TRƯỜNG ĐH KHOA HỌC TỰ NHIÊN, ĐHQG HCM

BÁO CÁO

PROJECT 3 – LINEAR REGRESSION

Sinh viên: Nguyễn Quốc Thắng

MSSV: 22127385

Lớp: 22CLC01

MỤC LỤC

1. Phân tích Khám phá dữ liệu:

1.1. Các thư viện sử dụng để trực quan hóa dữ liệu:

- Thư viện **Matplotlib:** giúp tạo các loại biểu đồ (đường, thanh, phân tán,...)

```
import matplotlib.pyplot as plt
from matplotlib.pyplot import box
```

- Thư viện **Seaborn:**
  - Tạo các biểu đồ thống kê như biểu đồ phân phối, biểu đồ hộp, biểu đồ nhiệt (heatmap),...
  - Tích hợp tốt với **Pandas DataFrame** để trực quan hóa dữ liệu.

```
import seaborn as sns # heatmap
```

1.2. Hiểu rõ dữ liệu:

a. Tải và hiển thị dữ liệu:

- Hàm **Read_csv :** hàm đọc và lưu dữ liệu dưới dạng 1 DataFrame của pandas

```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

- Hàm **Head():** hiển thị dữ liệu 5 dòng đầu:

```
train.head()
```

➢ **Kết quả:**

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	77	0	5	2	69.0
1	8	90	1	4	1	84.0
2	9	83	1	6	3	82.0
3	4	52	0	9	5	38.0
4	4	82	1	8	6	68.0

- Hàm **Tail():** Hiển thị dữ liệu 5 dòng cuối:

```
train.tail()
```

➢ **Kết quả:**

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	77	0	5	2	69.0
1	8	90	1	4	1	84.0
2	9	83	1	6	3	82.0
3	4	52	0	9	5	38.0
4	4	82	1	8	6	68.0

b. Kiểm tra thông tin dữ liệu:

- Hàm **infor():** Hàm này cung cấp một bản tóm tắt ngắn gọn về DataFrame, bao gồm kiểu dữ liệu của các cột, số lượng giá trị không null và dung lượng bộ nhớ.

```
train.info()
```

➢ **Kết quả:**

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9000 entries, 0 to 8999
Data columns (total 6 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Hours Studied                     9000 non-null   int64  
 1   Previous Scores                   9000 non-null   int64  
 2   Extracurricular Activities        9000 non-null   int64  
 3   Sleep Hours                       9000 non-null   int64  
 4   Sample Question Papers Practiced  9000 non-null   int64  
 5   Performance Index                 9000 non-null   float64
dtypes: float64(1), int64(5)
memory usage: 422.0 KB
```

- Hàm **describe():** Hàm này tạo ra các thống kê mô tả tóm tắt xu hướng trung tâm, độ phân tán và hình dạng của phân phối dữ liệu, loại trừ các giá trị NaN.

```
train.describe()
```

➢ **Kết quả:**

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
count	9000.000000	9000.000000	9000.000000	9000.000000	9000.000000	9000.000000
mean	4.976444	69.396111	0.493667	6.535556	4.590889	55.136333
std	2.594647	17.369957	0.499988	1.695533	2.864570	19.187669
min	1.000000	40.000000	0.000000	4.000000	0.000000	10.000000
25%	3.000000	54.000000	0.000000	5.000000	2.000000	40.000000
50%	5.000000	69.000000	0.000000	7.000000	5.000000	55.000000
75%	7.000000	85.000000	1.000000	8.000000	7.000000	70.000000
max	9.000000	99.000000	1.000000	9.000000	9.000000	100.000000

➢ **Nhận xét:**

- **Sự phân tán dữ liệu**: Các giá trị độ lệch chuẩn cao cho thấy dữ liệu có sự phân tán lớn, đặc biệt là ở các đặc trưng như "Hours Studied", "Previous Scores", "Sleep Hours", và "Sample Question Papers Practiced".
- **Giá trị ngoại lệ:** Các giá trị min và max cho thấy có sự khác biệt lớn giữa các học sinh, đặc biệt là ở các đặc trưng như "Previous Scores" và "Performance Index".
- **Phân phối dữ liệu:** Các phần trăm (25%, 50%, 75%) cho thấy sự phân phối của dữ liệu.

1.3. Phân tích đơn biến (Univariate Analysis):

a. Đặc trưng liên tục (Sử dụng boxplot):

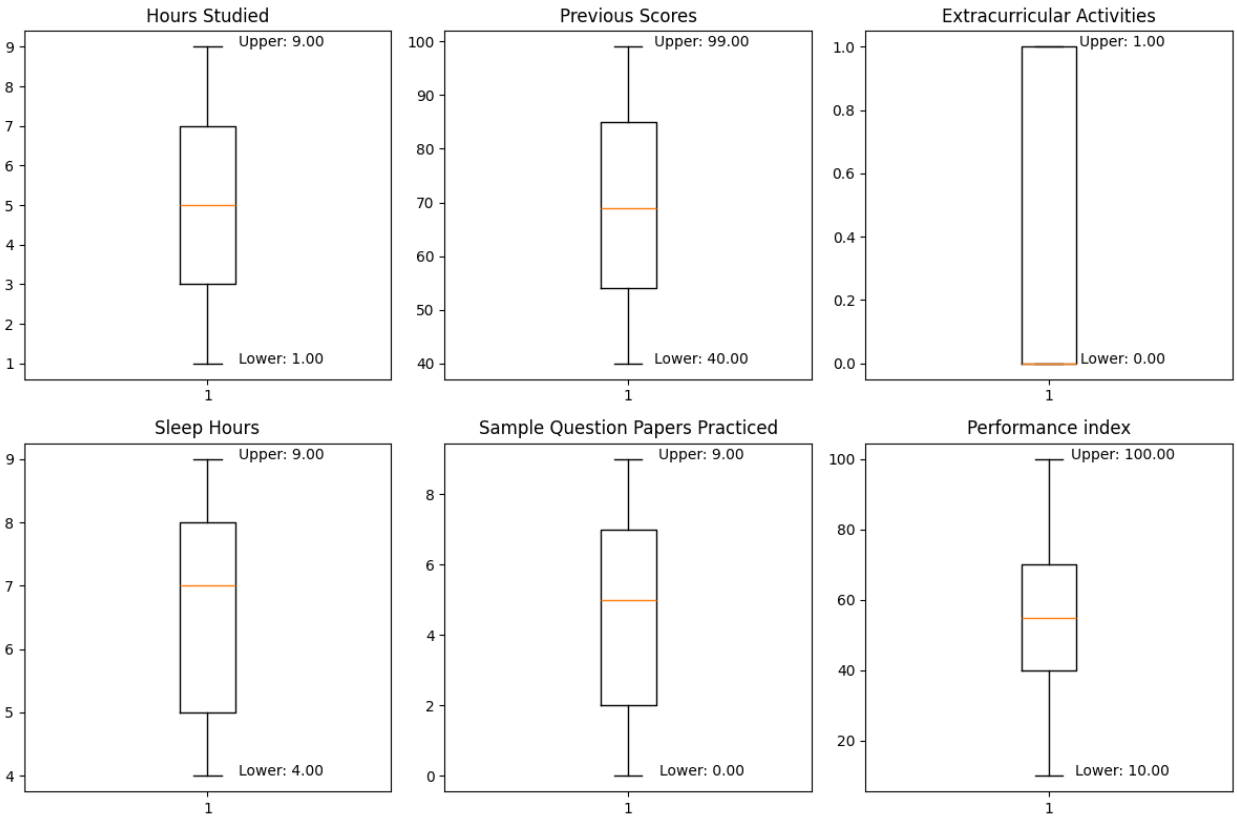
- Hàm **plt.subplots**: tạo một figure và một lưới các trục con (subplots).
- Hàm **plt.show**: hiển thị figure.

- Phương thức **axes[i, j].boxplot**: tạo một biểu đồ hộp (boxplot) trên trục con tại vị trí [i, j].
- Phương thức **axes[i, j].set_title**: đặt tiêu đề cho trục con tại vị trí [i, j].
- Phương thức **axes[i, j].text:** thêm văn bản vào trục con tại vị trí [i, j].
- Phương thức **fig.tight_layout**: điều chỉnh bố cục các trục con để chúng không bị chồng chéo.

```
fig, axes = plt.subplots(2, 3, figsize=(12, 8))

# Boxplot for Hours Studied
box = axes[0, 0].boxplot(X_train['Hours Studied'])
axes[0, 0].set_title('Hours Studied')
axes[0, 0].text(1.2, box['whiskers'][0].get_ydata()[1], f"Lower: {box['whiskers'][0].get_ydata()[1]:.2f}", ha='center')
axes[0, 0].text(1.2, box['whiskers'][1].get_ydata()[1], f"Upper: {box['whiskers'][1].get_ydata()[1]:.2f}", ha='center')
```

➢ **Kết quả:**



b. Đặc trưng phân loại:

- Hàm **plt.subplots.**
- Hàm **plt.show.**
- Phương thức **axes[i ,j].plot:** tạo một biểu đồ con tại vị trí [i, j] (được gọi thông qua **value_counts().plot –** trả về một Series chứa số lượng các giá trị duy nhất trong Series ban đầu).
- Phương thức .**settitle .**
- Phương thức **fig.tight_layout.**

- **Vẽ biểu đồ cột**

```
# Barplot for Hours Studied
barplot = X_train['Hours Studied'].value_counts().plot(kind='bar', title='Hours Studied', ax=axes[0, 0])
```

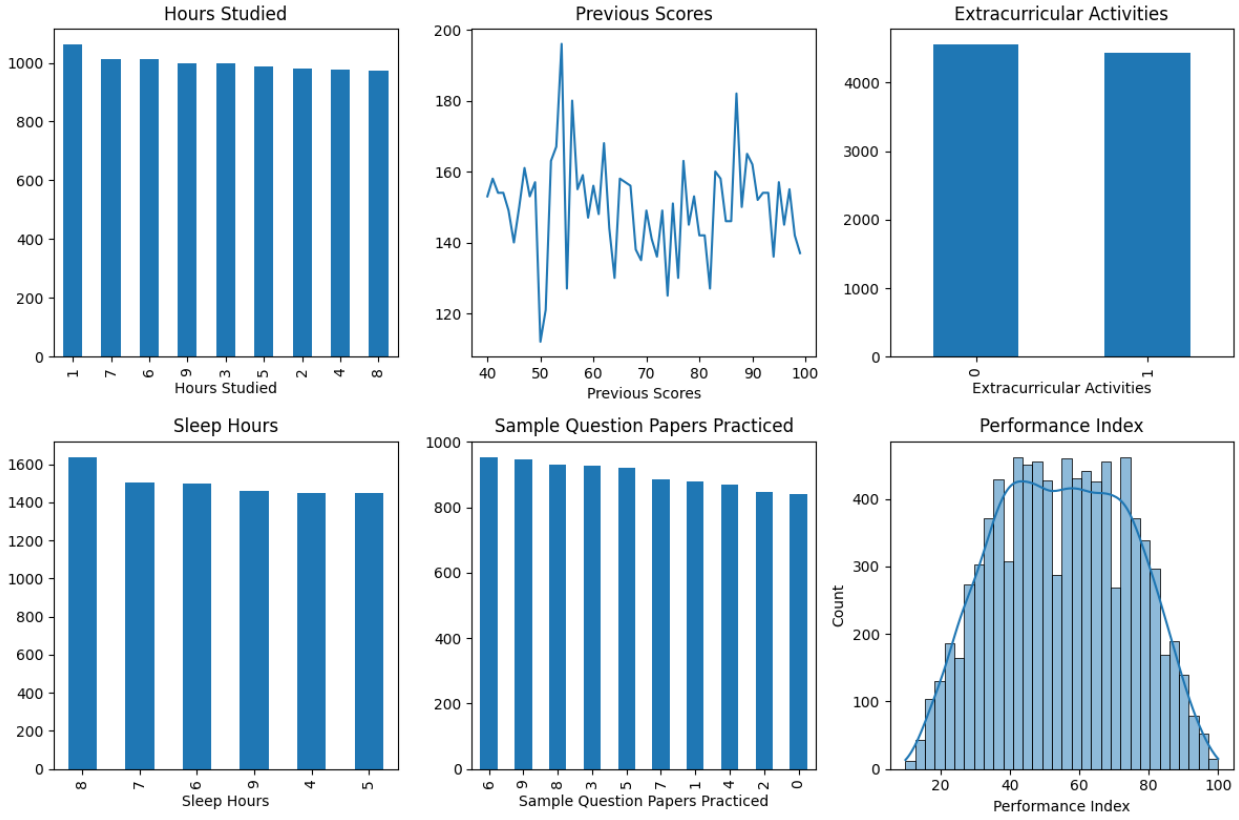
- **Vẽ biểu đồ đường**

```
# Barplot for Previous Scores
lineplot1 = X_train['Previous Scores'].value_counts().sort_index().plot(kind='line', title='Previous Scores', ax=axes[0, 1])
```

- **Vẽ biểu đồ phân phối và đường cong mật độ**
  - Sử dụng hàm histplot của thư viện Seaborn để vẽ biểu đồ phân phối.

```
# Scatter plot for Performance Index
histplot = sns.histplot(y_train, kde=True, ax=axes[1, 2])
axes[1, 2].set_title('Performance Index')
```

➢ **Kết quả:**



➢ **Nhận xét:**

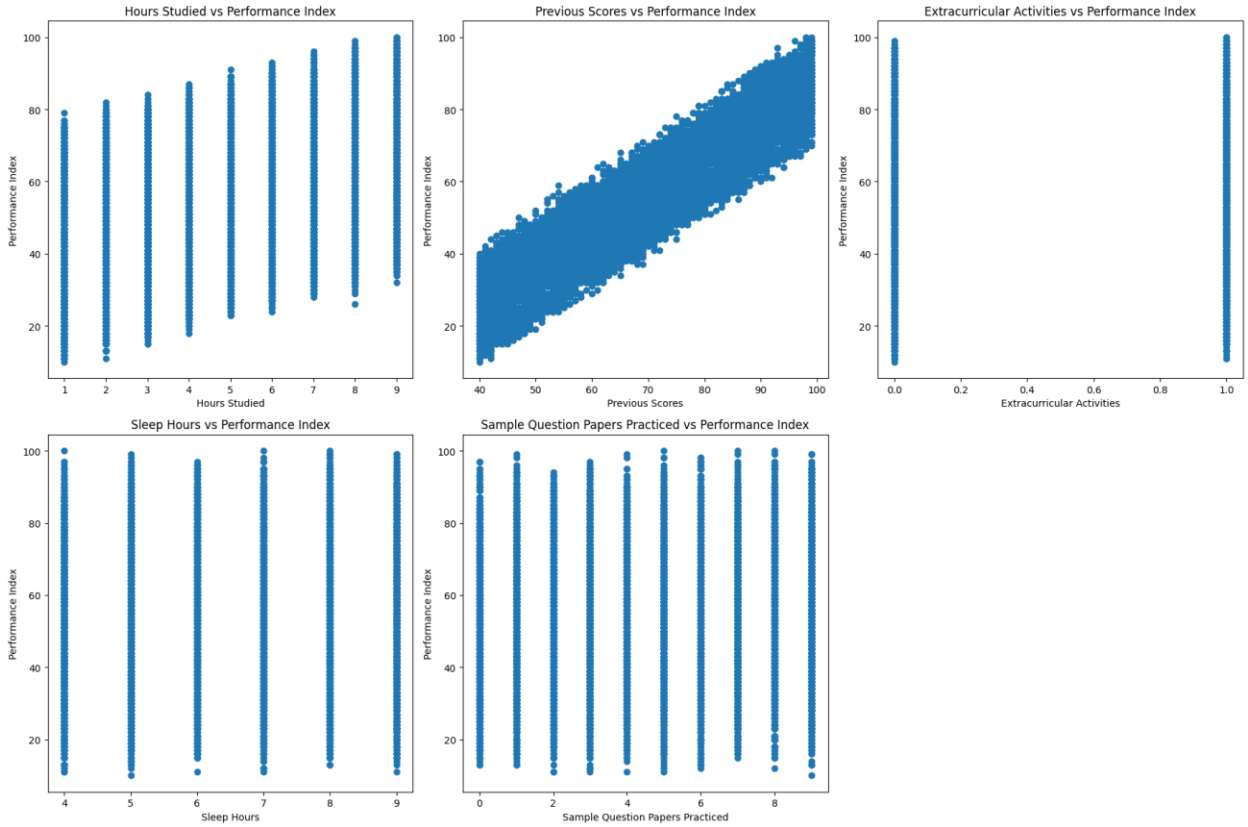
- Nhìn chung, ta thấy biểu đồ có sự phân bố đồng đều về số lượng học sinh đạt điểm cao vào trong các mức độ của 5 đặc trưng thu tập, chỉ trừ đặc trưng *'Previous Scores'* có sự phân bố không đồng đều.

1.4. Phân tích song biến (Bivariate Analysis):

- Cho biết mối tương quan giữa biến đặc trưng (*x1, x2, ... , x5*) với biến mục tiêu *y*.
- Hàm **scatter :** tạo một biểu đồ phân tán trên subplot.

```
# Scatter plot for Hours Studied
scatter1 = axes[0, 0].scatter(X_train['Hours Studied'], y_train)
axes[0, 0].set_title('Hours Studied vs Performance Index')
axes[0, 0].set_xlabel('Hours Studied')
axes[0, 0].set_ylabel('Performance Index')
```

➢ **Kết quả:**



➢ **Nhận xét:**

- Các đặc trưng *Extracurricular Activities, Sleep Hours, Sample Question Papers Practiced* hầu như phân bố đều và không có nhiều liên hệ đến biến mục tiêu *Performance Index*.
- Hai đặc trưng ***Hours Studied*** và ***Previous Scores*** có **xu hướng đồng biến** với biến mục tiêu ***Performance Index***, đặc biệt thể hiện rõ nhất chính là ***Previous Scores***.

Hệ số tương quan Pearson:

$$r = \frac{\sum(X - \overline{X})(Y - \overline{Y})}{\sqrt{\sum(X - \overline{X})^2 \sum(Y - \overline{Y})^2}}$$

Trong đó:

- r là hệ số tương quan Pearson.
- X và Y là các giá trị của hai biến.

- Hàm **corrwith**: cho biết mức độ và hướng của mối quan hệ tuyến tính giữa mỗi đặc trưng.

```
correlation = X_train.corrwith(y_train)
```

➢ **Kết quả:**

```
Hours Studied                     0.369148
Previous Scores                   0.914775
Extracurricular Activities        0.025637
Sleep Hours                       0.043980
Sample Question Papers Practiced  0.041088
dtype: float64
```

➢ **Nhận xét:**

- Đặc trưng *Previous Scores* có ảnh hưởng tới *Performance* nhất , sau đó là *Hours Studied*, còn lại không có ảnh hưởng nhiều.

1.5. Phân tích song biến (Multivariate Analysis):

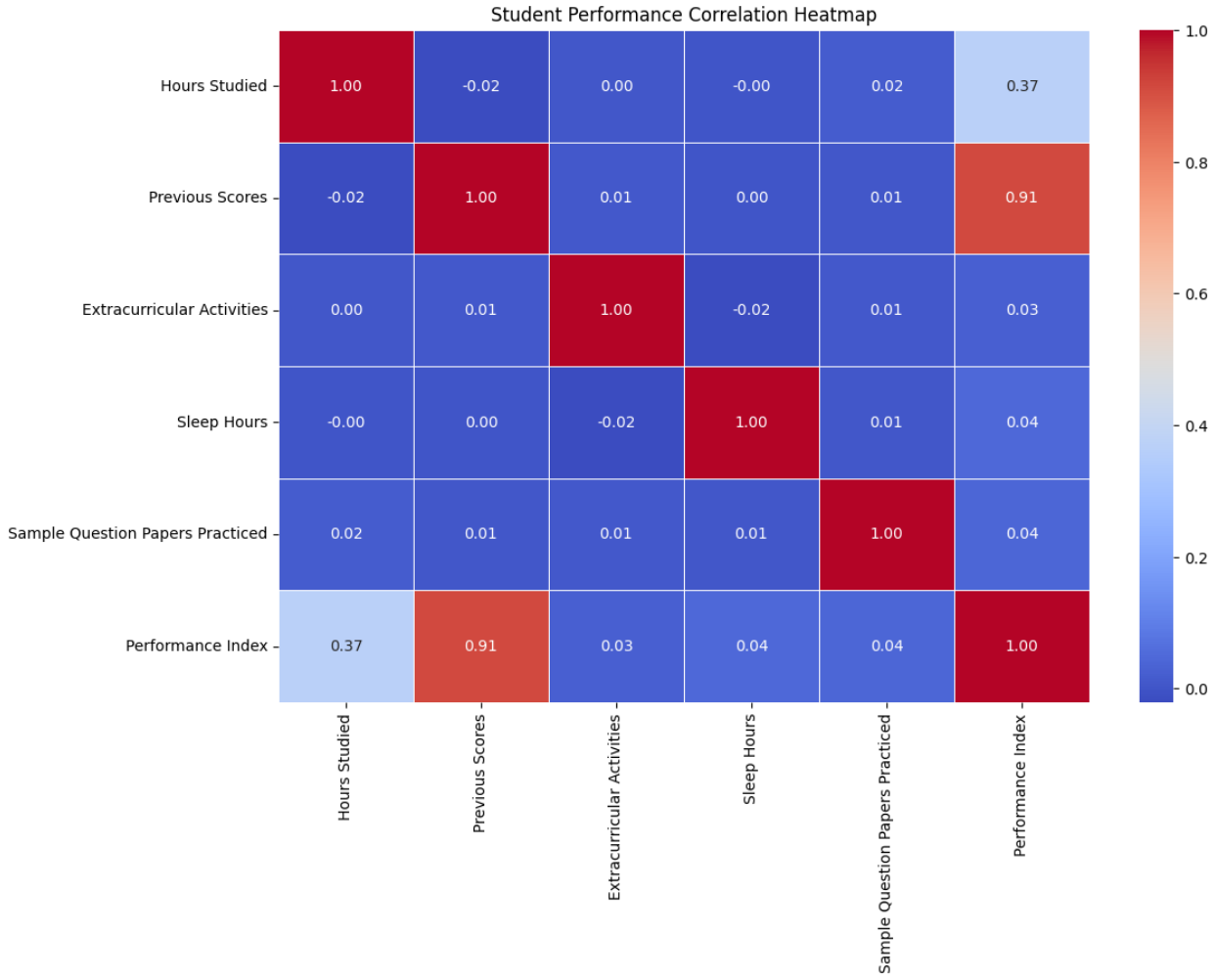
- Bản đồ nhiệt của ma trận tương quan:

  - Tạo một heatmap của ma trận tương quan để xem các mối tương quan giữa tất cả các cặp đặc trưng với nhau. Điều này giúp bạn xác định xem có những đặc trưng nào có mối tương quan mạnh mẽ với nhau hoặc với biến mục tiêu hay không.

  - **Student Perfromance HeatMap**:

    + Màu đỏ đậm: biểu diễn tương quan dương mạnh (gần 1).
    + Màu xanh đậm: biểu diễn tương quan âm mạnh (gần -1).
    + Màu trắng hoặc xanh nhạt: biểu diễn không có tương quan hoặc tương quan yếu.

- Hàm **heatmap** của **Seaborn**.



1.6. Kiểm tra giá trị thiếu (Missing values) và ngoại lệ (Outliers):

a. Giá trị thiếu (Missing value):

```
Hours Studied                     0
Previous Scores                   0
Extracurricular Activities        0
Sleep Hours                       0
Sample Question Papers Practiced  0
Performance Index                 0
dtype: int64
```

b. Ngoại lệ (Outliers):

- Phương thức **Quantile**: tính toán các phân vị

```
Q1 = X_train.quantile(0.25)
Q3 = X_train.quantile(0.75)
IQR = Q3 - Q1
outliers = ((X_train < (Q1 - 1.5 * IQR)) | (X_train > (Q3 + 1.5 * IQR))).sum()
```

- **Outliers**: tạo mặt nạ boolean cho các giá tị ngoại lai. Sum() tính tổng mặt nạ boolean để đếm số lượng giá trị ngoại lai.
    - **Kết quả:**

```
------------------First Quartile-------------------
Hours Studied                         3.0
Previous Scores                      54.0
Extracurricular Activities            0.0
Sleep Hours                           5.0
Sample Question Papers Practiced      2.0
Name: 0.25, dtype: float64
------------------Third Quartile-------------------
Hours Studied                         7.0
Previous Scores                      85.0
Extracurricular Activities            1.0
Sleep Hours                           8.0
Sample Question Papers Practiced      7.0
Name: 0.75, dtype: float64
------------------Total outliers-------------------
Hours Studied                        0
Previous Scores                      0
Extracurricular Activities           0
Sleep Hours                          0
Sample Question Papers Practiced     0
dtype: int64
```

2. Mô hình hồi quy tuyến tính:

- **Ý tưởng:** mô hình hóa mối quan hệ giữa một biến phụ thuộc y và các biến độc lập (X). Mục tiêu là tìm ra một đường thẳng (mặt phẳng trong trường hợp nhiều biến độc lập) tốt nhất để dự đoán giá trị của y dựa trên X.
- **Cài đặt:**
    - **Tách dữ liệu:** Dữ liệu ban đầu được tách thành 2 phần X_train và y_train để đưa vào huấn luyện.

```
X_train = train.iloc[:, :-1]
y_train = train.iloc[:, -1]

X_test = test.iloc[:, :-1]
y_test = test.iloc[:, -1]
```

    - Hàm **preprocess (Tiền xử lí dữ liệu):**

- Hàm này thêm một cột các giá trị 1 vào ma trận X để đại diện cho hệ số chặn (**intercept**) trong mô hình hồi quy tuyến tính, giúp mô hình có thể học được hệ số này mà không cần phải xử lý riêng biệt.
- Sử dụng hàm **hstack()** của thư viện **numpy**.

▪ Lớp **OLSLinearRegression:**

- Cài đặt mô hình sử dụng phương pháp **Ordinary Least Squares**. Gồm 3 hàm :

+ Hàm **fit(self, X, y):**

```
def fit(self, X, y):
    X_pinv = np.linalg.pinv(X)
    self.w = X_pinv @ y
    return self
```

X_pinv : tính toán ma trận khả nghịch của X.

W: tính toán trọng số w bằng cách nhân nó với ma trận khả nghịch của (X) với vector y.

+ Hàm **get_params(self):** trả về ma trận trọng số w.

```
def get_params(self):
    return self.w
```

+ Hàm **predict(self, X):** dự đoán giá trị y dựa trên ma trận X và vector trọng số w. Phép nhân của 2 ma trận này sẽ cho kết quả dự đoán.

```
def predict(self, X):
    return X @ self.w
```

3. Mô hình 1: Hồi quy tuyến tính sử dụng tất cả các đặc trưng:

- **Ý tưởng:** Dựa trên ý tưởng của OLS, đưa toàn bộ 5 đặc trưng vào huấn luyện
  - **Hàm dự đoán:**

$$\textit{Student Performance} = \theta_0 + \theta_1 * \textit{Hours Studied} + \theta_2 * \textit{Previous Scores} + \theta_3 * \textit{Extracurricular} + \theta_4 * \textit{Sleep Hours} + \theta_5 * \textit{Sample Question Papers Practiced}$$

- **Cài đặt mô hình:**
  - **Bước 1.** Tiền xử lí dữ liệu.
    - Tạo các bản sao của tập huấn luyện và kiểm tra.
    - Thêm cột 1 vào các tập huấn luyện **X_train** và tập test **X_test**, sử dụng hàm **preprocess.**

- **Bước 2.** Huấn luyện mô hình.
  - Gọi hàm **fit()** từ class **OLSLinearRegression** với tham số đầu vào là tập huấn luyện đặc trưng **X_train** và tập biến mục tiêu **y_train.** Ta thu được tập các trọng số **lr.**
- **Bước 3.** Dự đoán giá trị.
  - Gọi hàm predict từ class **OLSLinearRegression** với tham số đầu vào là tập huấn luyện đặc trưng **X_train .** Ta thu được tập các giá trị dự đoán **y_pred.**

- **Kết quả:**
  - **Tập trọng số:**

```
Weights: [-33.969   2.852   1.018   0.604   0.474   0.192]
```

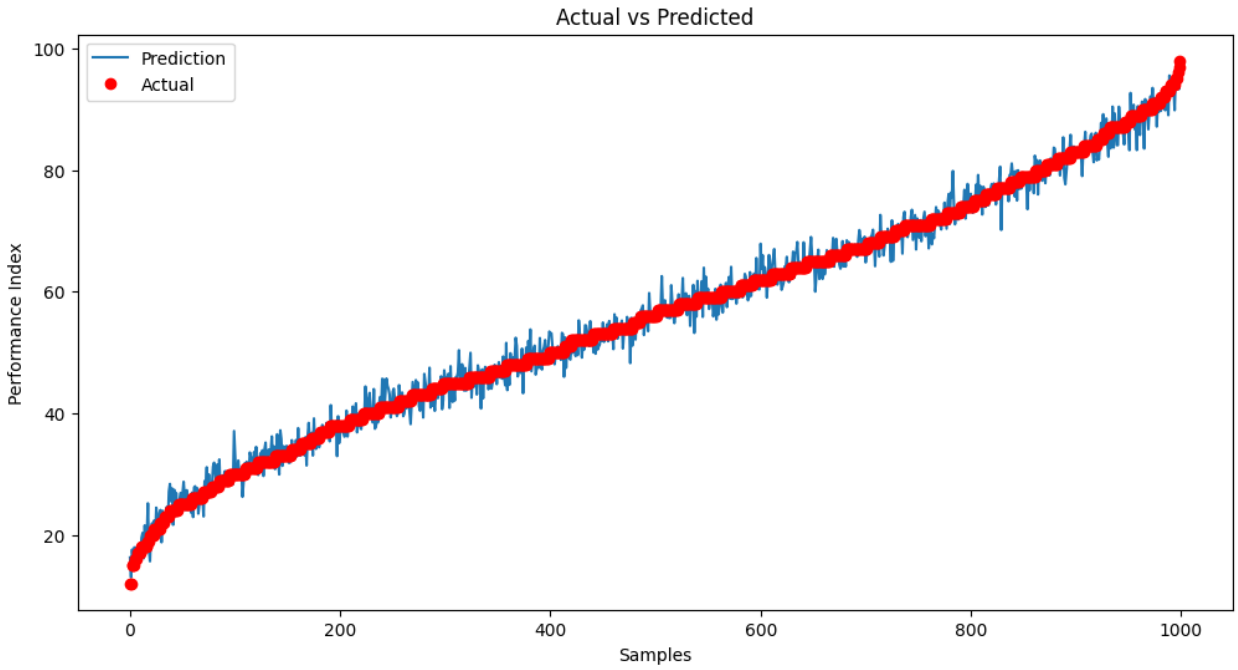
  - **Công thức hồi quy:**

Student Performance = -33.969 + 2.852*Hours Studied + 1.018 * Previous Scores + 0.604 * Extracurricular + 0.474 * Sleep Hours + 0.192 * Sample Question Papers Practiced

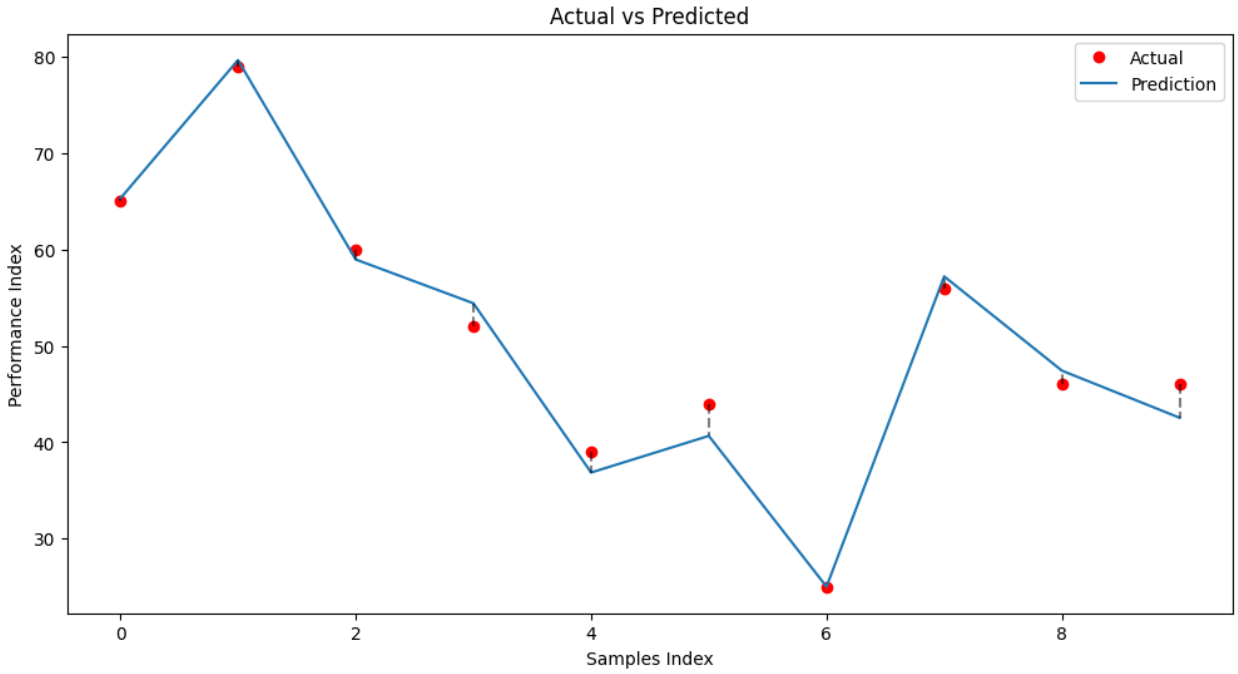
  - **Giá trị dự đoán:**

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index (Prediction)
0	7.0	74.0	0.0	8.0	1.0	65.298
1	6.0	89.0	1.0	8.0	8.0	79.665
2	3.0	79.0	0.0	8.0	1.0	58.979
3	7.0	62.0	0.0	8.0	8.0	54.430
4	5.0	51.0	0.0	7.0	7.0	36.864

  - **Trực quan hóa:**



Trên toàn tập dữ liệu



Trên 10 dòng dữ liệu đầu

- **Tính MAE trên tập dữ liệu Test**

```
def MAE(y_true, y_pred):
    return np.mean(np.abs(y_true - y_pred))
```

➢ Kết quả:

```
MAE on test set:  1.5956569999999999
```

4. Mô hình 2: Hồi quy tuyến tính sử dụng 1 trong các đặc trưng kết hợp với Cross-validation

4.1. Thuật toán Cross-validation:

- **Ý tưởng:** chia dữ liệu thành k phần (folds) và thực hiện huấn luyện, kiểm tra trên các phần này.
- **Cài đặt:**
  - **Bước 1.** ***Xáo trộn dữ liệu***

  - Để đảm bảo dữ liệu được phân phối ngẫu nhiên , ta xáo trộn dữ liệu đầu vào trước khi chia thành các phần. (**Chú ý**: chỉ xáo trộn 1 lần trước khi chia)
  - Hàm **shuffle_data:** `def shuffle_data(X_train, y_train):`

    + **Input: X_train** và **y_train** – tập biến đặc trưng và biến cục bộ tương ứng.

    + **Output: X_train** và **y_train** tương ứng sau khi đã được xáo trộn.

    + **Gồm các bước:**

      ✦ **Bước a.** Kiểm tra tính hợp lệ của dữ liệu đầu vào .

      Sử dụng `assert X_train.shape[0] == y_train.shape[0]` để đảm bảo số lượng mẫu trong **X_train** và **y_train** là bằng nhau. Nếu không, hàm sẽ dừng và báo lỗi.

      ✦ **Bước b.** Tạo một hoán vị cố định của các chỉ số.

      Sử dụng `fixed_permutation = np.arange(X_train.shape[0])` : đây là chỉ số của các mẫu trong dữ liệu.

      ✦ **Bước c.** Xáo trộn các chỉ số.

      Sử dụng `np.random.default_rng(42).shuffle(fixed_permutation)` sử dụng bộ sinh số ngẫu nhiên với seed là 42 để xáo trộn mảng chỉ số. Hạt giống cố định này đảm bảo rằng chỉ xáo trộn 1 lần tập dữ liệu và mỗi lần chạy hàm sẽ giống nhau.

      ✦ **Bước d.** Trả về dữ liệu được xáo trộn.

  - **Bước 2.** ***Chia dữ liệu thành k phần***.

  - Hàm **k_fold_split:** `def k_fold_split(X, y, k):`

+ **Input**: X – dữ liệu đầu vào (features) ; y – nhãn tương ứng với dữ liệu đầu vào ; k – số lượng phần muốn chia.

+ **Output**: **X_folds , y_folds** – danh sách chứa k phần của dữ liệu đầu vào và nhãn tương ứng.

+ **Các bước thực hiện:**

- **Bước a.** Tính kích thước mỗi phần = Số lượng mẫu // k.
- **Bước b.** Khởi tạo 2 danh sách rỗng để lưu trữ.
- **Bước c.** Sử dụng vòng lặp for để chia dữ liệu và nhãn thành k phần dựa trên chỉ số i và fold_size.

- **Bước 3.** ***Huấn luyện và kiểm tra mô hình.***

```
def cross_validation(X_train, y_train, k):
```

- Tạo tập huấn và tập kiểm tra cho fold hiện tại.

```
X_train_cv = np.concatenate([X_folds[j] for j in range(k) if j != i])
y_train_cv = np.concatenate([y_folds[j] for j in range(k) if j != i])
X_val_cv = X_folds[i]
y_val_cv = y_folds[i]
```

- Gọi hàm huấn luyện.

```
lr = train_linear_regression(X_train_cv, y_train_cv)
```

- Dự đoán trên tập kiểm tra.

```
y_pred = lr.predict(preprocess(X_val_cv))
```

- Tính toán MAE cho fold hiện tại.

```
mae = MAE(y_val_cv, y_pred)
mae_folds.append(mae)
```

- Trả về giá trị MAE trung bình cho tất cả các folds.

```
np.mean(mae_folds)
```

- **Bước 4.** ***Tính toán lỗi trung bình.***

- Hàm **MAE**

4.2. Thuật toán Cross-validation trong Hồi quy tuyến tính với từng đặc trưng:

- **Ý tưởng:** Dựa theo ý tưởng của Cross-validation đã mô tả ở phần trên.
- **Cài đặt:**

+ **Bước 1.** Tạo nơi lưu trữ cho các mae.

```
k_values = [5, 10, 15]
mae_hours = []
mae_previous = []
mae_extra = []
mae_sleep = []
mae_sample = []
```

+ **Bước 2.** Tiền xử lý dữ liệu. Gồm : Xáo trộn dữ liệu và tách từng đặc trưng ra để huấn luyện và kiểm tra.

```
X_train_2b, y_train_2b = shuffle_data(X_train_2b, y_train_2b)
```

```
X_train_2b_hours = X_train_2b['Hours Studied'].values.reshape(-1, 1)
X_train_2b_previous = X_train_2b['Previous Scores'].values.reshape(-1, 1)
X_train_2b_extra = X_train_2b['Extracurricular Activities'].values.reshape(-1, 1)
X_train_2b_sleep = X_train_2b['Sleep Hours'].values.reshape(-1, 1)
X_train_2b_sample = X_train_2b['Sample Question Papers Practiced'].values.reshape(-1, 1)
```

+ **Bước 3**. Chạy thuật toán Cross-validation với từng đặc trưng.

```
mae_hours.append(cross_validation(X_train_2b_hours, y_train_2b, k).round(5))
mae_previous.append(cross_validation(X_train_2b_previous, y_train_2b, k).round(5))
mae_extra.append(cross_validation(X_train_2b_extra, y_train_2b, k).round(5))
mae_sleep.append(cross_validation(X_train_2b_sleep, y_train_2b, k).round(5))
mae_sample.append(cross_validation(X_train_2b_sample, y_train_2b, k).round(5))
```

+ **Bước 4**. In kết quả

- **Kết quả:**

```
__________________MAE with cross-validation (k = 5)__________________

Hours Studied: 15.45611
Previous Scores: 6.61867
Extracurricular Activities: 16.20415
Sleep Hours: 16.19334
Sample Question Papers Practiced: 16.18969


__________________MAE with cross-validation (k = 10)_________________

Hours Studied: 15.45387
Previous Scores: 6.6185
Extracurricular Activities: 16.19994
Sleep Hours: 16.19173
Sample Question Papers Practiced: 16.18775


__________________MAE with cross-validation (k = 15)_________________

Hours Studied: 15.45302
Previous Scores: 6.61836
Extracurricular Activities: 16.19843
Sleep Hours: 16.18951
Sample Question Papers Practiced: 16.18806
```

- **Nhận xét:**

  + Hầu như không có sự khác biệt nhiều giữa 3 loại k (5,10,15).

  + **Previous Scores** là đặc trưng có chỉ số MAE nhỏ nhất nên có thể thấy nó là **đặc trưng tốt nhất** cho việc huấn luyện và dự đoán .

4.3. Mô hình 3: Hồi quy tuyến tính sử dụng đặc trưng tốt nhất trên toàn tập:

- **Đặc trưng tốt nhất**: *Previous Scores.*
- Thực hiện huấn luyện với mô hình **Hồi quy tuyến tính** với dữ liệu đầu vào là đặc trưng *Previous Scores* trên toàn tập dữ liệu.

Công thức hồi quy:

$$\text{Student Performance} = -14.989 + 1.011 * PreviousScores$$

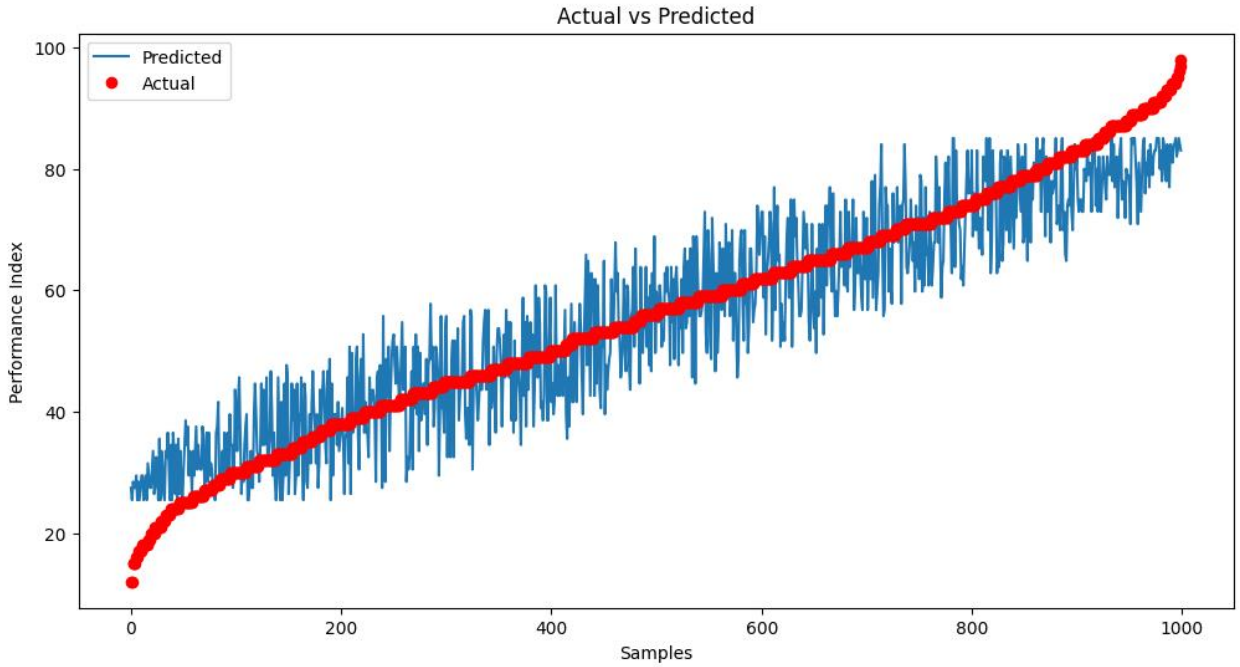
- **Kết quả:**
  - **Giá trị MAE:**

```
MAE with best feature (test_set):  6.544
```

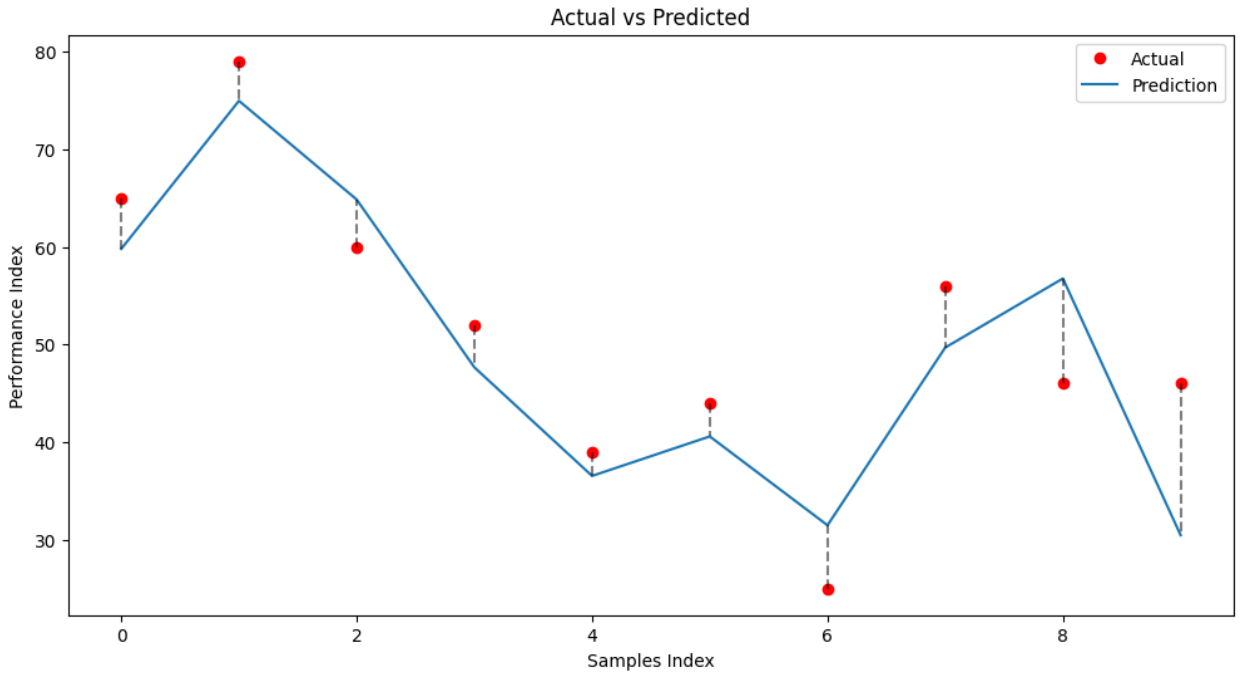
- **Giá trị dự đoán:** Trên 5 dòng dữ liệu đầu

	Previous Scores	Performance Index (Prediction)
0	74.0	59.789
1	89.0	74.946
2	79.0	64.841
3	62.0	47.663
4	51.0	36.547

- **Trực quan hóa giá trị dự đoán và giá trị thực tế:**



Trên toàn bộ dữ liệu



Trên 10 dòng dữ liệu đầu

5. Thiết kế mô hình cho kết quả tốt nhất:

5.1. Các hàm được sử dụng:

- **Hàm** `def cross_validation_v2(X_train, y_train, k)` **:** thực hiện việc chia dữ liệu huấn luyện thành k phần (folds) và sau đó thực hiện huấn luyện mô hình hồi quy tuyến tính trên từng tập dữ liệu huấn luyện được tạo ra từ các phần này. Điểm khác biệt là nó sẽ trả về lr thay vì **MAE** so với **Cross_validation_v1.**

5.2. Mô hình 4

- **Ý tưởng:** Sử dụng mô hình hồi quy tuyến tính với hàm dự đoán như sau:

- Công thức hồi quy:

$$\text{Student Performance} = Theta0 + Theta1 * HoursStudied + Theta2 * PreviousScores$$

- **Cài đặt:**
  - **Bước 1.** Tiền xử lý gồm tạo bản sao và lấy 2 đặc trưng *Hours Studied* và *Previous Scores* ra để huấn luyện.

```
X_train_2c_v1 = X_train_2c_v1[['Previous Scores', 'Hours Studied']]
X_test_2c_v1 = X_test_2c_v1[['Previous Scores', 'Hours Studied']]
```

- **Bước 2.** Huấn luyện mô hình.

```
lr = cross_validation_v2(X_train_2c_v1.values, y_train_2c_v1.values, k)
```

- **Bước 3.** Tính giá trị dự đoán.

```
y_pred_2c_v1 = predict_v2(X_test_2c_v1, lr)
```

- **Bước 4.** Tính MAE.

```
mae = MAE(y_test_2c_v1, y_pred_2c_v1)
```

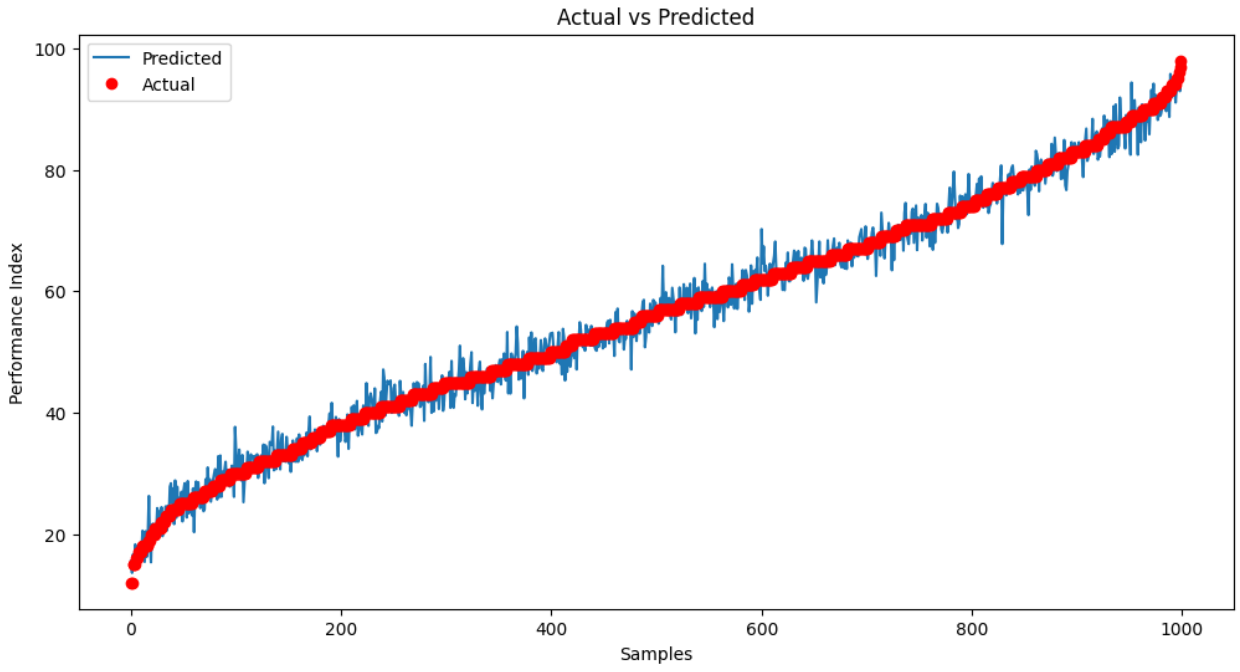
- **Kết quả:**
  - **Công thức hồi quy:**

$$\text{Student Performance} = -29.66729353 + 1.01773677 * PreviousScores + 2.84945817 * HoursStudied$$

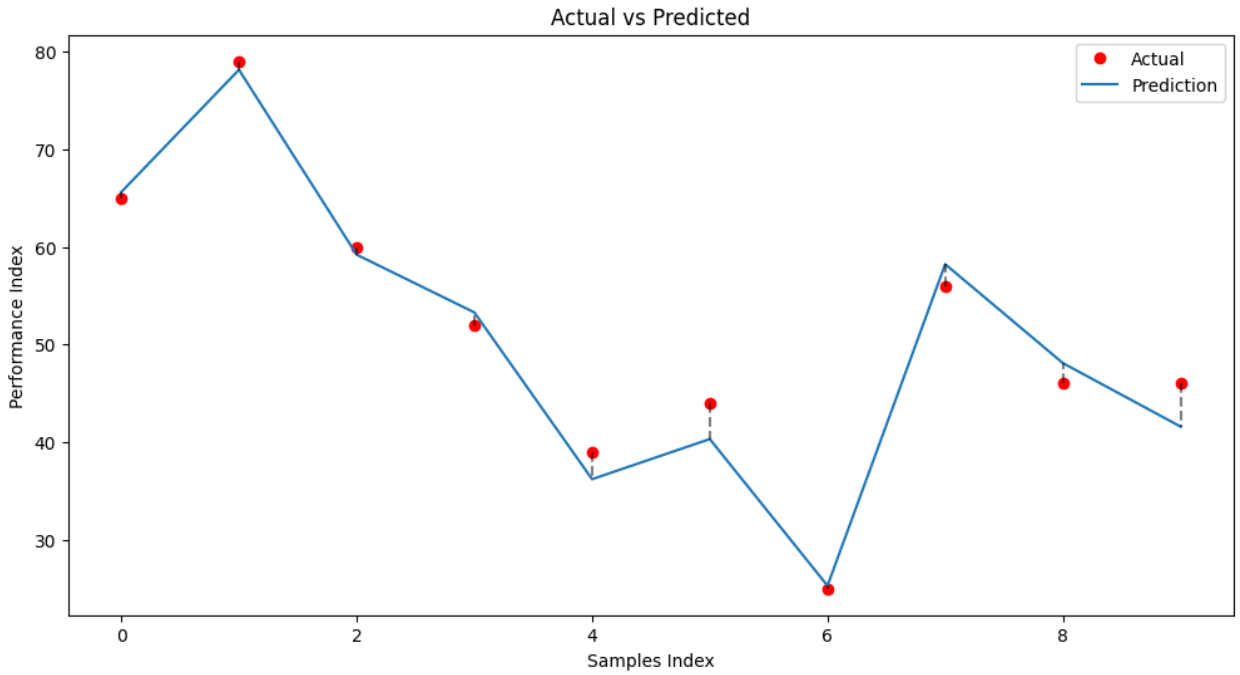
  - **Giá trị MAE:**

```
MAE with 2 best features (test_set):  1.8400519999999998
```

  - **Trực quan hóa giá trị dự đoán và giá trị thực tế:**



Trên toàn tập dữ liệu



Trên 10 dòng dữ liệu đầu

5.3. Mô hình 5

- **Ý tưởng:** Sử dụng mô hình hồi quy tuyến tính với hàm dự đoán như sau:

Student Performance = θ_0+ θ_1***Hours Studied*** + θ_2 * ***Previous Scores*** + θ_3 * ***Extracurricular*** + θ_4 * ***Sleep Hours*** + θ_5 * ***Sample Question Papers Practiced*** + θ_6***Hours Studied * Previous Scores***

- **Cài đặt:**
  - **Bước 1.** Tiền xử lý gồm thêm 1 đặc trưng *Hours Studied * Previous Scores* vào tập huấn luyện huấn luyện.

```
X_train_2c_v2['Hours Studied * Previous Scores'] = X_train_2c_v2['Hours Studied'] * X_train_2c_v2['Previous Scores']
X_test_2c_v2['Hours Studied * Previous Scores'] = X_test_2c_v2['Hours Studied'] * X_test_2c_v2['Previous Scores']
```

  - **Bước 2.** Huấn luyện mô hình.

```
k = 5
cross_val_results = []
lr = cross_validation_v2(X_train_2c_v2, y_train_2c_v2, k)
```

  - **Bước 3.** Tính giá trị dự đoán.

```
X_test_2c_v2 = preprocess(X_test_2c_v2)
y_pred_2c_v2 = predict_v2(X_test_2c_v2, lr)
```

- **Bước 4.** Tính MAE.

```
mae = MAE(y_test_2c_v2, y_pred_2c_v2)
```

- **Kết quả:**
  - **Công thức hồi quy:**

Student Performance = -34.036 + 2.866 * Hours Studied + 1.019 * Previous Scores + 0.604 * Extracurricular + 0.473 * Sleep Hours + 0.192 * Sample Question Papers Practiced + 0.0002 * (Hours Studied * Previous Scores)

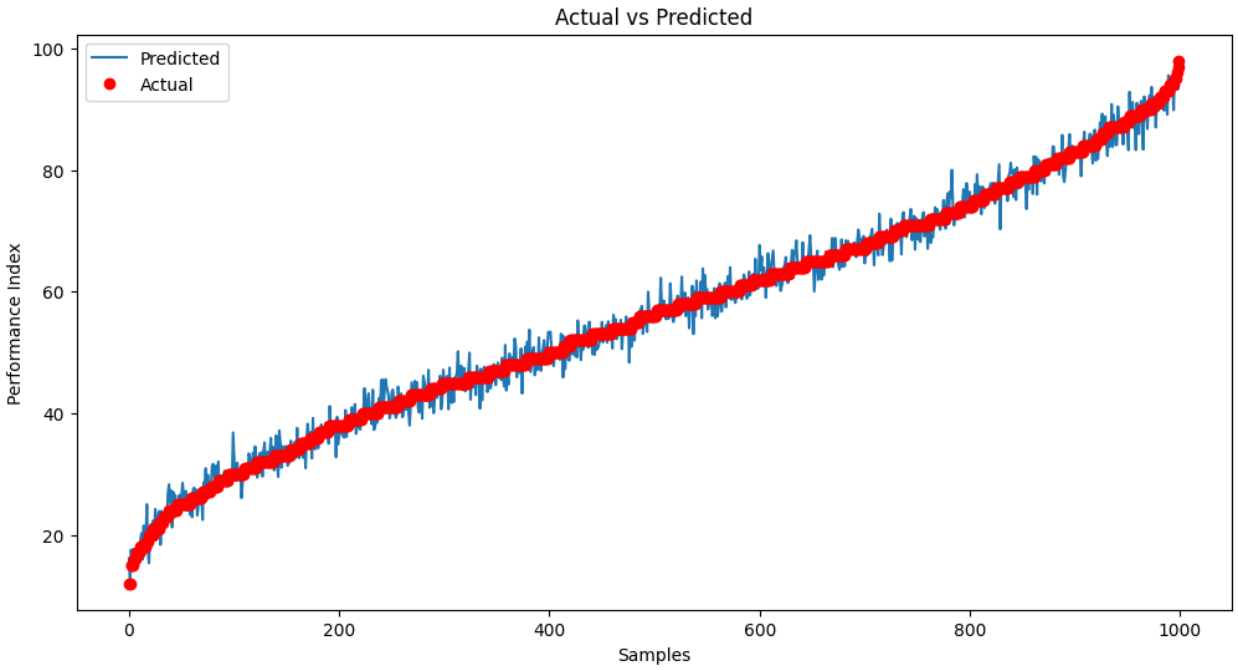
  - **Giá trị MAE:**

```
MAE (test_set):  1.594931
```

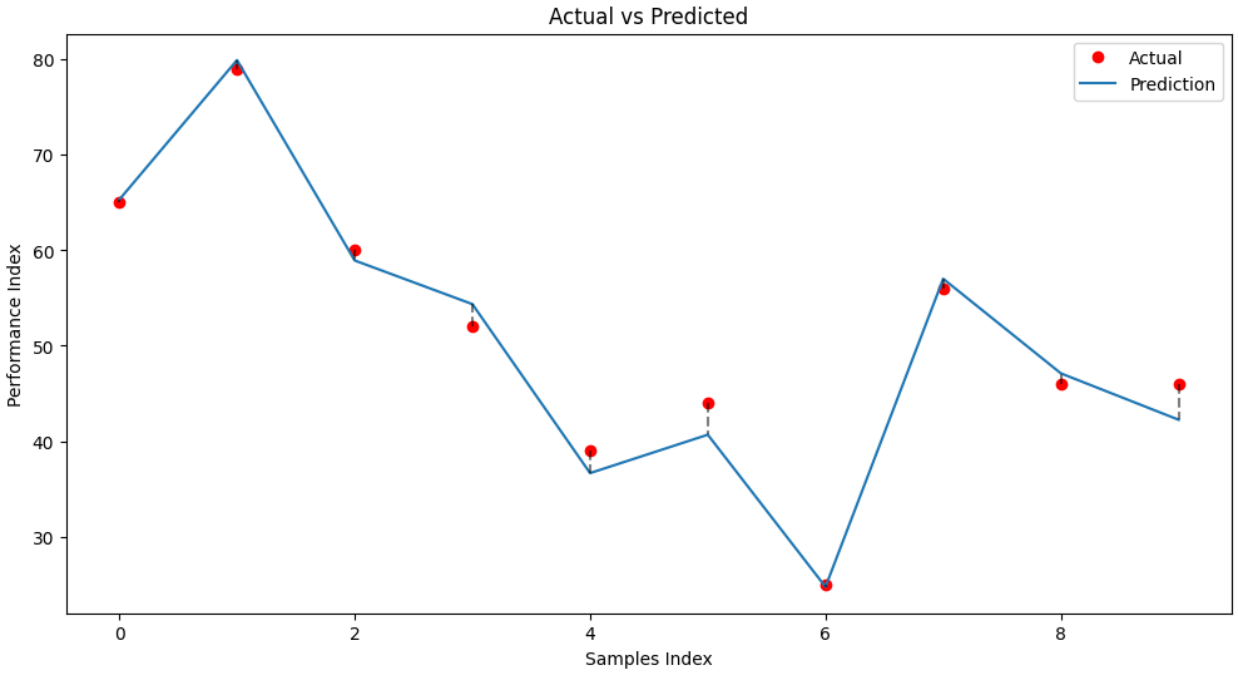
  - **Giá trị dự đoán:**

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Studied Hours * Sample Practiced	Performance Index (Prediction)
0	7.0	74.0	0.0	8.0	1.0	518.0	65.314126
1	6.0	89.0	1.0	8.0	8.0	534.0	79.947378
2	3.0	79.0	0.0	8.0	1.0	237.0	58.941804
3	7.0	62.0	0.0	8.0	8.0	434.0	54.364329
4	5.0	51.0	0.0	7.0	7.0	255.0	36.647328

  - **Trực quan hóa giá trị dự đoán và giá trị thực tế:**



Trên toàn tập dữ liệu



Trên 10 dòng dữ liệu đầu

5.4. Mô hình 6

- **Ý tưởng:** Sử dụng mô hình hồi quy tuyến tính với hàm dự đoán như sau:

Student Performance = θ_0+ θ_1***Hours Studied*** + θ_2 ***** ***Previous Scores*** + θ_3 ***** ***Extracurricular*** + θ_4 ***** ***Sleep Hours*** + θ_5 ***** ***Sample Question Papers Practiced*** + θ_6***Hours Studied** * **Previous Scores*** + θ_6***Previous** * **Previous Scores***

- **Cài đặt:**
  - **Bước 1.** Tiền xử lý gồm thêm 2 đặc trưng *Hours Studied* * *Previous Scores* và *Previous Scores* * *Previous Scores* vào tập huấn luyện huấn luyện.

```
X_train_2c_v3['Hours Studied * Previous Scores'] = X_train_2c_v3['Hours Studied'] * X_train_2c_v3['Previous Scores']
X_test_2c_v3['Hours Studied * Previous Scores'] = X_test_2c_v3['Hours Studied'] * X_test_2c_v3['Previous Scores']

X_test_2c_v3['Previous scores * Previous Scores'] = X_test_2c_v3['Previous Scores'] * X_test_2c_v3['Previous Scores']
X_train_2c_v3['Previous scores * Previous Scores'] = X_train_2c_v3['Previous Scores'] * X_train_2c_v3['Previous Scores']
```

  - **Bước 2.** Huấn luyện mô hình.

```
k = 5
cross_val_results = []
lr = cross_validation_v2(X_train_2c_v3, y_train_2c_v3, k)
```

  - **Bước 3.** Tính giá trị dự đoán.

```
X_test_2c_v3 = preprocess(X_test_2c_v3)
y_pred_2c_v3 = predict_v2(X_test_2c_v3, lr)
```

  - **Bước 4.** Tính MAE.

```
mae = MAE(y_test_2c_v2, y_pred_2c_v2)
```

- **Kết quả:**
  - **Công thức hồi quy:**

Student Performance = -34.391 + 2.879 * Hours Studied + 1.028 * Previous Scores + 0.602 * Extracurricular + 0.481 * Sleep Hours + 0.193 * Sample Question Papers Practiced + 0.0002 * (Hours Studied * Previous Scores)

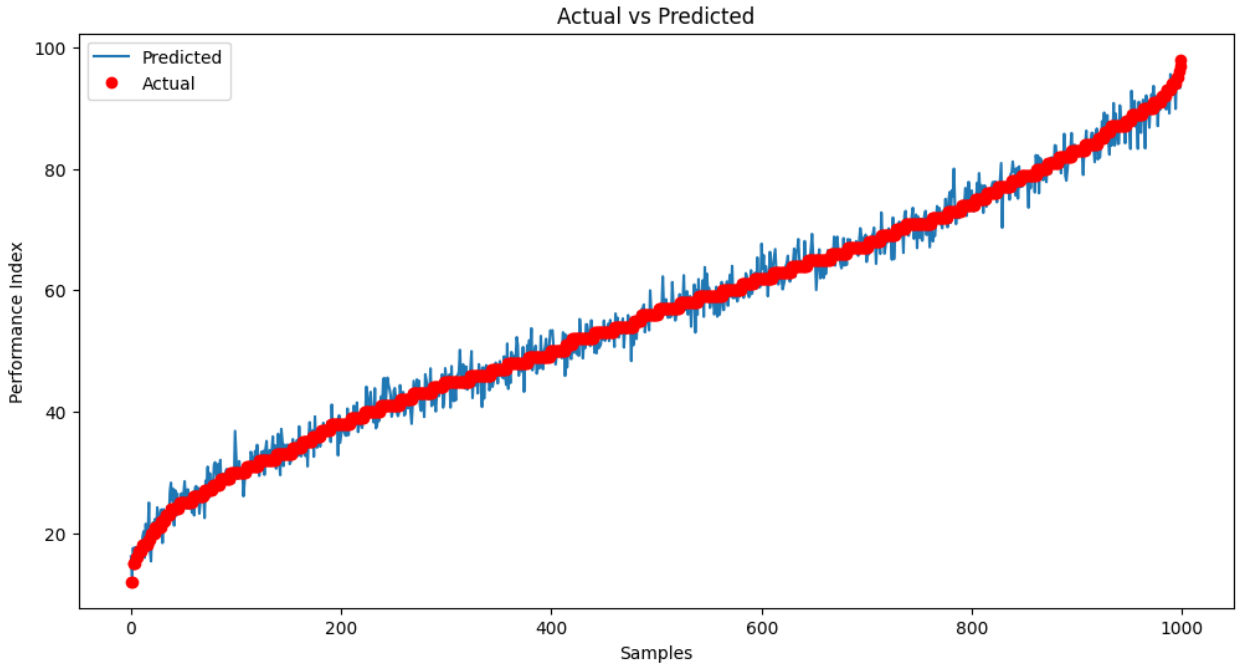
  - **Giá trị MAE:**

```
MAE (test_set):  1.5947060000000002
```

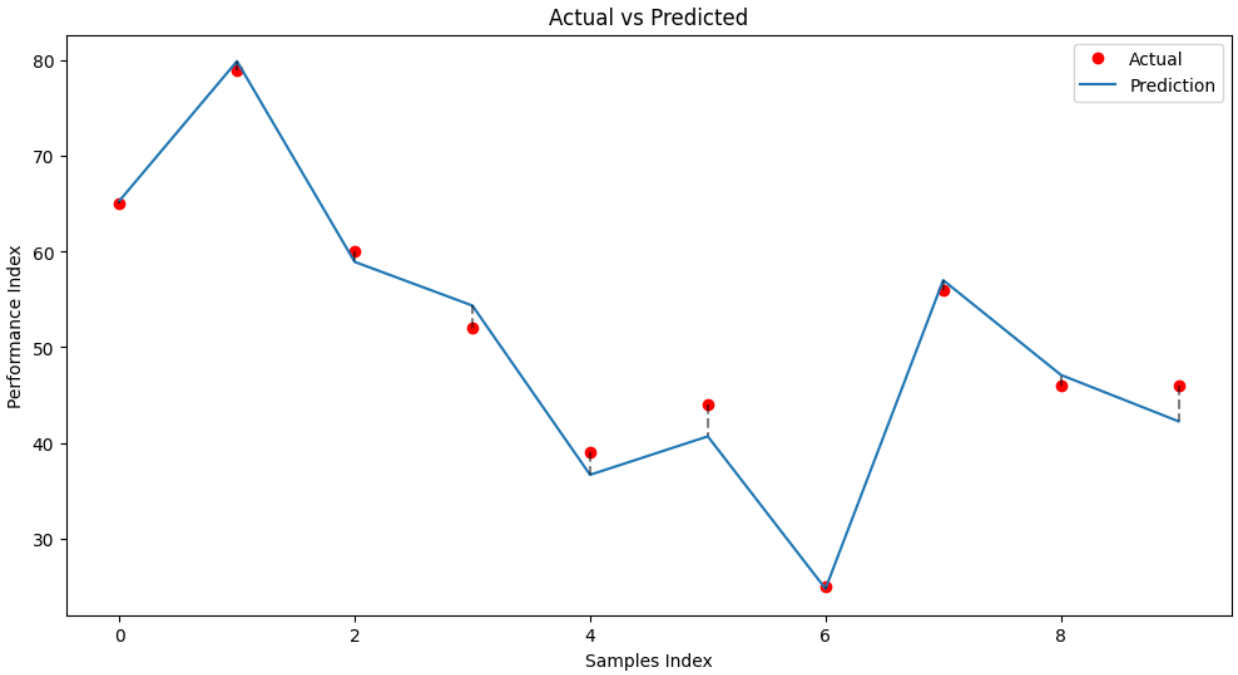
  - **Giá trị dự đoán:**

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Studied Hours * Sample Practiced	Performance Index (Prediction)
0	7.0	74.0	0.0	8.0	1.0	518.0	65.314126
1	6.0	89.0	1.0	8.0	8.0	534.0	79.947378
2	3.0	79.0	0.0	8.0	1.0	237.0	58.941804
3	7.0	62.0	0.0	8.0	8.0	434.0	54.364329
4	5.0	51.0	0.0	7.0	7.0	255.0	36.647328

- **Trực quan hóa giá trị dự đoán và giá trị thực tế:**



Trên toàn tập dữ liệu



Trên 10 dòng dữ liệu đầu

5.5. Mô hình tốt nhất

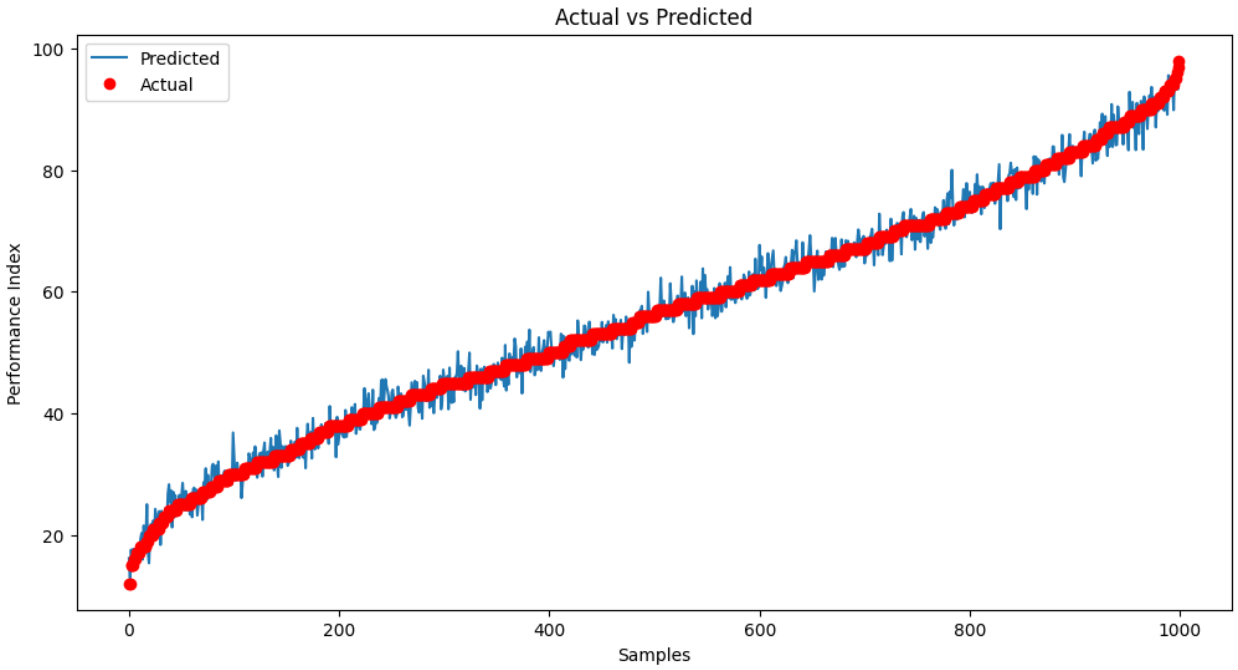
- Lựa chọn **mô hình 6** là mô hình tốt nhất để chạy trên toàn tập dữ liệu.
- **Kết quả:**
  - **Giá trị MAE:**

```
MAE (test_set):  1.595279
```

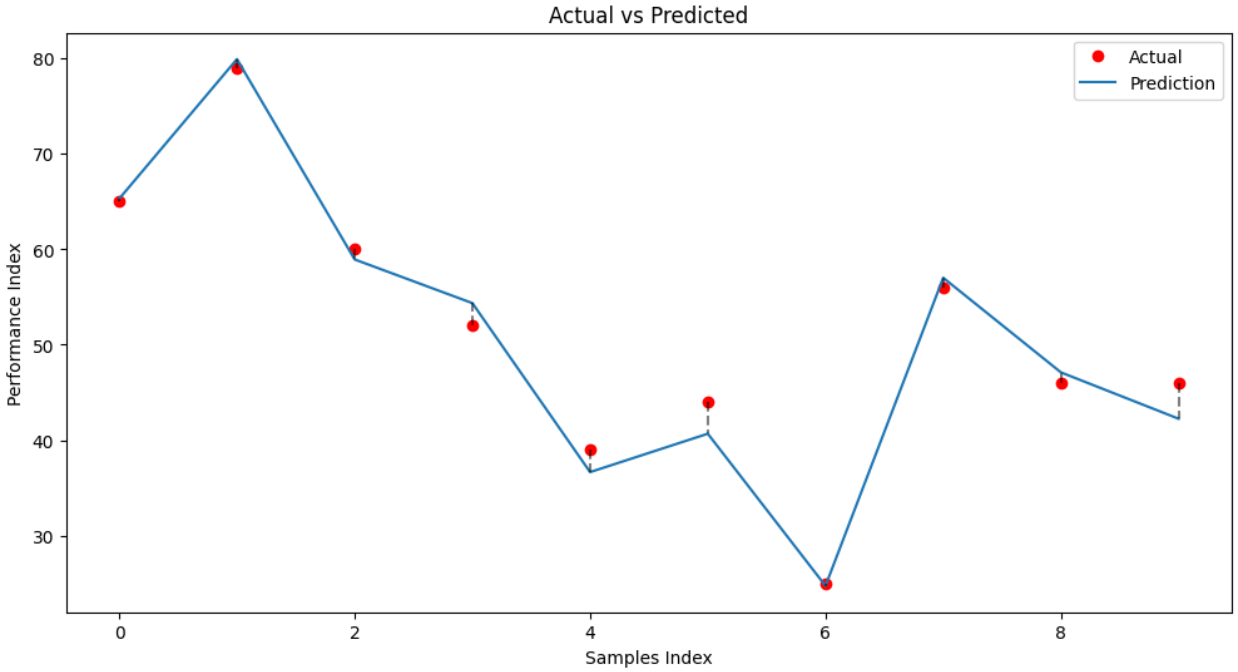
  - **Giá trị dự đoán:**

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Studied Hours * Sample Practiced	Sample Practiced * Sample Practiced	Performance Index (Prediction)
0	7.0	74.0	0.0	8.0	1.0	518.0	5476.0	65.316
1	6.0	89.0	1.0	8.0	8.0	534.0	7921.0	79.655
2	3.0	79.0	0.0	8.0	1.0	237.0	6241.0	58.998
3	7.0	62.0	0.0	8.0	8.0	434.0	3844.0	54.451
4	5.0	51.0	0.0	7.0	7.0	255.0	2601.0	36.861

  - **Trực quan hóa giữa Giá trị dự đoán và Giá trị thực tế:**



Trên toàn tập dữ liệu



Trên 10 dòng dữ liệu đầu

6. Mức độ hoàn thành:

STT	Công việc	Mức độ hoàn thành (%)
1	Phân tích và khám phá dữ liệu	100
2a	Xây dựng mô hình: Sử dụng toàn bộ 5 đặc trưng	100
2b	Xây dựng mô hình: Sử dụng duy nhất 1 đặc trưng	100
2b	Xây dựng mô hình với đặc trưng tốt nhất	100
2c	Xây dựng 3 mô hình	100
2c	Xây dựng mô hình tốt nhất	100

7. Tài liệu tham khảo:

1. https://machinelearningcoban.com/2016/12/28/linearregression/
2. Sách Deep Learning Cơ Bản – Nguyễn Thanh Tuấn – Chương 2: Machine Learning