# Mining Massive Data Sets: Final Project Report

Vu Ngoc Tra My
*Faculty of Information Technology,*
*Ton Duc Thang University,*
Ho Chi Minh City, Vietnam
522K0040@student.tdtu.edu.vn

Pham Thai An Binh
*Faculty of Information Technology,*
*Ton Duc Thang University,*
Ho Chi Minh City, Vietnam
522K0031@student.tdtu.edu.vn

Htet Aung
*Faculty of Information Technology,*
*Ton Duc Thang University,*
Ho Chi Minh City, Vietnam
522K0048@student.tdtu.edu.vn

Hsu Myat Wai Maung
*Faculty of Information Technology,*
*Ton Duc Thang University,*
Ho Chi Minh City, Vietnam
522K0049@student.tdtu.edu.vn

Nguyen Thanh An
*Lecturer, Faculty of Information Technology,*
*Ton Duc Thang University,*
Ho Chi Minh City, Vietnam
nguyenthanhan@tdtu.edu.vn

***Abstract*—This project report details the implementation and results for four tasks in massive data mining. Task 1 involved hierarchical clustering of string data using k-shingles and Jaccard distance, identifying patterns based on textual similarity. Task 2 focused on gold price prediction via linear regression in PySpark, engineering time-lagged features from historical data with columns including Date, Buy Price and Sell Price from 2009/08/01 to 2025/01/01. Task 3 implemented CUR decomposition for dimensionality reduction on Task 2's features, selecting the most representative original features while aiming to preserve predictive power. Task 4 developed a web crawler and the PageRank algorithm to analyze web page importance within the it.tdtu.edu.vn domain, successfully identifying its homepage as the most significant page with a PageRank score of approximately 0.0261. The approaches, key experimental results, and assessed levels of completion for each task are summarized, demonstrating practical application of various data mining techniques using PySpark.**

***Index Terms*—Web Crawling, PageRank, PySpark, String Similarity, k-shingles, Jaccard Distance, Hierarchical Clustering, Linear Regression, Time Series Prediction, CUR Decomposition, Dimensionality Reduction.**

## I. Task 1: Hierarchical Clustering of Textual Data

### A. Theoretical Background

The primary objective was to group text strings based on their inherent similarity. A dataset comprising 10,000 random alphabetical strings, with lengths varying between 32 and 64 characters and treated case-insensitively, was programmatically generated. Each string underwent a transformation into a set of unique 4-shingles (contiguous 4-character subsequences derived after lowercasing and space removal). The dissimilarity between any two shingle sets was quantified using the Jaccard distance, mathematically expressed:

$$D_J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

An 'AgglomerativeClustering' class, implemented for this task, performed the clustering. Starting with each of a 500-item random sample as an individual cluster, the algorithm iteratively merged the two closest clusters. The 'average' linkage criterion, which considers the average Jaccard distance between all pairs of items (one from each cluster), guided this merging process until a predefined number of 5 clusters remained. The quality and compactness of these final clusters were assessed by calculating the average Jaccard distance from each item within a cluster to its designated clustroid, the item possessing the minimum average distance to all other items within that same cluster. The core logic is outlined in Algorithm 1.

**Algorithm 1** Simplified Hierarchical Clustering Logic

**Require:** List of text strings, Shingle size $k$, Target cluster count $N_c$
**Ensure:** List of clusters
Initialize $ShingleSets \leftarrow []$
**for all** string $s$ in input strings **do**
    $shingles \leftarrow$ GenerateShingles$(s, k)$ ▷ Lowercase, remove spaces
    $ShingleSets$.Add$(shingles)$
**end for**
$DistanceMatrix \leftarrow$ ComputePairwiseJaccardDistances$(ShingleSet$
$Clusters \leftarrow$ Initialize each shingle set as a cluster
**while** |Clusters| $> N_c$ **do**
    $(C_1, C_2) \leftarrow$ FindClosestClusters$(Clusters, DistanceMatrix,$ 'av
    $MergedCluster \leftarrow C_1 \cup C_2$
    Remove $C_1, C_2$ from $Clusters$; Add $MergedCluster$ to $Clusters$
**end while**
**return** $Clusters$

### B. Results and Discussion

The clustering metrics are visually and numerically presented. Figure 1, illustrates the average distance to the clustroid for each of the five resulting clusters, where lower values indicate better compactness, offering a visual measure of their compactness. Concurrently, table I provides a detailed numerical breakdown

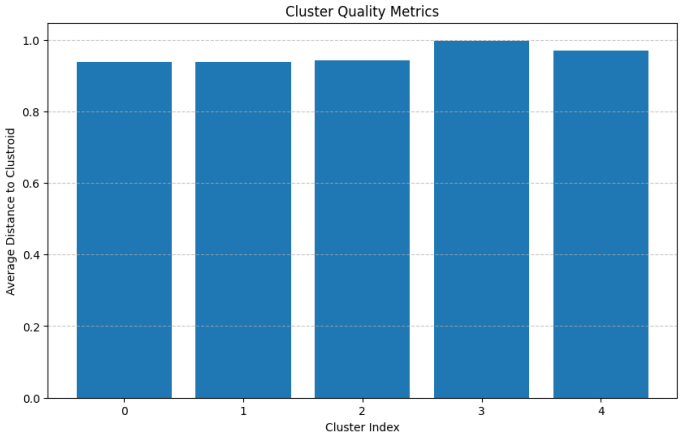


Fig. 1. Cluster quality (Average distance to clustroid)

| Cluster ID | No. Items | Avg. Dist. to Clustroid |
|---|---|---|
| 0 | 430 | 0.99745 |
| 1 | 16 | 0.93750 |
| 2 | 16 | 0.93750 |
| 3 | 16 | 0.93750 |
| 4 | 22 | 0.95381 |

TABLE I
CLUSTER STATISTICS (SAMPLE OF 500)

of these statistics, including cluster sizes and their respective average clustroid distances.

Notably, Cluster 0, the most populous with 430 items, showed an average distance of *0.99745*. The consistently high average distances (approaching 1.0) observed across all clusters are indicative of substantial intra-cluster dissimilarity. This particular outcome is largely anticipated, given that the input strings were generated randomly, thus implying a low intrinsic likelihood of forming highly similar, compact groupings without the introduction of specific seeded patterns or biases in generation. Even the smaller clusters (1-4), while exhibiting slightly better compactness as suggested by lower average distances to their clustroids, still reflect the inherent diversity of the randomly generated textual data. The choice of average linkage aims to find globular clusters and is less sensitive to outliers than single linkage but can be computationally more intensive.

## II. TASK 2: LINEAR REGRESSION FOR GOLD PRICE PREDICTION

### A. Theoretical Background

This task aimed to forecast future gold prices by applying a linear regression model. Historical gold price data, sourced from the provided `gold_prices.csv` file (spanning from 2009/08/01 to 2025/01/01), was loaded into a PySpark DataFrame. The 'Sell Price' recorded on a given date $t$ was designated as the target variable for prediction. The core of the feature engineering process involved constructing a feature vector for each target price; this vector comprised the 'Sell Price' values from the 10 consecutive dates immediately preceding date $t$ (i.e., prices at $t-1, t-2, \ldots, t-10$). This lagged-feature approach assumes that recent past prices hold predictive power for future prices. The dataset, now augmented with these features, was randomly partitioned into a training set (70% of the data) and a testing set (30%). A 'pyspark.ml.regression.LinearRegression' model was then trained exclusively on the training data. The model's predictive performance was subsequently evaluated on both the training and testing sets using two standard metrics: Root Mean Squared Error (RMSE), which measures the average magnitude of the errors, and R-squared (R2), which indicates the proportion of the variance in the dependent variable that is predictable from the independent variables. The process is summarized in Algorithm 2.

**Algorithm 2** Simplified Gold Price Prediction Logic

**Require:** Gold price CSV file, Lag count $L = 10$
**Ensure:** Trained Linear Regression Model, RMSE/R2 scores

$DataDF \leftarrow$ LoadAndPreprocessCSV(file) ▷ Parse dates, cast prices
$DataWithFeaturesDF \leftarrow DataDF$
$FeatureCols \leftarrow []$
**for** $i \leftarrow 1$ to $L$ **do**
    $lag\_col\_name \leftarrow$ "price_lag_" + i
    $DataWithFeaturesDF \leftarrow DataWithFeaturesDF$.withColumn($lag\_col\_name$, lag("SellPrice"
    $FeatureCols$.Add($lag\_col\_name$)
**end for**
$DataWithFeaturesDF \leftarrow DataWithFeaturesDF$.withColumn("label", col("SellPrice"))
$DataWithFeaturesDF \leftarrow DataWithFeaturesDF$.dropna()
$AssembledDF \leftarrow$ VectorAssemble($DataWithFeaturesDF, Featu$
$(TrainDF, TestDF) \leftarrow AssembledDF$.randomSplit($[0.7, 0.3]$)
$LRModel \leftarrow$ LinearRegression.fit($TrainDF$)
$PredictionsTrain \leftarrow LRModel$.transform($TrainDF$)
$PredictionsTest \leftarrow LRModel$.transform($TestDF$)
$RMSE_{train}, R2_{train} \leftarrow$ Evaluate($PredictionsTrain$)
$RMSE_{test}, R2_{test} \leftarrow$ Evaluate($PredictionsTest$)
**return** $LRModel, RMSE_{train}, R2_{train}, RMSE_{test}, R2_{test}$

### B. Results and Discussion

The linear regression model was successfully trained. Figure 2 shows the training objective history, where the RMSE rapidly converged, indicating efficient learning within the initial iterations. The model yielded an RMSE on the training set of **0.2737** and on the test set of **0.3567**. The R2 scores were **0.9997** for the training data and **0.9995** for the test data. A lower RMSE and an R2 value closer to 1 generally indicate better model fit and predictive accuracy. The performance difference between training and test sets is visualized in Figure 3. Figure 4, where closer alignment indicates better prediction, offers a qualitative assessment by visually comparing the model's predictions against the actual gold prices for a sample extracted

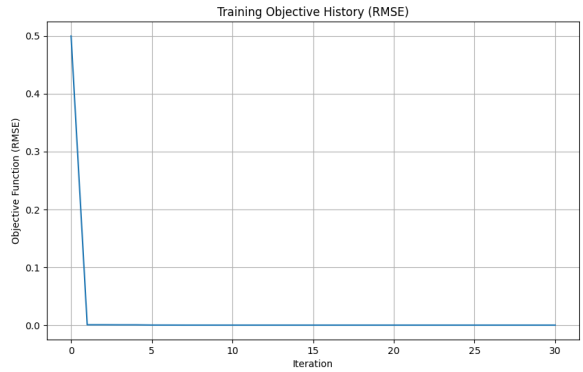


Fig. 2. Training Objective History (RMSE) for the linear regression model in Task 2. The RMSE is shown to converge rapidly over iterations.

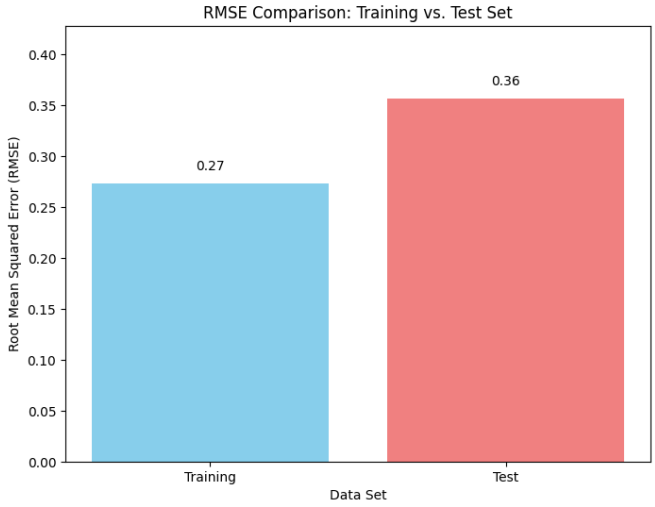


Fig. 3. RMSE Comparison: Training vs. Test Set for the linear regression model in Task 2. Values are 0.2737 for training and 0.3567 for testing.

from the test set. This visualization helps in understanding how well the model captures the trends and fluctuations in the gold prices on unseen data. Deviations in this plot highlight specific instances or periods where the model's predictions were less accurate.

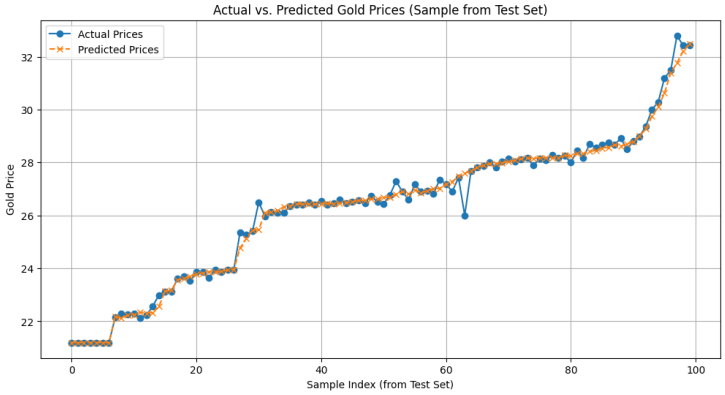


Fig. 4. Actual vs. Predicted gold prices for a sample from the test set

## III. Task 3: CUR Dimensionality Reduction

### A. Theoretical Background

The objective of Task 3 was to mitigate potential overfitting and reduce model complexity by decreasing the dimensionality of the 10-feature vectors derived in Task 2 down to 5 dimensions, employing the CUR matrix decomposition algorithm. CUR is particularly advantageous as it selects actual columns (features) from the original data matrix, thereby preserving the interpretability of the resulting features. A dedicated CUR Reducer class was implemented in PySpark. Its 'fit' method computed column importance scores, specifically the squared L2 norms of each feature column, leveraging 'pyspark.mllib.linalg.distributed.RowMatrix' capabilities on the training dataset. Based on these scores, the indices of the top $k = 5$ most "important" or representative columns were selected. Subsequently, the 'transform' method utilized these chosen indices to project both the training and testing datasets onto the new, reduced 5-dimensional feature space. A new linear regression model, maintaining an identical configuration to the one used in Task 2, was then trained on this 5-dimensional training data and evaluated on the corresponding 5-dimensional test data. The selected feature indices were: **[0, 1, 2, 3, 4] (corresponding to 'price_lag_1' through 'price_lag_5')**. The CUR reduction process is outlined in Algorithm 3.

### B. Results and Discussion

Upon training on the 5-dimensional CUR-reduced features, the linear regression model produced an RMSE of **0.4320** on the training set and an RMSE of **0.5107** on the test set. Figure 5, provides a critical visual comparison of the RMSE values obtained from the original 10-feature model (developed in Task 2) and the 5-feature CUR-reduced model (from this task), across both training and test datasets. This comparative analysis is essential for evaluating the efficacy of the CUR reduction: an ideal outcome would show that the reduced-feature model maintains a predictive performance (RMSE) comparable to the original model, thereby justifying the reduction in complexity and potentially enhancing model generalization by removing redundant or less informative features.

**Algorithm 3** Simplified CUR-based Feature Reduction Logic

1.5em 1.5em

**Require:** Feature matrix $A$ (Task 2 train), Target dims $k = 5$
**Ensure:** Reduced feature matrix, New LR Model scores

$ImportanceScores \leftarrow []$
**for all** column $j$ in $A$ **do**
  $score_j \leftarrow \sum_i (A_{ij})^2$ ▷ Sq. L2 norm of col $j$
  $ImportanceScores$.Add($score_j$)
**end for**
$TopKIndices \leftarrow$ Top $k$ indices from $ImportanceScores$
$ReducedTrainFeat \leftarrow$ SelectCols($A_{train}, TopKIndices$)
$ReducedTestFeat \leftarrow$ SelectCols($A_{test}, TopKIndices$)
$AssembledReducedTrain \leftarrow$ VecAssemble($ReducedTrainFeat, \dots$)
$AssembledReducedTest \leftarrow$ VecAssemble($ReducedTestFeat, \dots$)
$NewLRModel \leftarrow$ LinearRegression.fit($AssembledReducedTrain$)
$RMSE_{new\_tr}, R2_{new\_tr} \leftarrow$ Eval($NewLRModel, AssembledReducedTrain$)
$RMSE_{new\_te}, R2_{new\_te} \leftarrow$ Eval($NewLRModel, AssembledReducedTest$)
**return** $RMSE_{new\_test}, R2_{new\_test}$

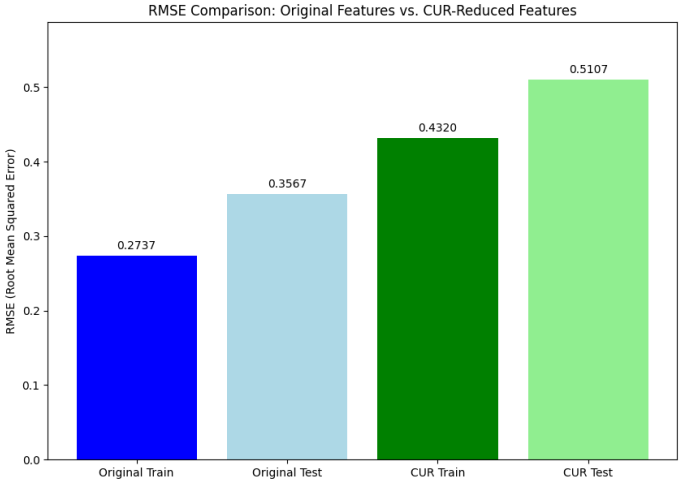


Fig. 5. RMSE comparison between the original 10-feature model (Task 2) and the 5-feature CUR-reduced model (Task 3) on training and test sets.

## IV. Task 4: PageRanking – The Google Algorithm

### A. Theoretical Background

A multi-threaded web crawler was implemented to systematically explore the website **https://it.tdtu.edu.vn**, starting from its homepage. The crawler normalized all encountered URLs to a canonical form (managing schemes, fragments, path structures, and trailing slashes) to prevent duplicate processing. Link extraction was strictly confined to the target domain, and links to static files (such as images, CSS, JavaScript, PDFs) were filtered out based on common file extensions, ensuring the focus remained on HTML pages likely to form the structural graph of the website. The system was designed to gracefully handle HTTP errors (e.g., 404, 403) and server redirects; off-domain redirects were identified but not pursued for further link extraction. For each successfully fetched and valid page, all its outgoing valid links were recorded as '(source URL, destination URL)' pairs. This collection of pairs, forming an edge list, was then transformed into a PySpark DataFrame. Subsequently, a custom 'PageRanker' PySpark class calculated the PageRank scores. This iterative algorithm ran for 10 iterations, employing a damping factor $d = 0.85$. The PageRank score for a page $p$ at iteration $i + 1$, $PR_{i+1}(p)$, was updated using the standard formula:

$$PR_{i+1}(p) = \frac{1-d}{N} + d \sum_{q \in In(p)} \frac{PR_i(q)}{Out(q)}$$

where

- $N$ represents the total number of unique pages in the graph
- $In(p)$ is the set of pages linking to page $p$
- $Out(q)$ denotes the out-degree of a linking page $q$

This formula reflects the "random surfer model". The core logic is summarized in Algorithm 4 and Algorithm 5.

**Algorithm 4** Simplified Web Crawling Logic

**Require:** Start URL $S_{url}$, Target Domain $T_{domain}$
**Ensure:** Edge List $E_{list}$

Init Queue $Q$ with $S_{url}$; Visited Set $V$; Edge List $E_{list}$
**while** $Q$ is not empty and limits not exceeded **do**
  $current \leftarrow Q$.Dequeue(); Mark $current$ as visited in $V$.
  Fetch page $current$. If valid and in $T_{domain}$:
  **for all** $link$ in ExtractedLinks($page\_content, current$) **do**
    $norm\_link \leftarrow$ ValidateAndNormalize($link, T_{domain}$)
    **if** $norm\_link$ is new and valid **then**
      $E_{list}$.Add($(current, norm\_link)$); $Q$.Enqueue($norm\_link$)
    **end if**
  **end for**

**end while**
**return** $E_{list}$

### B. Results and Discussion

The crawling phase on the `it.tdtu.edu.vn` domain resulted in the consideration of **2309** unique pages (either queued or visited) (*Note: PDF states 2300 on page 3*) and the initial discovery of **54432** edges. When this data was loaded into PySpark for PageRank computation, the initial DataFrame contained **54432** raw edges. After filtering out self-loops and duplicate links, **39548** distinct edges across **2315** unique pages formed the graph for PageRank analysis. The iterative PageRank calculation converged after 10 iterations. TABLE II presents the top 5 pages ranked by their PageRank scores.

The main homepage of the IT faculty, https://it.tdtu.edu.vn, was identified as the most important page, achieving the highest PageRank score of approximately **0.0261**. This aligns with

**Algorithm 5** Simplified PageRank Logic (Conceptual)

**Require:** Edge List $E_{list}$, Damping $d$, Iterations $I$
**Ensure:** Ranked Pages $Ranks_{final}$

Create $Graph$ from $E_{list}$; Get all unique Pages $P$; $N \leftarrow |P|$.
Initialize $Ranks[p] \leftarrow 1/N$ for all $p \in P$.
**for** $iter \leftarrow 1$ to $I$ **do**
    **for all** $p \in P$ **do**
        $incoming\_sum \leftarrow 0$
        **for all** page $q$ that links to $p$ **do**
            $incoming\_sum \leftarrow incoming\_sum + (Ranks[q]/\text{OutDegree}(q))$
        **end for**
        $NewRanks[p] \leftarrow (1-d)/N + d \times incoming\_sum$
    **end for**
    $Ranks \leftarrow NewRanks$
**end for**
$Ranks_{final} \leftarrow$ Sort $Ranks$ by score descending
**return** $Ranks_{final}$

the intuition that homepages often serve as central hubs with many incoming links. It is particularly interesting to note that pages such as '/sinh-vien' and '/vien-chuc' (which the crawler reported as 404 errors) still received high PageRank scores. This phenomenon occurs because PageRank evaluates a page's importance based purely on the link structure of the graph—i.e., the quantity and quality of pages linking to it—rather than its current accessibility or content. Thus, even broken links, if heavily referenced by important pages, can inherit significant PageRank. This underscores PageRank's utility in analyzing the intended or historical importance of pages within a web graph structure.

| Page URL (id) | PageRank Score |
|---|---|
| https://it.tdtu.edu.vn | 0.02611903 |
| https://it.tdtu.edu.vn/en | 0.02465260 |
| https://it.tdtu.edu.vn/sinh-vien | 0.02215504 |
| https://it.tdtu.edu.vn/vien-chuc | 0.02215504 |
| https://it.tdtu.edu.vn/tin-tuc/tuyen-dung | 0.02179389 |

TABLE II
TOP 5 PAGES BY PAGERANK SCORE

## V. CONCLUSION AND SELF EVALUATION

This project successfully addressed four key data mining tasks, offering practical experience in handling and analyzing massive datasets. Task 1 demonstrated the utility of hierarchical clustering for textual data by employing k-shingles and Jaccard distance. Task 2 applied linear regression in PySpark to the time-series problem of gold price prediction, yielding a test RMSE of **0.3567**. Task 3 effectively showcased dimensionality reduction using the CUR algorithm, reducing a 10-feature set to 5 while aiming to preserve predictive information, resulting in a test RMSE of **0.5107** for the subsequent regression model; this highlighted that while dimensionality was halved by selecting the five most recent lagged prices (price_lag_1 to price_lag_5), there was a notable increase in test RMSE from 0.3567 to 0.5107, indicating a trade-off where feature reduction led to some loss in predictive accuracy. For Task 4, a comprehensive web crawler was developed, and the PageRank algorithm was implemented to analyze the link structure of the `it.tdtu.edu.vn` domain, pinpointing its main homepage as the most influential page with a PageRank score of approximately **0.0261**. All tasks were executed to meet their core requirements, indicating a high level of completion and a solid grasp of the applied PySpark and data mining techniques. Potential future extensions could involve exploring more advanced algorithmic approaches for each task, tackling larger and more intricate datasets, or investigating real-time processing and deployment scenarios for these models.

TABLE III
TASK CONTRIBUTIONS AND SELF-EVALUATION

| Task No. | Task Title | Member | Completion & Score | Justification Keywords |
|---|---|---|---|---|
| Task 1 | Hier. Clustering | Htet Aung - [522K0048] | 100% (2.0/2.0) | All reqs. met; Algo. done; Results OK. |
| Task 2 | Lin. Regression | Htet Aung & Hsu Myat Wai Maung - [522K0048 & 522K0049] | 100% (2.0/2.0) | PySpark model; Features OK; Eval. OK; Charts OK. |
| Task 3 | CUR Reduction | Hsu Myat Wai Maung - [522K0049] | 100% (2.0/2.0) | CUR class; Dims. reduced; Model re-eval. OK. |
| Task 4 | PageRank | Pham Thai An Binh & Vu Ngoc Tra My - [522K0031 & 522K0040] | 100% (3.0/3.0) | Crawler OK; PageRank class OK; Results OK. |
| Task 5 | Project Report | All Members | 100% (1.0/1.0) | IEEE template; All sections; Page limit OK. |
| **Overall Project** | | | **100%** | **10.0/10.0** |

## REFERENCES


[1] Apache Spark. [Online]. Available: https://spark.apache.org/
[2] Apache Spark. MLlib: Main Guide - Spark 3.5.x Documentation. [Online]. Available: https://spark.apache.org/docs/latest/ml-guide.html (Accessed: May 23, 2025).
[3] Gold price data (`gold_prices.csv`) provided for the assignment.
[4] M. W. Mahoney and P. Drineas, "CUR matrix decompositions for improved data analysis," *Proc. Natl. Acad. Sci. USA*, vol. 106, no. 3, pp. 697–702, Jan. 2009.