MÔN DỮ LIỆU LỚN

ĐỒ ÁN CUỐI KÌ

Profit Prediction of companies

Linear Regression

GVHD : Ths. Nguyễn Hồ Duy Tri

Nhóm : 04

LỚP : IS405.N11.HTCL

THÀNH VIÊN

Nguyễn Cao Khoa

Hồ Trọng Khang

Lê Tuấn Khanh

Lê Tiến Vinh

NỘI DUNG

MỤC TIÊU & HƯỚNG TIẾP CẬN

01

Tổng quan về đề tài

- Vấn đề chi phí bỏ ra và lợi nhuận thu lại đang là vấn đề lớn đối với các công ty startup hiện tại để phát triển sản phẩm. Các công ty startup hiện tại đang đứng trên bờ vực phá sản vì thiếu hụt tài chính.
- Nhóm tiến hành nghiên cứu dự đoán lợi nhuận dựa trên chi phí bỏ ra của các công ty, giúp công ty có cái nhìn khách quan về thị trường tiêu hao hiện tại và đưa ra mức chi phí đầu tư phù hợp.

Mục tiêu

- Hiểu và nắm được dữ liệu bài toán.
- Có thể xử lý, làm sạch dữ liệu ban đầu thành nguồn dữ liệu có giá trị phục vụ giải quyết vấn đề đặt ra.
- Khám phá, phân tích và trực quan hóa tập dữ liệu.
- Tự xây dựng được mô hình **Linear Regression** để dự đoán lợi nhuận và chỉ số đánh giá độ chính xác của mô hình,
- So sánh, đánh giá được sự hiệu quả, tính chính xác, ưu nhược điểm giữa các mô hình **Linear Regression** tự xây dựng với thư viện Mllib.

Hướng tiếp cận

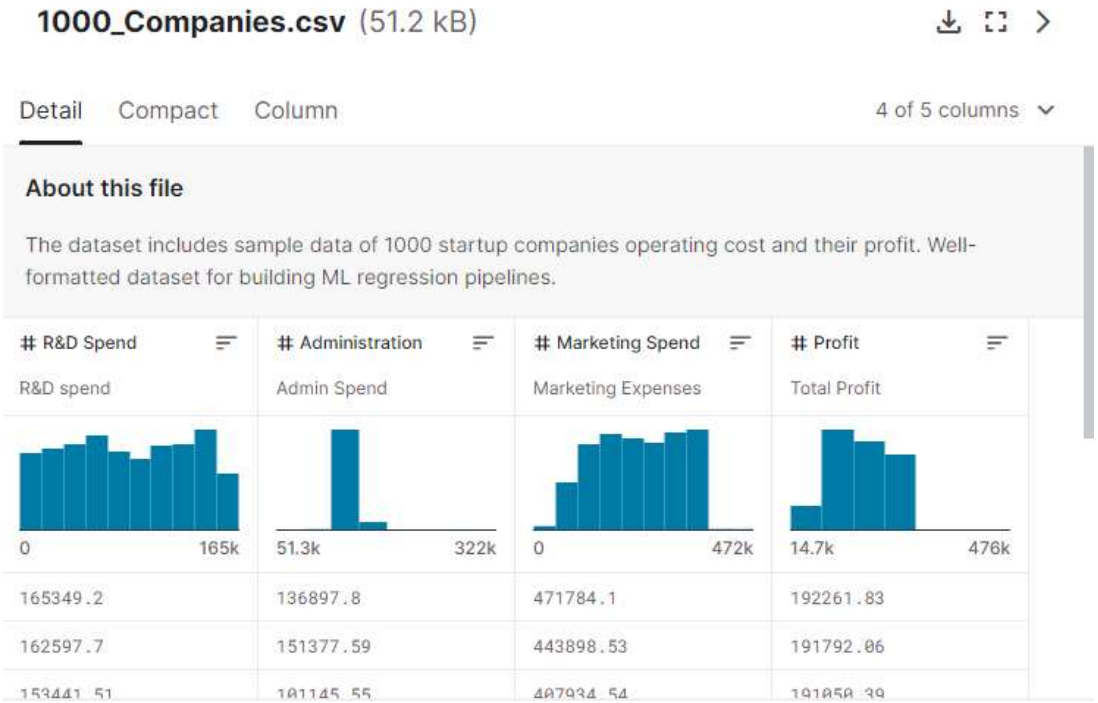
1. Xác định rõ mục tiêu trước và sau của dự án.
2. Tiến hành tìm nguồn và thu thập dữ liệu.
3. Thực hiện chọn và xử lý dữ liệu bằng cách lọc các dữ liệu bị lỗi, thiếu, và các dữ liệu ngoại lệ.
4. Tiến hành phân tích và khám phá dữ liệu bằng cách trực quan hóa, biểu diễn mối liên hệ giữa lợi nhuận với các thuộc tính khác.
5. Tiến hành xây dựng thuật toán và chỉ số đánh giá mô hình dự đoán lợi nhuận của các công ty.
6. So sánh kết quả, tính chính xác giữa các mô hình và tổng kết dự án.

NGUỒN & DỮ LIỆU THU THẬP

02

Nguồn gốc tập dữ liệu

- Dữ liệu được lấy từ **Kaggle** gồm 1000 công ty Startup về các chi phí đầu tư và lợi nhuận



Tổng quan tập dữ liệu

- Tên dataset: 1000_companies_profit
- Kích thước: 51.2 KB
- Loại tệp tin khi tải xuống từ Kaggle: .csv
- Tần suất cập nhật: Hàng tháng
- Link dataset: Companies_Profit

Tổng quan tập dữ liệu

Thống kê dataset

3.1.Kiểu dữ liệu các cột

```
[ ]  # Kiểu dữ liệu các cột
     df_com.printSchema()
```

```
root
 |-- R&D Spend: double (nullable = true)
 |-- Administration: double (nullable = true)
 |-- Marketing Spend: double (nullable = true)
 |-- State: string (nullable = true)
 |-- Profit: double (nullable = true)
```

Tổng quan tập dữ liệu

Thống kê dataset

3.2.Kiểm tra các thông tin cột

```
[ ]  # Mô tả các giá trị cột
     df_com.describe().show()
```

```
+-------+------------------+------------------+-----------------+----------+-----------------+
|summary|         R&D Spend|    Administration|  Marketing Spend|     State|           Profit|
+-------+------------------+------------------+-----------------+----------+-----------------+
|  count|              1000|              1000|             1000|      1000|             1000|
|   mean|        81668.9272|122963.89761169997|226205.0584188303|      null|119546.1646556102|
| stddev|46537.567891489154|12613.927534630999|91578.39354210423|      null|42888.63384847687|
|    min|               0.0|          51283.14|              0.0|California|          14681.4|
|    max|          165349.2|         321652.14|         471784.1|  New York|         476485.43|
+-------+------------------+------------------+-----------------+----------+-----------------+
```

XỬ LÝ & LÀM SẠCH DỮ LIỆU

03

Kiểm tra dữ liệu bị thiếu

3.3.Kiểm tra dữ liệu bị thiếu

```
[ ]  # Kiểm tra Missing Data trên df_com Dataframe
     df_com.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in df_com.columns]).show()
```

```
+---------+--------------+---------------+-----+------+
|R&D Spend|Administration|Marketing Spend|State|Profit|
+---------+--------------+---------------+-----+------+
|        0|             0|              0|    0|     0|
+---------+--------------+---------------+-----+------+
```

=> không có giá trị thiếu hay null

Phân loại đặc trưng (Features)

Chọn các cột cần cho việc dự đoán.

Các thuộc tính được chọn:

- R&D Spend: Là đặc trưng để dự đoán lợi nhuận.
- Profit: Là thuộc tính quyết định cho mô hình.

Kiểm tra dữ liệu các cột

RDSpend

3.6.Kiểm tra giá trị cột RDSpend

```
[ ]  # Kiểm tra các giá trị của RD Spend
     df.select('RDSpend').distinct().orderBy('R&D Spend').show(1000,False)
```

```
+---------+
|RDSpend  |
+---------+
|0.0      |
|542.05   |
|1000.23  |
|1215.0   |
|1269.0   |
|1315.46  |
|1634.0   |
|1681.0   |
|2296.0   |
|2307.0   |
|2603.0   |
|2626.0   |
|2638.0   |
|2682.0   |
|2696.0   |
|3089.0   |
|3111.0   |
|3372.0   |
|3574.0   |
|3737.0   |
|3797.0   |
```

Kiểm tra dữ liệu các cột

Profit

```
# Kiem tra cac gia tri cua Profit
df.select('Profit').distinct().orderBy('Profit').show(1000,False)
```

```
+-----------+
|Profit     |
+-----------+
|14681.4    |
|35673.41   |
|42559.73   |
|49490.75   |
|50070.86316|
|50116.99489|
|50428.81124|
|50468.96294|
|50994.35213|
|51003.74933|
|51256.61958|
|51276.26828|
|51286.51978|
|51324.1086 |
|51336.06868|
|51671.80519|
|51690.5996 |
|51913.56964|
|52086.1365 |
|52225.38599|
|52276.64348|
|52325.33808|
|52481.67341|
|52609.81711|
|52731.98078|
|53177.92087|
|53225.76119|
|53395.76517|
|53483.75718|
```

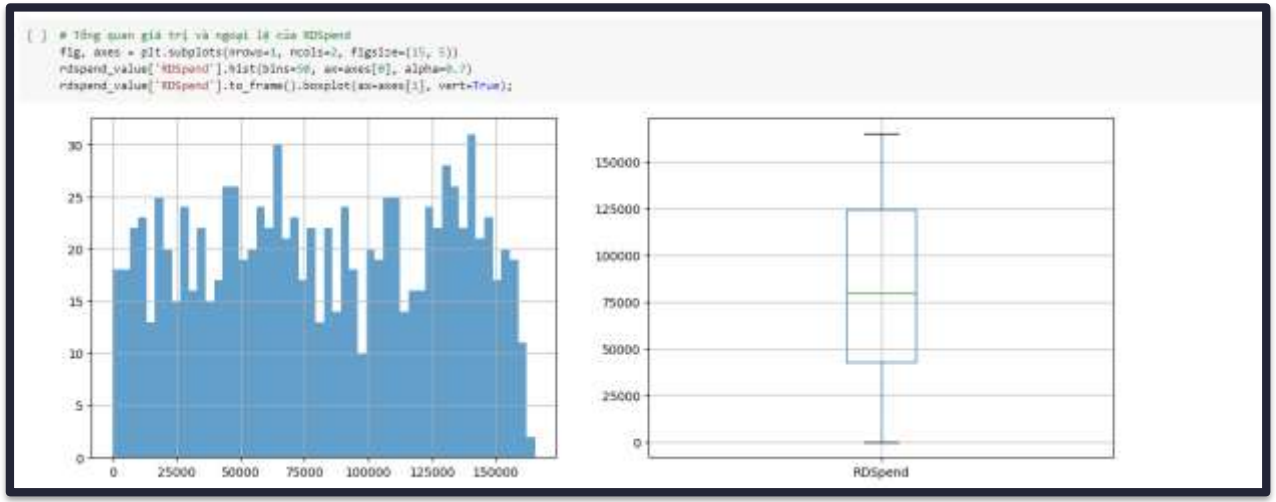
Kiểm tra Outliers

RDSpend

1. Sử dụng hàm map để lấy giá trị của cột RDSpend và đưa vào 1 Dataframe
2. Trực quan hóa giá trị tổng quan và ngoại lệ của thuộc tính RDSpend

```
[ ]  # Lấy các giá trị của RDSpend
     rdspend = df.rdd.map(lambda p: (p.RDSpend)).collect()
```

```
[ ]  # Đưa các giá trị vừa lấy vào dataframe
     rdspend_value = DataFrame({'RDSpend':rdspend})
```



Kiểm tra Outliers

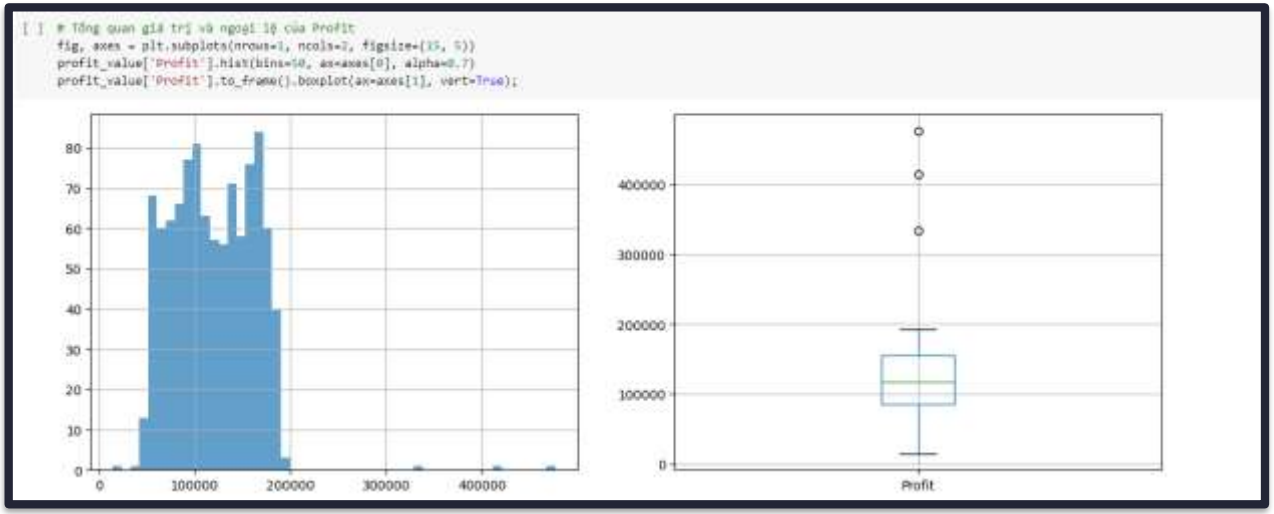
Profit

1. Sử dụng hàm map để lấy giá trị của cột Profit và đưa vào 1 Dataframe
2. Trực quan hóa giá trị tổng quan và ngoại lệ của thuộc tính Profit

```
[ ]  # Lấy các giá trị của Profit
     profit = df.rdd.map(lambda p: (p.Profit)).collect()
```

```
[ ]  # Đưa các giá trị vừa lấy vào dataframe
     profit_value = DataFrame({'Profit':profit})
```

```
[ ]  # Tổng quan giá trị và ngoại lệ của Profit
     fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(15, 5))
     profit_value['Profit'].hist(bins=50, ax=axes[0], alpha=0.7)
     profit_value['Profit'].to_frame().boxplot(ax=axes[1], vert=True);
```



Kiểm tra Outliers

Profit

Kiểm tra cột Profit với dữ liệu > 300000

```
[ ]  df.filter((df['Profit'] > 300000)).orderBy('Profit').show(10,True)
```

```
+---------+---------+
|  RDSpend|   Profit|
+---------+---------+
|128456.23|333962.19|
|100275.47|413956.48|
|161181.72|476485.43|
+---------+---------+
```

PHÂN TÍCH & KHÁM PHÁ DỮ LIỆU

04

Trực quan hóa dữ liệu

Lấy ra các giá trị của RDSpend và Profit và đưa vào Dataframe.

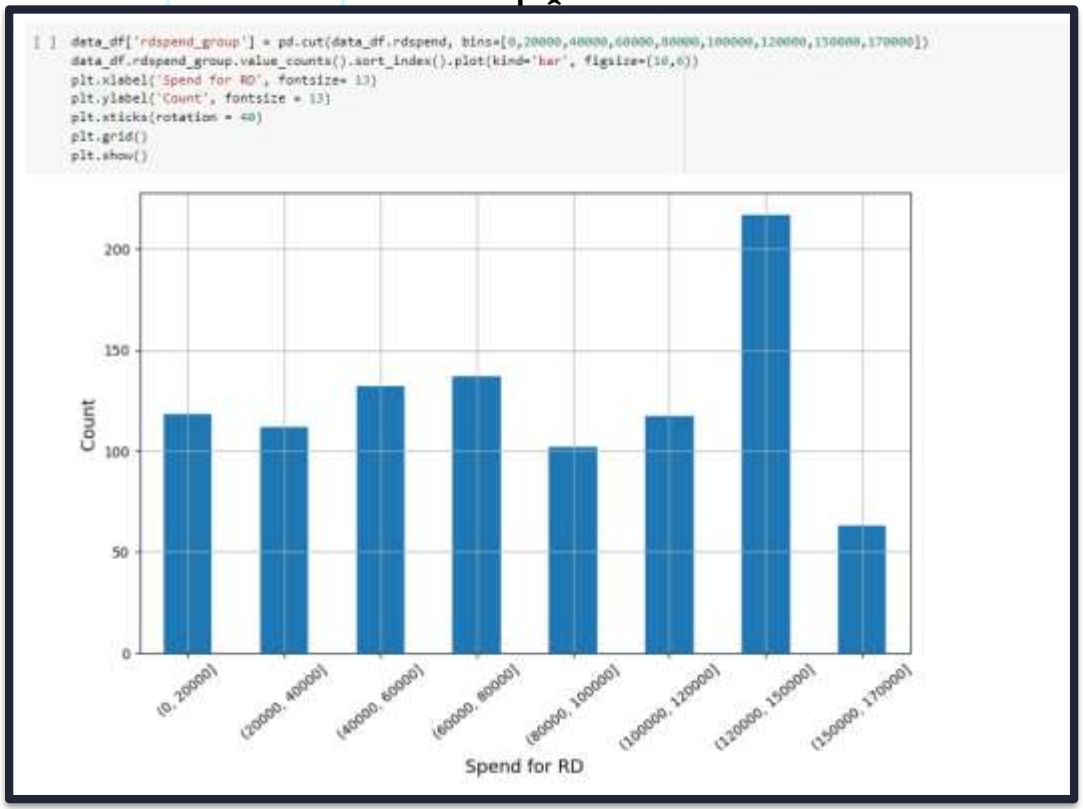
```
[ ]  # Lấy các giá trị của RDSpend và Profit
     x = df.rdd.map(lambda p: (p.RDSpend)).collect()
     y = df.rdd.map(lambda p: (p.Profit)).collect()
```

```
[ ]  # Đưa các giá trị vừa lấy vào dataframe
     data_df = DataFrame({'rdspend': x, 'profit': y})
```

Trực quan hóa dữ liệu

Trực quan hóa các chi phí bỏ ra để nghiên cứu và phát triển sản

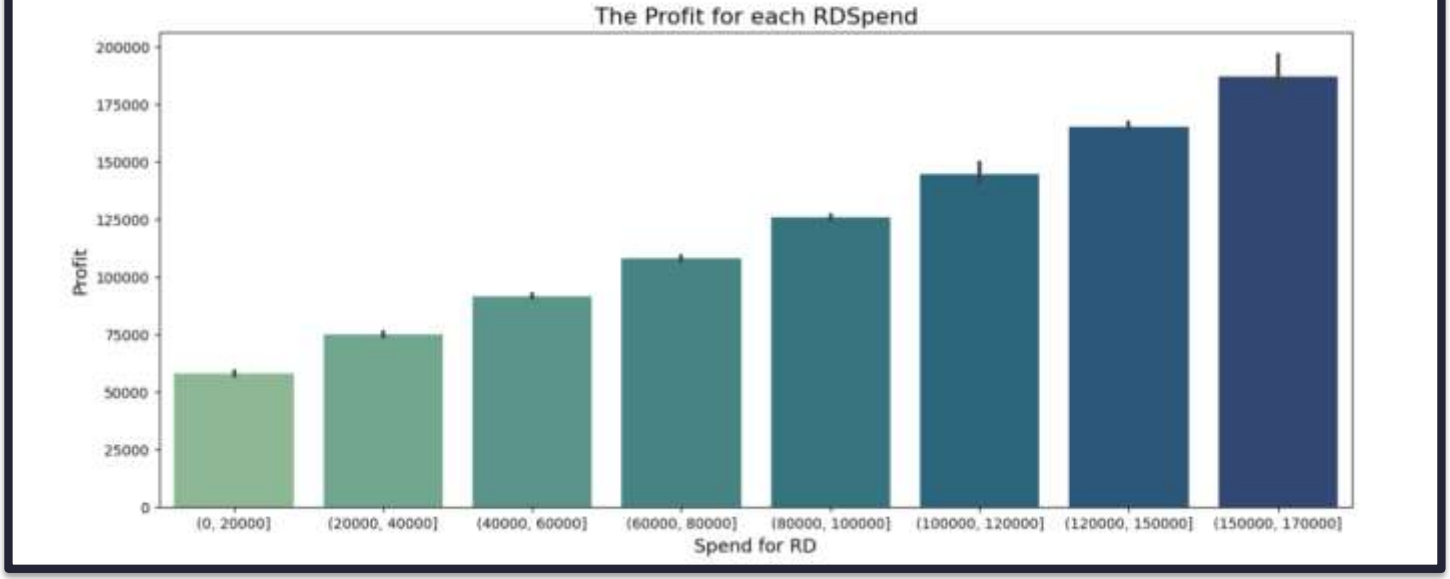
```
[ ] data_df['rdspend_group'] = pd.cut(data_df.rdspend, bins=[0,20000,40000,60000,80000,100000,120000,150000,170000])
    data_df.rdspend_group.value_counts().sort_index().plot(kind='bar', figsize=(10,6))
    plt.xlabel('Spend for RD', fontsize= 13)
    plt.ylabel('Count', fontsize = 13)
    plt.xticks(rotation = 40)
    plt.grid()
    plt.show()
```



Trực quan hóa dữ liệu

Trực quan hóa giữa chi phí (RDSpend) và lợi nhuận (Profit)

```
[ ] plt.figure(figsize=(15, 6))
    sns.barplot(data = data_df, x='rdspend_group', y='profit', palette='crest')
    plt.title('The Profit for each RDSpend', fontsize= 16)
    plt.xlabel('Spend for RD', fontsize= 13)
    plt.ylabel('Profit', fontsize = 13)
    plt.show();
```



CHUẨN BỊ DỮ LIỆU

05

Chuẩn bị dữ liệu

Phân chia dữ liệu thành 2 phần Feature(X) và Target(Y)

```
[ ]  df.show()

+---------+---------+
|  RDSpend|   Profit|
+---------+---------+
| 165349.2|192261.83|
| 162597.7|191792.06|
```

```
[ ]  y.show(10)

+---------+
|   Profit|
+---------+
|192261.83|
|191792.06|
|191050.39|
|182901.99|
|166187.94|
|156991.12|
|156122.51|
| 155752.6|
|152211.77|
|149759.96|
+---------+
only showing top 10 rows
```

```
[ ]  # Cắt dữ liệu thành hai phần X và y.
     df.createOrReplaceTempView("split_data")
     X = spark.sql("SELECT DOUBLE(RDSpend) FROM split_data")
     y = spark.sql("SELECT DOUBLE(Profit) FROM split_data")
```

Chuẩn bị dữ liệu

Chia dữ liệu Test Train theo tỉ lệ Test 30% và Train 70%

```
[ ]  X_train, X_test = X.randomSplit([0.7, 0.3], seed = 2)
```

```
[ ]  X_train.describe().show()

+-------+------------------+
|summary|          RDSpend|
+-------+------------------+
|  count|               698|
|   mean|  81301.123252149|
| stddev|46348.20018060621|
|    min|               0.0|
|    max|          165349.2|
+-------+------------------+
```

```
[ ]  y_train.describe().show()

+-------+-----------------+
|summary|           Profit|
+-------+-----------------+
|  count|              698|
|   mean|119157.5400676075|
| stddev| 42650.1715799624|
|    min|          14681.4|
|    max|         476485.43|
+-------+-----------------+
```

```
[ ]  y_test.describe().show()

+-------+------------------+
|summary|           Profit|
+-------+------------------+
|  count|               302|
|   mean|120444.37645172192|
| stddev|43492.754490076855|
|    min|         50070.86316|
|    max|         413956.48|
+-------+------------------+
```

```
[ ]  X_test.describe().show()

+-------+-----------------+
|summary|          RDSpend|
+-------+-----------------+
|  count|              302|
|   mean|82519.01711920531|
| stddev|47038.62548681213|
|    min|           1215.0|
|    max|         162597.7|
+-------+-----------------+
```

XÂY DỰNG THUẬT TOÁN

06

Phương trình mô quy tuyến tính

Mô hình hồi quy tuyến tính đơn giản là một đường được xác định bởi các hệ số được ước tính từ dữ liệu huấn luyện. Sau khi các hệ số được ước tính, chúng ta có thể sử dụng chúng để đưa ra dự đoán.

Phương trình đưa ra dự đoán với mô hình **hồi quy tuyến tính** như sau:

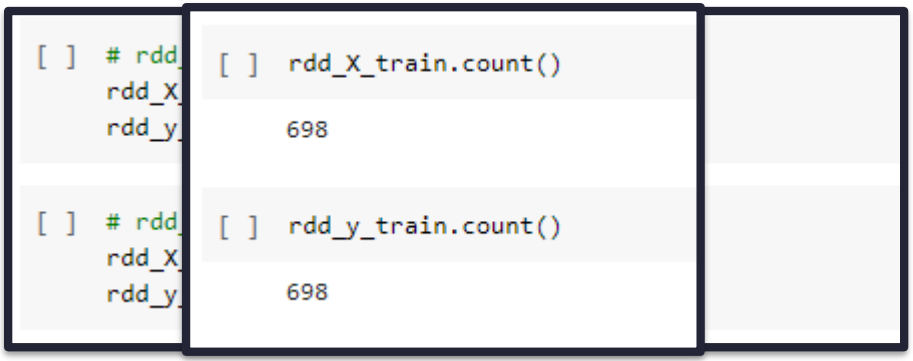
Công thức: $y = b0 + b1 * x$

Trong đó:

- B0 là chỉ số chặn của đường hồi quy.
- B1 là hệ số góc của đường hồi quy.

Hướng làm 1

1. Tạo mô hình hồi quy tuyến tính – Linear Regression



Đưa dữ liệu Test và Train lên RDD theo dạng list để tiến hành xử lý

HƯỚNG LÀM 1

Hướng làm 1

1. Tạo mô hình hồi quy tuyến tính – Linear Regression

Xây dựng hàm tính trung bình của cả biến đầu vào và đầu ra từ training data

Giá trị trung bình
Công thức:

$$\bar{x} = \frac{sum(x)}{count(x)}$$

```
[ ]  # Tính giá trị trung bình của danh sách số
     def mean(data):
         sum = data.reduce(lambda a, b: a + b)
         return sum / data.count()
```

Hướng làm 1

1. Tạo mô hình hồi quy tuyến tính – Linear Regression

Tính hệ số B1

Công thức tổng tính hệ số B1:

$$B1 = \frac{\sum_{i=1}^{m}(xi - \bar{x})(yi - \bar{y})}{\sum_{i=1}^{m}(xi - \bar{x})^2}$$

Công thức tính mẫu số B1:

$$\sum_{i=1}^{m}(xi - \bar{x})^2$$

```
[ ]  # Tính mẫu số B1 (Slope)
     def deno_slope(data, mean):
         deno_slope = data.map(lambda a: (a - mean) ** 2)
         result = deno_slope.reduce(lambda a, b: a + b)
         return result
```

Hướng làm 1

1. Tạo mô hình hồi quy tuyến tính – Linear Regression

Tính hệ số B1

```
[ ]  # Chuyển rdd_X_train thành danh sách các giá trị
     rdd_X_train_1 = rdd_X_train.map(lambda a: a[0])

[ ]  # Danh sách các giá trị
```

```
[ ]  print(">>>> X_train: Giá trị trung bình - Mean = " + str(mean_X_train) + "; Mẫu số B1 - Deno Slope = " + str(deno_X_train))
     print(">>>> y_train: Giá trị trung bình - Mean = " + str(mean_y_train) + "; Mẫu số B1 - Deno Slope = " + str(deno_y_train))

>>>> X_train: Giá trị trung bình - Mean = 81301.123252149; Mẫu số B1 - Deno Slope = 1497264495007.14
>>>> y_train: Giá trị trung bình - Mean = 119157.5400676075; Mẫu số B1 - Deno Slope = 1267868883652.7605
```

Tính mẫu số và Mean(y)

```
     42559.73,
     49490.75,
     50116.99489,
     50428.81124,
     50468.96294,
     51003.74933,
     51256.61958,
     51276.26828]

[ ]  # Giá trị trung bình của y_train
     mean_y_train = mean(rdd_y_train_1)

[ ]  # Giá trị mẫu số của y_train
     deno_y_train = deno_slope(rdd_y_train_1, mean_y_train)
```

Hướng làm 1

1. Tạo mô hình hồi quy tuyến tính – Linear Regression

Tính hệ số B1

Công thức tổng tính hệ số B1:

$$B1 = \frac{\sum_{i=1}^{m}(xi - \bar{x})(yi - \bar{y})}{\sum_{i=1}^{m}(xi - \bar{x})^2}$$

```
[ ]  # Giá trị tử số của X và y
     nume_slope_xy = nume_slope(rdd_X_train_1, mean_X_train, rdd_y_train_1, mean_y_train)
     print(">>>> Tử số B1 - Nume Slope = " + str(nume_slope_xy))

     >>>> Tử số B1 - Nume Slope = 1320279826734.3691
```

Công thức tính tử số B1:

$$\sum_{i=1}^{m}(xi - \bar{x})(yi - \bar{y})$$

```
[ ]  # Tính tử số B1 (Slope)
     def nume_slope(X, mean_X, y, mean_y):
         result = 0.0
         common_rdd = rdd_X_train_1.zip(rdd_y_train_1)
         nume_slope = common_rdd.map(lambda a: (a[0] - mean_X) * (a[1] - mean_y))
         result += nume_slope.reduce(lambda a, b: a + b)
         return result
```

Hướng làm 1

1. Tạo mô hình hồi quy tuyến tính – Linear Regression

Tính hệ số B0 và B1

Công thức tính B0: $$B0 = \bar{y} - B1 * \bar{x}$$

Công thức tính hệ số:

```
[ ]  # Tính toán các hệ số B0 và B1
     def coefficients(X, y):
         X_mean, y_mean = mean(X), mean(y)
         b1 = nume_slope(X, X_mean, y, y_mean) / deno_slope(X, X_mean)
         b0 = y_mean - b1 * X_mean
         return [b0, b1]
```

```
[ ]  # Hệ số của mô hình
     coef = coefficients(rdd_X_train_1, rdd_y_train_1)
     print(">>>> Hệ số - B0, B1 = " + str(coef))

     >>>> Hệ số - B0, B1 = [47466.64425153793, 0.8817946536079941]
```

Hướng làm 1

1. Tạo mô hình hồi quy tuyến tính – Linear Regression

Xây dựng hàm đoán lợi nhuận dựa trên mô hình hồi quy tuyến tính

```
[ ]  # Mô hình hồi quy tuyến tính để dự đoán lợi nhuận
     def simple_linear_regression(X_train, y_train, X_test):
         coef = coefficients(X_train, y_train)
         b0 = coef[0]
         b1 = coef[1]
         yhat = X_test.map(lambda a: a * b1 + b0)
         return yhat
```

Hướng làm 1

2. Ứng dụng mô hình hồi quy tuyến tính trên tập dữ liệu Test

Chuyển RDD Test thành danh sách các giá trị (dạng List)

```
[ ]  # Chuyển rdd_X_test thành danh sách các giá trị
     rdd_X_test_1 = rdd_X_test.map(lambda a: a[0])
```

Hướng làm 1

2. Ứng dụng mô hình hồi quy tuyến tính trên tập dữ liệu Test

Thời gian chạy của mô hình

```
[ ]  # Prediction lưu trữ kết quả dự đoán lợi nhuận
     start = time.time()
     prediction = simple_linear_regression(rdd_X_train_1, rdd_y_train_1, rdd_X_test_1)
     end = time.time()
     run_time_HL1 = end - start
     print(">>>> Run time:" + str(run_time_HL1) + "(s)")

     >>>> Run time:0.6315128803253174(s)
```

Hướng làm 1

2. Ứng dụng mô hình hồi quy tuyến tính trên tập dữ liệu Test

Kết quả dự đoán của mô hình

```
[ ]  # Xuất kết quả dự đoán lợi nhuận trên mô hình hồi quy tuyến tính
     prediction_result = prediction.collect()
     DataFrame({'prediction_predict':prediction_result})
```

	prediction_predict
0	48538.024756
1	48948.941064
2	49831.617513
3	50190.507937
4	50761.910872
...	...
297	187840.416954
298	187844.825927
299	188092.610225
300	188329.812987
301	190844.426800

302 rows × 1 columns

Hướng làm 1

2. Ứng dụng mô hình hồi quy tuyến tính trên tập dữ liệu Test

Lấy giá trị của y_test

```
[ ]  # Chuyển rdd_y_test thành danh sách các giá trị
     rdd_y_test_1 = rdd_y_test.map(lambda a: a[0])

[ ]  rdd_y_test_1.take(10)

     [50070.86316,
      50994.35213,
      51336.06868,
      51690.5996,
      52276.64348,
      53225.76119,
      53649.48971,
      54205.63339,
      54244.9308,
      54932.63535]
```

Hướng làm 1

3. Tạo và ứng dụng chỉ số đánh giá RMSE – Root Mean Square Error

Ước tính Sai số toàn phương trung bình căn – Root Mean Square Error.

Công thức:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(y_1 - y_2)^2}{n}}$$

Trong đó:

y_1: giá trị ước lượng

y_2: biến độc lập

n = (N – k – 1)

N: số tổng lượng quan sát

k: tổng lượng biến

```
[ ]  from math import sqrt
     # Hàm tính toán RMSE
     def rmse_metric(actual, predicted):
         common_rdd = actual.zip(predicted)
         prediction_error = common_rdd.map(lambda a: a[0] - a[1])
         # prediction_error.collect()
         prediction_error_square = prediction_error.map(lambda a: a ** 2)
         sum_error = prediction_error_square.reduce(lambda a, b: a + b)
         # print(sum_error)
         mean_error = sum_error / float(actual.count())
         return sqrt(mean_error)
```

Hướng làm 1

3. Tạo và ứng dụng chỉ số đánh giá RMSE – Root Mean Square Error

```
[ ]  # Hàm đánh giá thuật toán hồi quy trên tập dữ liệu huấn luyện
     def evaluate_algorithm(X_train, y_train, X_test, y_test, algorithm):
         predicted = algorithm(X_train, y_train, X_test)
         rmse = rmse_metric(y_test, predicted)
         return rmse
```

Hàm đánh giá thuật toán trên tập huấn luyện

```
[ ]  # Đánh giá thuật toán hồi quy trên tập dữ liệu huấn luyện
     evaluate_model_HL1 = evaluate_algorithm(rdd_X_train_1, rdd_y_train_1, rdd_X_test_1, rdd_y_test_1, simple_linear_regression)
     print(">>>> Root Mean Squared Error = " + str(evaluate_model_HL1))

     >>>> Root Mean Squared Error = 12920.51372601141
```

Đánh giá thuật toán

Hướng làm 1

4. Trực quan hóa và so sánh kết quả với dự đoán thực tế

```
[ ]  # So sánh giữa giá trị Profit dự đoán và giá trị gốc
     compare = DataFrame({'Origin': y_test_origin, 'Predict': prediction_result})
     compare.head(10)
```

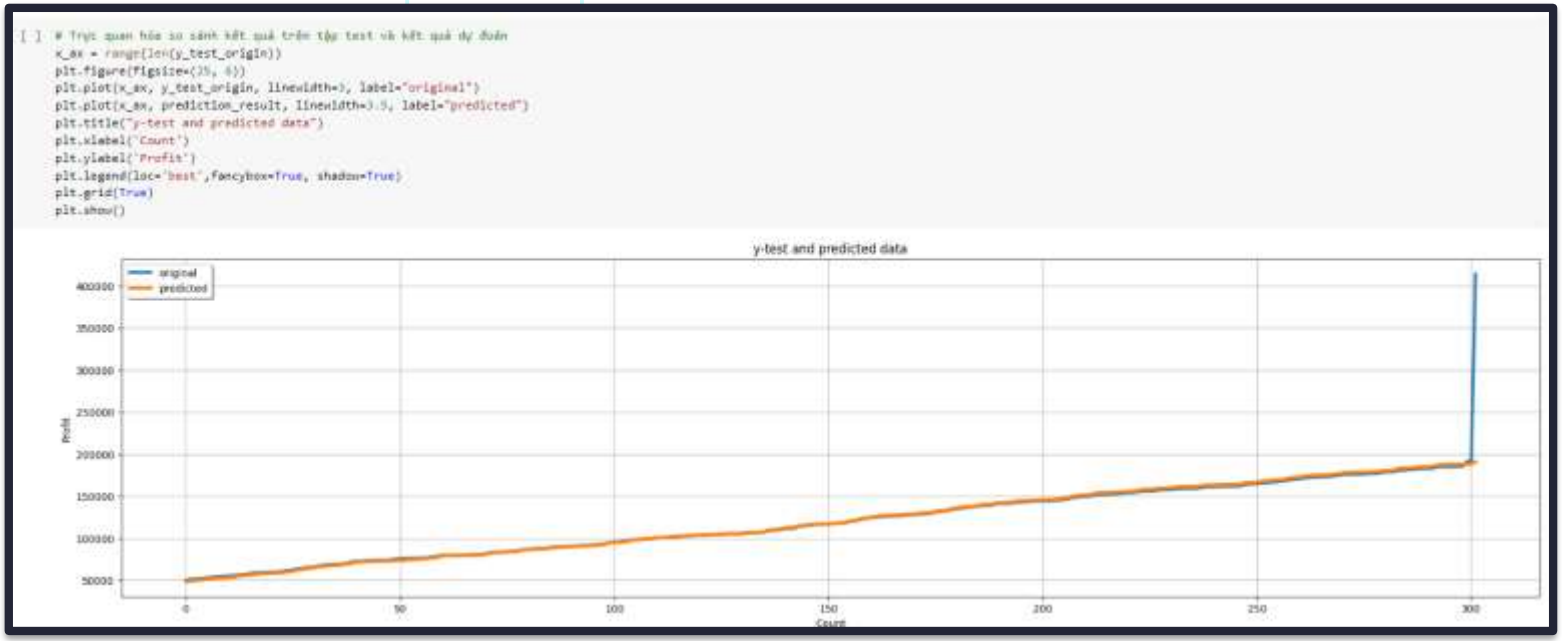
	Origin	Predict
0	50070.86316	48538.024756
1	50994.35213	48948.941064
2	51336.06868	49831.617513
3	51690.59960	50190.507937
4	52276.64348	50761.910872
5	53225.76119	51745.111911
6	53649.48971	52060.794397
7	54205.63339	52716.849619
8	54244.93080	52805.910879
9	54932.63535	52903.790086

So sánh kết quả dự đoán trên tập test và tập dự đoán

Hướng làm 1

4. Trực quan hóa và so sánh kết quả với dự đoán thực tế

```
[ ] # Trực quan hóa so sánh kết quả trên tập test và kết quả dự đoán
    x_ax = range(len(y_test_origin))
    plt.figure(figsize=(15, 4))
    plt.plot(x_ax, y_test_origin, linewidth=3, label="original")
    plt.plot(x_ax, prediction_result, linewidth=3.5, label="predicted")
    plt.title("y-test and predicted data")
    plt.xlabel('Count')
    plt.ylabel('Profit')
    plt.legend(loc='best',fancybox=True, shadow=True)
    plt.grid(True)
    plt.show()
```

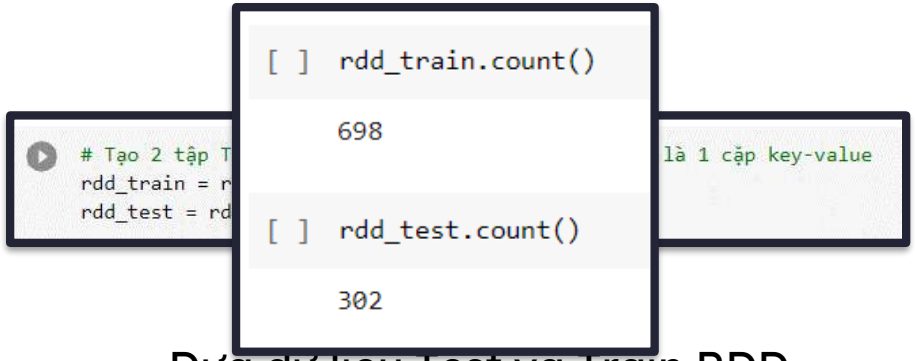


Trực quan hóa giá trị dự đoán và giá trị thực tế

HƯỚNG LÀM 2

Hướng làm 2

1. Tạo mô hình hồi quy tuyến tính – Linear Regression



Đưa dữ liệu Test và Train RDD theo dạng list từng cặp các **key-value** để tiến hành xử lý

Hướng làm 2

1. Tạo mô hình hồi quy tuyến tính – Linear Regression

Với Phương trình của **đường hồi quy** được biểu diễn như sau:

$$h(x_i) = \beta_0 + \beta_1 . x_i \quad (1)$$

Chúng ta cần phải tính:

$$\beta_1 = \frac{SS_{xy}}{SS_{xx}} \quad | \quad \beta_0 = \bar{y} - \beta_1 . \bar{x}$$

và

$$SS_{xy} = \sum_{i=1}^{n} y_i x_i - n . \bar{x} . \bar{y} \quad | \quad SS_{xx} = \sum_{i=1}^{n} (x_i)^2 - n (\bar{x}_i)^2$$

Công thức tính phương trình đường hồi quy và các hệ số liên quan

Hướng làm 2

1. Tạo mô hình hồi quy tuyến tính – Linear Regression

Với Phương trình của **đường hồi quy** được biểu diễn như sau:

$$h(x_i) = \beta_0 + \beta_1 . x_i \quad (1)$$

Chúng ta cần phải tính:

$$\beta_1 = \frac{SS_{xy}}{SS_{xx}} \quad | \quad \beta_0 = \bar{y} - \beta_1 . \bar{x}$$

và

$$SS_{xy} = \sum_{i=1}^{n} y_i x_i - n . \bar{x} . \bar{y} \quad | \quad SS_{xx} = \sum_{i=1}^{n} (x_i)^2 - n (\bar{x}_i)^2$$

```
[ ]  # Tính toán giá trị trung bình của mảng
     # Đầu vào sẽ là 1 tuple (key-value)
     def estimate_coef(data):
         # Số lượng phần tử trong m
         n = data.count()

         # Trung bình của mảng bao gồm các keys
         sum_x = data.keys().reduce(lambda a, b: a + b)
         mean_x = sum_x/n
         # Trung bình của mảng bao gồm các values
         sum_y = data.values().reduce(lambda a, b: a + b)
         mean_y = sum_y/n

         # Tính toán giá trị cross-deviation (độ lệch chéo) và deviation của x và y
         sum_xy = data.map(lambda x: (x[0],x[0]*x[1])).values().reduce(lambda x,y: x+y)
         SS_xy = sum_xy - n*mean_x*mean_y

         sum_xx = data.map(lambda x: (x[0],x[0]*x[0])).values().reduce(lambda x,y: x+y)
         SS_xx = sum_xx - n*mean_x*mean_x

         # Tính toán hệ số hồi quy
         b_1 = SS_xy / SS_xx
         b_0 = mean_y - b_1*mean_x

         return (b_0, b_1)
```

Hướng làm 2

1. Tạo mô hình hồi quy tuyến tính – Linear Regression

Tính hệ số B0 và B1

```
[ ]  # Hệ số ước lượng (estimating coefficients)
     b = estimate_coef(rdd_train)
     print("Hệ số ước lượng:\nb_0 = {} \nb_1 = {}".format(b[0], b[1]))

     Hệ số ước lượng:
     b_0 = 47466.644251537975
     b_1 = 0.8817946536079935
```

Giá trị B0 và B1 tính ra

Hướng làm 2

1. Tạo mô hình hồi quy tuyến tính – Linear Regression

Xây dựng hàm đoán lợi nhuận dựa trên mô hình hồi quy tuyến tính

```
[ ]  # Hàm dự đoán - trả về các giá trị y
     def simple_linear_regression(train, test):
         b_0, b_1 = estimate_coef(train)
         predictions = test.keys().map(lambda x: x*b_1 + b_0)
         return predictions
```

Hướng làm 2

2. Ứng dụng mô hình hồi quy tuyến tính trên tập dữ liệu Test

Thời gian chạy của mô hình

```
[ ]  start = time.time()
     prediction = simple_linear_regression(rdd_train, rdd_test)
     end = time.time()
     run_time_HL2 = end - start
     print(">>>> Run time:" + str(run_time_HL2) + "(s)")

     >>>> Run time:0.5445091724395752(s)
```

Hướng làm 2

3. Tạo và ứng dụng chỉ số đánh giá RMSE – Root Mean Square Error

Ước tính Sai số toàn phương trung bình căn – Root Mean Square Error.

Công thức:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(y_1 - y_2)^2}{n}}$$

```
[ ]  # Hàm tính toán RMSE
     def rmse_metric(actual, predict):
         rdd_zip = actual.zip(predict)

         predict_error = rdd_zip.map(lambda a: a[0] - a[1])
         predict_error_square = predict_error.map(lambda a: a**2)

         sum_error = predict_error_square.reduce(lambda a,b: a+b)
         #print(sum_error)

         mean_error = sum_error/float(actual.count())

         rmse = math.sqrt(mean_error)
         return rmse
```

Hướng làm 2

3. Tạo và ứng dụng chỉ số đánh giá RMSE – Root Mean Square Error

```
[ ]  # Hàm đánh giá thuật toán hồi quy trên tập dữ liệu huấn luyện
     def evaluate_algorithm(train, test):
         y_test = test.values()
         y_predict = simple_linear_regression(train, test)
         rmse = rmse_metric(y_test, y_predict)
         return rmse
```

Hàm đánh giá thuật toán trên tập Train

```
[ ]  # Đánh giá thuật toán hồi quy trên tập dữ liệu huấn luyện
     import math
     evaluate_model_HL2 = evaluate_algorithm(rdd_train, rdd_test)
     print("Chỉ số RMSE: {} ".format(evaluate_model_HL2))

     Chỉ số RMSE: 12920.51372601141
```

Đánh giá thuật toán

Hướng làm 2

4. Trực quan hóa và so sánh kết quả với dự đoán thực tế

```
[ ]  compare = DataFrame({'Origin': y_test, 'Predict': y_predict})
     compare.head(10)
```

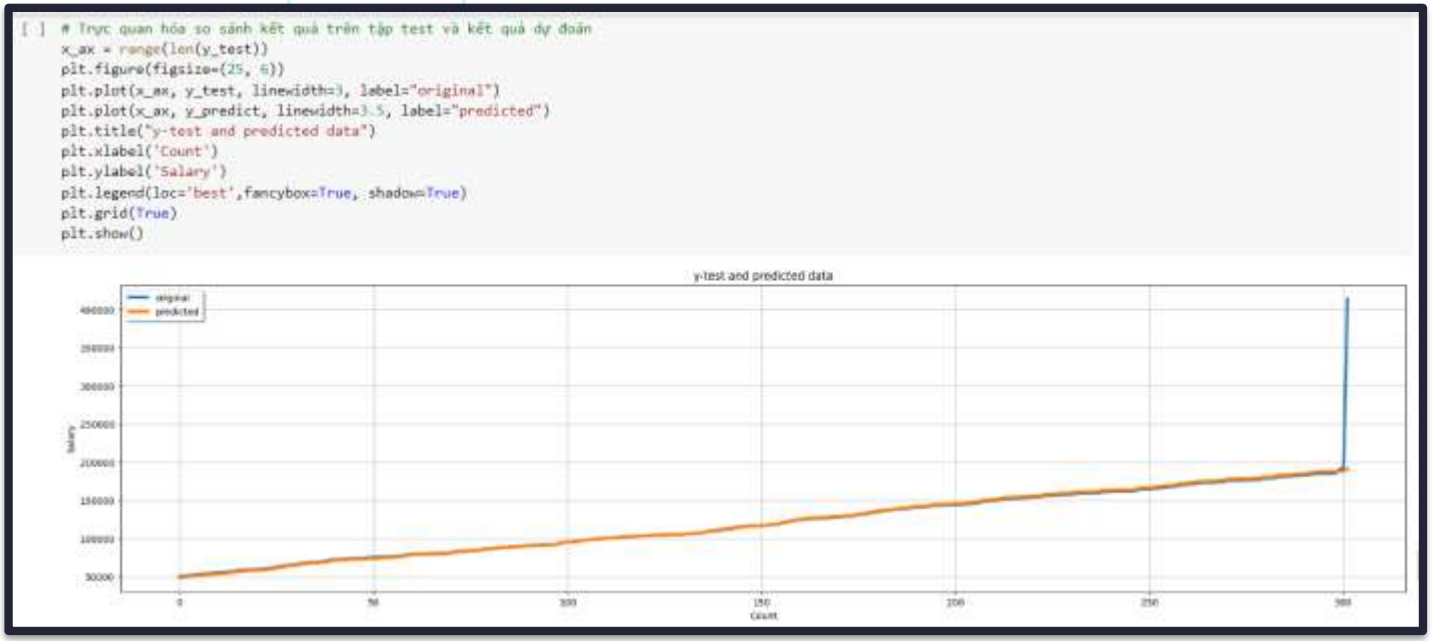
	Origin	Predict
0	50070.86316	48538.024756
1	50994.35213	48948.941064
2	51336.06868	49831.617513
3	51690.59960	50190.507937
4	52276.64348	50761.910872
5	53225.76119	51745.111911
6	53649.48971	52060.794397
7	54205.63339	52716.849619
8	54244.93080	52805.910879
9	54932.63535	52903.790086

So sánh kết quả dự đoán trên tập test và tập dự đoán

Hướng làm 2

4. Trực quan hóa và so sánh kết quả với dự đoán thực tế

```
[ ] # Trực quan hóa so sánh kết quả trên tập test và kết quả dự đoán
    x_ax = range(len(y_test))
    plt.figure(figsize=(25, 6))
    plt.plot(x_ax, y_test, linewidth=3, label="original")
    plt.plot(x_ax, y_predict, linewidth=3.5, label="predicted")
    plt.title("y-test and predicted data")
    plt.xlabel('Count')
    plt.ylabel('Salary')
    plt.legend(loc='best',fancybox=True, shadow=True)
    plt.grid(True)
    plt.show()
```



Trực quan hóa giá trị dự đoán và giá trị thực tế

HƯỚNG LÀM 3

Sử dụng thư viện MLlib

Hướng làm 3

1. Chuyển đổi RDD sang Dataframe

```
[ ]  # Zip X_test và y_test lưu vào rdd_data_MML
     rdd_data_MML = rdd_X_test_1.zip(rdd_y_test_1)

[ ]  # Tên cột khi chuyển đổi từ RDD sang Dataframe
     nameColumns = ["RDSpend", "label"]

[ ]  # Chuyển đổi rdd_data_MML sang Dataframe và lưu vào data_MML
     data_MML = rdd_data_MML.toDF(nameColumns)
```

Hướng làm 3

2. Chuyển đổi giá trị cột Feature thông qua Vector Assembler

```
[ ]  # Thêm lớp LinearRegression từ thư viện máy học
     from pyspark.ml.linalg import Vectors
     from pyspark.ml.feature import VectorAssembler

     assembler = VectorAssembler(
         inputCols=["RDSpend"],
         outputCol="features")
     data_MML = assembler.transform(data_MML)
```

```
[ ]  data_MML.show()
```

```
+-------+----------+---------+
|RDSpend|     label| features|
+-------+----------+---------+
| 1215.0|50070.86316| [1215.0]|
| 1681.0|50994.35213| [1681.0]|
| 2682.0|51336.06868| [2682.0]|
| 3089.0| 51690.5996| [3089.0]|
| 3737.0|52276.64348| [3737.0]|
| 4852.0|53225.76119| [4852.0]|
| 5210.0|53649.48971| [5210.0]|
| 5954.0|54205.63339| [5954.0]|
| 6055.0| 54244.9308| [6055.0]|
| 6166.0|54932.63535| [6166.0]|
| 7214.0|55227.36587| [7214.0]|
| 7251.0|55623.75707| [7251.0]|
| 7959.0|56138.89476| [7959.0]|
| 9063.0|56788.15621| [9063.0]|
| 9261.0|56991.47755| [9261.0]|
| 9960.0|57893.60924| [9960.0]|
|10758.0|58605.23395|[10758.0]|
|11624.0|59225.44949|[11624.0]|
|11977.0|59328.81874|[11977.0]|
|12052.0|59342.48741|[12052.0]|
```

```
[ ]  data_MML.describe().show()
```

```
+-------+-----------------+------------------+
|summary|          RDSpend|             label|
+-------+-----------------+------------------+
|  count|              302|               302|
|   mean|82519.01711920531|120444.37645172192|
| stddev|47038.62548681213|43492.754490076855|
|    min|           1215.0|       50070.86316|
|    max|         162597.7|         413956.48|
+-------+-----------------+------------------+
```

Hướng làm 3

3. Ứng dụng mô hình hồi quy tuyến tính trên tập dữ liệu Test

```
[ ] from pyspark.ml.regression import LinearRegression

    start = time.time()
    #Tạo biến LinearRegression để sử dụng
    lr = LinearRegression()

    #Tạo ra hai mô hình tương ứng với hai tham số
    modelA = lr.fit(data_MNL, {lr.regParam:0.0})

    predictionsA = modelA.transform(data_MNL)
    end = time.time()
    run_time_HL3 = end - start
    print(">>>> Run time:" + str(run_time_HL3) + "(s)")

    22/12/19 03:53:18 WARN Instrumentation: [53b965f9] regParam is zero, which might cause numerical instability and overfitting.
    >>>> Run time:3.170637369155884(s)
```

Hướng làm 3

3. Ứng dụng mô hình hồi quy tuyến tính trên tập dữ liệu Test

```
[ ] #Hiển thị thông tin của hai mô hình
    print(">>>> Model intercept: {}, coefficient: {}".format(modelA.intercept, modelA.coefficients[0]))

    >>>> Model intercept: 47601.040120002785, coefficient: 0.882746049028887

[ ] predictionsA.show()

+-------+----------+---------+------------------+
|RDSpend|     label| features|        prediction|
+-------+----------+---------+------------------+
| 1215.0|50070.86316| [1215.0]| 48673.57656957279|
| 1681.0|50994.35213| [1681.0]|49084.936228420214|
| 2682.0|51336.06868| [2682.0]| 49968.56502349805|
| 3089.0| 51690.5996| [3089.0]| 50327.84266545277|
| 3737.0|52276.64348| [3737.0]|50899.862105223445|
| 4852.0|53225.76119| [4852.0]| 51884.12394989056|
| 5210.0|53649.48971| [5210.0]|52200.147035442875|
| 5954.0|54205.63339| [5954.0]| 52856.91009592031|
| 6055.0| 54244.9308| [6055.0]| 52946.06744687222|
| 6166.0|54932.63535| [6166.0]| 53044.05225831442|
| 7214.0|55227.36587| [7214.0]| 53969.17011769661|
| 7251.0|55623.75707| [7251.0]| 54001.83172151068|
| 7959.0|56138.89476| [7959.0]|54626.815924223076|
| 9063.0|56788.15621| [9063.0]|55601.367562350875|
| 9261.0|56991.47755| [9261.0]|55776.151280058584|
| 9960.0|57893.60924| [9960.0]| 56393.19076832972|
|10758.0|58605.23395|[10758.0]| 57097.62211545471|
|11624.0|59225.44949|[11624.0]| 57862.88019391366|
|11977.0|59328.81874|[11977.0]| 58173.68954922083|
|12052.0|59342.48741|[12052.0]| 58239.89550289798|
+-------+----------+---------+------------------+
only showing top 20 rows
```

Thêm giá trị dự đoán vào Data Frame

Hướng làm 3

4. Tạo và ứng dụng chỉ số đánh giá RMSE – Root Mean Square Error

```
[ ]  #Thêm lớp RegressionEvaluator từ thư viện máy học
     from pyspark.ml.evaluation import RegressionEvaluator

     #Sử dụng phương pháp tính Root Mean Squared Error để đánh giá kết quả mô hình
     evaluator = RegressionEvaluator(metricName="rmse")
     evaluate_model_HL3 = evaluator.evaluate(predictionsA)
     print(">>>> Model: Root Mean Squared Error = " + str(evaluate_model_HL3))

     >>>> Model: Root Mean Squared Error = 12918.682233714082
```

Hướng làm 3

5. Trực quan hóa và so sánh kết quả dự đoán với thực tế

```
[ ]  predictionsA.createOrReplaceTempView("predictionsA")
     prediction_result_MML = spark.sql("SELECT label, prediction FROM predictionsA")
```

```
[ ]  # Đưa giá trị Salary test và Salary dự đoán vào DataFrame để tiến hành so sánh và trực quan hóa
     prediction_result_MML.toPandas().head()
```

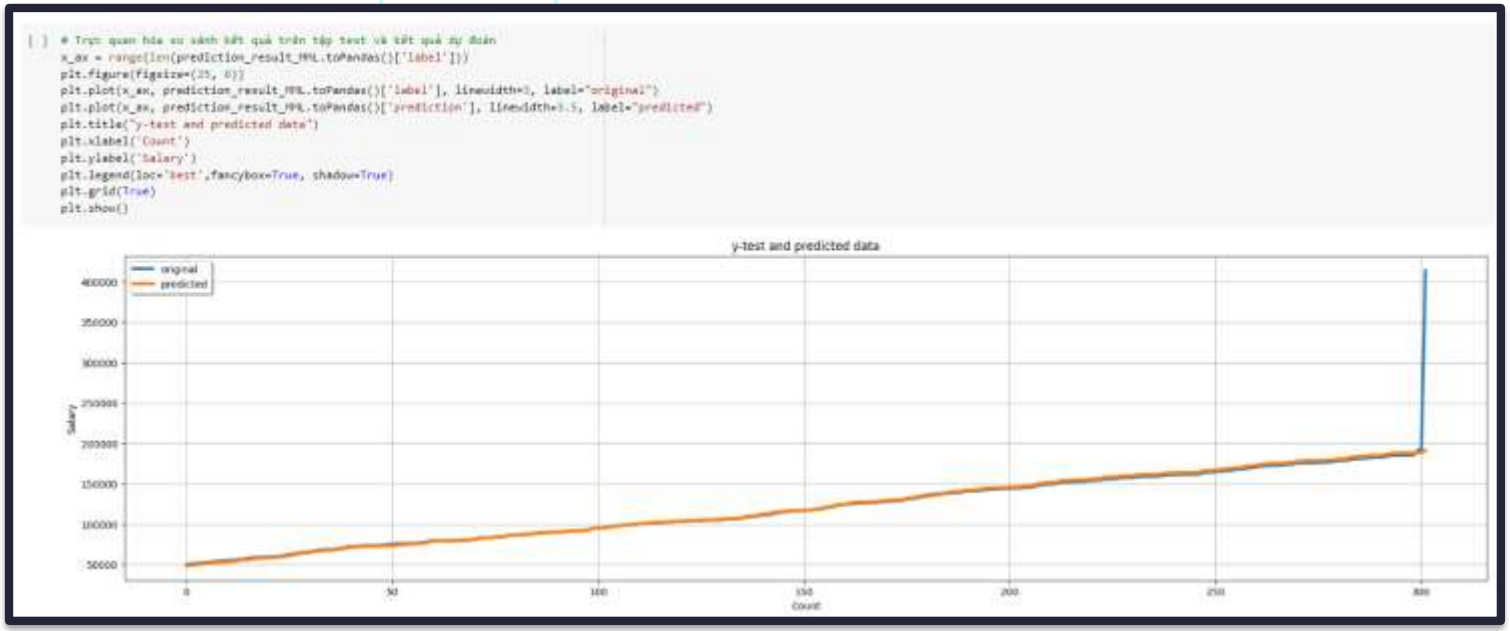
	label	prediction
0	50070.86316	48673.576570
1	50994.35213	49084.936228
2	51336.06868	49968.565023
3	51690.59960	50327.842665
4	52276.64348	50899.862105

Đưa giá trị test và dự đoán vào Data Frame để tiến hành so sánh và trực quan hóa

Hướng làm 3

5. Trực quan hóa và so sánh kết quả dự đoán với thực tế

```
[ ] # Trực quan hóa so sánh kết quả trên tập test và kết quả dự đoán
    x_ax = range(len(prediction_result_MYL.toPandas()['label']))
    plt.figure(figsize=(25, 6))
    plt.plot(x_ax, prediction_result_MYL.toPandas()['label'], linewidth=3, label="original")
    plt.plot(x_ax, prediction_result_MYL.toPandas()['prediction'], linewidth=3.5, label="predicted")
    plt.title("y-test and predicted data")
    plt.xlabel('Count')
    plt.ylabel('Salary')
    plt.legend(loc='best',fancybox=True, shadow=True)
    plt.grid(True)
    plt.show()
```

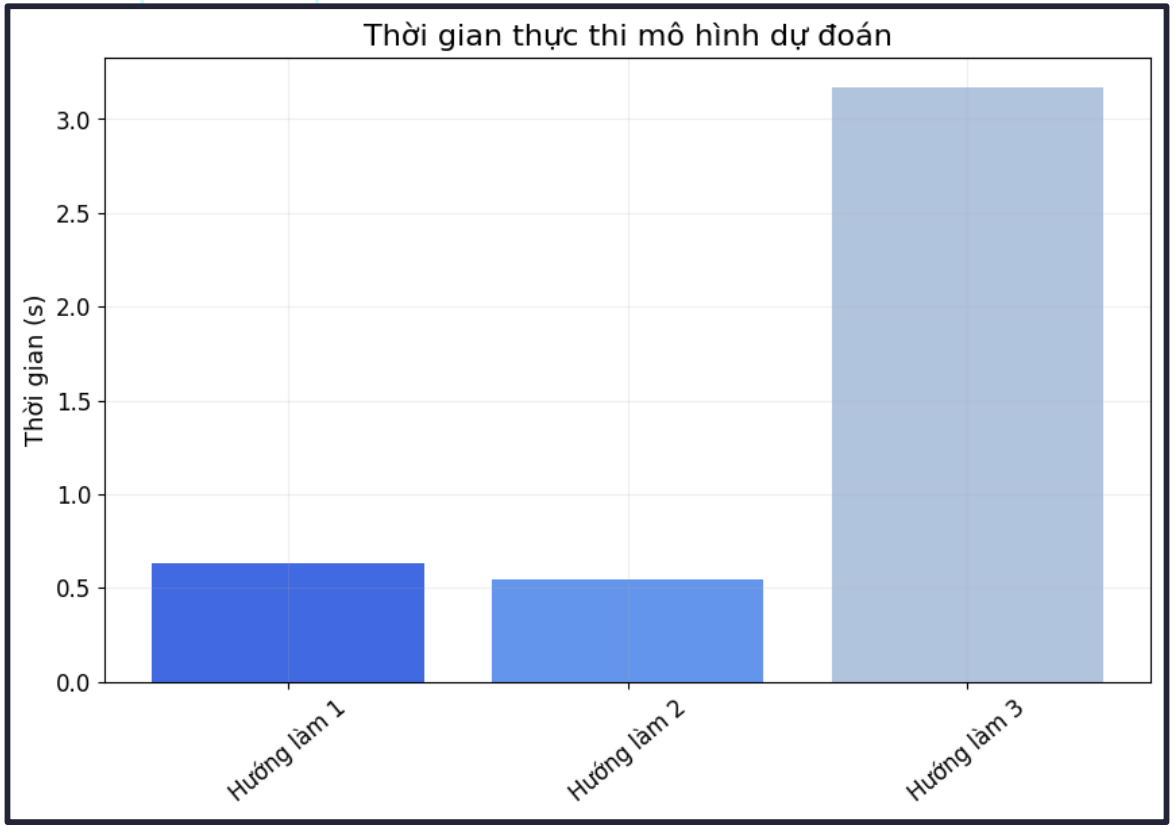


Trực quan hóa giá trị dự đoán và giá trị thực tế

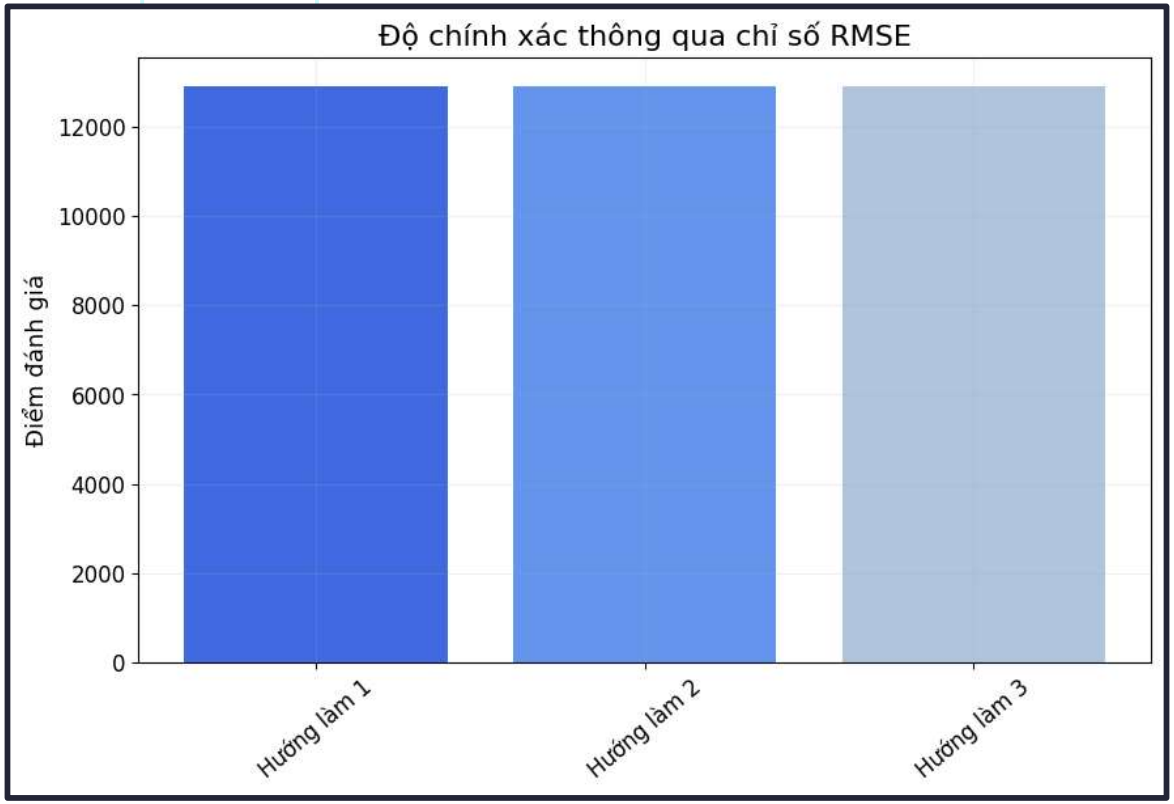
SO SÁNH THUẬT TOÁN

07

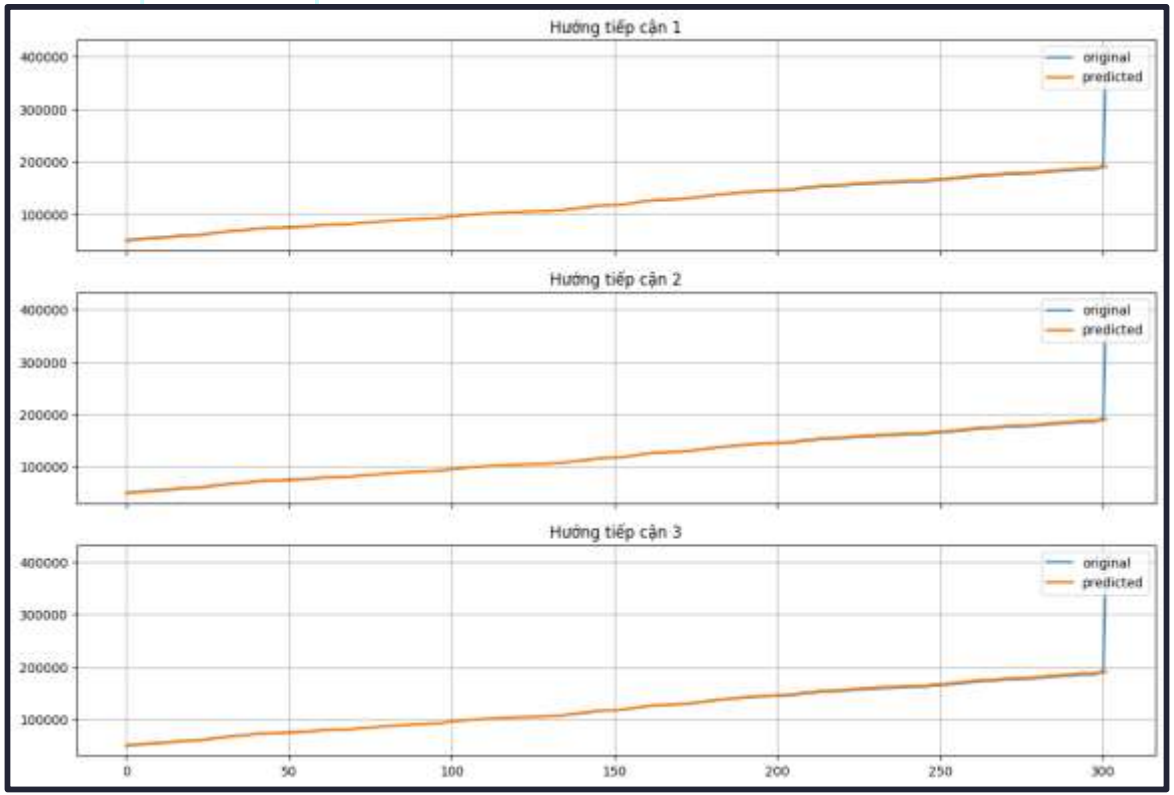
SO SÁNH THỜI GIAN CHẠY



SO SÁNH ĐỘ CHÍNH XÁC QUA CHỈ SỐ RMSE



SO SÁNH ĐỘ CHÍNH XÁC QUA TRỰC QUA DỮ LIỆU



KẾT LUẬN

08

ƯU VÀ NHƯỢC ĐIỂM CÁC HƯỚNG LÀM

• Thời gian thực thi dự đoán giữa hướng làm 1 và 2 nhanh hơn so với hướng làm 3.

• Độ chính xác, đúng đắn giữa giá trị dự đoán và giá trị kiểm thử trên 3 hướng làm gần như tương đồng với nhau.

HƯỚNG PHÁT TRIỂN

• Mở rộng mô hình Hồi quy tuyến tính đơn giản (Simple Linear Regression) với 2 hệ số thành mô hình Hồi quy tuyến tính phức tạp hơn với nhiều đối số truyền vào (Multiple Linear Regression) để có thể đưa ra được những dự đoán khách quan hơn.

• Xây dựng thêm một số mô hình dự đoán đơn (Single Model) chẳng hạn như Decision Tree,.. và mô hình tập hợp (Embedded Model) chẳng hạn như Random Forest. XGBoots để so sánh tính chính xác, khả năng dự đoán của các mô hình trên cùng một tập dữ liệu.

CÁM ƠN THẦY VÀ CÁC BẠN ĐÃ LẮNG NGHE