**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM**
**KHOA CÔNG NGHỆ THÔNG TIN**

---


# TOÁN ỨNG DỤNG VÀ THỐNG KÊ
# ĐỒ ÁN 3
# LINEAR REGRESSION

**Giảng viên hướng dẫn:**

Vũ Quốc Hoàng
Nguyễn Văn Quang Huy
Ngô Đình Hy
Phan Thị Phương Uyên

**Sinh viên thực hiện:**

21127232 – Nguyễn Thanh Bình

Thành phố Hồ Chí Minh, Tháng 08/2023

# MỤC LỤC

# I. Khái quát đồ án và ý tưởng thực hiện

- Linear Regression (Hồi quy tuyến tính) là một trong những thuật toán cơ bản và phổ biến nhất của Supervised Learning (Học có giám sát), trong đó đầu ra dự đoán là liên tục.
- Thuật toán này thích hợp để dự đoán các giá trị đầu ra là các đại lượng liên tục như doanh số hay giá cả thay vì cố gắng phân loại chúng thành các đại lượng rời rạc như màu sắc và chất liệu của quần áo, hay xác định đối tượng trong một bức ảnh là mèo hay chó, …[4]
- Phương pháp OLS (Ordinary Least Squares) là một phương pháp ước lượng tham số trong mô hình hồi quy tuyến tính. Phương pháp này sẽ lựa chọn các hệ số hồi quy alpha và beta sao cho bình phương sai số của mô hình ước lượng là nhỏ nhất. [7]
- Đồ án này được xây dựng dựa trên kiến thức về hồi quy tuyến tính và phương pháp OLS. Từ một bộ dữ liệu cho trước được thu thập tại Ấn Độ, nơi có hơn 6000 cơ sở đào tạo kỹ thuật công nghệ với khoảng 2,9 triệu sinh viên đang học tập, mục tiêu đồ án là tìm hiểu các yếu tố quyết định mức lương và việc làm của các kỹ sư ngay sau khi tốt nghiệp. [6]

# II. Mô tả các hàm và thư viện được sử dụng

## 1. Các thư viện được sử dụng

Trong đồ án này, ta sẽ sử dụng các thư viện bao gồm:

- numpy: sử dụng cho việc tính toán và xử lí ma trận. Một số hàm sẽ được sử dụng của thư viện numpy như *np.array(), np.mean(), np.matmul(), …*
- pandas: sử dụng cho việc đọc các file csv input (gồm hai file là train.csv và test.csv) vào các dataframes. Một số hàm và thuộc tính của thư viện pandas được sử dụng như *iloc, corr(), …*
- matplotlib và seaborn: sử dụng cho việc trực quan hóa dữ liệu, vẽ biểu đồ tương quan. Một số hàm được sử dụng như: *plt.subplots(), sns.heatmap(), …*
- copy: sử dụng hàm *deepcopy()* để copy dữ liệu, tạo một bản sao độc lập với bản cũ.

## 2. Mô tả các hàm

Trong đồ án này, để dễ dàng trong việc quản lí, tận dụng tối đa việc tái sử dụng code, ta cài đặt 2 class chính bao gồm: *OLSLinearRegression* dùng cho việc tính toán mô hình theo phương pháp OLS [5] và *KFoldCrossValidation* dùng cho việc thực hiện phương pháp K-fold Cross Validation.

### 2.1 Hàm OLSLinearRegression.fit() [5]

- Đầu vào: *X*: ma trận dataframe, *y*: vector dataframe
- Đầu ra: Đối tượng hiện tại được khởi tạo từ class *OLSLinearRegression*.
- Các bước thực hiện:
  - Sử dụng hàm *addOnesColumn()* để thêm cột 1 vào phía trước ma trận *X*.
  - Tính phép nhân ma trận $X^T X$ bằng hàm *np.matmul()*.
  - Tính ma trận nghịch đảo của ma trận $X^T X$ bằng cách sử dụng hàm *np.linalg.inv()*. Kết quả thu được là $(X^T X)^{-1}$
  - Sau đó, tiếp tục nhân ma trận kết quả với $X^T$ thu được $(X^T X)^{-1} X^T$

- Cuối cùng nhân tiếp với vector *y* để thu được *w* (tham số của mô hình).

**2.2 Hàm OLSLinearRegression.get_params()** [5]

- Đầu vào: None
- Đầu ra: Hàm này trả về thuộc tính *w* của đối tượng *OLSLinearRegression*, là tham số của mô hình có được sau khi thực hiện quá trình data fitting.

**2.3 Hàm OLSLinearRegression.predict()** [5]

- Đầu vào: *X*: ma trận dataframe
- Đầu ra: ma trận *y* dự đoán sau khi tính tích vô hướng của *X* với *w*
- Các bước thực hiện:
  - Sử dụng *np.array()* để chuyển *X* thành numpy array sau đó dùng hàm *addOnesColumn()* để thêm cột 1 vào phía trước ma trận *X*.
  - Tính tích vô hướng của *X* với *w* bằng cách dùng hàm *ravel() và np.sum()*.

**2.4 Hàm KFoldCrossValidation.__init__()**

- Đầu vào:
  - *k*: số lượng fold được chia
  - *folds*: mảng các fold, mỗi fold chứa các cột là bộ các đặc trưng không trùng nhau của các mô hình đang xét và giá trị mục tiêu (kết quả đầu ra của hàm *split_data_into_k_folds*)
  - *models*: mảng các model đang xét bằng phương pháp K-fold Cross Validation, mỗi model là một mảng chứa tên các đặc trưng.
- Đầu ra: None
- Các bước thực hiện:
  - Khởi tạo các thuộc tính *k, folds, models* để lưu trữ các tham số đầu vào cho đối tượng của class *KFoldCrossValidation.*
  - Thêm một thuộc tính *model_names* để lưu trữ tên của các models. Ví dụ như model chứa 2 đặc trưng tên *A* và *B* thì tên của model sẽ là "*A,B*". Việc đặt tên cho model dựa trên các đặc trưng được thực hiện bằng việc sử dụng hàm *get_model_name()*, được mô tả rõ hơn bên dưới.
  - Thêm một thuộc tính *mae_values* kiểu *dictionary* để lưu trữ giá trị MAE của từng model với *key* là tên của model và *value* là giá trị MAE tương ứng (khởi tạo ban đầu là 0)

**2.5 Hàm KFoldCrossValidation.fit()**

- Đầu vào: None
- Đầu ra: None
- Các bước thực hiện:
  - Sử dụng vòng lặp để duyệt qua từng fold trong thuộc tính *folds*, sau đó ứng với mỗi fold lặp qua từng model trong thuộc tính *models.*
  - Với từng model, ta sẽ thực hiện quá trình xây dựng mô hình hồi quy từ các đặc trưng của model đó và giá trị mục tiêu lấy ra từ fold đang xét bằng cách sử dụng hàm *fit()* của class *OLSLinearRegression.*

- Sau đó tiếp tục sử dụng hàm *predict()* của class đó để dự đoán giá trị mục tiêu trên chính tập dữ liệu train của fold hiện tại. Từ đó tính toán giá trị MAE và cộng dồn vào MAE của mô hình hiện tại trong thuộc tính *mae_values.*

**2.6 Hàm KFoldCrossValidation.calc_avg_mae()**

- Đầu vào: None.
- Đầu ra: None
- Các bước thực hiện: Hàm này dùng để tính giá trị trung bình của các giá trị MAE bằng cách lấy tổng các MAE đã được tính toán và lưu trữ trong thuộc tính *mae_values* chia cho *k* sau đó cập nhập lại cho *mae_values.*

**2.7 Hàm KFoldCrossValidation.get_mae_values()**

- Đầu vào: None.
- Đầu ra: ma trận dataframe gồm các cột là tên Model và giá trị MAE, các dòng là tên của từng model cụ thể và giá trị MAE tương ứng của mô hình đó.
- Các bước thực hiện:
  - Tính giá trị trung bình MAE của từng mô hình bằng hàm *calc_avg_mae()*
  - Tạo một list rỗng *mae_results*
  - Duyệt qua từng tên của model nằm trong thuộc tính model_names và thêm tên model kèm theo giá trị MAE trung bình tương ứng vào trong *mae_results*
  - Dùng hàm *pd.DataFrame()* để chuyển đổi *mae_results* thành dataframe với các cột có nhãn là 'Model' và 'MAE', sau đó trả về bên ngoài.

**2.8 Hàm KFoldCrossValidation.get_best_model()** [8]

- Đầu vào: None.
- Đầu ra: Giá trị MAE nhỏ nhất và tên của mô hình tương ứng.
- Các bước thực hiện:
  - Sử dụng hàm *min()* và *lambda* của python để có thể tìm kiếm mô hình có giá trị MAE nhỏ nhất trong thuộc tính *mae_values*.

**2.9 Hàm addOnesColumn()**

- Đầu vào: *X*: ma trận dataframe.
- Đầu ra: ma trận *X* sau khi đã được thêm cột 1 vào phía trước
- Các bước thực hiện:
  - Sử dụng hàm *np.ones()* để tạo ra ma trận toàn 1 với số dòng bằng số dòng của ma trận *X* và số cột là 1.
  - Sau đó dùng hàm *np.concatenate()* để thực hiện việc nối ma trận toàn 1 đã tạo với ma trận *X* theo axis=1 (theo chiều các cột) và trả về bên ngoài.

**2.10 Hàm split_data_into_k_folds()**

- Đầu vào:
  - *data_frame*: ma trận dataframe**.**

  - *k:* số lượng các fold được chia.
  - *models:* mảng các model được xét, mỗi model là một mảng chứa tên các đặc trưng của model đó.
- Đầu ra: mảng các fold được chia thành công, mỗi fold chứa các cột là bộ các đặc trưng không trùng nhau của các mô hình đang xét và giá trị mục tiêu.
- Các bước thực hiện:
  - Tạo một bản deep copy của *data_frame* bằng cách sử dụng hàm *copy()* của thư viện pandas và một list rỗng *folds*.
  - Tính số lượng dòng trong *data_frame* lưu vào biến n, số lượng dòng của từng fold là $\frac{n}{k}$ lưu vào biến m.
  - Sử dụng hàm *sample()* của pandas để shuffle dữ liệu trong dataframe trước khi chia thành các fold.
  - Sử dụng hàm *deepcopy()* để tạo bản sao cho *models* sau đó dùng *np.concatenate()* kết hợp với thuộc tính *flat* để flatten *models* vì nó là một mảng hai chiều các đặc trưng. Sau đó chuyển kết quả thu được thành kiểu *set* bằng cách dùng hàm *set()* để tự lọc bỏ đi các đặc trưng trùng nhau và chuyển thành *list* lại bằng hàm *list()*.
  - Sau khi đã có được mảng các đặc trưng không trùng nhau lấy từ các model, ta sẽ lặp qua *k* lần, tại mỗi lần lặp thứ *i* ta thêm vào list *folds*, một dataframe được lấy từ dòng *i*m* đến *min(i*m + m, n)* của tập *data_frame*, kèm theo đó chỉ lấy các cột là các đặc trưng nằm trong mảng các đặc trưng và giá trị mục tiêu (cụ thể trong bài này là *Salary*).

### 2.11 Hàm get_model_name()

- Đầu vào: *model*: mô hình cần được lấy tên.
- Đầu ra: tên của mô hình đầu vào
- Các bước thực hiện: Hàm này sử dụng phương thức *join()* của string để có thể nối tên các đặc trưng của mô hình thành một chuỗi duy nhất ngăn cách bởi một kí tự tùy chọn.

### 2.12 Hàm mae()

- Đầu vào:
  - *y*: vector dataframe của các giá trị mục tiêu thực tế.
  - y_hat: vector dataframe của các giá trị mục tiêu được dự đoán từ mô hình.
- Đầu ra: Giá trị MAE – Mean Absolute Error (Độ lỗi tuyệt đối trung bình) được tính theo công thức [5]

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- Các bước thực hiện:
  - Chuyển hai vector dataframe *y* và *y_hat* thành numpy array

- Sử dụng hàm *ravel()* của numpy để flatten hai numpy array vừa thu được và tính hiệu giữa chúng.
- Sử dụng *np.abs()* để tính trị tuyệt đối của hiệu vừa thu được. Sau đó tính trung bình bằng hàm *np.mean()*.

# III. Thực hiệu các yêu cầu và nhận xét kết quả

## 1. Yêu cầu 1a: Sử dụng toàn bộ 11 đặc trưng đầu tiên để xây dựng mô hình.

- Mô tả yêu cầu:
  - Ở yêu cầu 1a, ta cần phải thực hiện việc huấn luyện mô hình 1 lần duy nhất cho toàn bộ 11 đặc trưng đầu tiên (`Gender`, `10percentage`, `12percentage`, `CollegeTier`, `Degree`, `collegeGPA`, `CollegeCityTier`, `English`, `Logical`, `Quant`, `Domain`) cho toàn bộ tập huấn luyện (train.csv).
  - Sau đó thực hiện tính toán mô hình hồi quy, đưa ra công thức
  - Cuối cùng mang mô hình hồi quy tính toán được đi thực hiện dự đoán giá trị mục tiêu trên tập kiểm tra, từ đó tính độ lỗi MAE và ghi nhận kết quả.
- Các bước thực hiện:
  - Đầu tiên ta sử dụng hàm *deepcopy()* của thư viện *copy* để tạo ra một bản sao từ tập *X_train* (Dataframe chứa các đặc trưng huấn luyện)
  - Sau đó sử dụng *iloc[:, :11]* để lấy ra được 11 đặc trưng đầu tiên lưu vào *X_train_1a*
  - Sử dụng hàm *OLSLinearRegression.fit()* để thực hiện quá trình fit mô hình và thu về kết quả là công thức đường hồi quy lưu vào *lr_1a*, tiếp tục sử dụng *lr_1a.get_params()* để lấy ra các tham số.
  - Sau khi đã có công thức hồi quy, ta sử dụng hàm *predict()* đã được cài đặt để dự đoán đầu ra cho tập kiểm tra với 11 đặc trưng đầu tiên. Từ đó sử dụng hàm *mae()* tính toán độ lỗi và trả về kết quả.
- Kết quả:
  - Mô hình hồi quy tuyến tính tìm được:

$$\begin{aligned} Salary = {} & 49248.09 - 23183.33 * Gender + 702.767 * 10percentage \\ & + 1259.019 * 12percentage - 99570.608 * CollegeTier \\ & + 18369.962 * Degree + 1297.532 * collegeGPA \\ & - 8836.727 * CollegeCityTier + 141.76 * English \\ & + 145.742 * Logical + 114.643 * Quant \\ & + 34955.75 * Domain. \end{aligned}$$

  - Giá trị MAE = 105052.529

## 2. Yêu cầu 1b: Sử dụng duy nhất 1 đặc trưng tính cách để xây dựng mô hình

- Mô tả yêu cầu:
  - Ở yêu cầu 1b, ta vẫn thực hiện việc xây dựng mô hình giống như câu 1a, tuy nhiên vì có tổng cộng 5 đặc trưng tính cách gồm: `conscientiousness`, `agreeableness`, `extraversion`, `nueroticism`, `openess_to_experience` do đó ta sẽ có 5 mô hình cho từng đặc trưng.

  - Sử dụng phương pháp K-fold Cross Validation để tìm ra đặc trưng tốt nhất trong các đặc trưng tính cách, tức là tìm ra mô hình tốt nhất trong 5 mô hình và đưa ra công thức.
  - Sau khi đã tìm được mô hình tốt nhất, ta đem mô hình đó đi huấn luyện lại trên toàn bộ tập huấn luyện, sau đó dự đoán giá trị mục tiêu trên tập kiểm tra và ghi nhận giá trị độ lỗi.
- Các bước thực hiện:
  - Tạo mảng các mô hình *models_1b*, mỗi model trong *models_1b* là một mảng chứa 1 đặc trưng tính cách và khai báo số lượng fold được chia *k_1b = 5*
  - Sử dụng hàm *split_data_into_k_folds()* để shuffle và chia bộ dữ liệu train ban đầu thành 5 folds, lưu vào *folds_1b*
  - Tạo ra một đối tượng *k_cross_validation_1b* từ class *KFoldCrossValidation* với 3 tham số truyền vào là *k_1b, folds_1b* và *models_1b.*
  - Sau đó sử dụng phương thức *fit()* trên đối tượng để thực hiện quá trình data fitting trên từng mô hình với từng fold, tính toán độ lỗi MAE của từng mô hình và lưu giá trị đó vào mae_values.
  - Dùng phương thức *get_best_model()* của đối tượng để lấy ra mô hình tốt nhất (đặc trưng tốt nhất), in kết quả mô hình tốt nhất ra màn hình và dùng *get_mae_values()* để lấy ra kết quả MAE trung bình của từng mô hình.
  - Cuối cùng, ta thực hiện các bước xây dựng mô hình tương tự như ở câu 1a cho mô hình của đặc trưng tốt nhất trên toàn bộ tập train, sau đó dự đoán giá trị mục tiêu và ghi nhận lại kết quả.
- Kết quả:
  - Đặc trưng tốt nhất: nueroticism
  - Bảng 5 kết quả tương ứng cho 5 mô hình từ K-fold Cross Validation (lấy trung bình)

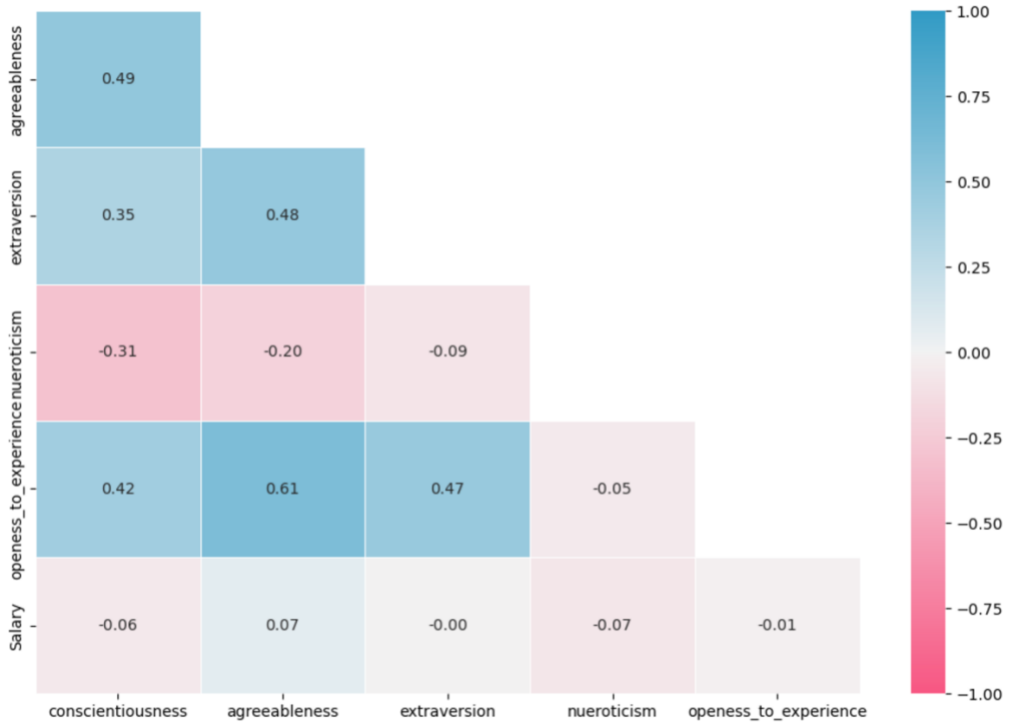
| STT | Mô hình với 1 đặc trưng | MAE |
|---|---|---|
| 1 | conscientiousness | 124563.730 |
| 2 | agreeableness | 123453.074 |
| 3 | extraversion | 123919.757 |
| 4 | nueroticism | 123367.981 |
| 5 | openess_to_experience | 124089.216 |

  - Mô hình của đặc trưng tốt nhất:

$$Salary = 304647.553 - 16021.494 * nueroticism$$

  - Giá trị MAE = 119361.917
- Nhận xét: Trong 5 đặc trưng tính cách trên, có hai đặc trưng có độ tốt xấp xỉ nhau là ‘agreeableness’ và ‘nueroticism’. Tuy nhiên vì sau nhiều lần chạy, tần suất ra đặc trưng ‘nueroticism’ nhiều hơn so với ‘agreeableness’ do đó ta chọn ‘nueroticism’ là đặc trưng tốt nhất.

- Giải thích: Đặc trưng tính cách ‘nueroticism’ là đặc trưng đặc trưng tốt nhất bởi vì khi thực hiện trực quan hóa dữ liệu bằng heatmap (hình 1) để thấy được sự tương quan giữa các đặc trưng tính cách với giá trị mục tiêu Salary, ở dòng cuối cùng, ta có thể thấy được có hai đặc trưng có độ tương quan cao nhất (lấy giá trị tuyệt đối) là ‘nueroticism’ và ‘agreeableness’ (lần lượt là -0.07 và 0.07). Về phương pháp tìm ma trận tương quan và trực quan hóa bằng seaborn sẽ được trình bày rõ hơn ở mục 1d.




*Hình 1: Biểu đồ tương quan giữa 5 đặc trưng tính cách và Salary*

### 3. Yêu cầu 1c: Sử dụng duy nhất 1 đặc trưng tính cách để xây dựng mô hình

- Mô tả yêu cầu:
  - Ở yêu cầu 1c, ta thực hiện tương tự như câu 1c nhưng thay vì 5 đặc trưng tính cách thì ta sẽ dùng 3 đặc trưng về ngoại ngữ, logic và định lượng (`English`, `Logical`, `Quant`).
  - Sử dụng phương pháp K-fold Cross Validation để tìm ra đặc trưng tốt nhất trong các 3 đặc trưng.
  - Sau khi đã tìm được mô hình tốt nhất, ta đem mô hình đó đi huấn luyện lại trên toàn bộ tập huấn luyện, sau đó dự đoán giá trị mục tiêu trên tập kiểm tra và ghi nhận giá trị độ lỗi.
- Các bước thực hiện:
  - Tạo mảng các mô hình *models_1c*, mỗi model trong *models_1c* là một mảng chứa 1 đặc trưng trong 3 đặc trưng đã nêu và khai báo số lượng fold được chia *k_1c = 5*.

- Sử dụng hàm *split_data_into_k_folds()* để shuffle và chia bộ dữ liệu train ban đầu thành 5 folds, lưu vào *folds_1c*
- Tạo ra một đối tượng *k_cross_validation_1c* từ class *KFoldCrossValidation* với 3 tham số truyền vào là *k_1c, folds_1c* và *models_1c.*
- Sau đó sử dụng phương thức *fit()* trên đối tượng để thực hiện quá trình data fitting trên từng mô hình với từng fold, tính toán độ lỗi MAE của từng mô hình và lưu giá trị đó vào mae_values.
- Dùng phương thức *get_best_model()* của đối tượng để lấy ra mô hình tốt nhất (đặc trưng tốt nhất), in kết quả mô hình tốt nhất ra màn hình và dùng *get_mae_values()* để lấy ra kết quả MAE trung bình của từng mô hình.
- Cuối cùng, ta thực hiện các bước xây dựng mô hình tương tự như ở câu 1a cho mô hình của đặc trưng tốt nhất trên toàn bộ tập train, sau đó dự đoán giá trị mục tiêu và ghi nhận lại kết quả.

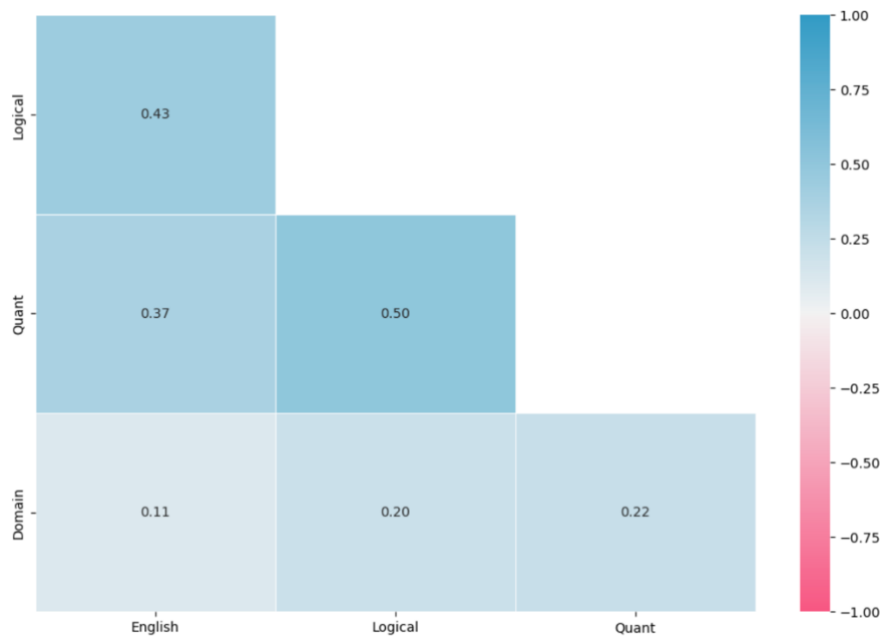
- Kết quả:
  - Đặc trưng tốt nhất: Quant
  - Bảng 3 kết quả tương ứng cho 3 mô hình từ K-fold Cross Validation (lấy trung bình)

| STT | Mô hình với 1 đặc trưng | MAE |
|---|---|---|
| 1 | English | 120745.287 |
| 2 | Logical | 119979.877 |
| 3 | Quant | 117132.631 |

  - Mô hình của đặc trưng tốt nhất:

$$Salary = 117759.729 + 368.852 * Quant$$

  - Giá trị MAE = 108814.059
- Nhận xét: Trong 3 đặc trưng ngoại ngữ, logic và định lượng thì MAE của định lượng là nhỏ nhất do đó nó có ảnh hưởng nhiều nhất lên giá trị mục tiêu. Vì thế Quant là đặc trưng tốt nhất.
- Giải thích: Đặc trưng định lượng ‘Quant’ là đặc trưng đặc trưng tốt nhất bởi vì khi thực hiện trực quan hóa dữ liệu bằng heatmap (hình 2) để thấy được sự tương quan giữa các đặc trưng với giá trị mục tiêu Salary, ở dòng cuối cùng, ta có thể thấy được đặc trưng Quant có độ tương quan cao nhất (0.22).




*Hình 2: Biểu đồ tương quan giữa 3 đặc trưng (ngoại ngữ, logic, định lượng) và Salary*

4. **Yêu cầu 1d: Sinh viên tự xây dựng mô hình, tìm mô hình cho kết quả tốt nhất**
   - Mô tả yêu cầu:
     - Ở yêu cầu 1d, ta cần phải tự xây dựng mô hình (tối thiểu 3 mô hình), đồng thời các mô hình đó phải khác mô hình ở 1a, 1b và 1c.
     - Sử dụng phương pháp K-fold Cross Validation để tìm ra đặc trưng tốt nhất trong các 3 đặc trưng.
     - Sau khi đã tìm được mô hình tốt nhất, ta đem mô hình đó đi huấn luyện lại trên toàn bộ tập huấn luyện, sau đó dự đoán giá trị mục tiêu trên tập kiểm tra và ghi nhận giá trị độ lỗi.
   - Ý tưởng xây dựng mô hình:
     - Để xây dựng một mô hình tốt, trước hết ta cần phải chọn lọc được các đặc trưng tốt để đưa vào mô hình và loại bỏ các đặc trưng không liên quan, tuy nhiên để làm được điều đó ta cần có phương pháp chọn lọc, có rất nhiều phương pháp khác nhau nhưng trong bài này ta sẽ sử dụng phương pháp chọn lọc dựa trên hệ số tương quan Pearson (*Pearson Correlation Coefficient*) – một phương pháp thuộc *Filter Methods*.
     - Ý tưởng của phương pháp *Pearson Correlation Coefficient*: [2]
       - Sự tương quan là thước đo của mối quan hệ tuyến tính giữa hai hoặc nhiều biến. Thông qua độ tương quan, ta có thể dự đoán một biến từ biến khác. Các đặc trưng được chọn là những đặc trưng có độ tương quan cao nhất với giá trị mục tiêu.
       - Nếu như hai biến tương quan mạnh với nhau, ta có thể dự đoán một biến từ biến còn lại. Do đó, mô hình có thể chỉ cần một trong hai chứ không cần cả hai.

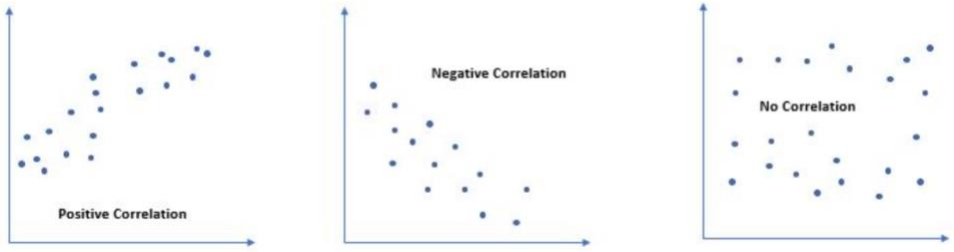
- Công thức tính hệ số tương quan Pearson: [1]

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \ \sum(y_i - \bar{y})^2}}$$

Trong đó:

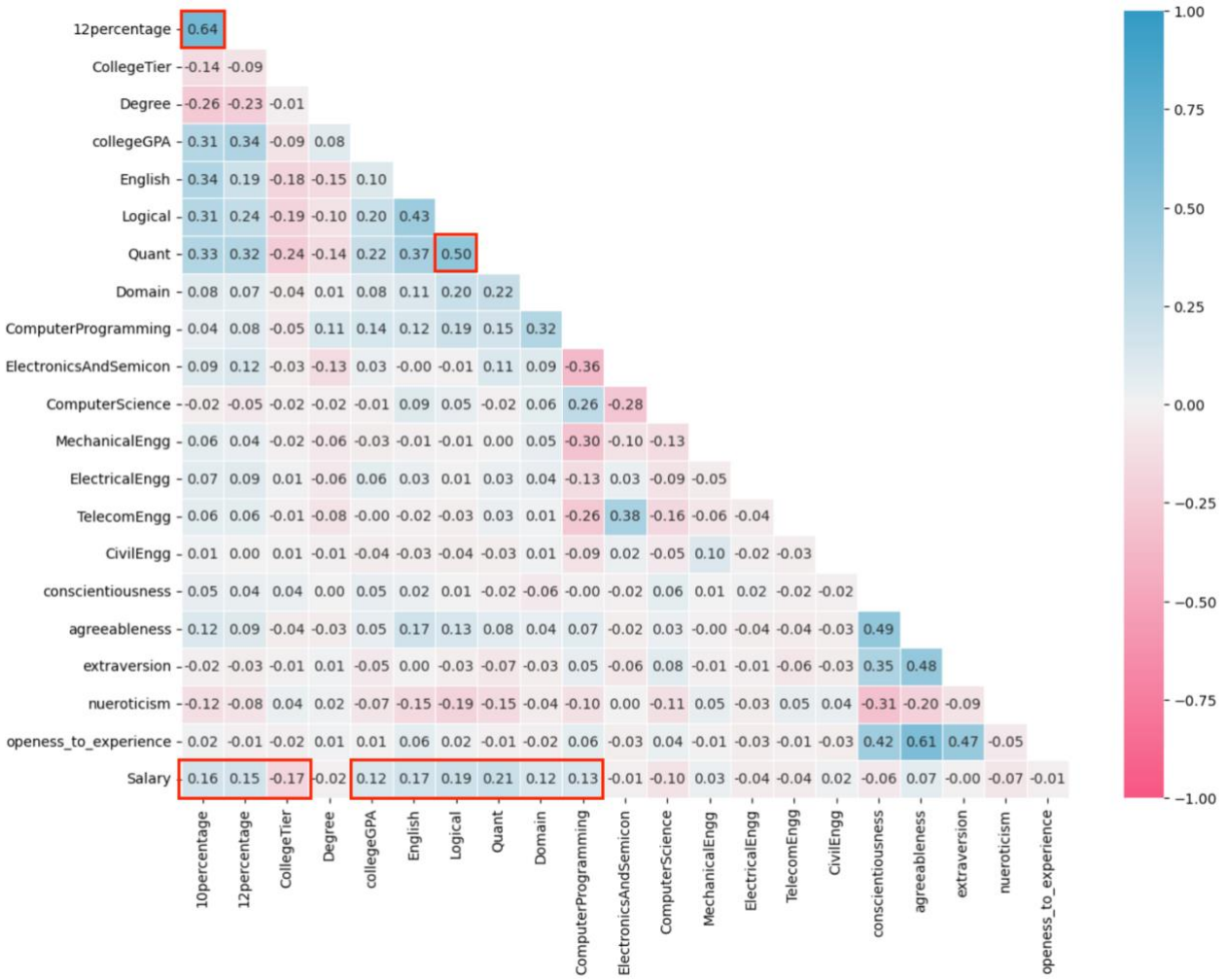
$r$: là hệ số tương quan Pearson
$x_i$: là giá trị của biến x thứ i trong mẫu
$y_i$: là giá trị của biến y thứ i trong mẫu
$\bar{x}$: là giá trị trung bình của các biến x
$\bar{y}$: là giá trị trung bình của các biến y

- Giá trị của hệ số tương quan nằm trong khoảng từ -1 đến 1, hệ số bằng 0 tức là không có tương quan giữa hai biến, hệ số lớn hơn 0 tức là tương quan dương, ngược lại nếu hệ số nhỏ hơn 0 thì là tương quan âm.




*Hình 3: Các loại tương quan (dương, âm và không tương quan)*

- Các bước thực hiện:
  - Sử dụng hàm *drop()* để loại bỏ các đặc trưng không liên quan
  - Sử dụng hàm *corr()* của thư viện pandas để tính toán ma trận tương quan và lưu vào *train_corr*, sau đó sử dụng hàm *heatmap()* của thư viện seaborn để trực quan hóa ma trận tương quan.




*Hình 4: Trực quan hóa ma trận tương quan bằng heatmap*

- Nhìn vào dòng cuối cùng của biểu đồ heatmap, ta lựa chọn ra các đặc trưng có độ tương quan lớn (giá trị tuyệt đối) so với Salary (trong bài này cụ thể là > 0.1), ta có 9 đặc trưng được chọn như sau: 'Quant', 'Logical', 'English', 'CollegeTier', '10percentage', '12percentage', 'ComputerProgramming', 'collegeGPA', 'Domain'. Ta có mô hình đầu tiên gồm 9 đặc trưng.
- Ta lần lượt loại bỏ dần 1 đặc trưng trong các cặp có độ tương quan cao trong mô hình đầu tiên để được các mô hình có số lượng đặc trưng ít dần đi 1 đặc trưng.
- Cặp đầu tiên là 'Quant' và 'Logical' có độ tương quan 0.5, ta loại bỏ 'Logical' và thu được mô hình có 8 đặc trưng.
- Cặp tiếp theo là '12percentage' và '10percentage' có độ tương quan 0.64, ta loại bỏ '10percentage' và thu được mô hình 7 đặc trưng.
- Sau khi đã chọn được 3 mô hình như trên, ta lần lượt thêm vào *features_1d,* sau đó thực hiện các quá trình chọn lọc mô hình tốt nhất, dự đoán giá trị mục tiêu, ghi nhận độ lỗi tương tự như các yêu cầu 1b và 1c.

- Kết quả:
  - Mô hình tốt nhất: Sử dụng 8 đặc trưng [‘Quant’, ‘English’, ‘CollegeTier’, ‘10percentage’, ‘12percentage’, ‘ComputerProgramming’, ‘collegeGPA’, ‘Domain’]
  - Bảng 3 kết quả tương ứng cho 3 mô hình từ K-fold Cross Validation (lấy trung bình)

| STT | Mô hình | MAE |
|---|---|---|
| 1 | Sử dụng 9 đặc trưng (‘Quant’, ‘Logical’, ‘English’, ‘CollegeTier’, ‘10percentage’, ‘12percentage’, ‘ComputerProgramming’, ‘collegeGPA’, ‘Domain’) | 113596.355 |
| 2 | Sử dụng 8 đặc trưng (‘Quant’, ‘English’, ‘CollegeTier’, ‘10percentage’, ‘12percentage’, ‘ComputerProgramming’, ‘collegeGPA’, ‘Domain’) | 113118.505 |
| 3 | Sử dụng 7 đặc trưng (‘Quant’, ‘English’, ‘CollegeTier’, ‘12percentage’, ‘ComputerProgramming’, ‘collegeGPA’, ‘Domain’) | 113430.847 |

  - Mô hình tốt nhất:

$$Salary = 86401.590 + 160.650 * Quant + 160.980 * English - 101319.000 * CollegeTier + 653.340 * 10percentage + 1052.560 * 12percentage + 73.950 * ComputerProgramming + 1169.590 * collegeGPA + 27517.570 * Domain$$

  - Giá trị MAE = 103953.053

## IV. Tài liệu tham khảo

[1] analyticsvidhya.com/blog/2021/01/beginners-guide-to-pearsons-correlation-coefficient
[2] analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/what-is-feature-selection-in-machine-learning?
[3] CodeXplore - Trực quan hóa dữ liệu với Seaborn và Python
[4] tutorials.aiclub.cs.uit.edu.vn/index.php/2021/04/24/linear-regression-là-gì?
[5] Lab04 – Ths Phan Thị Phương Uyên
[6] Project03 – Ths Phan Thị Phương Uyên
[7] https://stataguide.wordpress.com/2020/04/13/mo-hinh-hoi-quy-ols/
[8] https://stackoverflow.com/questions/3282823/get-the-key-corresponding-to-the-minimum-value-within-a-dictionary