TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KỲ

XỬ LÝ DỮ LIỆU LỚN

Người hướng dẫn: **GV NGUYỄN THÀNH AN**
Người thực hiện: **ĐINH THỊ NGỌC PHƯỢNG – 52100923**
NGUYỄN ĐÔNG TRIỀU-52100337
NGUYỄN MINH PHÚ - 52100920
NGUYỄN TIẾN ĐẠT-52100177
ĐINH PHÚ QUỐC - 52100927
Lớp: **21050301**
Khóa: **25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KỲ

XỬ LÝ DỮ LIỆU LỚN

Người hướng dẫn: **GV NGUYỄN THÀNH AN**
Người thực hiện: **ĐINH THỊ NGỌC PHƯỢNG – 52100923**
NGUYỄN ĐÔNG TRIỀU-52100337
NGUYỄN MINH PHÚ - 52100920
NGUYỄN TIẾN ĐẠT-52100177
ĐINH PHÚ QUỐC - 52100927
Lớp: **21050301**
Khóa: **25**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Để hoàn thành được bài báo cáo này, lời đầu tiên chúng em xin cảm ơn chân thành đến Ban Giám hiệu nhà Trường Đại học Tôn Đức Thắng đã tạo điều kiện tốt nhất cho sinh viên về cơ sở vật chất, với rất nhiều tài liệu, sách tham khảo cùng với hệ thống thư viện hiện đại, thuận lợi cho việc nghiên cứu và tìm kiếm thông tin để hoàn thành bài báo cáo của mình Tiếp đến, Chúng em xin gửi lời cảm ơn sâu sắc đến thầy Nguyễn Thành An, giảng viên đảm nhiệm thực hành môn Xử lý dữ liệu lớn. Thầy là một giảng viên rất tận tâm với sinh viên, thầy đã truyền đạt không chỉ trong những kiến thức ở tài liệu mà còn mở rộng các kỹ năng trong môn học giúp chúng em đi xa hơn trong cách phân tích các cấu trúc tổ chức dữ liệu. Bằng sự nhiệt tình và nghiêm túc của mình, thầy luôn mang đến cho sinh viên những điều kiện tốt nhất trong thời gian học cùng thầy. Chính vì những điều này đã giúp chúng em có được tinh thần học tập hiệu quả và nghiêm túc trong quá trình học. Nhờ đó môn học này mang tính vận dụng thực tế cao gắn liền với nhu cầu thực tiễn cho chúng em và các bạn sinh viên khi theo học ngành Công nghệ thông tin.

Tuy nhiên, thật lòng chúng em vẫn có mặt hạn chế về kiến thức vì thế bài báo cáo sẽ không tránh khỏi những sai sót. Mong thầy có thể xem xét và đóng góp ý kiến để bài báo cáo của em được hoàn thiện hơn.

Lời cuối cùng, chúng em kính chúc thầy có thật nhiều sức khỏe, thành công, hạnh phúc trong sự nghiệp trồng người và tiếp tục dìu dắt nhiều thế hệ học trò kế tiếp đi đến những bến đỗ tri thức

ĐỒ ÁN ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của Giảng viên Nguyễn Thành An. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 15 tháng 05 năm 2024

Tác giả

(ký tên và ghi rõ họ tên)

Đinh Thị Ngọc Phượng

Nguyễn Đông Triều

Nguyễn Minh Phú

Phạm Tiến Đạt

Đinh Phú Quốc

TÓM TẮT

Báo cáo này là bài cuối kỳ môn "Xử lý ngôn ngữ tự nhiên", với 5 yêu cầu chính cùng các nội dung chính như sau:

Yêu cầu 1: Phân cụm dữ liệu

Sử dụng pyspark.sql để khai thác dữ liệu, matplotlib.pyplot để vẽ biểu đồ, và thuật toán k-Means với k = 10, trong đó một số điểm dữ liệu được gán trọng số cao hơn. Sau đó, tính trung bình khoảng cách đến centroid và vẽ biểu đồ cột.

Yêu cầu 2: Giảm số chiều với SVD

Sử dụng pyspark và SVD để giảm số chiều dữ liệu từ 784 xuống 3, chọn ngẫu nhiên 100 điểm, rồi dùng kết quả phân cụm và matplotlib.pyplot để vẽ biểu đồ 3D của 100 điểm này.

Yêu cầu 3: Khuyến nghị sản phẩm với Collaborative Filtering

Chia dữ liệu thành tập training và test theo tỷ lệ 7:3, sử dụng pyspark và ALS để khảo sát MSE với số lượng người dùng "tương đồng" từ 10 đến 20. Chạy inference và vẽ biểu đồ cột trực quan hóa MSE.

Yêu cầu 4 : Dự đoán giá chứng khoán.

Xây dựng mô hình Linear Regression với pyspark để dự đoán biên độ giá chứng khoán, huấn luyện trên tập training, đánh giá trên tập test, tính Mean Square Error cho cả hai tập, và vẽ biểu đồ cột bằng matplotlib.pyplot để hiển thị các giá trị MSE.

Yêu cầu 5: Phân loại đa lớp với pyspark

Xây dựng mô hình phân loại đa lớp với pyspark. Vẽ biểu đồ cột đôi với matplotlib.pyplot để thể hiện độ chính xác của ba mô hình (Multi-layer Perceptron, Random Forest, Linear Support Vector Machine) trên tập training và test.

MỤC LỤC

YÊU CẦU 1: PHÂN CỤM DỮ LIỆU

1.1 Mô tả bài toán.

Yêu cầu: Sử dụng tập dữ liệu **mnist_mini.csv**:

- Cài đặt thuật toán k-Means (pyspark.ml.clustering.KMeans) và giá trị k = 10, trong đó các điểm dữ liệu tại dòng 0, 1, 2, 3, 4, 7, 8, 11, 18, 61 được gán trọng số gấp 100 lần các điểm dữ liệu khác.
- Tính trung bình khoảng cách từ các điểm dữ liệu tới centroid trong các cluster. Vẽ biểu đồ cột để trực quan hoá kết quả.

Mô tả đầu vào: Dữ liệu hình ảnh ký số viết tay trong tập MNIST.

- 10000 dòng dữ liệu.
- Mỗi dòng chứa 785 số nguyên

  o Số thứ nhất: loại ký số (0, 1, 2, 3, ..., 9)

  o 784 số còn lại là pixel của ảnh grayscale 28 x 28.

1.2 Giải quyết bài toán

Bài toán được xây dựng thông qua lớp MNISTDataAnalyzer, với các hàm sau:

- _init_: lớp MNISTDataAnalyzer được khởi tạo với các tham số: đường dẫn đến tập dữ liệu và số pixel của bức ảnh.
- explore_data: hàm này dùng để trực quan các bức ảnh đầu vào.
- run_kmeans: hàm có chức năng là xử lý dữ liệu đầu vào và triển khai thuật toán Kmeans.
- correct_predictions: hàm này dùng để trả về số lượng mẫu được dự đoán chính xác.
- visualization: đây là hàm thực hiện trực quan trung bình khoảng cách từ các điểm dữ liệu đến centroid trong cluster.

1.2.1 Phương thức khởi tạo.

Các thuộc tính của lớp MNISTDataAnalyzer:

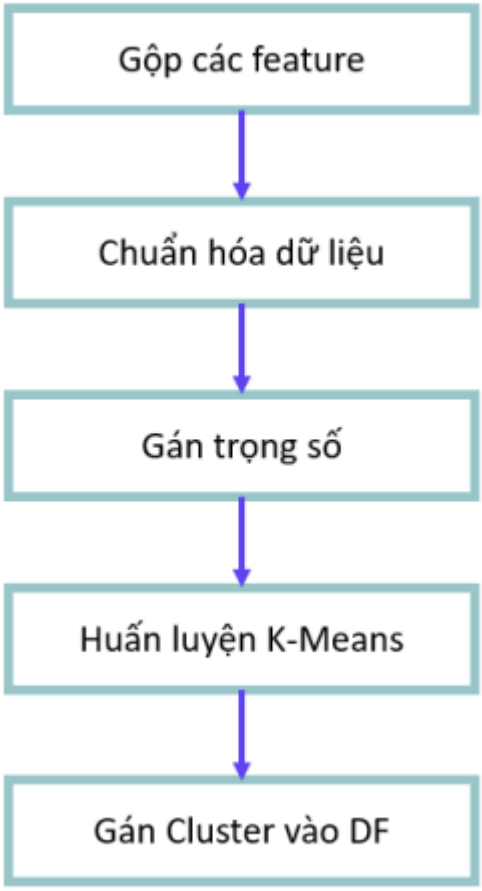
- self.num_pixel: số pixel của ảnh, được lưu trữ lại từ tham số đầu vào.
- self.spark: để khởi tạo và lấy về một SparkSession trong Spark.
- self.df: dùng để lưu trữ tập dữ liệu đầu vào dưới dạng DataFrame với 785 cột: cột đầu tiên là ‘label’ lưu trữ lại ký số (0, 1, 2,…,9); còn 784 cột còn lại để lưu trữ giá trị của các pixel (giá trị từ 0 đến 255).
- self.model: lưu trữ mô hình KMeans
- self.clustered: để lưu trữ DataFrame chứa dữ liệu phân cụm sau khi Kmeans được triển khai.
- self.k: lưu trữ số cluster.

1.2.2 Phương thức run_kmeans.

Các tham số đầu vào của run_kmeans:

- k: số cluster người dùng muốn phân cụm.
- weighted_indices: mảng chứa vị trí các dòng dữ liệu được gán trọng số.
- weight_value: giá trị trọng số mà người dùng muốn gán.

Các bước thực hiện:



Hình 1.1: Các bước thực hiện trong run_kmeans.

Bước 1: Gộp 784 feature chứa 784 pixel thành một features duy nhất bằng cách sử dụng hàm VectorAssembler.

```
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
weighted_df = assembler.transform(weighted_df)
```

Bước 2: Chuẩn hóa dữ liệu, từ "features" mà chúng ta đã gộp ở trên ta sẽ tiến hành chuẩn hóa bằng MinMaxScaler. Thuật toán này sẽ chuẩn hóa tập dữ liệu về khoảng từ 0 đến 1.

```
scaler = MinMaxScaler(inputCol="features", outputCol="scaledFeatures")
weighted_df = scaler.fit(weighted_df).transform(weighted_df)
```

Bước 3: Gán trọng số, để thuận tiện thì trước tiên ta cần gán index cho từng hàng trong tập dữ liệu. Từ các index này, ta sẽ so sáng chúng với các vị trí cần gán trọng số được truyền vào từ biến weighted_indices. Từ đó các hàng được gán trọng số sẽ mang giá trị weight_value, còn lại sẽ là 1.

Bước 4: Huấn luyện Kmeans, trước tiên, ta cần khởi tạo mô hình KMeans từ thư viện pyspark.ml.clustering.KMeans với các tham số khởi tạo là k (số cluster) và featuresCol là cột chon làm feature cho quá trình phân cụm (cột "scaledFeatures").

```
kmeans = KMeans(k=k, featuresCol="scaledFeatures")
```

Tiếp đó, ta sẽ thiết lập trọng số cho các điểm dữ liệu từ cột trọng số mà ta đã gán ở bước 3, bằng cách dùng hàm setWeightCol() trên biến kmeans với tham số truyền vào là tên của cột trọng số.

```
kmeans.setWeightCol("weightCol")
```

Cuối cùng là đưa tập dữ liệu vào để tiến hành quá trình huấn luyện.

```
self.model = kmeans.fit(weighted_df)
```

Bước 5: Gán các cluster dự đoán vào DataFrame, sử dụng phương thức transform trên biến model để thực hiện.

```
clusters = self.model.transform(weighted_df)
```

Sau bước này DataFrame sẽ được gán thêm một cột tên là “prediction” biểu diễn các cluster dự đoán dữ liệu.

label	scaledFeatures	id	weightCol	prediction
7	(784,[0,1,2,3,4,5...	1	100.0	3
2	(784,[0,1,2,3,4,5...	2	100.0	5
1	(784,[0,1,2,3,4,5...	3	100.0	1
0	(784,[0,1,2,3,4,5...	4	100.0	6
4	(784,[0,1,2,3,4,5...	5	100.0	9
1	(784,[0,1,2,3,4,5...	6	1.0	1
4	(784,[0,1,2,3,4,5...	7	1.0	3
9	(784,[0,1,2,3,4,5...	8	100.0	9
5	(784,[0,1,2,3,4,5...	9	100.0	8
9	(784,[0,1,2,3,4,5...	10	1.0	0
0	(784,[0,1,2,3,4,5...	11	1.0	7
6	(784,[0,1,2,3,4,5...	12	100.0	8
9	(784,[0,1,2,3,4,5...	13	1.0	3
0	(784,[0,1,2,3,4,5...	14	1.0	7
1	(784,[0,1,2,3,4,5...	15	1.0	1
5	(784,[0,1,2,3,4,5...	16	1.0	5
9	(784,[0,1,2,3,4,5...	17	1.0	9
7	(784,[0,1,2,3,4,5...	18	1.0	3
3	(784,[0,1,2,3,4,5...	19	100.0	5
4	(784,[0,1,2,3,4,5...	20	1.0	9

Hình 1.2 Kết quả dự đoán của KMeans trong cột "prediction".

1.2.3 Phương thức correct_prediction.

Đây là phương thức dùng để tra về số lượng điểm dữ liệu được dự đoán đúng.

Được thực hiện bằng cách so sánh label và prediction trên cũng một hàng, sau đó dùng hàm count() để trả về số lượng.

```
def correct_predictions(self):
    correct = self.clustered.where(self.clustered.label == self.clustered.prediction).count()
    return correct
```

```
number of correct prediction are  3120
```

1.2.4 Phương thức visualization.

Đây là phương thức trực quan quá khoảng cách trung bình từ các điểm dữ liệu tới các centroids trong các clusters.

Trước hết ta cần phải tính các khoảng cách trung bình theo các bước sau:

Bước 1: tìm Dataframe chứa hai cột là các clusters và centroid tương ứng với các clusters đó. Thực hiện bằng cách lấy các centroids từ hàm clusterCenters() trên biến model.

```
centroids = self.model.clusterCenters()
```

Tiếp đến là tạo ra một DataFrame với các centroid vừa tìm được và tạo thêm một cột đánh số thứ tự thể hiện các clusters gọi là "clusters".

Bước 2: Tính khoảng cách từ các điểm dữ liệu đến centroid của điểm dữ liệu đó, bằng cách thực hiện tuần tự quá trình sau:

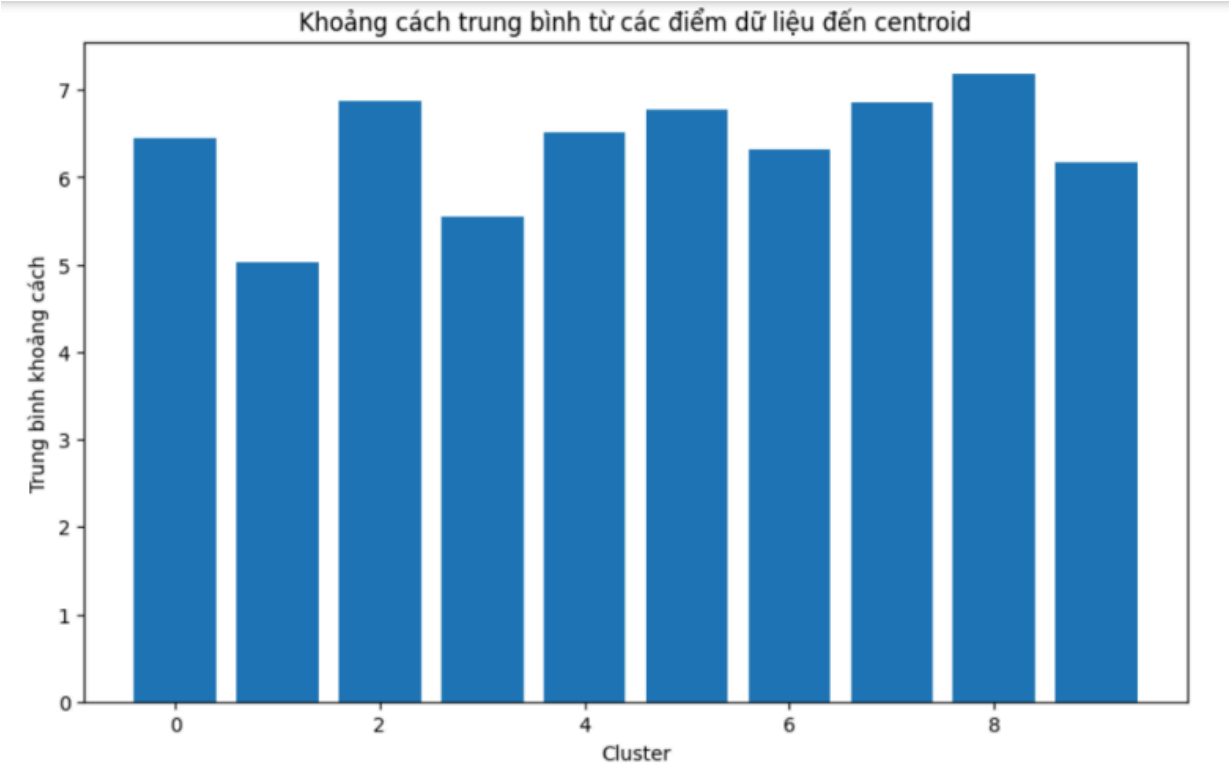
- join DataFrame chứa kết quả dự đoán với DataFrame chứa các centroids ở bước 1 lại với nhau dựa vào giá trị "prediction" với "clusters". Từ đó ta có được một DataFrame mà mỗi hàng đều có centroid của cluster mà nó thuộc về.
- Tính khoảng cách bằng cách tính căn bậc 2 của Vectors.squared_distance trên "scaledFeatures" với giá trị của centroids.
- Lọc chọn các cột cần thiết như "prediction" (cluster dự đoán) và "distance_to_centroid" (khoảng cách ta vừa tìm được).

Bước 3: Tính trung bình các khoảng cách trong một cluster. Ta sẽ nhóm các "prediction" lại với nhau và tình trung bình "distance_to_centroid" trong cùng một nhóm.

prediction	avg_distance
0	6.450506557647026
1	5.02880467839939
2	6.868201295301388
3	5.5535157685707555
4	6.515950961919523
5	6.78245671409697
6	6.322893116368711
7	6.860956425230379
8	7.188780807293526
9	6.177680057509396

Hình 1.3: Kết quả khoảng cách trung bình từ các điểm đến centroid.

Cuối cùng ta sẽ trực quan kết quả này lên biểu đồ cột với trục hoành là 10 clusters và trục tung là khoảng cách trung bình từ các điểm dữ liệu đến centroid của chúng thuộc vào.



Hình 1.4: Biểu đồ cột thể hiện khoảng cách trung bình từ các điểm dữ liệu đến centroid.

YÊU CẦU 2: GIẢM THIỂU SỐ CHIỀU VỚI SVD

1.1 Mô tả bài toán.

Yêu cầu: Sử dụng tập dữ liệu **mnist_mini.csv**:

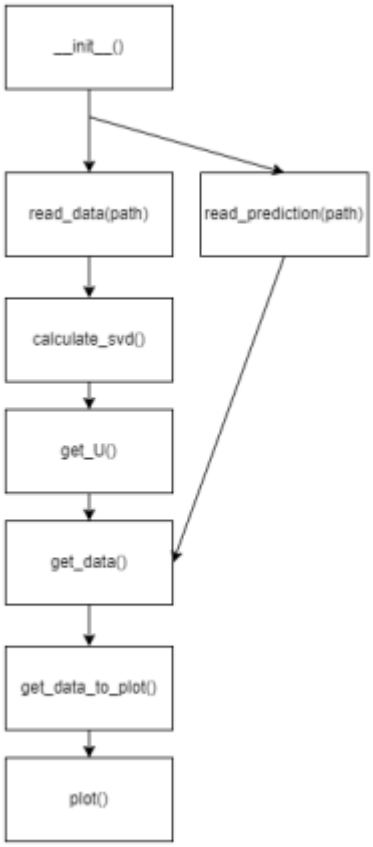
- Giảm số chiều của dữ liệu xuống còn 3.
- Chọn ngẫu nhiên 100 điểm dữ liệu. Sử dụng kết quả dự đoán ở câu . Vẽ biểu đồ 3D mô tả phân bố của 100 điểm này trong không gian.

1.2 Giải quyết bài toán

Bài toán được xây dựng thông qua lớp SVD, với các hàm sau:

- _init_: phương thức khởi tạo
- read_data: phương thức để đọc dữ liệu từ file liệu mnist_mini.csv.
- read_prediction: phương thức để đọc dữ liệu dự đoán từ câu 1.
- calculate_svd: giảm thực hiện thuật toán SVD trên tập dữ liệu mnist_mini, số chiều xuống còn 3.
- get_U: lấy ra Left Singular Vector
- get_data: với lấy ra dữ liệu sau khi đã giảm chiều, kết hợp giá trị đó giá trị dữ đoán ở câu 1.
- get_data_to_plot: lấy ra 100 điểm dữ liệu ngẫu nhiên để tiến hành plot
- plot: plot dữ liệu 100 điểm dữ liệu ngẫu nhiên vừa lấy

1.2.1 Quy trình chạy thuật toán



Hình 2.1: Quy trình thực hiện các phương thức trong lớp SVD

Việc chia nhỏ thành các phương thức khác nhau giúp tăng khả năng tùy chỉnh cho lớp, tránh việc phải chạy lại thuật toán svd nhiều lần mà vẫn có thể lấy được những bộ sample khác nhau để thực hiện được plot dữ liệu.

1.2.2 Một số phương thức chính

a. Phương thức khởi tạo

Tham số: svd = None. Dùng cho mục đích phát triển, nếu được cũng cấp svd, bước calculate_svd sẽ được bỏ qua và sử dụng svd được cung cấp.

Khởi tạo một số thuộc tính cho lớp, khởi tạo các giá trị này bằng None, sau khi thực hiện các phương thức tương ứng, các thuộc tính này sẽ được cập nhật giá trị:

+ df : dữ liệu đọc từ mnist.csv lưu dưới dạng csv.

+ label_df: dữ liệu prediction đọc từ câu 1.

+ svd: Kết quả thu được sau khi thực hiện thuật toán svd lên df, gồm 3 ma trận U, s, v

+ U_df: Left singular vector, vì yêu cầu của bài toán là giảm số chiều của pixel xuống còn 3 và giữ nguyên số lượng document, nên ta sẽ sử dụng U (mxr) để thu được kết quả này.

+ data: kết quả gộp của U và prediction lấy từ phương thức read_prediction(path). Mỗi dòng của data gồm 4 thuộc tính, với 3 cột đầu tiên tượng trưng cho giá trị x, y, z lấy từ U_df, cột thứ 4 là giá trị label được dự đoán lấy từ label_df.

+ plot_data: Một subset của data, dùng để plot dữ liệu lên biểu đồ 3D.

b. Phương thức calculate_SVD

Sử dụng phương thức calculateSVD của lớp RowMatrix được cung cấp trong thư viện pyspark.

Ở đây, ta cần phải chuyển DataFrame ban đầu sang dạng RowMatrix mới có thể sử dụng phương thức này.

```
number_of_columns = len(self.df.columns)

df = self.df

rdd = df.rdd.map(lambda row: \
Vectors.dense([row[f"_c{i}"] for i in range(number_of_columns)]))

mat = RowMatrix(rdd)


svd = mat.computeSVD(3, computeU=True)
```

Hình 2.2 Chuyển đổi DataFrame thành RowMatrix và thực hiện tính toán SVD

c. Phương thức get_U

Sau khi thực hiện tính toán SVD, ta sẽ thu được một svd object, bao gồm U, tuy nhiên, vì U ở dạng RowMatrix, ta cần chuyển đổi từ RowMatrix sang DataFrame để tiếp tục tính toán.

```
U = self.svd.U
self.U_df = U.rows.map(lambda row: Row(*(float(x)\
 for x in row.toArray()))).toDF(["x", "y", "z"])
```

Hình 2.3.Chuyển đổi U từ RowMatrix sang DataFrame

d. Phương thức get_data

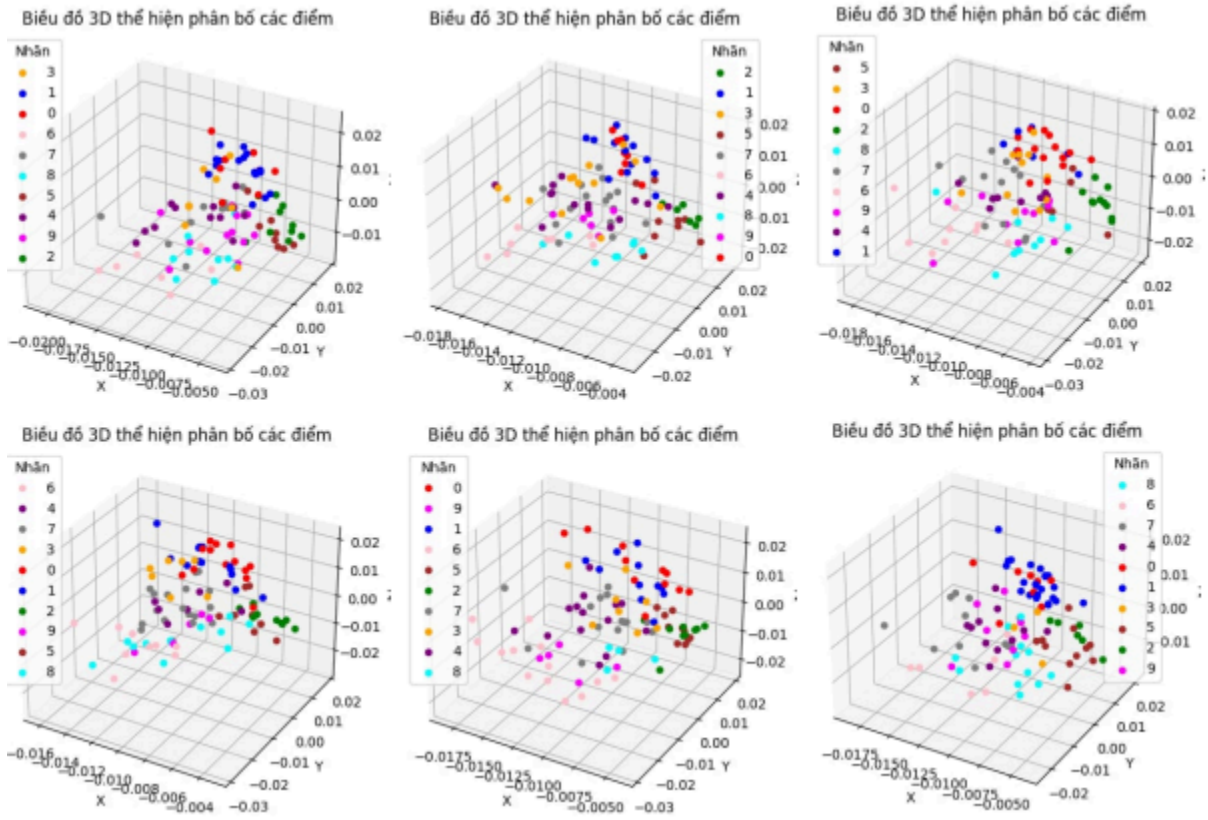
Sau khi thực hiện tính toán svd, ta thu được một DataFrame mới với kích thước nhỏ hơn nhiều so với ban đầu và đã có thể an toàn để collect về, ta sẽ thực hiện collect U_df về , sau đó kết hợp với giá trị prediction ở hàng tương ứng thu được từ câu 1, từ đó ta sẽ thu được 1 mảng 2 chiều (m x 4) với m là số lượng document và 4 cột (x, y, z, label).

e. Phương thức plot

Sau khi thực hiện phương thức get_data_to_plot (phương thức này sẽ lấy ngẫu nhiên 100 hàng từ data). Ta sẽ thực hiện quá trình plot lên biểu đồ 3D.

Ở đây, mỗi nhãn ta sẽ gán cho nó một màu sắc tương ứng với label, giá trị trên không gian sẽ là 3 giá trị (x,y,z)

Ta thu được kết quả như sau:



Hình 2.4 : Kết quả thu được khi thực hiện plot 100 điểm dữ liệu ngẫu nhiên lên biểu đồ 3D

YÊU CẦU 3: KHUYẾN NGHỊ SẢN PHẨM VỚI COLLABORATIVE FILTERING

1.1 Mô tả bài toán.

Yêu cầu: Sử dụng tập dữ liệu **ratings2k.csv**:

- Sử dụng pyspark và thuật toán ALS để khảo sát hiệu suất của mô hình theo độ đo Mean
- Squared Error (MSE) với các giá trị số lượng người dùng "tương đồng" trong đoạn [10; 20].
- Chạy inference để minh hoạ hoạt động của mô hình.
- Vẽ biểu đồ cột để trực quan hoá các giá trị sai số MSE.

1.2 Giải quyết bài toán

1.2.1. Hàm load_and_split_data(self, test_ratio=0.3)

Chức năng: Thực hiện đọc dữ liệu và chia dữ liệu thành hai phần train và test.

Thực hiện đọc dữ liệu

```
data = self.spark.read.csv(self.ratings_file, header=True, inferSchema=True)
```

Hình 3.1 Đọc dữ liệu

Và chia dữ liệu thành hai phần train và test với tỉ lệ dữ liệu trong phần test được quy định bởi test_ratio .

```
(self.training_data, self.test_data) = data.randomSplit([1 - test_ratio, test_ratio])
```

Hình 3.2 Chia dữ liệu

1.2.2. Hàm train_model(self, rank, max_iter=10, reg_param=0.1)

Chức năng: Giúp khởi tạo mô hình ALS và thực hiện train mô hình trên train data.

Hàm này sẽ giúp khởi tạo mô hình ALS với các tham số khởi tạo:

- **maxIter**=**max_iter**: Số lượng vòng lặp tối đa (epochs) cho việc huấn luyện, mặc định là 10.
- **regParam**=**reg_param**: Hệ số điều chuẩn (regularization parameter), mặc định là 0.1.
- **rank**=**rank**: Số lượng yếu tố ẩn (latent factors) trong mô hình.
- **serCol**="user": Tên cột chứa ID người dùng trong tập dữ liệu đầu vào
- **itemCol**="item": Tên cột chứa ID sản phẩm trong tập dữ liệu đầu vào
- **ratingCol**="rating": Tên cột chứa thông tin đánh giá (rating) trong tập dữ liệu đầu vào
- **coldStartStrategy**="drop": Chiến lược xử lý dữ liệu chưa thấy trong quá trình huấn luyện (cold-start), ở đây là bỏ qua các dự đoán cho những dữ liệu đó.

```
als = ALS(maxIter=max_iter, regParam=reg_param, rank=rank, userCol="user",
          itemCol="item", ratingCol="rating", coldStartStrategy="drop")
```

Hình 3.3 Khởi tạo mô hình ALS

1.2.3.Hàm evaluate_model(self)

Chức năng: Đánh giá hiệu suất mô hình trên cả tập dữ liệu huấn luyện (training_data) và tập dữ liệu kiểm tra (test_data) theo độ đo MSE.

1.2.4. Hàm plot_mse(self, train_mse_values, test_mse_values)

Chức năng: Vẽ biểu đồ so sánh giá trị MSE (Mean Squared Error) của tập dữ liệu huấn luyện và tập dữ liệu kiểm tra cho các giá trị tham số **rank** khác nhau trong mô hình ALS.

1.2.5. Hàm recommend_for_user

Chức năng: Tạo ra item đề xuất cho một người dùng cụ thể (user_id) dựa trên mô hình ALS đã được huấn luyện.

Các bước thực hiện:

a. *Tạo DataFrame chứa một người dùng*

```
user_subset = self.spark.createDataFrame([(user_id,)], ["user"])
```

Hình 3.4 DataFrame chứa một người dùng

b. *Lấy khuyến nghị cho người dùng*

```
recommendations = self.model.recommendForUserSubset(user_subset, num_recommendations)
```

Hình 3.5 Lấy khuyến nghị cho người dùng

c. *Chuyển đổi các khuyến nghị thành định dạng dễ đọc và hiển thị*

```
recommendations = recommendations.select("user",
                                         explode("recommendations")
                                         .alias("recommendation"))
recommendations = recommendations.select("user",
                                         "recommendation.item",
                                         "recommendation.rating")
```

Hình 3.6 Lấy khuyến nghị cho người dùng

```
# Hiển thị khuyến nghị
recommendations.show(truncate=False)
```

Hình 3.7 Hiển thị khuyến nghị cho người dùng

1.2.6. Hàm run(self)

Chức năng: Sử dụng các hàm được trình bày ở trên để thực hiện

- Huấn luyện mô hình với các giá trị rank khác nhau và tính toán MSE.
- Vẽ biểu đồ so sánh MSE của tập huấn luyện và tập kiểm tra.
- Đưa ra các khuyến nghị cho một số người dùng cụ thể.

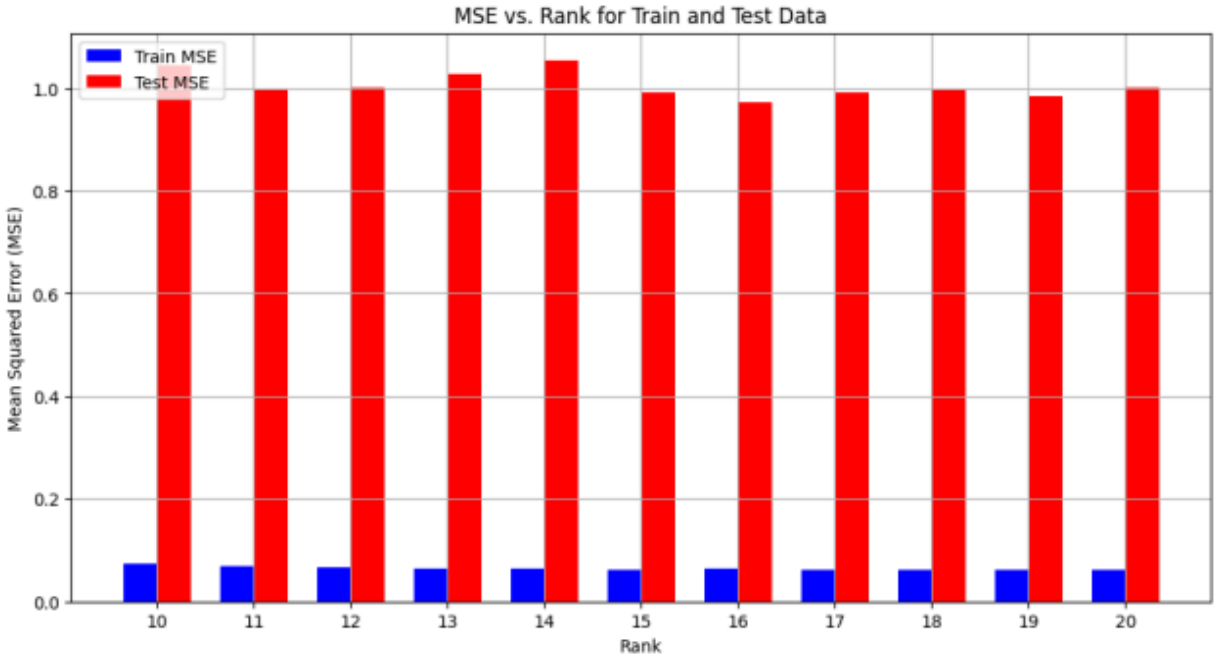
Kết quả của hàm như sau:

Kết quả đánh giá MSE trên training_data và test_data với giá trị rank thuộc khoảng [10,20]

```
Rank: 10, Train MSE: 0.07114943961013666, Test MSE: 1.2545634962317758
Rank: 11, Train MSE: 0.06601208193954475, Test MSE: 1.2154045891478833
Rank: 12, Train MSE: 0.06352040844449992, Test MSE: 1.241752488749613
Rank: 13, Train MSE: 0.0634881464343109, Test MSE: 1.1851011834425735
Rank: 14, Train MSE: 0.06207706520272037, Test MSE: 1.2402474053075536
Rank: 15, Train MSE: 0.06007378930430928, Test MSE: 1.198749413854865
Rank: 16, Train MSE: 0.05958955410115725, Test MSE: 1.2002075267485965
Rank: 17, Train MSE: 0.059684493691241555, Test MSE: 1.1971307327658962
Rank: 18, Train MSE: 0.058429828900684114, Test MSE: 1.2175196509096209
Rank: 19, Train MSE: 0.059276816572469464, Test MSE: 1.2050979068173673
Rank: 20, Train MSE: 0.05882472610287887, Test MSE: 1.192493082951519
```

Hình 3.8 Kết quả đánh giá MSE

Biểu đồ so sánh MSE của tập huấn luyện và tập kiểm tra



Hình 3.9 Biểu đồ so sánh MSE của tập huấn luyện và tập kiểm tra

Inference khuyến nghị cho user = {1,5}, với 5 item được khuyến nghị cho mỗi user

user	item	rating
1	352	4.9729304
1	168	4.8322177
1	39	4.1583924
1	324	4.05625
1	21	3.7489126

Hình 3.10 khuyến nghị cho user =1

user	item	rating
5	187	4.935555
5	36	4.8949347
5	356	4.8000045
5	397	4.744708
5	324	4.5986457

Hình 3.11 khuyến nghị cho user =5

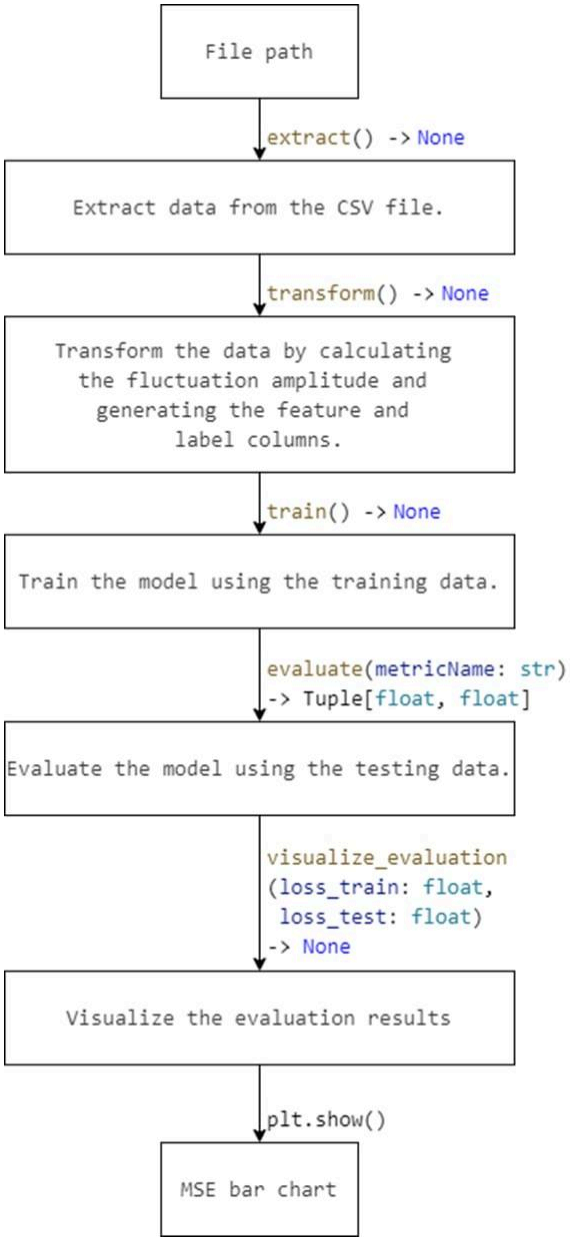
YÊU CẦU 4: DỰ ĐOÁN GIÁ CHỨNG KHOÁN

4.1 Mô tả bài toán.

Yêu cầu: Sử dụng tập dữ liệu **stockHVN2022.csv**:

- Cho biên độ dao động giá chứng khoán k ngày liền trước của mã HVN, dự đoán biên độ của ngày tiếp theo sử dụng mô hình Linear Regression (pyspark)
- Tính ra sai số Mean Square Error trên tập training và test với mô hình đã huấn luyện.
- Sử dụng matplotlib.pyplot vẽ biểu đồ cột thể hiện giá trị Mean Square Error trên tập training và test.

4.2 Giải quyết bài toán



Hình 4.1 Sơ đồ dự đoán giá chứng khoán

4.2.1 Lớp StockPriceAmplitudePrediction

Lớp đối tượng `StockPriceAmplitudePrediction` được thiết kế để dự đoán biên độ dao động giá cổ phiếu của một mã cổ phiếu nào đó. Lớp này sử dụng mô hình hồi quy tuyến tính `LinearRegression` để dự đoán biên độ dao động giá cổ phiếu. Mô tả các thuộc tính và phương thức của lớp `StockPriceAmplitudePrediction` được thể hiện trong bảng dưới đây:

StockPriceAmplitudePrediction
+ StockPriceAmplitudePrediction(file_path: str, K: int = 5, is_percentage: bool = True)
+ extract()
+ plot_data(NO_ROWS_TO_SHOW: int = 10000, title: str = "Stock Graph")
+ transform()
+ train()
+ evaluate(metricName: str = "mse"): Tuple[float, float]
+ visualize_evaluation(loss_train: float, loss_test: float)
+ run()

Bảng 4.1: Dự Đoán Biên Độ Giá Chứng Khoán

Các thuộc tính của lớp

o file_path: str - Đường dẫn tới tệp dữ liệu.

o K: int - Số lượng ngày trước để dự đoán biên độ dao động giá cổ phiếu.

o is_percentage: bool - Flag xác định biên độ dao động giá cổ phiếu được tính theo đơn vị phần trăm hay không.

o spark: SparkSession - Đối tượng SparkSession.

o model: Model - Mô hình hồi quy tuyến tính.

o df: DataFrame - DataFrame chứa toàn bộ dữ liệu.

o training_data: DataFrame - DataFrame chứa dữ liệu huấn luyện.

o testing_data: DataFrame - DataFrame chứa dữ liệu kiểm tra.

o featuresCol: str - Tên cột chứa đặc trưng dùng để train mô hình và dữ liệu để dự đoán (X).

o labelCol: str - Tên cột chứa nhãn dùng để train mô hình (y).

o date_col: str - Tên cột chứa ngày.

o price_col: str - Tên cột chứa giá cổ phiếu theo ngày.

o test_predictions: DataFrame - DataFrame chứa dự đoán của mô hình trên tập kiểm tra.

Các thuộc tính của lớp

extract(): extract(): Trích xuất dữ liệu từ tệp, đọc vào DataFrame và chuyển đổi cột ngày thành kiểu dữ liệu ngày tháng.

plot_data(NO_ROWS_TO_SHOW, title): Vẽ biểu đồ giá cổ phiếu theo ngày, giới hạn số dòng và đặt tiêu đề.

transform(): Tính biên độ dao động giá cổ phiếu, tạo cột mới cho biên độ dao động của K ngày trước và ngày tiếp theo, chia dữ liệu thành tập huấn luyện và kiểm tra.

train(): Huấn luyện mô hình hồi quy tuyến tính trên tập huấn luyện.

evaluate(metricName): Đánh giá mô hình, trả về Mean Squared Error (MSE) của tập huấn luyện và kiểm tra.

visualize_evaluation(loss_train, loss_test): Trực quan hóa kết quả đánh giá MSE của tập huấn luyện và kiểm tra.

run(): Chạy toàn bộ quy trình dự đoán biên độ dao động giá cổ phiếu, bao gồm trích xuất, chuyển đổi và huấn luyện.

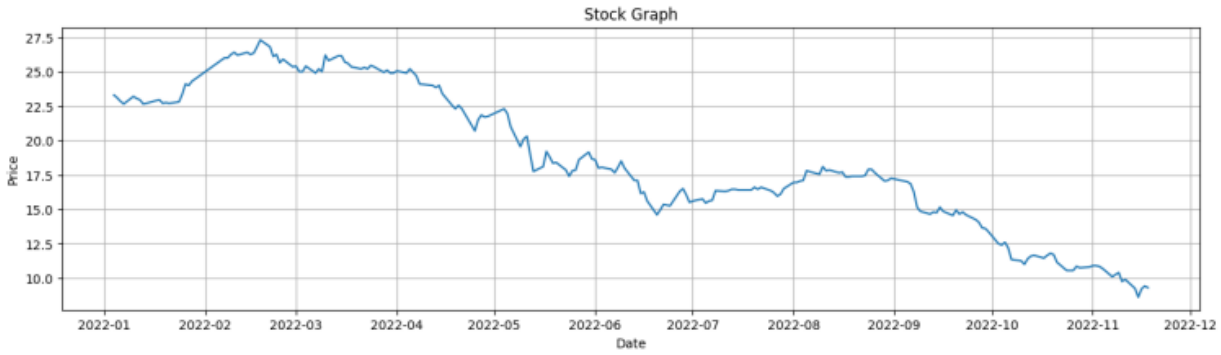
4.2.2 Sử dụng lớp StockPriceAmplitudePrediction

1. Khởi tạo một đối tượng ***StockPriceAmplitudePrediction()*** với tham số **bắt buộc** *file_path* là đường dẫn tới tệp dữ liệu cần dự đoán và các tham số khác nếu cần.

2. Gọi phương thức ***run()*** để chạy toàn bộ quy trình dự đoán biên độ dao động giá cổ phiếu.

3. Gọi các phương thức khác để trực quan hóa dữ liệu, huấn luyện mô hình, đánh giá mô hình, v.v. Như ở đây tôi thực hiện:

- *`.plot_data()`* - Vẽ biểu đồ để trực quan dữ liệu giá cổ phiếu đầu vào.



Hình 4.2. Biểu đồ giá cổ phiếu HVN

- *`.evaluate()`* - Đánh giá mô hình và lần lượt lưu kết quả đánh giá vào `*loss_train*` và `*loss_test*`.

o *`.visualize_evaluation()`* - Trực quan hóa kết quả đánh giá mô hình, biểu đồ cột thể hiện kết quả đánh giá Mean Squared Error (MSE) của tập huấn luyện và tập kiểm tra.

Hình 4.3 Biểu đồ MSE của tập huấn luyện và tập kiểm tra

o *`.test_predictions.show()`* - Hiển thị dự đoán của mô hình trên tập kiểm tra.

```
+----------+-----+--------------------+--------------------+--------------------+--------------------+
|      Ngay|  HVN|         fluctuation|      ampl_Kd_before|          ampl_next|          prediction|
+----------+-----+--------------------+--------------------+--------------------+--------------------+
|2022-07-01| 15.6|0.006451612903225784|[-0.0342679127725...|0.006451612903225784|0.008579916685473676|
|2022-07-04|15.75|0.009615384615384638|[0.00645161290322...|0.009615384615384638|0.004753977842311...|
|2022-07-05|15.45|-0.01904761904761...|[0.00961538461538...|-0.01904761904761...|-0.00777426943257257|
|2022-07-06| 15.6|0.009708737864077693|[-0.0190476190476...|0.009708737864077693|-0.01118983966620...|
|2022-07-07|15.65|0.003205128205128...|[0.00970873786407...|0.003205128205128...|0.003307230591493...|
|2022-07-08|16.35|  0.0447284345047924|[0.00320512820512...|  0.0447284345047924|-0.00332000399618021|
|2022-07-11| 16.3|-0.00305810397553...|[0.04472843450479...|-0.00305810397553...|-6.72697183292789...|
|2022-07-12|16.35|0.003067484662576...|[-0.0030581039755...|0.003067484662576...|-0.00956332650430...|
|2022-07-13|16.45|0.006116207951070...|[0.00306748466257...|0.006116207951070...|-1.04471842602637...|
|2022-07-14|16.45|                 0.0|[0.00611620795107...|                 0.0|0.003752061721746179|
|2022-07-15| 16.4|-0.00303951367781...|[0.0,0.0061162079...|-0.00303951367781...|-0.00447455046936...|
|2022-07-18| 16.4|                 0.0|[-0.0030395136778...|                 0.0|-0.00241730684757...|
|2022-07-19| 16.4|                 0.0|[0.0,-0.003039513...|                 0.0|-0.00129821377956...|
|2022-07-20| 16.6|0.012195121951219686|[0.0,0.0,-0.00303...|0.012195121951219686|-0.00283812266974...|
|2022-07-21|16.45|-0.00903614457831...|[0.01219512195121...|-0.00903614457831...|-0.00174395599361...|
|2022-07-22| 16.6| 0.009118541033434 78|[-0.0090361445783...| 0.009118541033434 78|-0.00593936620197...|
|2022-07-25|16.35|-0.01506024096385542|[0.00911854103343...|-0.01506024096385542|3.352234010622068...|
|2022-07-26| 16.2|-0.00917431192660...|[-0.0150602409638...|-0.00917431192660...|-0.00460436117354...|
|2022-07-27|15.95| -0.0154320987654321|[-0.0091743119266...| -0.0154320987654321|-0.00214264673401...|
|2022-07-28| 16.1|0.009404388714733676|[-0.0154320987654...|0.009404388714733676|-0.00188058308567...|
+----------+-----+--------------------+--------------------+--------------------+--------------------+
only showing top 20 rows
```

Hình 4.4.DataFrame chứa dự đoán của mô hình trên tập kiểm tra

YÊU CẦU 5: PHÂN LOẠI ĐA LỚP VỚI PYSPARK

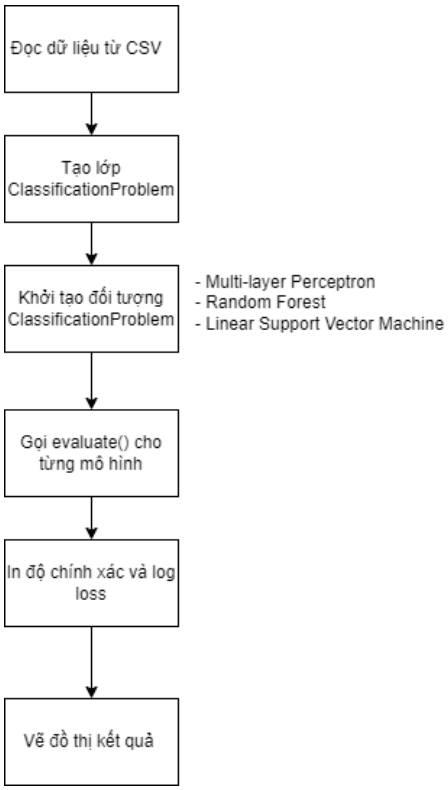
5.1 Mô tả bài toán.

Yêu cầu: Sử dụng tập dữ liệu **mnist_mini.csv**:

- Xây dựng mô hình phân loại đa lớp với pyspark:
  - Multi-layer Perceptron
  - Random Forest
  - Linear Support Vector Machine

Vẽ biểu đồ cột đôi với matplotlib.pyplot để thể hiện độ chính xác của ba mô hình trên tập training và test.

5.2 Giải quyết bài toán



Hình 5.1 Sơ đồ bài toán

5.2.1. Đọc dữ liệu

Dữ liệu được đọc từ tệp CSV bằng Spark

5.2.2. Định nghĩa lớp ClassificationProblem

Lớp ClassificationProblem bao gồm các phương thức để xử lý dữ liệu, xây dựng mô hình, huấn luyện mô hình, và đánh giá mô hình.

- Khởi tạo lớp (__init__)
  - Khởi tạo các thuộc tính để lưu trữ mô hình và dữ liệu, bao gồm các mô hình MLP, RF và LSVC, cùng với dữ liệu huấn luyện và kiểm thử.
- Tạo mô hình (create_model)
  - Gọi các phương thức tạo mô hình con để tạo ra các mô hình MLP, RF và LSVC.
- Tạo mô hình MLP (create_model_MLP)
  - Tạo mô hình MLP với các lớp ẩn xác định và số lần lặp là 100. MLP có cấu trúc nhiều lớp perceptron, phù hợp cho bài toán phân loại phức tạp.
- Tạo mô hình RF (create_model_RF)
  - Tạo mô hình Random Forest với 10 cây quyết định. Sử dụng Pipeline để tiền xử lý dữ liệu bao gồm việc chuyển đổi nhãn và các đặc trưng.
- Tạo mô hình LSVC (create_model_LSVC)
  - Tạo mô hình Linear Support Vector Classifier với số lần lặp là 100 và tham số điều chuẩn là 0.1. LSVC phù hợp cho bài toán phân loại nhị phân.

5.2.3. Tiền xử lý dữ liệu

- Tiền xử lý dữ liệu chính (data_progressing)
  - Chuyển đổi các cột đặc trưng thành vector đặc trưng và chia dữ liệu thành tập huấn luyện và kiểm thử với tỷ lệ 60:40.
- Tiền xử lý dữ liệu cho LSVC (data_progressing_LSVC)

- Chuyển đổi nhãn thành nhãn nhị phân để phù hợp với yêu cầu của mô hình LSVC.

5.2.4. Huấn luyện mô hình

Phương thức train huấn luyện các mô hình MLP, RF và LSVC trên tập dữ liệu huấn luyện đã được tiền xử lý.

5.2.5. Tính toán Cross Entropy

Phương thức cross_entropy tính toán độ đo cross entropy để đánh giá chất lượng của mô hình MLP và RF. Cross entropy đo lường sự khác biệt giữa phân phối xác suất dự đoán và phân phối thực tế.

5.2.6. Đánh giá mô hình

Phương thức evaluate đánh giá độ chính xác của các mô hình trên tập kiểm thử và huấn luyện bằng cách sử dụng MulticlassClassificationEvaluator.

5.2.7. Chạy bài toán

Phương thức run thực hiện toàn bộ quy trình từ tiền xử lý dữ liệu, tạo mô hình, đến huấn luyện mô hình. Sau đó, độ chính xác và cross entropy của các mô hình được in ra để đánh giá.

5.2.8.Kết quả và đánh giá

- Độ chính xác trên tập kiểm thử và huấn luyện:
  - Kết quả được đánh giá bằng cách so sánh độ chính xác của các mô hình MLP, RF và LSVC trên cả tập kiểm thử và tập huấn luyện.
- Cross Entropy:
  - Cross entropy của các mô hình MLP và RF được tính toán để đánh giá mức độ phù hợp của các mô hình với dữ liệu.

PHÂN CÔNG CÔNG VIỆC

MSSV	Email	Họ và tên	Công việc	Hoàn thành
52100337	52100337@student.tdtu.edu.vn	Nguyễn Đông Triều	Câu 1	100%
52100927	52100927@student.tdtu.edu.vn	Đinh Phú Quốc	Câu 2	100%
52100177	52100177@student.tdtu.edu.vn	Nguyễn Tiến Đạt	Câu 3	100%
52100920	52100920@student.tdtu.edu.vn	Nguyễn Minh Phú	Câu 4	100%
52100923	52100923@student.tdtu.edu.vn	Đinh Thị Ngọc Phượng	Câu 5 + report	100%

TÀI LIỆU THAM KHẢO

[1] Tác giả: chưa xác định, “pyspark.sql.DataFrame”. [Trực tuyến]. Địa chỉ: https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.html [Truy cập 18/05/2024].

[2] Tác giả: chưa xác định, “pyspark library”. [Trực tuyến]. Địa chỉ: https://spark.apache.org/docs/latest/api/python/ [Truy cập 18/05/2024]

[3] Tác giả: chưa xác định, “Singular Value Decomposition (SVD)”. [Trực tuyến]. Địa chỉ: https://spark.apache.org/docs/latest/ml-clustering.html#singular-value-decomposition-svd [Truy cập 18/05/2024].

[4] Tác giả: chưa xác định, “Alternating Least Squares (ALS)”. [Trực tuyến]. Địa chỉ: https://spark.apache.org/docs/latest/ml-collaborative-filtering.html#alternating-least-squares-als [Truy cập 18/05/2024].

[5] Tác giả: chưa xác định, “Multi-layer Perceptron Classifier”. [Trực tuyến]. Địa chỉ: https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier [Truy cập 18/05/2024].

[6] Tác giả: chưa xác định, “Random Forest Classifier”. [Trực tuyến]. Địa chỉ: https://spark.apache.org/docs/latest/ml-classification-regression.html#random-forest-classifier [Truy cập 18/05/2024].

[7] Tác giả: chưa xác định, “Linear Support Vector Machine”. [Trực tuyến]. Địa chỉ: https://spark.apache.org/docs/latest/ml-classification-regression.html#linear-support-vector-machine [Truy cập 18/05/2024].