Faculty of Information Technology
University of Science, VNU HCMC

Project 03
Linear Regression
Report

Prepared by
Thien Huynh Duc - 21127693
August 21, 2023

Table of Contents

1. Project Introduction

1.1. Information

Student

Full name: Huynh Duc Thien
ID: 21127693
Contact: hdthien21@clc.fitus.edu.vn
Course: MTH00057 - Applied Math and Statistics
Class: 21CLC07

Lecturer

Teacher, Nguyen Van Quang Huy
Teacher, Ngo Dinh Hy
Teacher, Tran Ha Son
Teacher, Nguyen Dinh Thuc

Project

Linear Regression

1.2. Linear Regression

This project focuses on exploring linear regression, a statistical technique used for analyzing relationships between variables. The objective is to develop predictive models that accurately estimate a target variable based on independent variables. Through data collection, preprocessing, exploratory data analysis, model development, evaluation, and interpretation, we aim to deepen our understanding of linear regression and its practical applications. By building accurate models, we can gain insights into relationships, make informed decisions, and support various domains with valuable predictive capabilities.

2. Completion levels evaluation

No.	Work	Completion (%)
1	Requirement a	
	• Train the model using the first 11 features	100
	• Express the formula for the regression model	100
	• Report result on test set	100
	• Comments	100

2	Requirement b	
	• Select best personality feature	100
	• Train best_personality_feature_ model and express formula	100
	• Report result on test set	100
	• Comments	100
3	Requirement c	
	• Select best skill feature	100
	• Train best_skill_feature_model and express formula	100
	• Report result on test set	100
	• Comments	100
4	Requirement d	
	• Approaches	100
	• Train my_best_model and express formula	100
	• Report result on test set	100
	• Comments	100

3. Utilities

For this project on linear regression, we utilized several libraries and implemented various functions to facilitate our analysis and modeling. Additionally, some data preprocessing steps will be applied to optimize the model training process for better results.

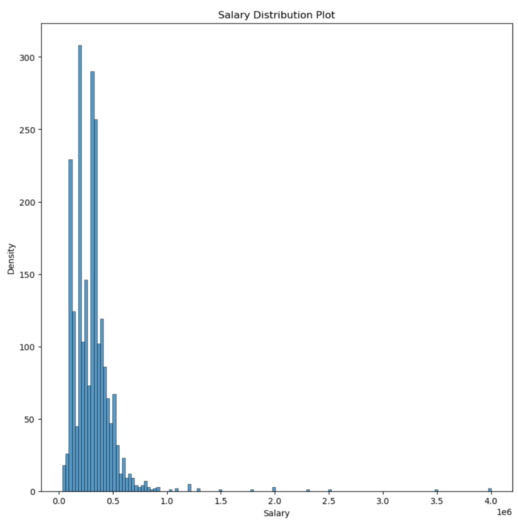
3.1. Libraries and Functions Used

- numpy (imported as np): NumPy is a fundamental library for numerical computing in Python. It provides powerful tools for working with arrays and matrices, along with a collection of mathematical functions. It is commonly used for tasks such as numerical operations, array manipulation, linear algebra, and random number generation.

- pandas (imported as pd): Pandas is a popular data manipulation and analysis library in Python. It provides data structures like DataFrames, which are efficient for handling structured data. Pandas offers functions for data cleaning, transformation, merging, slicing, and aggregation. It is widely used for data preprocessing and exploratory data analysis.
- seaborn (imported as sns): Seaborn is a data visualization library built on top of matplotlib. It provides a high-level interface for creating visually appealing and informative statistical graphics. Seaborn simplifies the process of creating common types of plots, such as scatter plots, bar plots, box plots, and heatmaps. It also offers additional features like color palettes, themes, and statistical estimation.
- matplotlib.pyplot (imported as plt): Matplotlib is a widely used plotting library in Python. The pyplot module provides a collection of functions that allow you to create various types of plots, customize their appearance, and add labels, titles, and legends. It is a versatile tool for visualizing data and generating publication-quality figures.
- sklearn.linear_model.LinearRegression: LinearRegression is a class from the scikit-learn library that implements linear regression models. Linear regression is a supervised learning algorithm used for predicting a continuous target variable based on one or more input features. The LinearRegression class provides methods for fitting the model to training data, making predictions, and assessing model performance.
- sklearn.metrics.mean_absolute_error: mean_absolute_error is a function from scikit-learn's metrics module. It calculates the mean absolute error (MAE) between the predicted values and the true values of the target variable. MAE is a common evaluation metric for regression problems, measuring the average absolute difference between the predicted and actual values. Lower MAE indicates better model performance.
- sklearn.model_selection.cross_val_score: cross_val_score is a function from scikit-learn's model_selection module. It performs cross-validation, which is a technique for evaluating the performance of a model by splitting the data into multiple subsets, training the model on different subsets, and computing the evaluation metric (such as MAE) on each subset. cross_val_score returns an array of scores, allowing you to assess the model's performance across different folds of the data.

3.2. Data preprocessing

Data of the 'train.csv' and 'test.csv' are read, and then stored in 'train' and 'test' variables.



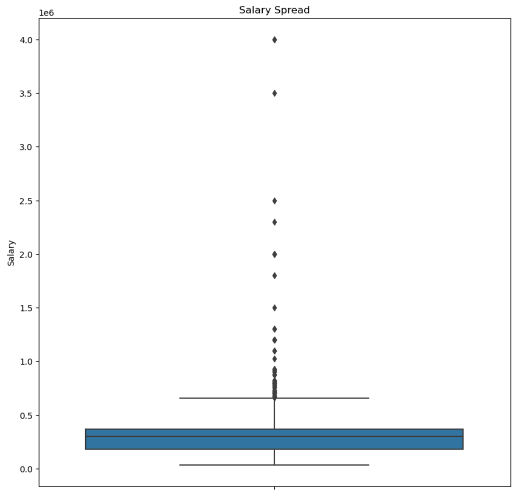


Figure 1: Scatter plot and histogram of data before processing (detail)

Now we will perform some operations to streamline the data, serving the model training process:

- **Remove rows with Nan**

```
train = train.dropna()
```

In pandas, the dropna() function is used to drop rows or columns that have missing values. By default, it removes any row that contains at least one missing value. In this case, the code removes rows with missing values from the train DataFrame and assigns the modified DataFrame back to the variable train. After executing this code, the DataFrame train will no longer contain any rows with missing values.

- **Remove outliers**

```
median = train['Salary'].median()
mad = np.median(np.abs(train['Salary'] - median))
modified_z_scores = 0.6745 * (train['Salary'] - median) / mad

threshold = 3.5

outliers = train[modified_z_scores.abs() > threshold]

train = train[modified_z_scores.abs() <= threshold]
```

The provided code utilizes the modified Z-score method to detect and eliminate outliers. Outliers refer to observations that deviate significantly from the majority of the data and can adversely affect analysis or modeling. The modified Z-score method identifies outliers by measuring the number of median absolute deviations an observation that deviates from the median. By utilizing this code, you can identify and remove outliers from the 'Salary' column in the 'train' DataFrame, resulting in a more precise and dependable analysis or modeling procedure.

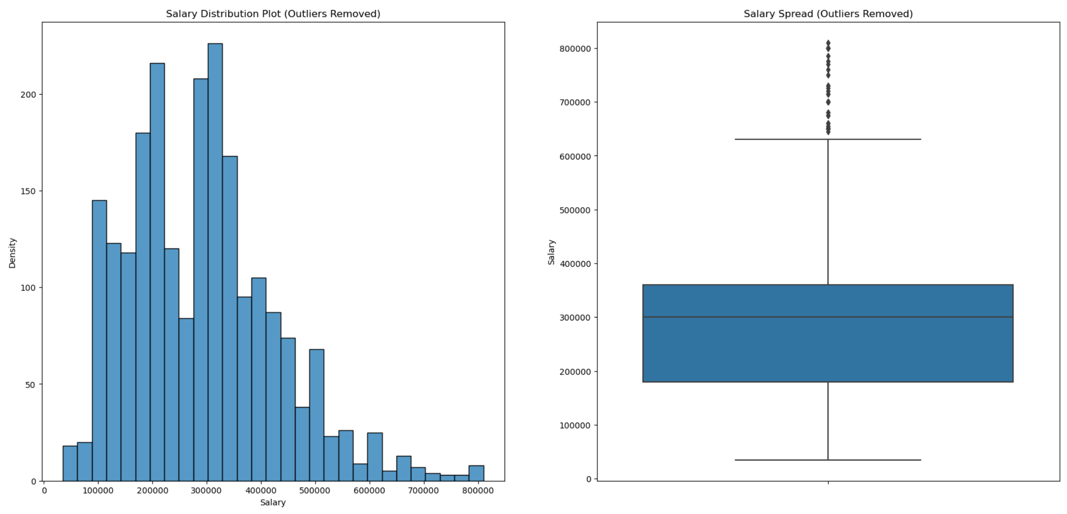


Figure 2: Scatter plot and histogram of data after processing (detail)

4. Requirement a

```
selected_features = ['Gender', '10percentage', '12percentage',
'CollegeTier', 'Degree', 'collegeGPA', 'CollegeCityTier', 'English',
'Logical', 'Quant', 'Domain']

X_train_1a = train[selected_features]
y_train_1a = train['Salary']

X_test_1a = test[selected_features]
y_test_1a = test['Salary']
```

Before starting the model training, the necessary data of the first 11 defined features and the 'Salary' label are extracted from the previously read train and test dataframes. The extracted data is then stored in 'X_train_1a', 'y_train_1a', 'X_test_1a', and 'y_test_1a'.

- **Train the model using the first 11 features**

```
model = LinearRegression()
model.fit(X_train_1a, y_train_1a)
```

In this code segment, the LinearRegression() function from the scikit-learn library is utilized to train a linear regression model.

- model = LinearRegression(): An instance of the LinearRegression() class is created and assigned to the variable model. This object represents the linear regression model.
- model.fit(X_train_1a, y_train_1a): The fit() method of the model object is called with the standardized training features (X_train_1a) and the corresponding target variable (y_train_1a). This method fits the linear regression model to the training data by estimating the coefficients that minimize the sum of squared differences between the predicted and actual target values. The model learns the relationships between the standardized features and the target variable.

After executing this code, the model object contains a trained linear regression model that can be used to make predictions on new data or analyze the relationships between the selected features and the target variable.

- **Express the formula for the regression model**

```
intercept = model.intercept_
coefficients = model.coef_

formula = f"Salary = {intercept:.3f}"

for feature, coefficient in zip(selected_features, coefficients):
    formula += f" + {coefficient:.3f} * {feature}"

print("Regression model formula:\n")
print(formula)
```

We retrieve the intercept and coefficients of a trained linear regression model, then constructs the formula for the regression model by combining the intercept and the contribution of each feature. The formula represents the relationship between the selected features and the predicted target variable. The resulting formula is printed, providing insights into how each feature affects the predicted target variable.

- **Report result on test set**

```
y_test_pred = model.predict(X_test_1a)

mae = mean_absolute_error(y_test_1a, y_test_pred)
print("Mean Absolute Error (MAE):", mae)
```

The mean_absolute_error function, a metric commonly used in regression tasks to evaluate the accuracy of a model's predictions. It measures the average absolute difference between the predicted and actual values of the target variable.

By using the mean_absolute_error function, we can quantitatively assess the performance of a regression model by measuring the average absolute deviation of its predictions from the true values.

- **Comments**

  Model Formula: The regression model formula indicates how each feature contributes to the predicted salary. The intercept is approximately -15,687.505. The coefficients for the features show the estimated impact of each feature on the salary prediction. For example, a higher value in '10percentage' or '12percentage' tends to positively influence the predicted salary, while a higher value in 'CollegeTier' or 'CollegeCityTier' has a negative impact.

  Feature Importance: From the coefficients, we can observe that features like 'Gender', 'CollegeTier', 'CollegeCityTier' have relatively larger coefficients, due to their categorical nature. Features such as 'Degree', 'collegeGPA', 'English', 'Logical', 'Quant', and 'Domain' also contribute to the salary prediction but with smaller coefficients.

  Model Performance: The Mean Absolute Error (MAE) is a measure of the model's prediction accuracy. In this case, the calculated MAE is approximately 101172.87638093019. The MAE represents the average absolute difference between the predicted and actual salaries. A lower MAE indicates better accuracy, so the model may have some room for improvement in accurately predicting salaries based on the given features.

5. Requirement b

Firstly, we need to declare variables to store the best feature and its corresponding MAE value. Additionally, we select features for training the model and shuffle the data for training purposes.

```
best_feature = None
best_mae = None

train_shuffled = train.sample(frac=1, random_state=42)

selected_features = ['conscientiousness', 'agreeableness', 'extraversion',
'nueroticism', 'openess_to_experience']
```

- **Select the best feature**

```
for ft in selected_features:
    X_train_1b = train_shuffled[[ft]]
    y_train_1b = train_shuffled['Salary']

    model = LinearRegression()

    mae_scores = -cross_val_score(model, X_train_1b, y_train_1b,
scoring='neg_mean_absolute_error', cv=5)
    cur_mae = mae_scores.mean()

    if best_mae is None or cur_mae < best_mae:
        best_feature = ft
        best_mae = avg_mae

    print(f"MAE for feature '{ft}': {cur_mae}")

print(f"\nBest Feature: {best_feature}")
print(f"MAE for Best Feature: {best_mae}")
```

In the above code, the program evaluates the performance of each feature using cross-validation and selects the feature with the lowest mean absolute error (MAE) as the best feature in the selected_features.

The cross_val_score function is used to perform cross-validation with the LinearRegression model. It calculates the negative mean absolute error (neg_mean_absolute_error) for the current feature using 5-fold cross-validation (k=5), the corresponding MAE value of the feature is then printed out for comparison. After the loop, the best_feature and best_mae variables hold the feature with the lowest MAE across all features. The best feature has been selected for future training.

- **Train best_personality_feature_model and express formula**

```
X_train_best = train[[best_feature]]
X_test_best = test[[best_feature]]

best_personality_feature_model = LinearRegression()
best_personality_feature_model.fit(X_train_best, y_train)
```

```
intercept_best = best_personality_feature_model.intercept_
coefficient_best = best_personality_feature_model.coef_[0]

formula_best = f"Salary = {intercept_best:.3f} + {coefficient_best:.3f} *
{best_feature}"
```

```
print("\nRegression model formula for the best feature:\n")
print(formula_best)
```

From the selected best_feature, new trained model '*best_personality_feature_model*' give out the corresponding formula of the regression model, showing the relationship between the best_feature and the salary.

- **Report result on test set**

```
y_test_pred_best = best_personality_feature_model.predict(X_test_best)
mae_best = mean_absolute_error(y_test, y_test_pred_best)
print("\nMean Absolute Error (MAE) for the best feature:", mae_best)
```

By using the mean_absolute_error function, we can quantitatively assess the performance of a regression model by measuring the average absolute deviation of its predictions from the true values.

- **Comments**

The model was trained and evaluated using five different features: conscientiousness, agreeableness, extraversion, nueroticism, and openess_ to_experience. Each feature was individually evaluated using mean absolute error (MAE) as the performance metric. The lower the MAE, the better the model's prediction accuracy.

  - This formula indicates that for each unit increase in 'nueroticism', the predicted salary is estimated to decrease by approximately 10111.747. The intercept term of 289134.240 represents the estimated salary when the 'nueroticism' feature is zero.
  - Based on the provided MAE values, the feature 'nueroticism' achieved the lowest MAE of 107,719.61023647434, indicating that it performed the best among the features in predicting the salary.
  - The MAE values for the other features are relatively close, ranging from 108,084.87297639923 to 108,494.47210710093. This suggests that these features, namely 'conscientiousness', 'agreeableness', 'extraversion', and 'openess_to_experience', have similar predictive power for the salary.
  - It's important to note that the MAE measures the average absolute difference between the predicted and actual salary values. A lower MAE indicates better prediction accuracy. In this case, the difference in MAE between the best feature ('nueroticism') and the other features is relatively small.

The hypothesis is that nueroticism may be related to factors such as personality, psychology, or work attitude of an individual, and these factors can influence the salary they receive. This implies that individuals with higher levels of nueroticism are more likely to receive lower salaries due to a range of psychological factors such as stress, anxiety, or difficulties in their job.

6. Requirement c

Firstly, we select features for training the model.

```
selected_features = ['English', 'Logical', 'Quant']
```

- **Select the best feature**

```
for ft in selected_features:
    X_train_1c = train_shuffled[[ft]]
    y_train_1c = train_shuffled['Salary']

    model = LinearRegression()

    mae_scores = -cross_val_score(model, X_train_1c, y_train_1c,
scoring='neg_mean_absolute_error', cv=5)
    cur_mae = mae_scores.mean()

    if best_mae is None or cur_mae < best_mae:
        best_feature = ft
        best_mae = avg_mae

    print(f"MAE for feature '{ft}': {cur_mae}")

print(f"\nBest Feature: {best_feature}")
print(f"MAE for Best Feature: {best_mae}")
```

In the above code, the program evaluates the performance of each feature using cross-validation and selects the feature with the lowest mean absolute error (MAE) as the best feature in the selected_features.

The cross_val_score function is used to perform cross-validation with the LinearRegression model. It calculates the negative mean absolute error (neg_mean_absolute_error) for the current feature using 5-fold cross-validation (k=5), the corresponding MAE value of the feature is then printed out for comparison. After the loop, the best_feature and best_mae variables hold the feature with the lowest MAE across all features. The best feature has been selected for future training.

- **Train best_skill_feature_model and express formula**

```
X_train_best = train[[best_feature]]
X_test_best = test[[best_feature]]

best_skill_feature_model = LinearRegression()
best_skill_feature_model.fit(X_train_best, y_train)
```

```
intercept_best = best_skill_feature_model.intercept_
coefficient_best = best_skill_feature_model.coef_[0]

formula_best = f"Salary = {intercept_best:.3f} + {coefficient_best:.3f} *
{best_feature}"
print("\nRegression model formula for the best feature:\n")
print(formula_best)
```

From the selected best_feature, new trained model '*best_skill_feature_model*' give out the corresponding formula of the regression model, showing the relationship between the best_feature and the salary.

- **Report result on test set**

```
y_test_pred_best = best_personality_feature_model.predict(X_test_best)
mae_best = mean_absolute_error(y_test, y_test_pred_best)
print("\nMean Absolute Error (MAE) for the best feature:", mae_best)
```

By using the mean_absolute_error function, we can quantitatively assess the performance of a regression model by measuring the average absolute deviation of its predictions from the true values.

- **Comments**

The model was trained and evaluated using three different features: English, Logical, and Quant. Each feature was individually evaluated using mean absolute error (MAE) as the performance metric. The lower the MAE, the better the model's prediction accuracy.

- Among the features, Quant achieved the lowest MAE of 117,353.84, indicating that it performed the best in predicting the salary.
- The regression model formula for the best feature, Quant, suggests that for each unit increase in Quant, the predicted salary increases by approximately 368.85. The intercept of the model is 117,759.73.
- Comparing the MAE values, the difference between the best feature (Quant) and the other features is noticeable. English has an MAE of 120,728.60, and Logical has an MAE of 119,932.50.

The Quant feature serves as a proxy for a person's analytical and problem-solving skills, which are highly valued in many industries. Individuals with strong quantitative abilities are likely to excel in roles that require numerical analysis, financial modeling, statistical analysis, or technical expertise. Employers may reward individuals with higher salaries for their quantitative abilities due to the perceived value and demand for these skills in the job market.

7. Requirement d

- **Approaches**

  For the given requirement d, there are several approaches mentioned in this solution.

  - Option 1: Select top 10 features

```python
all_features = ['Gender', '10percentage', '12percentage',
'CollegeTier', 'Degree', 'collegeGPA', 'CollegeCityTier',
'English', 'Logical', 'Quant', 'Domain', 'ComputerProgramming',
'ElectronicsAndSemicon', 'ComputerScience', 'MechanicalEngg',
'ElectricalEngg', 'TelecomEngg', 'CivilEngg', 'conscientiousness',
'agreeableness', 'extraversion', 'nueroticism',
'openess_to_experience']
mae_features = []

for feature in all_features:
    X_train = train[[feature]]
    y_train = train['Salary']

    model = LinearRegression()
    mae_scores = -cross_val_score(model, X_train, y_train,
scoring='neg_mean_absolute_error', cv=5)
    mae = mae_scores.mean()

    mae_features.append((feature, mae))

mae_features.sort(key=lambda x: x[1])
top_features = [feature for feature, _ in mae_features[:10]]

print("Top 10 Features:")
for feature, mae in mae_features[:10]:
    print(f"{feature} -- {mae}")
```

From the given 23 features, proceed selecting out the top 10 features that give the best performance (top 10 features with smallest MAE).

At each iteration, we create a LinearRegression model and calculate the MAE value with k-fold Cross Validation method. From the sorted MAE values, we select the top 10 features that exhibit the best contribution to the target value 'Salary'. Each selected feature will be printed along with its corresponding MAE value.

```
X_train = train[top_features]
y_train = train['Salary']
model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_train)
mae = mean_absolute_error(y_train, y_pred)
print(f'\nMAE for dataframe with top 10 features: {mae}')
```

Afterward, we proceed to train the model using the selected features and evaluate its output results and the corresponding MAE value.

- Option 2: Select suitable features from visualized data

```
for feature in all_features:
    plt.scatter(train[feature], train['Salary'])
    plt.xlabel(feature)
    plt.ylabel('Salary')
    plt.title(f'Scatter plot of {feature} vs Salary')
    plt.show()
```

By creating these scatter plots, the code visually explores the relationships between each feature and the 'Salary' variable. This can help identify any patterns, trends, or potential correlations between the features and the target variable, providing insights into the data and potentially guiding feature selection or further analysis.

From the charts, we proceed to investigate and select to exclude some features that have a negative impact on model training due to the scattered points not tending to form a line with an acceptable deviation. However, the selected features include: '10percentage', '12percentage', 'collegeGPA', 'English', 'Logical', 'Quant', 'conscientiousness', 'agreeableness', 'extraversion', 'nueroticism', 'openess_to_experience'. From here, we proceed to train the model again and obtain results with a relatively stable MAE value.

- Option 3: Generate new features

```
modified_train = train.copy()

modified_train['engineering_skills'] =
modified_train[['ComputerProgramming', 'ElectronicsAndSemicon',
'ComputerScience', 'MechanicalEngg', 'ElectricalEngg',
'TelecomEngg', 'CivilEngg']].mean(axis=1)

modified_train['collegeGPA_squared'] = modified_train['collegeGPA']
** 2

modified_train['collegeGPA_cube'] = modified_train['collegeGPA'] **
3

modified_train['interaction_percentage'] =
modified_train['10percentage'] * modified_train['12percentage']

modified_train['10percentage_bins'] =
pd.cut(modified_train['10percentage'], bins=[0, 50, 75, 100],
labels=['low', 'medium', 'high'])

modified_train['12percentage_bins'] =
pd.cut(modified_train['12percentage'], bins=[0, 50, 75, 100],
labels=['low', 'medium', 'high'])
```

+ engineering_skills: This feature calculates the average of skills in various engineering fields such as ComputerProgramming, ElectronicsAndSemicon, ComputerScience, MechanicalEngg, ElectricalEngg, TelecomEngg, and CivilEngg. It represents an overall measure of engineering skills.
+ collegeGPA_squared: This feature calculates the square of the 'collegeGPA' feature. Squaring the GPA can capture non-linear relationships or potential quadratic effects.
+ collegeGPA_cube: This feature calculates the cube of the 'collegeGPA' feature. Cubing the GPA can capture non-linear relationships or potential cubic effects.
+ interaction_percentage: This feature represents the interaction between the '10percentage' and '12percentage' features. It calculates the product of these two features, which can capture potential interaction effects between the percentages.
+ 10percentage_bins: This feature discretizes the '10percentage' feature into three bins: 'low', 'medium', and 'high'. It uses the pd.cut() function to divide the values into bins based on the specified ranges.

+ 12percentage_bins: This feature discretizes the '12percentage' feature into three bins: 'low', 'medium', and 'high'. It also uses the pd.cut() function to divide the values into bins based on the specified ranges.

By creating these new features, the code incorporates additional information or transformations of existing features that can potentially improve the model's ability to capture complex relationships and patterns in the data. These features provide more insights and flexibility when training the model and enhance its predictive performance.

- **Train my_best_model and express formula**

```
my_best_model = LinearRegression()
my_best_model.fit(X_train_encoded, y_train)

selected_features = list(X_train_encoded.columns)
intercept = my_best_model.intercept_
coefficients = my_best_model.coef_

formula = f"Salary = {intercept:.3f}"

for feature, coefficient in zip(selected_features, coefficients):
    formula += f" + {coefficient:.3f} * {feature}"

print("Regression model formula:\n")
print(formula)
```

After selecting the best model approaches (option 3), we use the previously created features to train the 'my_best_model' model. Simultaneously, we calculate the corresponding formula of the regression model, showing the relationship between the features and the salary.

- **Report result on test set**

```
modified_test = test.copy()

modified_test['engineering_skills'] =
modified_test[['ComputerProgramming', 'ElectronicsAndSemicon',
'ComputerScience', 'MechanicalEngg', 'ElectricalEngg', 'TelecomEngg',
'CivilEngg']].mean(axis=1)
modified_test['collegeGPA_squared'] = modified_test['collegeGPA'] ** 2
modified_test['collegeGPA_cube'] = modified_test['collegeGPA'] ** 3
modified_test['interaction_percentage'] = modified_test['10percentage'] *
modified_test['12percentage']
```

```
modified_test['10percentage_bins'] =
pd.cut(modified_test['10percentage'], bins=[0, 50, 75, 100],
labels=['low', 'medium', 'high'])
modified_test['12percentage_bins'] =
pd.cut(modified_test['12percentage'], bins=[0, 50, 75, 100],
labels=['low', 'medium', 'high'])

X_test = modified_test.drop('Salary', axis=1)
X_test_encoded = pd.get_dummies(X_test, columns=['10percentage_bins',
'12percentage_bins'])

y_test_pred = my_best_model.predict(X_test_encoded)
mae = mean_absolute_error(y_test, y_test_pred)
print("\nMean Absolute Error (MAE) for the test dataset with
my_best_model:", mae)
```

The test dataset is modified in this code to align it with the feature engineering steps performed during the training phase. It is necessary to apply the same modifications to the test dataset as those applied to the training dataset to ensure consistency and fairness in evaluating the model's performance.

By creating new features or applying transformations to the test dataset that were previously done during training, we ensure that the model receives the same type of input it was trained on.

The purpose of this code is to assess the performance of the trained 'my_best_model' on unseen data (the test dataset) by calculating the MAE. It provides insights into how well the model generalizes to new data and helps evaluate its effectiveness in predicting the target variable.

- **Comments**
  - Model Formula: The regression model formula represents the relationship between the features and the predicted salary. Each coefficient in the formula indicates the impact of the corresponding feature on the predicted salary. Positive coefficients suggest a positive relationship, while negative coefficients indicate a negative relationship. The magnitude of the coefficients represents the strength of the impact. It appears that your model includes a wide range of features from different domains, such as education, personality traits, engineering skills, and more.
  - MAE for the Dataframe with Additional Features: The MAE for the dataframe with additional features is 93229.66136111609. This value represents the average absolute difference between the predicted salaries and the actual salaries in the training dataset. A lower MAE indicates better prediction accuracy, suggesting that the model, when trained on the given features, can capture a significant portion of the salary variance.

- MAE for the Test Dataset with my_best_model: The MAE for the test dataset with your best model is 98133.16432320676. This value represents the average absolute difference between the predicted salaries and the actual salaries in the test dataset. Comparing this MAE with the one for the training dataset, it seems that the model's predictive performance is slightly degraded when applied to unseen data. This could be due to overfitting, differences in the data distribution between the training and test datasets, or other factors.

Overall, the model seems to capture some meaningful relationships between the features and the salary prediction, as indicated by the lower MAE for the training dataset. However, the relatively higher MAE for the test dataset suggests that the model might not generalize well to unseen data. It's recommended to further evaluate the model using other metrics, such as RMSE or R-squared, and consider techniques like regularization, feature selection, or cross-validation to improve its performance and avoid overfitting. Additionally, thorough data analysis, feature engineering, and domain knowledge can help refine the model and potentially enhance its predictive capabilities.

Reference

[1] Visualize the relationship between features and salary (requirement d) : matplotlib.pyplot.scatter (link)

[2] K-fold cross-validation materials: Giới thiệu về k-fold cross-validation, *Trí tuệ nhân tạo*: (link)

[3] Enhance the accuracy of a Regression Model (requirement d): How to improve the accuracy of a Regression Model (link)

[4] Linear regression materials: Everything you need to Know about Linear Regression! (link)

–The end–