFrame({'Data in file': y_k_test, 'Predicted': y_k_test_pred})
print(k_predic.round(3))
```

- The k_predic DataFrame is printed, rounding the values to three decimal places.
- A DataFrame named k_coef is created to display the coefficient of the selected feature (feature_names[best_index]) in the trained model reg2. The k_coef DataFrame is printed, rounding the values to three decimal places.



```
k_coef = pd.DataFrame({'Feature': feature_names[best_index], 'Coefficient': reg2.coef_})
print(k_coef.round(3))
```

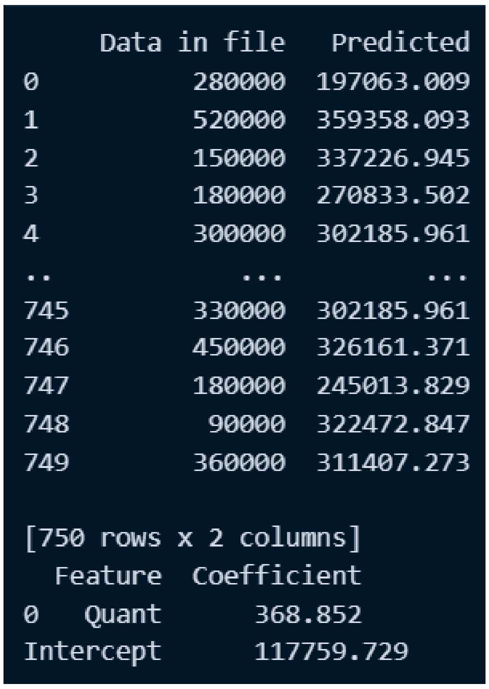
- The k_coef DataFrame is printed, rounding the values to three decimal places.

b. Result

- We aim at training a linear regression model and evaluating its performance using cross-validation. This is the result after performance:

No	Feature	Average MAE
0	English	120764.684
1	Logical	120069.276
2	Quant	117190.854

- After evaluating the performance of the trained model on unseen test data and analyzing the predicted values compared to the actual values. We can see the insights into the coefficient value of the selected feature in the linear regression model.



```
     Data in file    Predicted
0          280000  197063.009
1          520000  359358.093
2          150000  337226.945
3          180000  270833.502
4          300000  302185.961
..            ...         ...
745        330000  302185.961
746        450000  326161.371
747        180000  245013.829
748         90000  322472.847
749        360000  311407.273

[750 rows x 2 columns]
  Feature  Coefficient
0   Quant      368.852
Intercept      117759.729
```

- MAE of the best model generated**: 108814.06**
- Based on this model, we can generate a Regression formula:

$$\text{Salary} = 368.852 * `Quant` + 117759.729$$

c. Reflection & Comment

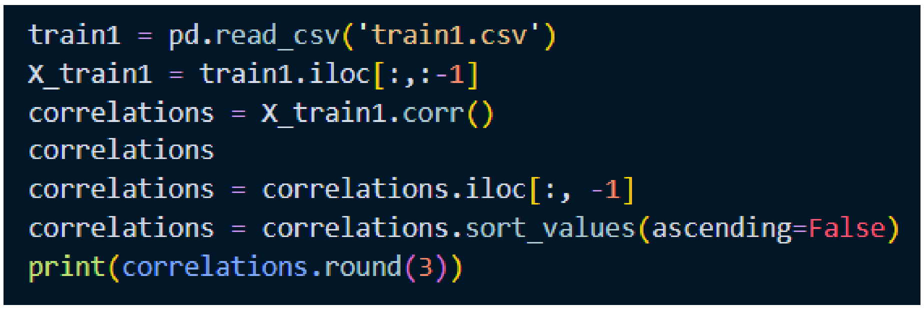
- After considering the 5 features modelling, we go to question whether with a smaller number of features can generate a better feature modelling. This time, we consider features relating to the language which is focused on three famous one: English, Logical and Quant. After performing the cross-validation, Quant has a MAE of 117190.854. This MAE is better than others with nearly 3000 smaller.

- With the better performance, we use Quant as the best feature to re-train with the whole dataset. After this, we have seen a MAE of 108814.06 which is better than the previous 5 features model. Hower, this model is still not good enough with the first 11 features model.
- After the second experiment with small number of features (3 features), we can partly conclude that decreasing the number of features can get a better version of model. However, this model still does not adapt to what we want. And we can raise questions whether in other dimensions, with higher features but these features affecting importantly to salary can get a good model. In the next part, we will test different models with different number of features as well as the importance of these features to salary rate.

4. Build our own models and search for the best model for optimal results

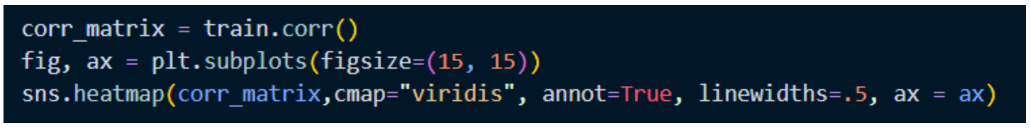
a. Implementation

- Firstly, we calculate the correlation between the features and the target variable.
    - As usual, we read the CSV file 'train.csv' into DataFrame train1.
    - X_train1 is created by selecting all columns of train1 except the last one, which is assumed to be the target variable.
    - The corr() function is applied to X_train1 to calculate the correlation matrix between the features.
    - The correlations of the features with the target variable are extracted by selecting the last column of the correlation matrix using .iloc[:, -1].
    - The correlations are sorted in descending order using sort_values(ascending=False).
    - The sorted correlations are printed, rounding the values to three decimal places.



```
train1 = pd.read_csv('train1.csv')
X_train1 = train1.iloc[:,:-1]
correlations = X_train1.corr()
correlations
correlations = correlations.iloc[:, -1]
correlations = correlations.sort_values(ascending=False)
print(correlations.round(3))
```

- Then, generates a heatmap visualization of the correlation matrix using the seaborn library.



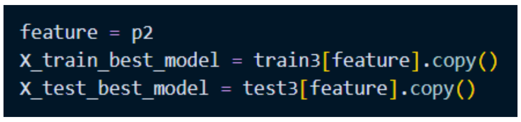
```
corr_matrix = train.corr()
fig, ax = plt.subplots(figsize=(15, 15))
sns.heatmap(corr_matrix,cmap="viridis", annot=True, linewidths=.5, ax = ax)
```

- We define a function **getCorrelation()** that identifies highly correlated columns in a DataFrame based on a given threshold. It then utilizes this function to create different sets of features for training models. Here's a breakdown of the code:
    - The getCorrelation function takes two parameters: df (the DataFrame to analyze) and threshold (the correlation threshold).

    - A set called correlated_cols is initialized to store the names of the highly correlated columns.
    - The correlation matrix is calculated using df.corr().
    - Two nested loops iterate over the upper triangle of the correlation matrix to check for correlations exceeding the given threshold:
        - If the absolute correlation value between two columns is greater than the threshold, the name of the column at index i is added to correlated_cols.
    - The correlated_cols set is printed to display the highly correlated column names.
    - The correlated_cols set is returned from the function.
- Next, we utilize the **getCorrelation()** function to create different sets of features for training models. Here's a breakdown of the subsequent code:
    - Four calls to getCorrelation are made with different threshold values (0.001, 0.2, 0.3, and 0.4) on the X_train DataFrame. The returned sets of highly correlated columns are assigned to variables p1, p2, p3, and p4, respectively.
    - An empty list called models1 is created to store information about the different models.
    - A loop iterates four times, creating different sets of features based on the highly correlated columns for each threshold value:
        - The highly correlated column names from p[i] are converted to a list and assigned to the variable index_set.
        - X_1_train and X_1_test are created by selecting columns from X_train and X_test based on index_set.
        - A list [Model i, X_1_train, X_1_test] is appended to models1, representing the model number and the corresponding training and test data.
- Then, after selecting models, we perform cross-validation on different models using the features and target variable from the 'train1.csv' dataset.
    - Firstly, we read the CSV file 'train.csv' into the DataFrame train_3.
    - The DataFrame train_3 is shuffled randomly using the sample(frac=1) method to ensure the data is randomized.
    - A list called model is created, containing the names of the models ('Model 1', 'Model 2', 'Model 3', 'Model 4').
    - An empty list called ARMSE is created to store the average root mean squared error (RMSE) values for each model.
    - A loop iterates four times, corresponding to the four models:
        - A new copy of the train_3 DataFrame is created and assigned to train_3_1.
        - The feature columns for the current model (p[i]) are selected and assigned to X_train_1.
        - The target variable column is selected and assigned to y_train_1.
        - A new instance of the LinearRegression model is created as rg_3_1.

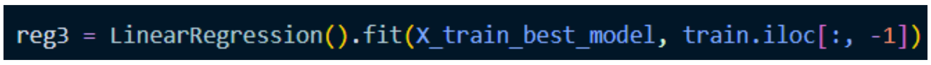
        - Cross-validation is performed using cross_val_score with 5-fold cross-validation (cv=5), using negative mean absolute error (scoring='neg_mean_absolute_error').
        - The absolute values of the cross-validation scores are taken, and the mean is calculated. The resulting mean value is appended to ARMSE.

- Then we use the selected features from the **'train.csv'** dataset to train a linear regression model and make predictions on the **'test.csv'** dataset.
    - The code reads the CSV files 'train.csv' and 'test.csv' into DataFrames train3 and test3, respectively.
    - The variable feature is assigned the value of p2, which represents the selected features for the best model.
    - The selected feature columns are extracted from the train3 DataFrame and assigned to X_train_best_model.
    - The selected feature columns are extracted from the test3 DataFrame and assigned to X_test_best_model.



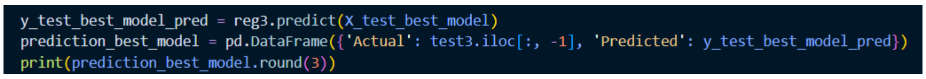
```
feature = p2
X_train_best_model = train3[feature].copy()
X_test_best_model = test3[feature].copy()
```

    - An instance of the LinearRegression model is created as reg3, and it is fitted using X_train_best_model as the training data and the last column of train3 as the target variable.



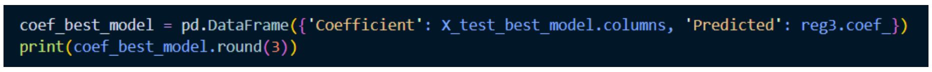
```
reg3 = LinearRegression().fit(X_train_best_model, train.iloc[:, -1])
```

    - Predictions are made on X_test_best_model using the trained model reg3, and the predicted values are stored in y_test_best_model_pred. A DataFrame prediction_best_model is created, combining the actual values from the last column of test3 and the predicted values. The DataFrame is then printed, rounding the values to three decimal places.



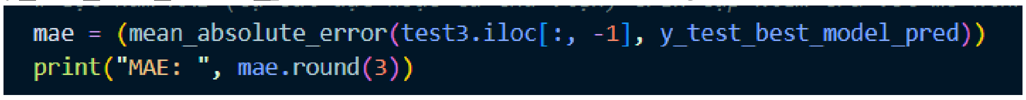
```
y_test_best_model_pred = reg3.predict(X_test_best_model)
prediction_best_model = pd.DataFrame({'Actual': test3.iloc[:, -1], 'Predicted': y_test_best_model_pred})
print(prediction_best_model.round(3))
```

    - Another DataFrame coef_best_model is created, combining the names of the selected features in X_test_best_model and their corresponding coefficients from reg3. The DataFrame is printed, rounding the values to three decimal places.



```
coef_best_model = pd.DataFrame({'Coefficient': X_test_best_model.columns, 'Predicted': reg3.coef_})
print(coef_best_model.round(3))
```

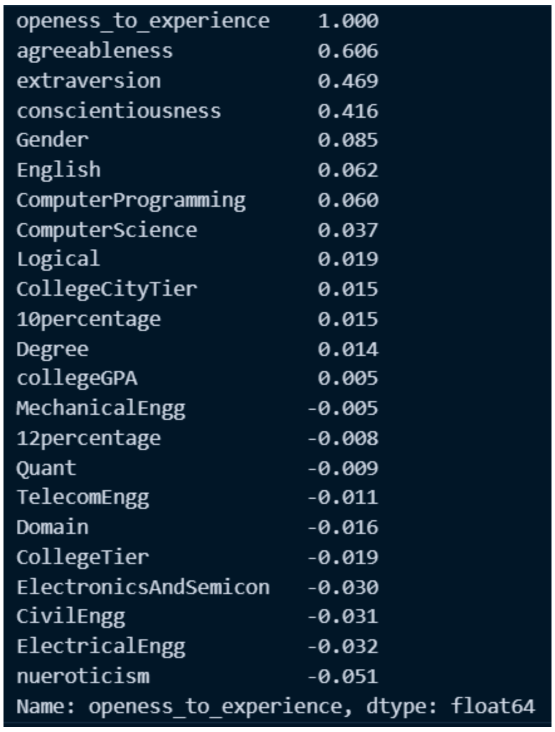
- Finally, we calculate and print the mean absolute error (MAE) between the actual values in the last column of the **'test.csv'** dataset and the predicted values **y_test_best_model_pred** obtained from the linear regression model.



```
mae = (mean_absolute_error(test3.iloc[:, -1], y_test_best_model_pred))
print("MAE: ", mae.round(3))
```

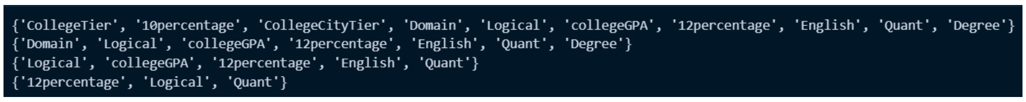
b. Result

- After identifying the correlations between each feature and the target variable in the 'train1.csv' dataset. By sorting the correlations in descending order, we can observe which features have the strongest positive or negative relationship with the target variable.



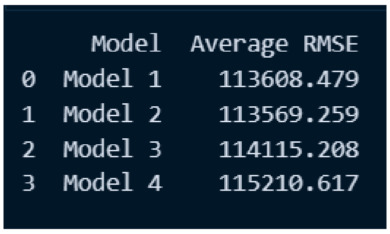
```
openess_to_experience    1.000
agreeableness            0.606
extraversion             0.469
conscientiousness        0.416
Gender                   0.085
English                  0.062
ComputerProgramming      0.060
ComputerScience          0.037
Logical                  0.019
CollegeCityTier          0.015
10percentage             0.015
Degree                   0.014
collegeGPA               0.005
MechanicalEngg          -0.005
12percentage            -0.008
Quant                   -0.009
TelecomEngg             -0.011
Domain                  -0.016
CollegeTier             -0.019
ElectronicsAndSemicon   -0.030
CivilEngg               -0.031
ElectricalEngg          -0.032
nueroticism             -0.051
Name: openess_to_experience, dtype: float64
```

- A heatmap generated for easily visualizing the correlations between the features in the train DataFrame. The color intensity represents the strength of the correlation, with brighter colors indicating stronger positive or negative correlations. The numeric values displayed within each cell provide the actual correlation coefficients.
- To analyze and select features based on their correlation with each other. It creates different sets of features for training models by selecting columns that have correlations above certain thresholds.



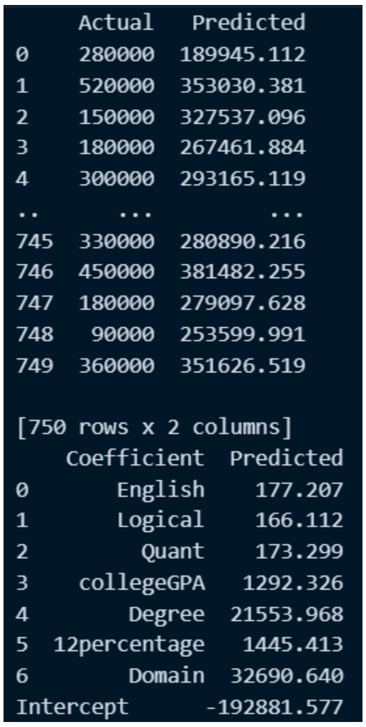
```
{'CollegeTier', '10percentage', 'CollegeCityTier', 'Domain', 'Logical', 'collegeGPA', '12percentage', 'English', 'Quant', 'Degree'}
{'Domain', 'Logical', 'collegeGPA', '12percentage', 'English', 'Quant', 'Degree'}
{'Logical', 'collegeGPA', '12percentage', 'English', 'Quant'}
{'12percentage', 'Logical', 'Quant'}
```

- After using cross-validation for each model, evaluating their performance using the mean absolute error. The average root mean squared error (ARMSE) values for each model are stored in the ARMSE list.



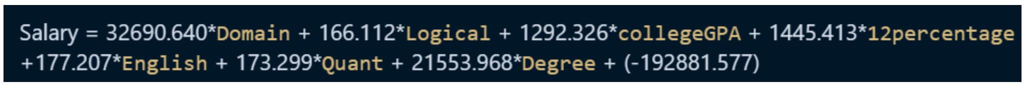
```
     Model  Average RMSE
0  Model 1    113608.479
1  Model 2    113569.259
2  Model 3    114115.208
3  Model 4    115210.617
```

- The predictions and the coefficients of the selected features are then printed for analysis.



```
     Actual   Predicted
0    280000  189945.112
1    520000  353030.381
2    150000  327537.096
3    180000  267461.884
4    300000  293165.119
..      ...         ...
745  330000  280890.216
746  450000  381482.255
747  180000  279097.628
748   90000  253599.991
749  360000  351626.519

[750 rows x 2 columns]
    Coefficient  Predicted
0       English    177.207
1       Logical    166.112
2         Quant    173.299
3     collegeGPA  1292.326
4        Degree  21553.968
5  12percentage   1445.413
6        Domain  32690.640
Intercept      -192881.577
```

- MAE of the best model generated: **105363.758**
- Based on this model, we have the Regression formulation for Salary:



```
Salary = 32690.640*Domain + 166.112*Logical + 1292.326*collegeGPA + 1445.413*12percentage
+177.207*English + 173.299*Quant + 21553.968*Degree + (-192881.577)
```

d. Reflection & Comment

- After viewing the visualization of other features affecting to the salary, we can see that these features divide in 2 main categories increase are: increase the salary (with ratio >= 0) and decrease the salary (with ratio < 0). In order to get the better model, we need to choose features with ratio > 0 include: **openess_to_experience, agreeableness, extraversion, conscientiousness, Gender, English, ComputerProgramming, ComputerScience, Logical, CollegeCityTier, 10percentage, Degree, collegeGPA**. For these figures, openness_to_experience is absolutely significant to Salary.

Correlation	Ratio	Feature
Weak correlation	0 - 0.3	**Logical, CollegeCityTier, 10percentage, Degree, collegeGPA**
Moderate correlation	0.3 – 0.6	**extraversion, conscientiousness, Gender, English, ComputerProgramming, ComputerScience**
Strong correlation	0.6 - 1	**openess_to_experience, agreeableness**

- In order to get the precise feature to build model, we use the **getCorrelation() with a threshold** parameter to define the range of features selected for a specific model.
  - For the first model, we choose all the features that have the positive **correlation** means that when one variable increases, the other variable also increases. (threshold = 0.001)
  - For the next three models, we increase the requirement to experiment whether this model can get a better performance with the threshold are 0.1, 0.2, 0.3 respectively.

- After building these models, we can see that the Average MAE is approximately 113000. There is no any difference between these models, but we can see that model number 2 get the best MAE at 113569.259 with take advantage of 7 features including: **Domain, Logical, collegeGPA, 12percentage, English, Quant, Degree.** As we claimed before, openess_to_experience, agreeableness is a strong correlation to Salary, but we don't use these features in this model. To conclude, the combination of positive correlation can generate a good model. The best model generated at MAE of 105363.758 is absolutely the best model we have generated in the whole project.

VII. REFERENCE

1. https://machinelearningcoban.com/2016/12/28/linearregression/#-dang-cua-linear-regression (For Requirement 1a, 1b, 1c, 1d)
2. https://www.analyticsvidhya.com/blog/2022/02/k-fold-cross-validation-technique-and-its-essentials/ (For Requirement 1b & 1c)
3. https://realpython.com/numpy-scipy-pandas-correlation-python/ (For Requirement 1d)