

**Vietnam General Confederation of Labor
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY**



FINAL PROJECT

MASSIVE DATA PROCESSING TECHNIQUES IN DATA SCIENCE

Instructor: **MSc. NGUYEN THANH AN**

Student: **Do Minh Quan - 521H0290**

Ho Huu An - 521H0489

Van Cong Nguyen Phong - 521H0287

Nguyen Le Phuoc Tien - 521H0514

Class : **21H50301**

Year : **25**

HO CHI MINH CITY, 2024

**Vietnam General Confederation of Labor
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY**



FINAL PROJECT

MASSIVE DATA PROCESSING TECHNIQUES IN DATA SCIENCE

Instructor: **MSc. NGUYEN THANH AN**

Student: **Do Minh Quan - 521H0290**

Ho Huu An - 521H0489

Van Cong Nguyen Phong - 521H0287

Nguyen Le Phuoc Tien - 521H0514

Class : **21H50301**

Year : **25**

HO CHI MINH CITY, 2024

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to MSc. Nguyen Thanh An from Ton Duc Thang University for his invaluable support and guidance throughout the Massive Data Processing Techniques In Data Science. His expertise in the field of Massive Data Processing Techniques In Data Science has been instrumental in shaping the direction of this project.

MSc. Nguyen Thanh An's commitment to excellence and his willingness to share his knowledge have significantly contributed to the success of this endeavor. His mentorship has not only enhanced our understanding of Pyspark, algorithm tackling enormous data but has also inspired me to explore new horizons in the realm of handling massive data.

We extend our heartfelt thanks to Mr. NGUYEN THANH AN for his continuous encouragement and support, which have played a pivotal role in the completion of this project. His dedication to fostering a learning environment has been a source of inspiration, and we are grateful for the opportunity to work under his guidance.

Ho Chi Minh city, 20th May, 2024

Author

(Sign and write full name)



Do Minh Quân



Hồ Hữu An



Văn Công Nguyên Phong



Nguyễn Lê Phước Tiến

THIS PROJECT WAS COMPLETED AT TON DUC THANG UNIVERSITY

We fully declare that this is our own project and is guided by Mr. NGUYEN THANH AN; The research contents and results in this topic are honest and have not been published in any form before. The data in the tables for analysis, comments and evaluation are collected by the author himself from different sources, clearly stated in the reference section.

Besides that, the project also uses a number of comments, assessments as well as data from other authors, other agencies and organizations, with citations and source annotations.

Should any frauds were found, we will take full responsibility for the content of our report. Ton Duc Thang University is not related to copyright and copyright violations caused by me during the implementation process (if any).

Ho Chi Minh city, 20th May, 2024

Author

(Sign and write full name)




Do Minh Quân



Hồ Hữu An



Văn Công Nguyên Phong



Nguyễn Lê Phước Tiến

SUMMARY

This essay delves into the realm of mining massive data by leveraging PySpark's capabilities, specifically focusing on DataFrames, SVD for dimensionality reduction, ALS for recommendation systems, Linear Regression for time series prediction, and various classifiers for multi-class classification. Each algorithm's strengths align with specific data analysis needs. The objective is to extract meaningful insights efficiently from vast datasets while maintaining scalability and performance.

Through the implementation of various data processing and analysis tasks using PySpark, several key lessons were learned. First, PySpark's distributed computing capabilities ensure scalability and performance, making it suitable for handling large datasets. Selecting the appropriate algorithm for each task, such as k-Means for clustering or ALS for recommendation systems, is crucial for achieving accurate results. Effective data preprocessing and feature engineering significantly impact analysis outcomes, highlighting the importance of preparing data thoroughly. Consistent evaluation using metrics like Mean Squared Error (MSE) and accuracy provides clear performance measures, aiding in model comparison. Visualization of results facilitates interpretation and uncovering insights.

Organizing code with Object-Oriented Programming (OOP) principles enhances readability and maintainability. Integrating different techniques, such as combining clustering with dimensionality reduction, leads to deeper insights. Empirical evaluation with real data validates model robustness and real-world applicability. Lastly, PySpark's flexibility and iterative model refinement underscore the importance of continuous learning and improvement in data analysis tasks.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	i
SUMMARY	iii
LIST OF ABBREVIATIONS	2
LIST OF FIGURES	3
LIST OF TABLES	4
CHAPTER 1 – INTRODUCTION	5
1.1 PySPark	5
1.2 Kmeans Clustering	5
1.3 SVD	6
1.4 ALS	8
1.5 Practical application	9
CHAPTER 2 – IMPLEMENT	11
2.1 Task 1: Clustering	11
2.2 Task 2: Dimensionality Deduction with SVD	12
2.3 Task 3: Recommendation with Collaborative Filtering	15
2.4 Task 4: Stock price regression	16
2.5 Task 5: Multi-class classification	19
CHAPTER 3 – EVALUATION	21
3.1 Task assignments	21
3.2 Self- assessment	21
3.3 Advantages versus disadvantages	21
CHAPTER 4 – REFERENCES	23

LIST OF ABBREVIATIONS

1. MSE : Mean Squared Error
2. OOP: Object-Oriented Programming
3. SVD: Singular Value Decomposition
4. ALS: Alternating Least Squares

LIST OF FIGURES

Figure 1: PySpark.....	5
Figure 2: KMeanss Clustering	6
Figure 3: SVD	7
Figure 4: Example of SVD in practical.....	8
Figure 5: Matrix Factorization - MF and Collaborative Filtering - CF	9

LIST OF TABLES

Table 1: Task assignments	21
Table 2: Self-Assessment	21

CHAPTER 1 – INTRODUCTION

1.1 PySPark

PySpark is an interface for Apache Spark in Python. With PySpark, you can write Python and SQL-like commands to manipulate and analyze data in a distributed processing environment.



Figure 1: PySpark

Apache Spark is written in Scala programming language. To support Python with Spark, Apache Spark Community released a tool, PySpark. Using PySpark, you can work with RDDs in Python programming language also. It is because of a library called Py4j that they are able to achieve this.

PySpark offers PySpark Shell which links the Python API to the spark core and initializes the Spark context. Majority of data scientists and analytics experts today use Python because of its rich library set. Integrating Python with Spark is a boon to them.

1.2 Kmeans Clustering

Let us understand the K-means clustering algorithm with its simple definition. A K-means clustering algorithm tries to group similar items in the form of clusters. The number of groups is represented by K. Let's take an example. Suppose you went to a vegetable shop to buy some vegetables. There you will see different kinds of vegetables. The one thing you will notice there that the vegetables will be arranged in a group of their types. Like all the carrots will be kept in one place, potatoes will be kept

with their kinds and so on. If you will notice here then you will find that they are forming a group or cluster, where each of the vegetables is kept within their kind of group forming the clusters.

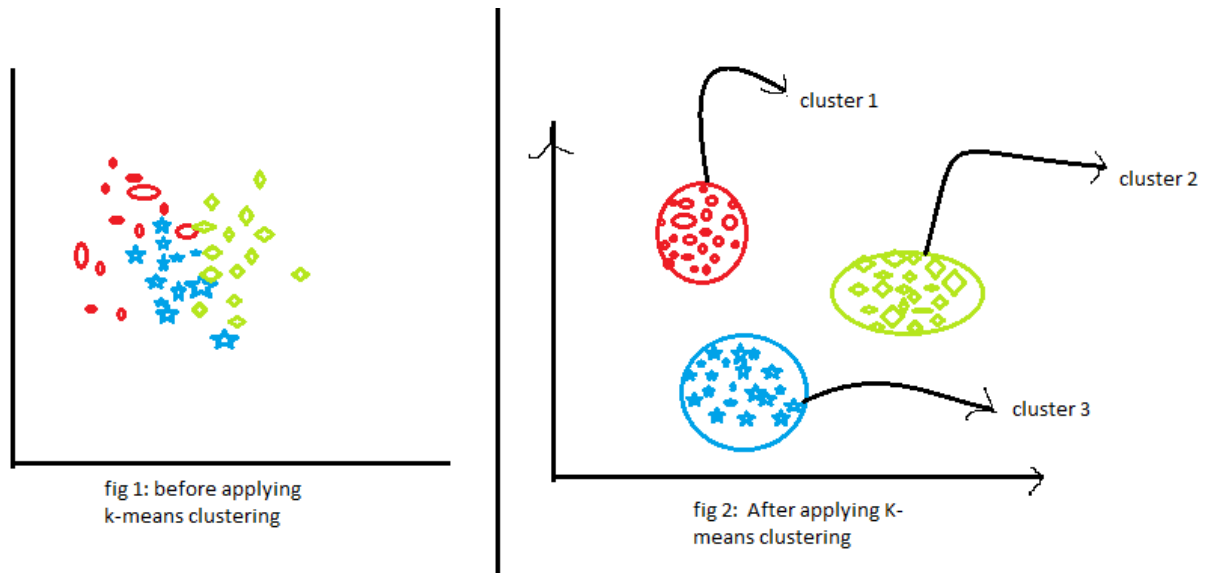


Figure 2: KMeanss Clustering

Now, look at the above two figures. what did you observe? Let us talk about the first figure. The first figure shows the data before applying the k-means clustering algorithm. Here all three different categories are messed up. When you will see such data in the real world, you will not be able to figure out the different categories.

Now, look at the second figure(fig 2). This shows the data after applying the K-means clustering algorithm. you can see that all three different items are classified into three different categories which are called clusters.

1.3 SVD

Mathematics Behind SVD

Consider an input data matrix of order $m \times n$ i.e. the matrix has m rows and n columns. For instance, if we think of this matrix as a document matrix then every row represents a document, and every column representing a word.

The idea behind SVD is to take this matrix of order $m \times n$ and represent it as a product of three matrices, further these three matrices would be having certain constraints associated with it.

$$A_{[m \times n]} = U_{[m \times r]} \Sigma_{[r \times r]} (V_{[n \times r]})^T$$

Figure 3: SVD

- A: Input data matrix — $m \times n$ matrix (eg. m documents, n terms)
- U: Left Singular Vectors — $m \times r$ matrix (m documents, r concepts)
- Σ : Singular Values — $r \times r$ diagonal matrix (strength of each 'concept') where r is rank of matrix A
- V: Right Singular Vectors — $n \times r$ matrix (n terms, r concepts)

An Example

Let us keep the above theorem and concepts of SVD in the back of our minds and understand how these concepts could be helpful in solving a real-life problem. Let us think about the Users to Movies matrix. Let's say we are Netflix or a user review website and we have a matrix that consists of users and movies. User-based on their interest would have watched and rated a set of movies that they have seen. A value of 1 depicts they did not like the movie and 5 represents that they really liked the movie. This means every column represents a different movie whereas every row corresponds to a different user.

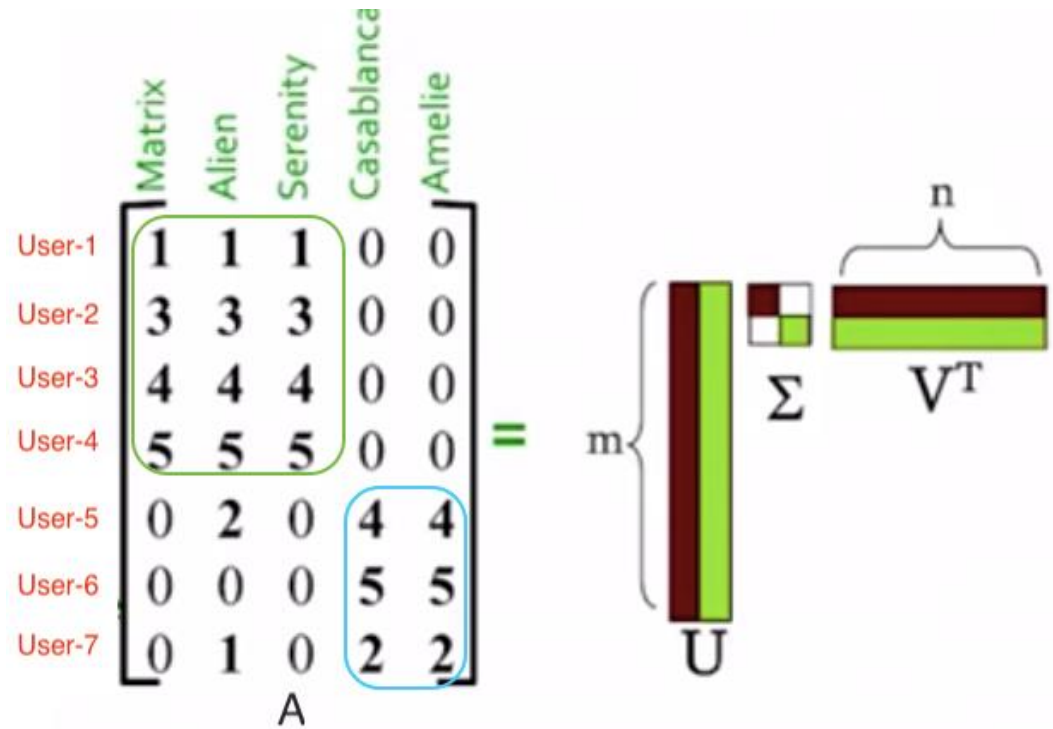


Figure 4: Example of SVD in practical

We would like to decompose matrix A into thinner and smaller matrices U , Σ , and V . Here we can notice we have set users who prefer watching Sci-fi movies and another set of users (bottom three users) who prefer watching non-Sci-fi movies.

1.4 ALS

Collaborative Filtering using Alternating Least Squares (ALS) is a popular technique in recommendation systems, which are essential for personalizing user experiences on platforms like e-commerce sites, streaming services, and social networks. ALS is an iterative optimization process where we for every iteration try to arrive closer and closer to a factorized representation of our original data.

We have our original matrix R of size $u \times i$ with our users, items and some type of feedback data. We then want to find a way to turn that into one matrix with users and hidden features of size $u \times f$ and one with items and hidden features of size $f \times i$. In

U and V we have weights for how each user/item relates to each feature. What we do is we calculate U and V so that their product approximates R as closely as possible: $R \approx U \times V$.

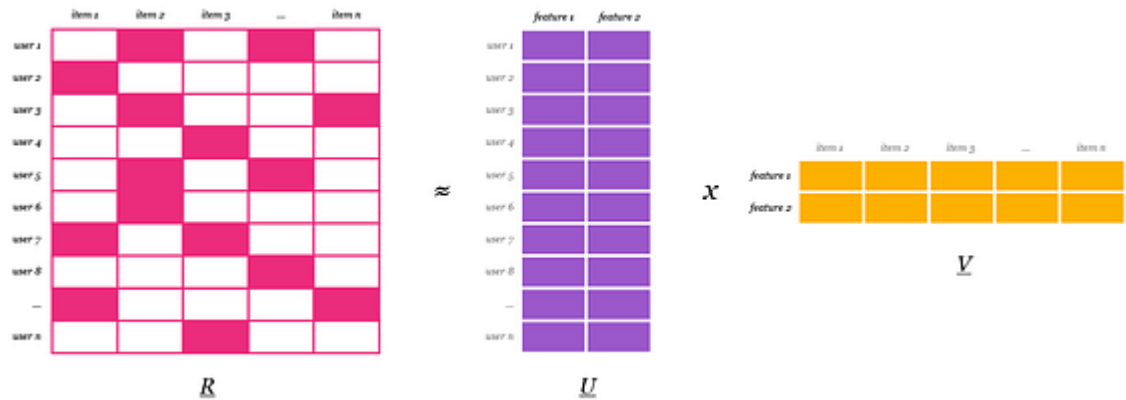


Figure 5: Matrix Factorization - MF and Collaborative Filtering - CF

By randomly assigning the values in U and V and using least squares iteratively we can arrive at what weights yield the best approximation of R. The least squares approach in it's basic forms means fitting some line to the data, measuring the sum of squared distances from all points to the line and trying to get an optimal fit by minimising this value.

1.5 Practical application

- **k-means Clustering**

Description: k-means clustering partitions users or items into k distinct clusters based on their feature vectors.

Application: Used for customer segmentation to provide personalized marketing strategies.

Example: In an e-commerce setting, customers are clustered based on their purchasing behavior. Each cluster represents a group of customers with similar shopping patterns, allowing for tailored marketing campaigns and promotions.

- **Singular Value Decomposition (SVD)**

Description: SVD factorizes the user-item interaction matrix into three matrices: U (users), Σ (singular values), and V (items). It reduces the dimensionality of the data while preserving the essential patterns.

Application: Utilized in recommendation systems to uncover latent features that explain the observed user-item interactions.

Example: Streaming services like Netflix use SVD to recommend movies. By reducing the dimensionality of the user-m

ovie rating matrix, SVD identifies latent factors representing user preferences and movie attributes, allowing for more accurate recommendations even with sparse data.

- **Alternating Least Squares (ALS)**

Description: ALS is a matrix factorization technique that iteratively optimizes the user and item matrices by fixing one matrix and solving for the other. It includes regularization to prevent overfitting.

Application: Applied in collaborative filtering to generate personalized recommendations based on user-item interaction histories.

Example: In e-commerce platforms like Amazon, ALS is used to recommend products. By analyzing the purchase history of users, ALS identifies patterns and suggests products that similar users have bought.

CHAPTER 2 – IMPLEMENT

2.1 Task 1: Clustering

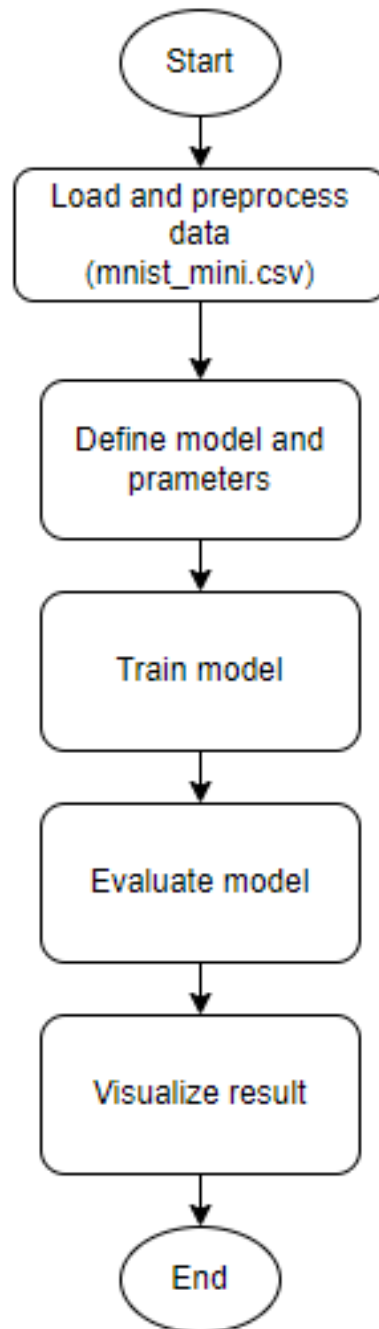


Figure 6: Flowchart of Clustering

Initialization: The class is initialized with the path to the data file, the number of clusters (k), and a list of weighted rows.

Data Loading: The `load_data` method reads the data from the specified file path using Spark's `read.csv` function.

Data Preprocessing: The `preprocess_data` method performs the following tasks:

- Assembles the feature columns into a single vector column named "features".
- Adds a row index column and a weight column based on the provided weighted rows list.

Clustering: The clustering method performs the k-means clustering using the KMeans estimator from Spark ML. It calculates the cluster centers, joins them with the transformed data, and computes the distance between each data point and its corresponding cluster center. Finally, it aggregates the sum of distances and counts for each cluster.

Plotting: The `plot_average_distance` method creates a bar plot showing the average distance for each cluster (prediction) using the Matplotlib library.

Run: The `run` method orchestrates the entire process by calling the `load_data`, `preprocess_data`, `clustering`, and `plot_average_distance` methods in sequence.

2.2 Task 2: Dimensionality Deduction with SVD

Initialization: The class is initialized with the path to the data file and the clustering result DataFrame.

Data Loading: The `load_data` method reads the data from the specified file path using Spark's `read.csv` function.

Data Preprocessing: The `preprocess_data` method assembles the feature columns into a single vector column named "features".

SVD: The svd method performs the following tasks:

- Computes the SVD of the data matrix using Spark's computeSVD function.
- Reconstructs the data matrix by multiplying the left singular vectors (U) and the diagonal matrix of singular values (S).
- Converts the reconstructed data matrix to a Spark DataFrame.
- Joins the reconstructed data with the clustering result DataFrame.
- Samples a small fraction (1%) of the data, limiting it to 100 rows.

Plotting: The plot_3d method creates a 3D scatter plot using Matplotlib, where each cluster is represented by a different color and marker. The plot displays the first three principal components of the reconstructed data.

Run: The run method orchestrates the entire process by calling the load_data, preprocess_data, svd, and plot_3d methods in sequence.

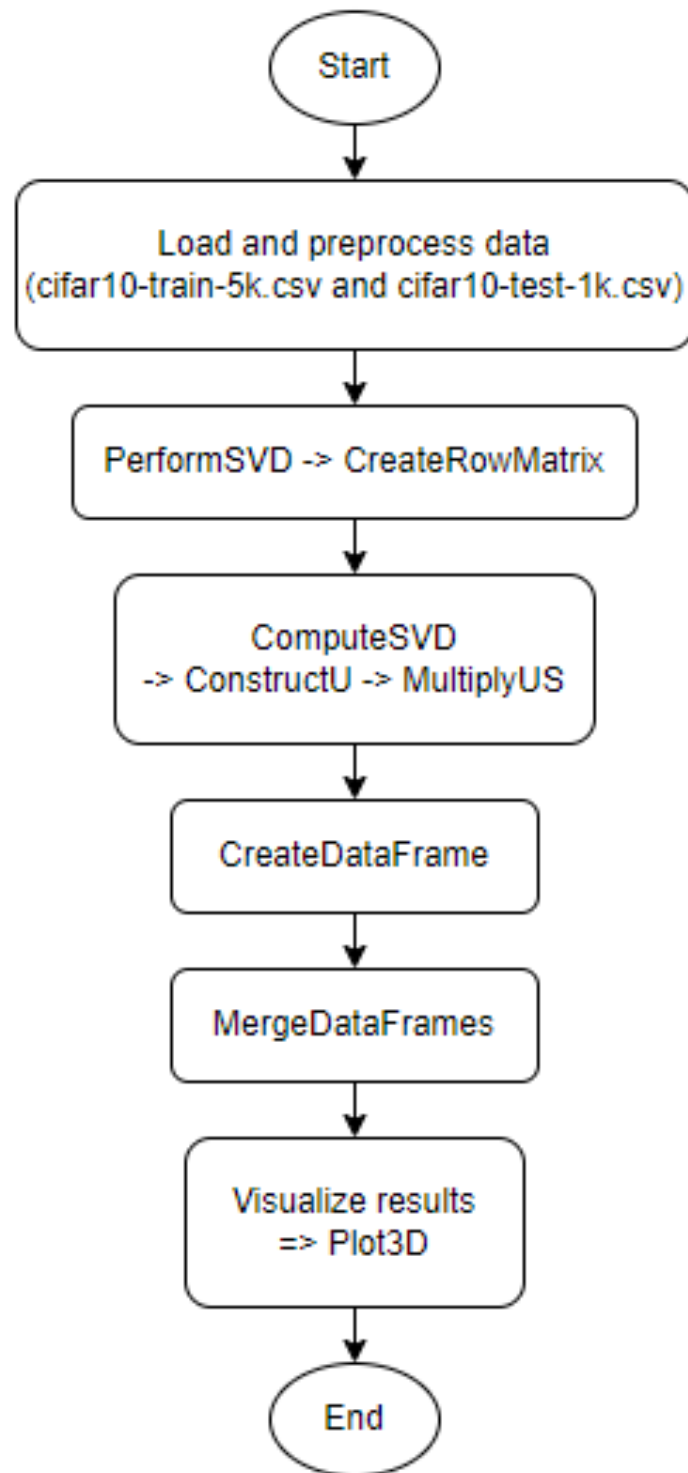


Figure 7: Flowchart of Dimensionality Deduction with SVD

2.3 Task 3: Recommendation with Collaborative Filtering

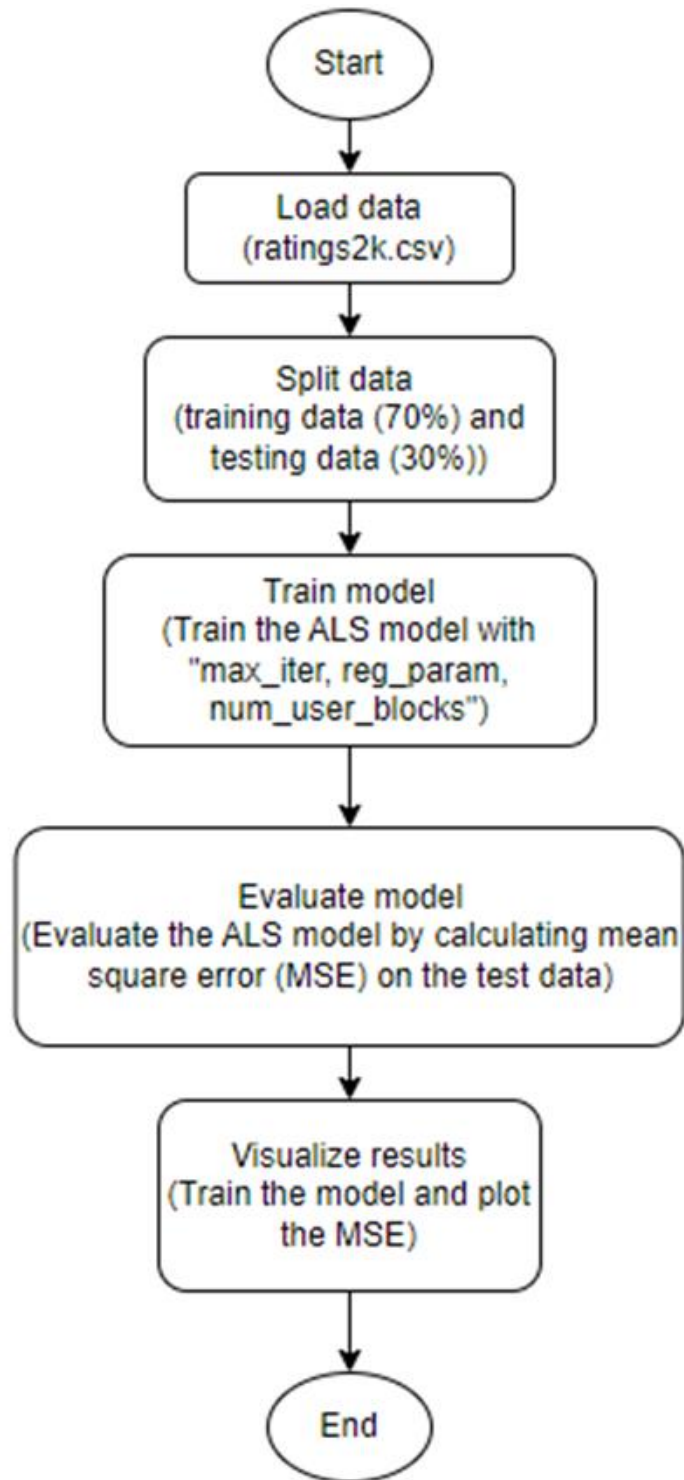


Figure 8: Flowchart of Recommendation with Collaborative Filtering

Initialization: The class is initialized with the path to the rating data file and a range of values for the number of user blocks to be considered during the training process.

Data Loading: The `load_data` method reads the rating data from the specified file path using Spark's `read.csv` function.

Data Splitting: The `split_data` method randomly splits the loaded data into training and testing sets, with a 70/30 ratio.

Model Training: The `train_model` method trains the ALS model on the training data, using the specified rank, maximum iterations, and regularization parameter.

Model Evaluation: The `evaluate_model` method evaluates the trained model's performance on the testing data by computing the Mean Squared Error (MSE) between the predicted and actual ratings.

Visualization: The `visualize_results` method trains multiple ALS models with different ranks (number of similar users) and plots the resulting MSE values against the rank values. This visualization helps in selecting an appropriate rank for the final model.

Run: The `run` method orchestrates the entire process by calling the `load_data`, `split_data`, and `visualize_results` methods in sequence.

2.4 Task 4: Stock price regression

Initialization: The class is initialized with the number of previous dates to consider for prediction and the path to the stock price data file.

Data Loading: The `load_data` method reads the stock price data from the specified file path using Spark's `read.csv` function.

Data Preprocessing: The `preprocess_data` method performs several data transformations, including converting the date column to a date format, calculating

price fluctuations, creating lagged features for the specified number of previous dates, and splitting the data into training and testing sets.

Model Training: The `train_model` method trains a linear regression model using Spark's `LinearRegression` estimator on the training data.

Model Evaluation: The `evaluate_model` method evaluates the trained model's performance on both the training and testing data by computing the Mean Squared Error (MSE) using Spark's `RegressionEvaluator`.

Visualization: The `plot_results` method creates a bar plot using Matplotlib to visualize the MSE values for both the training and testing datasets.

Run: The `run` method orchestrates the entire process by calling the `load_data`, `preprocess_data`, `train_model`, `evaluate_model`, and `plot_results` methods in sequence.

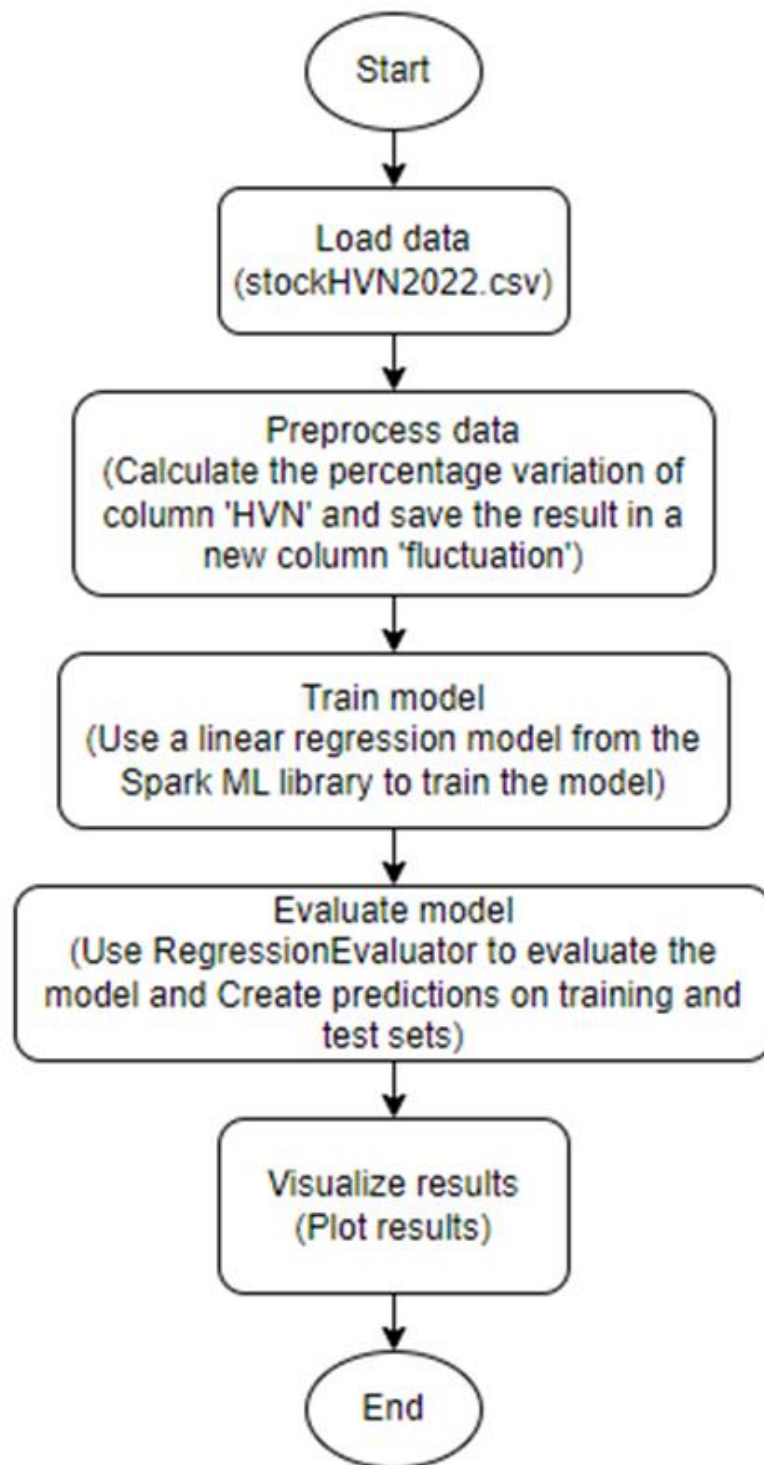


Figure 9: Flowchar of Stock price regression

2.5 Task 5: Multi-class classification

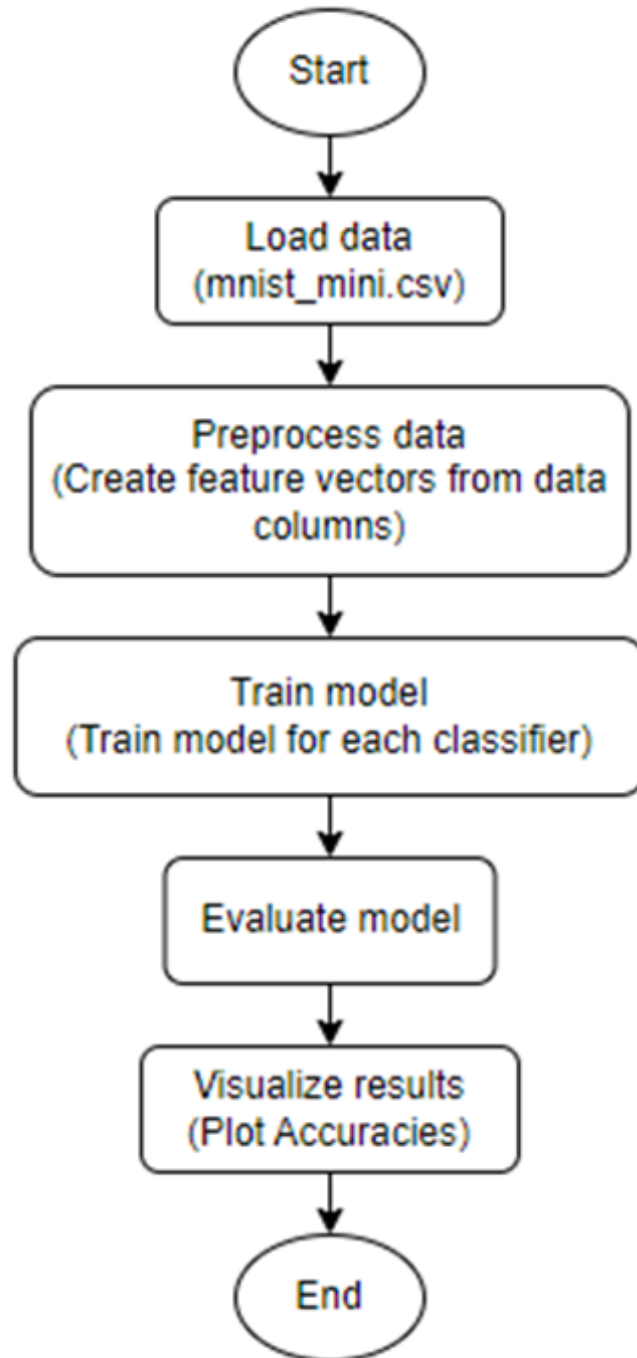


Figure 10: Flowchart of Multi-class classification

Initialization: The class is initialized with the path to the data file.

Data Loading: The `load_data` method reads the data from the specified file path using Spark's `read.csv` function.

Data Preprocessing: The `preprocess_data` method performs feature engineering by assembling the input columns into a single "features" column and renaming the target column as "label".

Classification: The `run_classification` method trains and evaluates multiple classification models, including Multilayer Perceptron (MLP), Random Forest, and Linear Support Vector Machine (SVM) with One-vs-Rest strategy. It splits the data into training and testing sets, trains each model on the training data, and computes the accuracy scores for both training and testing data.

Visualization: The `plot_accuracies` method creates a bar plot using Matplotlib to visualize the training and testing accuracies for each classification model.

Run: The `run` method orchestrates the entire process by calling the `load_data`, `preprocess_data`, `run_classification`, and `plot_accuracies` methods in sequence.

CHAPTER 3 – EVALUATION

3.1 Task assignments

Table 1: Task assignments

Full name	Email	Assigned tasks	Completion Percentage
Do Minh Quan	521H0290@student.tdtu.edu.vn	Task 3, 4	100%
Ho Huu An	521H0489@student.tdtu.edu.vn	Task 1, 2, 3, 4, 5, 6	100%
Van Cong Nguyen Phong	521H0287@student.tdtu.edu.vn	Task 4, 5	100%
Nguyen Le Phuoc Tien	521H0514@student.tdtu.edu.vn	Task 5, 6	100%

3.2 Self- assessment

Table 2: Self-Assessment

Requirement	Complete Percentage
Task 1	100%
Task 2	100%
Task 3	100%
Task 4	100%
Task 5	100%

3.3 Advantages versus disadvantages

Advantages :

- Diverse perspectives: Working in a group brings together individuals with diverse backgrounds, experiences, and ideas. This diversity can lead to

innovative solutions and a broader range of insights when tackling problems related to K-Means SVD, ALS algorithm and machine learning.

- Collaboration and synergy: Group work encourages collaboration among team members. They can share their knowledge, skills, and resources, leading to a synergy that enhances the overall quality of the project. Collaborative problem-solving can help overcome challenges more effectively.
- Reduced workload: Sharing the workload among team members can reduce the individual burden and prevent burnout. It allows team members to focus on specific aspects of the project, ensuring better attention to detail and overall project quality.

Disadvantages :

- Coordination and communication challenges: Managing a group project requires effective coordination and communication. It can be challenging to ensure that all team members are on the same page, have a clear understanding of the project goals, and are working towards them efficiently.
- Differences in work styles and commitment levels: Team members may have different work styles, levels of commitment, and priorities. This can lead to conflicts and disagreements within the group, affecting the overall progress and cohesion of the project.

CHAPTER 4 – REFERENCES

- [1] J. G. Deng, "Apache Spark Machine Learning Library (MLlib) Documentation," Apache Software Foundation, 2023. [Online]. Available: <https://spark.apache.org/docs/latest/ml-guide.html>. [Accessed: 15-May-2023].

- [2] J. G. Deng, "Classification and Regression - Apache Spark 3.5.1 Documentation," Apache Software Foundation, 2023. [Online]. Available: <https://spark.apache.org/docs/latest/ml-classification-regression.html>. [Accessed: 15-May-2023].

- [3] J. G. Deng, "Clustering - Apache Spark 3.5.1 Documentation," Apache Software Foundation, 2023. [Online]. Available: <https://spark.apache.org/docs/latest/ml-clustering.html>. [Accessed: 15-May-2023].

- [4] J. G. Deng, "Collaborative Filtering - Apache Spark 3.5.1 Documentation," Apache Software Foundation, 2023. [Online]. Available: <https://spark.apache.org/docs/latest/ml-collaborative-filtering.html>. [Accessed: 15-May-2023].

- [5] J. G. Deng, "Linear Regression - Apache Spark 3.5.1 Documentation," Apache Software Foundation, 2023. [Online]. Available: <https://spark.apache.org/docs/latest/ml-regression.html#linear-regression>. [Accessed: 15-May-2023].

- [6] J. G. Deng, "Dimensionality Reduction - Apache Spark 3.5.1 Documentation," Apache Software Foundation, 2023. [Online]. Available:

<https://spark.apache.org/docs/latest/ml-features.html#dimensionality-reduction>.
[Accessed: 15-May-2023].

- [7] J. Hunter, D. Hsu, and M. Droettboom, "Matplotlib: Python Plotting," Matplotlib Project, 2023. [Online]. Available: <https://matplotlib.org/>. [Accessed: 15-May-2023].
- [8] M. Zaharia et al., "Apache Spark: A Unified Engine for Big Data Processing," Communications of the ACM, vol. 59, no. 11, pp. 56-65, Nov. 2016.
- [9] J. G. Deng, "PySpark API Documentation," Apache Software Foundation, 2023. [Online]. Available: <https://spark.apache.org/docs/latest/api/python/index.html>. [Accessed: 15-May-2023].