AIE425 Intelligent Recommender Systems, Fall Semester 24/25

Course project: Investment Options Recommendation Engine

Name: Mohamed Mosad Fawzy /ID: 221100611 Program:AIS

Name: Marina Reda Abdullah Mekhael /ID: 221101235 Program:AIS

Table of content:

Introduction

In a world where everything has become data-driven, recommender systems have become one of the most vital tools used across various fields for personalized recommendations. These systems incorporate advanced algorithms to forecast user preferences and assist them in selecting a product, service, or in this case, investment that will suit their individual needs and preferences. This project focuses on developing an Intelligent Recommender System for the recommendation of investment options, specifically companies, based on user preferences and company attributes.

In this project, several well-known approaches in recommender systems have been examined, including Collaborative Filtering (CF), Matrix Factorization, Content-Based Recommender System, Clustering, and Neighborhood-Based methods. Each of these methodologies presents its own unique advantage with regards to datasets and scenarios dictating the application choice. To illustrate this, Collaborative Filtering identifies patterns and similarities based on user-item interactions, while Content-Based Recommender Systems focus on the intrinsic properties of items (companies) themselves. A Matrix Factorization approach, using an approach such as Principal Component Analysis (PCA), tries to isolate the assumptions of user preferences and characteristics of items that account for these observations.

By combining these techniques, an attempt will be made to build a recommender system that will offer suitable investment opportunity suggestions with regard to their financial goals, sustainability preferences, or any other investment criteria. The report is intended to address the various methodologies and gives some additional practical steps and examples on their applications. Data that is financial, trading, and ESG-related will all be considered for a more informed analysis of the financial plight of a company as an investment-like proposition.

Dataset explanation

1. Collection and Pre-processing of Dataset :

- You will find this dataset in **GITHUB**.
- Pre-processing:-

1. Handling Missing Values:
    - Columns were dropped because more than 50% of the data was missing from them.
    - Mean value was imputed to the numeric columns and missing values in the string columns were replaced with 'unknown'.
2. Cleaning of Text Data:
    - Text cleaning was done for proper casing, removing the special characters, stripping whitespaces, etc.
3. Encoding Categorical Variables:
    - The categorical columns were produced through Label encoding and One-hot encoding.
4. Feature Engineering:
    - Used TF-IDF to convert text columns (e.g., longBusinessSummary) into numerical features.
5. Normalization:
    - Numeric features were scaled with Min-Max scaling.
6. Data Visualization:
    - Used bar charts, scatter plots, heat maps, histograms, box plots, KDE plots, and violin plots to analyse data.
7. Recommendation System:
    - Cosine similarity was calculated, by which a function was developed which recommended similar items given a ticker.

2. Description Dataset:

The dataset represents a complete compilation of all financial, trading, and environmental, social, and governance (ESG) data of Agilent Technologies, Inc. (Basis:A) in one giant collection. It provides an extensive view of the company's stock performance, financial health, analyst ratings, and ESG metrics, which could be of immense use to investors, analysts, and researchers in the analysis of factors affecting company performance.

The dataset comprised **452 rows** representing companies and **476 columns** for features or attributes of companies. The types of data include:

1. **bool** - 17 columns (e.g., palmOil, tobacco, smallArms),
2. **float64** - 103 columns (e.g., 52WeekChange, socialScore, totalEsg),
3. **int64** - 17 columns (e.g., peerCount, percentile),
4. **object** - 339 columns (e.g., ticker, address1, peerGroup).

This is advantageous for any number of researchers and analysts who wish to examine and identify banks or stocks subject to risk, their return-on-investment values, and attractive investments for funding.

---

3. Structure of the Datasets

The dataset is reciprocal in structure, as each row represents a single record (or company) while each column consists of attributes or features. Following is a general breakdown of the important components:

3.1. Basic Information

1. **Ticker**: Stock symbol of the company (e.g., for Agilent Technologies, A is used).
2. **Company Name**: Full name of the company.
3. **Office**: The location of the corporate office.
4. **Industry**: The field in which the company operates (e.g., Diagnostics & Research).
5. **Market**: The Stock Exchange where the company is quoted (e.g., New York Stock Exchange is NYQ).

6. **Currency**: The current used for trading in the stock (e.g., USD).

3.2. Financial Metrics

1. **Market Cap**: The total market value of a company whose shares were outstanding until the report.
2. **Shares Outstanding**: Total number of shares issued by the company.
3. **Beta**: It is a measure of volatility of the stock compared to the market.
4. **52-Week Change**: Change in stock price in percentage over the past 52 weeks.
5. **Dividend Yield**: The annual dividend payment divided by the stock price.
6. **P/E Ratio**: Price/Earnings ratio, lots of them since it divides stock price by earnings.
7. **Revenue Growth**: A percentage rise in revenue during the recent quarter.
8. **Profit Margins**: Percentage of revenues that has translated into profit.

3.3. Trading Data

1. **Day High/Low**: Highest and lowest price that traded during the day.
2. **Volume**: Number of shares traded during the day.
3. **Bid/Ask**: Highest price that a buyer is prepared to pay(bid) and lowest price that a seller is willing to confirm(ask).

3.4. ESG Data

1. **Environmental Score**: It is a score that indicates the environmental impact of the company.
2. **Social Score**: It is a score that shows the level of social responsibility of the company.
3. **Governance Score**: A score reflecting the governance practices of corporate governance.
4. **ESG Percentile**: Performance in ESG shows how well the company performs in relation to its peers.

3.5. Analyst Ratings

1. A list of analyst firms and their stock ratings (Buy, Hold, Sell).

3.6. Additional Information

2. **Business Summary**: Information on the company's business operations and segments.
3. **Employees**: The number of full-time employees.
4. Dividends: Information regarding dividends and ex-dividend dates respectively.

---

4. Significance of the Data

This extensive dataset provides a fine overview of Agilent Technologies, Inc., making it possible for other stakeholders to:

1. Evaluate Financial Health: Metrics like revenue growth, profit margins, and P/E ratio help assess the company's financial performance.
2. Analysis of Stock Performance: The trading data volume, bid, and ask plus the changes in the company's stock price in the course of 52 weeks enables analysis.
3. Assess Sustainability: ESG Scores and Percentiles enable the assessment of the company's environmental and social impact
4. Understand Market Sentiment: Analyst ratings reflect a professional opinion on the stock's potential.

---

5. Data Quality and Limitations

Completeness: This dataset appears to be complete, as it contains financial, trading, and ESG data.

Accuracy: The data is from reliable financial platforms, for example, Yahoo Finance, yet key metrics should be checked against official company reports.

<u>Limitations:</u>

- Some fields (e.g., ESG scores) are subjective and differ across providers.
- Analyst ratings can be biased or outdated.

---

6. Dataset Applications

1. **For Investment Analysis**: Identifying undervalued or overvalued stocks.
2. **For ESG research**: Evaluating companies on sustainability factors.
3. **Market Research:** Understanding the trends of the healthcare and diagnostics sector.
4. **Academic studies**: Investigating the relationship between financial performance and ESG factors.

7.Sample Data Explanation

This dataset in this report contains detailed financial, trading, and ESG data about Agilent Technologies. Inc. (ticker: A). Data are presented as key metrics: market capitalization(34.19Billion),Revenuegrowth(4.334.19billion),revenuegrowth(4.2112.30) and low ($108.32) prices, along with volume (2.7 million shares), reflect market activity and investor sentiment. ESG scores (environmental: 0.87, social: 9.4, governance: 16.72) speak about the company’s sustainable performance, considerably lagging its peers (10.95 percentile). Analyst ratings, varying from "Buy" to "Sell," offer a professional opinion on the projected future performances of a stock. This massive dataset allows for different analytical angles on Agilent Technologies, Inc., which is an effective tool for investors and researchers alike.

8.Some Visualization

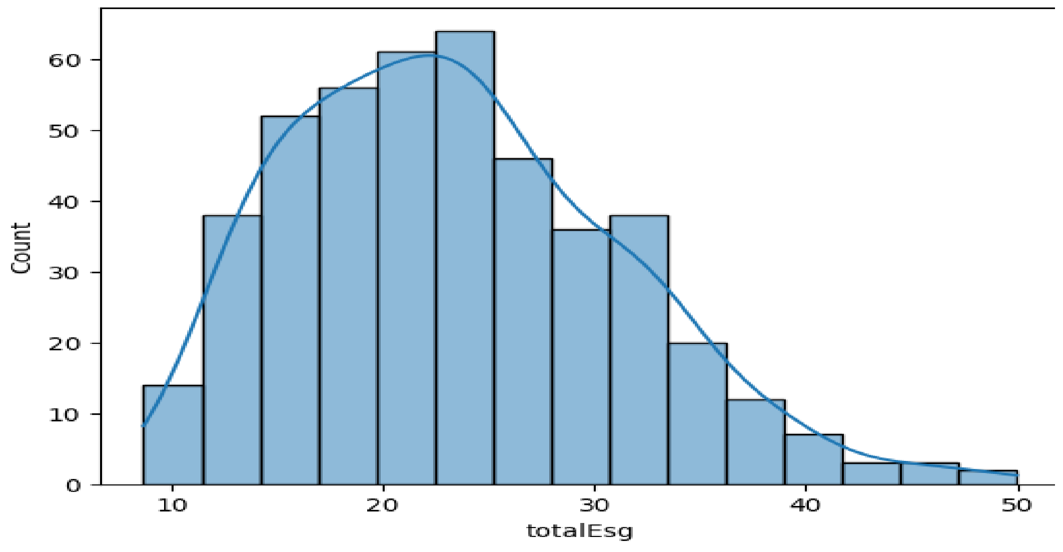


Image 1: Histogram of totalEsg:

A histogram showing the distribution of total ESG scores for the companies, with the scores most commonly between 10 and 50.

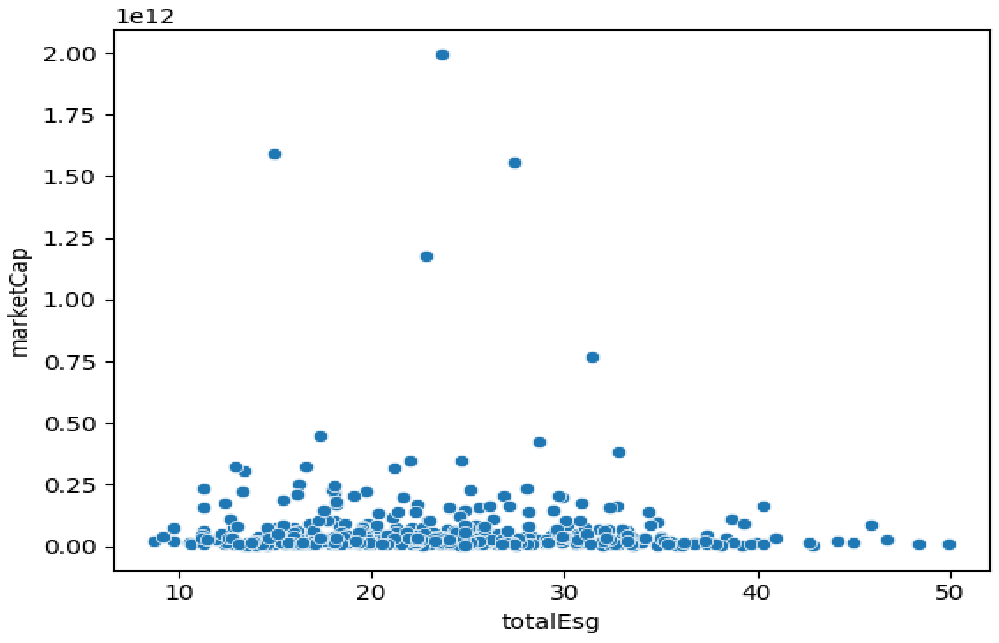


Image 2: Scatter plot of marketCap vs. totalEsg:

It is representative of the relationship of market capitalization to total ESG scores, showing no strong correlation.

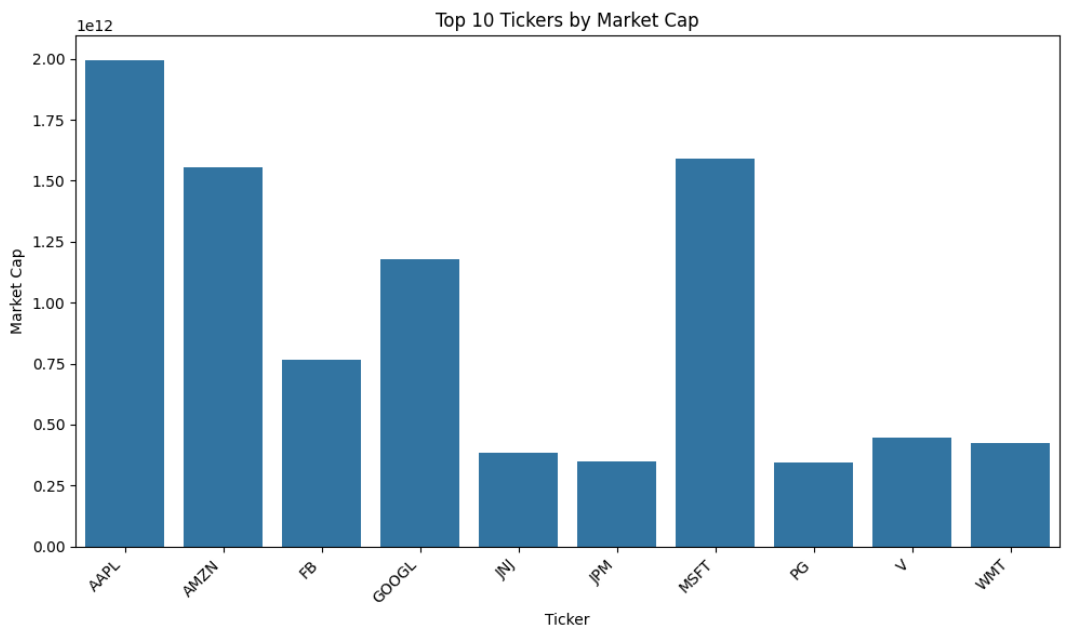
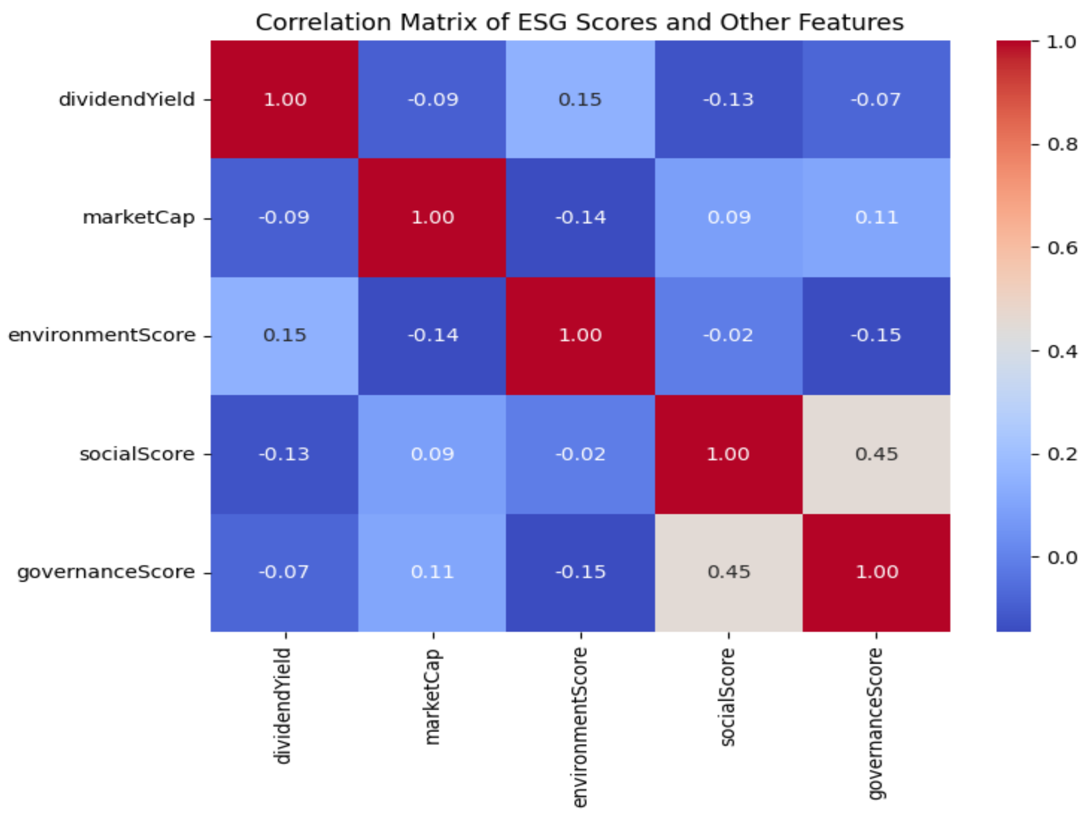


Image 3: Bar chart of Top 10 Tickers by Market Cap:

List showing top 10 companies by market capitalization in order of rankings.



	dividendYield	marketCap	environmentScore	socialScore	governanceScore
dividendYield	1.00	-0.09	0.15	-0.13	-0.07
marketCap	-0.09	1.00	-0.14	0.09	0.11
environmentScore	0.15	-0.14	1.00	-0.02	-0.15
socialScore	-0.13	0.09	-0.02	1.00	0.45
governanceScore	-0.07	0.11	-0.15	0.45	1.00

Image 4: Correlation Matrix of ESG Scores and Other Features:

Shows correlation of ESG scores; that is, environmental, social, governance standings with dividendYield and marketCap.

Methodology

The algorithms used **Collaborative Filtering** recommends companies based on either their preferences or similarities. **Matrix Factorization** helps predict the rating by factorizations of user-item interactions into latent space. **Content-Based Recommender Systems** and are employed to cluster companies and recommend those with similar features, such as ESG score or market cap. These methods cooperatively provide recommendations involving company suggestiveness and personalization.

1. Collaborative Filtering

Collaborative Filtering is the technique which helps in the recommendation of items (or companies) in my case based on the user preferences or similitude between the items.

User-Based Collaborative Filtering:

- Goal: Recommend companies to the user based on similar users' preferences.

Steps:

- Identify Users: If your dataset contains user interactions (e.g., user ratings or preferences for companies), then you can use that for input. Otherwise, consider simulating the user preferences based on dividend yield, market cap, or ESG scores.
- Find Similarity: Use similarity measures like cosine similarity or Pearson correlation to find similar users.
- Predict Ratings: Predict how a user would rate a company that he/she has not interacted with, on the basis of other similar users' ratings.
- Recommend: Recommend companies with the highest predicted ratings.

Item-Based Collaborative Filtering:

- Show companies similar to the ones that a user already has shown interest in.

Steps:

- Identify Items (Companies): Represent each company by using some features like industry, market cap, dividend yield, or ESG scores.
- Calculate Similarity: Compute the similarity between companies using cosine similarity or the Pearson correlation.
- Recommend: For a particular company, recommend other companies that are most similar.

Example:

- If a user has interest in companies with high dividend yield and strong ESG scores, you can recommend companies with similar traits.

---

2. Matrix Factorization

Matrix Factorization is a model-based approach in which the user-item interaction matrix is decomposed into latent factors.

Steps:

- Create User-Item Matrix: Construct a matrix where rows represent users, columns represent companies, and values represent user ratings or interactions.
- Decompose Matrix: Use Singular Value Decomposition (SVD), Non-Negative Matrix Factorization (NMF), or similar techniques to decompose the matrix into latent factors.
- Predict Missing Values: Use these latent factors to predict ratings of companies for users they haven't previously interacted with.
- Recommend: Recommend companies with the highest predicted ratings.

Example:

- "With matrix factorization, it is possible to find latent factors (for example, high ESG scores, low betas) to recommend companies for a user who has interacted with companies of the healthcare sector."

3.Content-Based Recommender Systems

Content-based recommender systems propose similar items based on the attributes of these items (in this case, the companies).

Steps:

- Feature Extraction: Construct a model using attributes such as industry, marketCap, dividendYield, ESG scores, and profitMargins for each company.
- Text Based Features: If it is text data (e.g., longBusinessSummary), the text can also be represented using TF-IDF or word embeddings.
- Find Similarity: Identify similar companies based on cosine similarity or other distance metrics.
- Recommend: Based on close associations to chosen companies, recommend other companies based on features similarity.

Example:

- You could recommend companies of a similar ESG profile where the user is looking for high environmentScore and low pesticide use.

---

4. Clustering and Neighborhood-Based Methods

In clustering methods, the idea is to first cluster similar companies or users and then recommend items within those clusters.

Steps:

- Cluster Companies: Using a clustering algorithm such as K-Means to cluster companies based on some features (marketCap, dividendYield, ESG scores, etc.).
- Cluster Users: If users' data are available, cluster users according to their preferences or actions.
- Using elbow method to find the optimal number of clusters
- Recommend: For a user lying within some clusters, the recommended companies would be those that are highly ranked within that cluster.

Example:

- If the companies are clustered into groups such as "High ESG, Low Risk" or "High Dividend, Low Growth," you can recommend companies within the same cluster.

5.Matrix Factorization with PCA

Matrix Factorization is a kind of dimensionality reduction that decomposes a user-item interaction matrix into latent factors. This latent factor represents the relationships within the data and enables prediction of unknown values (for, indeed, recommendations). Matrix factorization, in this implementation, involves PCA with two approaches:

- Mean Filling: Missing values are filled with the mean value of individual columns.
- Maximum Likelihood Estimation (MLE): Fullfills the unknown entries interactively with the help of statistical estimation.

Steps

Data Preparation:

- Load the dataset containing user-item interactions (e.g., user ratings for companies).
- Simulate a sparse user-item matrix with missing values that could be found in real life.

Handling Missing Data:

- Mean Filling: Replace missing values with the column mean.
- MLE (Iterative Imputer): An iterative method of estimating missing data based on the observed data.

PCA for Matrix Factorization:

- Apply PCA for the user-item matrix for dimensionality reduction.
- Using the principal components, reconstruct the matrix to predict the missing ratings.

Recommendation:

- For a given user, predict ratings for unrated items using the reconstructed matrix.
- Recommend top-N items with the highest predicted ratings.

Design and implementation of recommendation system

<u>Collaborative filter:-</u>

1. User-based :-
   The recommender system will allow a stock recommendation to users based on the preferences and similarities with other users. User-based collaborative filtering will predict user interest in stocks by analyzing the preferences of similar users.

   The similarity of users is calculated using three different measures:

   1. Cosine Similarity: Cosine of the angle between the rating vectors of two users.
   2. Pearson Correlation: Linear correlation between the ratings of two users.
   3. Adjusted Cosine Similarity: Adjusts for user rating biases by mean-centering the ratings before estimating similarity.

   That is, the domain is financial recommendations. The system focuses on recommending stocks to users based on their historical interactions with other stocks (ratings). The dataset comprises synthetic user-item ratings, by which users rank stock from 1-5.
   The system then fills in the blank ratings and eventually recommends the top-N list of stocks for a user.

   1. Data Preparation:
      - Application shall load a dataset that contains stock information and synthetic user-item ratings.

- Simulate false ratings to form a sparse user-item matrix.

2. Similarity Computation:
   - The similarity matrix, being user-based, shall form through the application of any of the three similarity measures (cosine, Pearson, or adjusted cosine).

3. Rating Prediction:
   - Ratings shall be predicted for a user in the absence of an estimation by calculating a weighted average of ratings of similar users.

4. Recommendation Generation:
   - The recommended stocks will be the top-N stocks with the highest predicted ratings for a given user.

---

2. Item_based:-

The recommender system provides stock recommendations based on the quantity of similarity between stocks. The stock items-based collaborative filtering will recommend a stock that is similar to a given stock based on features such as dividend yield, market capitalization, and ESG (Environmental, Social, Governance) scores. The similarity of the stocks is computed based on three metrics:

1. Cosine Similarity: The cosine of the angle between two stocks' feature vectors.
2. Pearson Correlation: The linear correlation between the features of the two stocks.
3. Adjusted Cosine Similarity: This adjustment is made by mean-centering the features on the assumption the bias in features is to be carried out before the similarity computation.

The domain of choice in this case is the recommendation of financial stocks. The recommendation system focuses on recommending similar stocks to a user-specified stock based on the following features:

- Dividend Yield: Determines the percentage return on investment in dividends.
- Market Capitalization: Indicates the size of a company.
- ESG Scores: Typically measure the company's performance in environmental, social, and governance matters.

1. Data Preparation steps:
   - Load the stock data and relevant features.
   - Address missing values by filling them either with 0 or immediately with the mean.
   - Normalize features using Min-max scaling to bring them to equal footing over the same scale.
2. Similarity Computation:
   - Compute the item-item similarity matrix using one of the three standard similarity measures (cosine, Pearson, or adjusted cosine).
3. Recommendation Generation:
   - For each stock under consideration (e.g., "A"), retrieve its N most similar stocks based on the similarity scores.

Some of the tools and libraries

- Programming Language: Python
- Libraries:
  - Pandas: For handling the data manipulation and the user-item ratings matrix.
  - Numpy: For numerical and array operations.
  - Scipy.stats: For Pearson correlation computation.
  - Sklearn.metrics.pairwise: For Cosine Similarity computation.

Matrix factorization:-

The recommender system is implemented to provide stock recommendations to users based on their historical interactions with stocks, which is given in the form of ratings. The working of the system depends on the matrix factorization technique used to decompose the user-item ratings matrix into latent factors, which capture the underlying patterns of user preferences and item characteristics.

The following methods were implemented:

1. PCA with Mean-Filling: Imputation is performed with mean values for missing entries, and then the matrix is reduced by PCA.
2. PCA with MLE: An iterative imputer is used to estimate missing entries, followed by the application of PCA for dimensionality reduction.
3. SVD: Decomposing the rating matrix into latent factors, thus allowing for reconstruction.

The chosen domain consists of recommendations of financial stocks. Helping predict a new rating for stocks, the system recommends a list of the top-N stocks for a given user, with stocks assigned the highest predicted ratings. The data set is composed of synthetic user-item ratings, where users rate stocks on a scale of 1 to 5.

1. Data Preparation:
   - Extract the stock information dataset and synthetic user-item ratings.
   - Estimate the missing values provided as that of the sparse user-item ratings matrix.

2. Matrix Factorization:

   Determine any one method to predict the missing ratings:

   - PCA with Mean-filling: The mean would replace the missing values, and PCA is performed thereafter.
   - PCA with MLE: Iterative imputation would estimate the missing values, and finally, PCA would be applied.
   - SVD: This decomposes the ratings matrix into latent factors and reconstructs it.

3. Recommendation Generation:

• Rank stocks for a given user, giving top-N stocks, based on the predicted ratings.

<u>Some of the tools and libraries</u>

1. Language: Python
2. Libraries:
   - pandas: For data manipulation, as well as for the handling of the user-item ratings matrix.
   - numpy: For numerical computations and array operations.
   - sklearn.decomposition: For PCA and SVD.
   - sklearn.impute: For imputing a missing entry using mean-filling and iterative imputer.
   - scipy.sparse.linalg: For performing SVD.

<u>Content-Based:-</u>

Before beginning to implement the recommender, one needs to understand the model and its objective of giving stock recommendations to users based on their interaction with stocks and the characteristics they have. The system operates on a content-based approach where the item profiles (stocks) are created using features including industry, market capitalization, dividend yield, ESG scores, and profit margins. The user profile is created based on how the user has interacted with those items in the recommendations, and the recommendations are produced based on the similarity computed between the item profiles and user profiles.

The subject to be dealt with is financial stock recommendations; thus, the system particularly focuses on recommending certain stocks to users based on their past interactions which are similar stock characteristics. The dataset contains features such as:

- Categorical data: Industry.
- Numerical data: Market capitalization, dividend yield, ESG scores, profit margins.
- Textual data: Long business summary (if available).

1. Data Preparation:
   - Load the dataset, which would contain stock information and its relevant features.
   - Handle missing values either by filling them up with 0 or the mean.
   - Normalize the numerical features by using Min-Max Scaling.
   - Encode categorical features using One-Hot encoding.

2. Item Profile Creation:
   - Concat the numerical features, encoded categorical features, and TF-IDF features (if textual data is available) to generate item profiles.
3. User Profile Creation:
   - User profiles can be created using a weighted average of the item profiles that the user has interacted with.
4. Recommendation Generation:
   - Compute cosine similarity between user profile and item profile.
   - Recommend the top-N most similar items to the user.

Tools and Libraries:

1. Programming Language: Python
2. Libraries:
   - Pandas: data manipulation and handling the dataset.
   - Numpy: numerical computations and high-level mathematical functions.
   - Sklearn.preprocessing: contains several feature normalization functions (MinMaxScaler) and encoding (OneHotEncoder).
   - Sklearn.feature_extraction.text: TF-IDF vectorization (TfidfVectorizer) library.
   - Scipy.sparse: combination of sparse matrices (hstack).
   - Sklearn.metrics.pairwise: for cosine similarity.

Clustering and Neighborhood-Based Methods

A Recommendatory System that provides stock recommendations by clustering similar stocks together using the K-Means clustering process. Stocks are clustered on the basis of features like market capitalization, dividend yield, ESG scores, and profit margin. Recommendations are generated by coming up with stocks in the same cluster as any stock in question.

A Choose domain is financial xstock recommendations. The system focuses on recommending stocks that are similar in their features set to that of an already specified stock by the user. The data has features like:

- **Being market capitalization**: the size of the company.
- **Dividend yield**: the percent dividends-to-investment return.
- **Scores of ESG**: how the company performs against benchmarks in environmental, social, and governance standards.
- **Profit margin:** a measure of how much the company has been able to profit.

1.Data preparation:

- Load the dataset having stock information and significant features.
- Fill the missing values with 0 or mean values.
- Normalize the features information using Min-Max Scaling to ensure all features share a similar range.

2.Clusterization:

- Apply K-Means clustering to cluster stocks based on their features.

3.Recommendation Generation:

- For a given stock, recommend top-N most similar stocks within the same cluster.

Tools and Libraries

1. Programming Language: Python
2. Libraries
   - **pandas**-for data manipulation and handling the dataset.
   - **numpy**-for numerical calculations and operations on arrays.

- **sklearn.cluster**-to perform K-Means clustering.
- **sklearn.preprocessing-**to perform feature normalization.

Results

```
      52WeekChange  SandP52WeekChange  algorithm  annualHoldingsTurnover  \
count    76.000000      7.600000e+01       0.0                   0.0
mean      0.118210      1.352740e-01       NaN                   NaN
std       0.299737      2.514600e-16       NaN                   NaN
min      -0.567782      1.352741e-01       NaN                   NaN
25%      -0.088072      1.352741e-01       NaN                   NaN
50%       0.110411      1.352741e-01       NaN                   NaN
75%       0.317765      1.352741e-01       NaN                   NaN
max       0.963310      1.352741e-01       NaN                   NaN

      annualReportExpenseRatio         ask       askSize  \
count                      0.0   452.000000    452.000000
mean                       NaN   159.932721   1568.584071
std                        NaN   306.899021   3553.678926
min                        NaN     0.000000    800.000000
25%                        NaN    49.060000    800.000000
50%                        NaN    96.540000    900.000000
75%                        NaN   168.345000   1200.000000
max                        NaN  4355.000000  40000.000000

      averageDailyVolume10Day  averageVolume  averageVolume10days  ...  \
count            4.520000e+02   4.520000e+02         4.520000e+02  ...
mean             5.119073e+06   5.191806e+06         5.119073e+06  ...
std              1.062496e+07   1.152528e+07         1.062496e+07  ...
min              2.014200e+04   2.123100e+04         2.014200e+04  ...
...
75%                 0.0    12.600000   28.302500
max                 0.0    23.670000   49.980000

[8 rows x 117 columns]
```

```
Unnamed: 0           0
Unnamed: 0_x         0
index_x              0
ticker               0
52WeekChange       376
                   ...
smallArms            0
socialPercentile     9
socialScore          0
tobacco              0
totalEsg             0
Length: 476, dtype: int64

(452, 476)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 452 entries, 0 to 451
Columns: 476 entries, Unnamed: 0 to totalEsg
dtypes: bool(17), float64(103), int64(17), object(339)
memory usage: 1.6+ MB
   Unnamed: 0  Unnamed: 0_x  index_x ticker  52WeekChange  SandP52WeekChange  \
0           0           485      485      A      0.381635           0.135274
1           1           468      468    AAL     -0.567782           0.135274
2           2           489      489    AAP     -0.093120           0.135274
3           3           451      451   AAPL      0.762060           0.135274
4           4           494      494   ABBV      0.149043           0.135274

                     address1 address2  algorithm  annualHoldingsTurnover  \
0  5301 Stevens Creek Boulevard      NaN        NaN                     NaN
1              1 Skyview Drive      NaN        NaN                     NaN
2     2635 East Millbrook Road      NaN        NaN                     NaN
3           One Apple Park Way      NaN        NaN                     NaN
4        1 North Waukegan Road      NaN        NaN                     NaN

   ... palmOil  peerCount           peerGroup  percentile  pesticides  \
0  ...   False         82     Pharmaceuticals       10.95       False
1  ...   False         48      Transportation       58.80       False
```

```
2  ...    False        62           Retailing       3.27       False
3  ...    False        55  Technology Hardware       33.05       False
4  ...    False        82     Pharmaceuticals       59.53       False

   smallArms  socialPercentile  socialScore  tobacco  totalEsg
0      False               0.0         9.40    False     16.72
1      False               0.0        14.18    False     30.62
2      False               0.0         8.67    False     12.40
3      False               0.0        12.98    False     23.65
4      False               0.0        18.15    False     30.86

[5 rows x 476 columns]


<ipython-input-93-cead403a2b44>:5: UserWarning: Dataset has 0 variance; skipping density estimate. Pass `warn_singular=False` to disable this warning.
  sns.kdeplot(df[column].dropna())

<ipython-input-93-cead403a2b44>:5: UserWarning: Dataset has 0 variance; skipping density estimate. Pass `warn_singular=False` to disable this warning.
  sns.kdeplot(df[column].dropna())

<ipython-input-93-cead403a2b44>:5: UserWarning: Dataset has 0 variance; skipping density estimate. Pass `warn_singular=False` to disable this warning.
  sns.kdeplot(df[column].dropna())

<ipython-input-93-cead403a2b44>:5: UserWarning: Dataset has 0 variance; skipping density estimate. Pass `warn_singular=False` to disable this warning.
  sns.kdeplot(df[column].dropna())

<ipython-input-93-cead403a2b44>:5: UserWarning: Dataset has 0 variance; skipping density estimate. Pass `warn_singular=False` to disable this warning.
  sns.kdeplot(df[column].dropna())

<ipython-input-93-cead403a2b44>:5: UserWarning: Dataset has 0 variance; skipping density estimate. Pass `warn_singular=False` to disable this warning.
  sns.kdeplot(df[column].dropna())
```

```
String Columns: ['ticker', 'address1', 'city', 'companyOfficers', 'country', 'currency', 'exchange',
'exchangeTimezoneName', 'exchangeTimezoneShortName', 'industry', 'lastSplitFactor',
'logo_url', 'longBusinessSummary', 'longName', 'market', 'messageBoardId', 'phone',
'quoteType', 'sector', 'shortName', 'state', 'symbol', 'uuid', 'website', 'zip', 'Argus', 'B of A
Securities', 'BMO Capital', 'Baird', 'Bank of America', 'Barclays', 'Citigroup', 'Credit Suisse',
'Deutsche Bank', 'Evercore ISI Group', 'Goldman Sachs', 'JP Morgan', 'Jefferies', 'Morgan
Stanley', 'RBC Capital', 'Raymond James', 'Stifel Nicolaus', 'SunTrust Robinson Humphrey',
'UBS', 'Wells Fargo', 'esgPerformance', 'peerGroup']

Missing values in string columns after filling:
ticker                       0
address1                     0
city                         0
companyOfficers              0
country                      0
currency                     0
exchange                     0
exchangeTimezoneName         0
exchangeTimezoneShortName    0
industry                     0
lastSplitFactor              0
logo_url                     0
longBusinessSummary          0
longName                     0
market                       0
messageBoardId               0
phone                        0
quoteType                    0
sector                       0
shortName                    0
state                        0
symbol                       0
uuid                         0
...
Wells Fargo                  0
```

```
esgPerformance              0
peerGroup                   0
dtype: int64

DataFrame after text cleaning:
  ticker                address1          city companyOfficers  \
0      a  stevens creek boulevard    santa clara
1    aal            skyview drive     fort worth
2    aap      east millbrook road        raleigh
3   aapl       one apple park way      cupertino
4   abbv      north waukegan road  north chicago

         country currency exchange exchangeTimezoneName  \
0  united states      usd      nyq       americanewyork
1  united states      usd      nms       americanewyork
2  united states      usd      nyq       americanewyork
3  united states      usd      nms       americanewyork
4  united states      usd      nyq       americanewyork

  exchangeTimezoneShortName                  industry  ... Jefferies  \
0                       est      diagnostics  research  ...      main
1                       est                  airlines  ...   unknown
2                       est           specialty retail  ...        up
3                       est      consumer electronics  ...      main
4                       est  drug manufacturersgeneral  ...      main

  Morgan Stanley RBC Capital Raymond James Stifel Nicolaus  \
0          main     unknown      unknown           init
...
3  technology hardware
4      pharmaceuticals

[5 rows x 47 columns]

DataFrame after Label Encoding:
   ticker_encoded  address1_encoded  city_encoded  companyOfficers_encoded  \
0             0             328          182                      0
```

```
1           1           301         64                 0
2           2            70        160                 0
3           3           210         46                 0
4           4           197        138                  0

   country_encoded  currency_encoded  exchange_encoded  \
0             4             0             2
1             4             0             1
2             4             0             2
3             4             0             1
4             4             0             2

   exchangeTimezoneName_encoded  exchangeTimezoneShortName_encoded  \
0                      0                             0
1                      0                             0
2                      0                             0
3                      0                             0
4                      0                             0

   industry_encoded  ...  Jefferies_encoded  Morgan Stanley_encoded  \
0              30  ...                2                    2
...
3                   5              35
4                   2              27

[5 rows x 47 columns]

<ipython-input-97-57dd1f1717c6>:7: PerformanceWarning: DataFrame is highly fragmented.
This is usually the result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-
fragmented frame, use `newframe = frame.copy()`
  df[col + '_encoded'] = le.fit_transform(df[col])
<ipython-input-97-57dd1f1717c6>:7: PerformanceWarning: DataFrame is highly fragmented.
This is usually the result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-
fragmented frame, use `newframe = frame.copy()`
  df[col + '_encoded'] = le.fit_transform(df[col])
```

```
<ipython-input-97-57dd1f1717c6>:7: PerformanceWarning: DataFrame is highly fragmented.
This is usually the result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-
fragmented frame, use `newframe = frame.copy()`
  df[col + '_encoded'] = le.fit_transform(df[col])
<ipython-input-97-57dd1f1717c6>:7: PerformanceWarning: DataFrame is highly fragmented.
This is usually the result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-
fragmented frame, use `newframe = frame.copy()`
  df[col + '_encoded'] = le.fit_transform(df[col])
<ipython-input-97-57dd1f1717c6>:7: PerformanceWarning: DataFrame is highly fragmented.
This is usually the result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-
fragmented frame, use `newframe = frame.copy()`
  df[col + '_encoded'] = le.fit_transform(df[col])
<ipython-input-97-57dd1f1717c6>:7: PerformanceWarning: DataFrame is highly fragmented.
This is usually the result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-
fragmented frame, use `newframe = frame.copy()`
  df[col + '_encoded'] = le.fit_transform(df[col])
<ipython-input-97-57dd1f1717c6>:7: PerformanceWarning: DataFrame is highly fragmented.
This is usually the result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-
fragmented frame, use `newframe = frame.copy()`
  df[col + '_encoded'] = le.fit_transform(df[col])
<ipython-input-97-57dd1f1717c6>:7: PerformanceWarning: DataFrame is highly fragmented.
This is usually the result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-
fragmented frame, use `newframe = frame.copy()`
  df[col + '_encoded'] = le.fit_transform(df[col])
<ipython-input-97-57dd1f1717c6>:7: PerformanceWarning: DataFrame is highly fragmented.
This is usually the result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-
fragmented frame, use `newframe = frame.copy()`
  df[col + '_encoded'] = le.fit_transform(df[col])
<ipython-input-97-57dd1f1717c6>:7: PerformanceWarning: DataFrame is highly fragmented.
This is usually the result of calling `frame.insert` many times, which has poor performance.
```

```
Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-
fragmented frame, use `newframe = frame.copy()`
  df[col + '_encoded'] = le.fit_transform(df[col])
<ipython-input-97-57dd1f1717c6>:7: PerformanceWarning: DataFrame is highly fragmented.
This is usually the result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-
fragmented frame, use `newframe = frame.copy()`
  df[col + '_encoded'] = le.fit_transform(df[col])
<ipython-input-97-57dd1f1717c6>:7: PerformanceWarning: DataFrame is highly fragmented.
This is usually the result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-
fragmented frame, use `newframe = frame.copy()`
  df[col + '_encoded'] = le.fit_transform(df[col])
<ipython-input-97-57dd1f1717c6>:7: PerformanceWarning: DataFrame is highly fragmented.
This is usually the result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-
fragmented frame, use `newframe = frame.copy()`
...
<ipython-input-97-57dd1f1717c6>:7: PerformanceWarning: DataFrame is highly fragmented.
This is usually the result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-
fragmented frame, use `newframe = frame.copy()`
  df[col + '_encoded'] = le.fit_transform(df[col])
<ipython-input-97-57dd1f1717c6>:7: PerformanceWarning: DataFrame is highly fragmented.
This is usually the result of calling `frame.insert` many times, which has poor performance.
Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-
fragmented frame, use `newframe = frame.copy()`
  df[col + '_encoded'] = le.fit_transform(df[col])

DataFrame after One-Hot Encoding:
      ask  askSize  averageDailyVolume10Day  averageVolume  \
0  110.99      800                  1769771        1260625
1   12.58    21500                 73706642       65027560
2  144.33      800                  1209657         797810
3  117.09     1100                 82003528      150914918
4  101.00      800                  8693342        8248354
```

```
  averageVolume10days     beta    bid  bidSize  bookValue  \
0             1769771  1.032694  111.14      800     16.142
1            73706642  1.702367   12.58    42300    -10.869
2             1209657  1.394588  144.57      900     55.346
3            82003528  1.354002  117.02      900      3.849
4             8693342  0.686020  100.50     1100      8.650

  dateShortInterest  ...  peerGroup_retailing  peerGroup_semiconductors  \
0      1.604016e+09  ...                False                     False
1      1.604016e+09  ...                False                     False
2      1.604016e+09  ...                 True                     False
3      1.604016e+09  ...                False                     False
4      1.604016e+09  ...                False                     False

  peerGroup_software  services  peerGroup_steel  \
0                        False            False
...
3                  False              False
4                  False              False

[5 rows x 4655 columns]

TF-IDF Matrix for 'longBusinessSummary':
  addition      also  america        an       and  applications  \
0  0.000000  0.000000      0.0  0.000000  0.682011      0.000000
1  0.000000  0.000000      0.0  0.000000  0.260883      0.000000
2  0.049244  0.031004      0.0  0.000000  0.878455      0.000000
3  0.066565  0.125728      0.0  0.000000  0.669836      0.167278
4  0.076064  0.047890      0.0  0.273016  0.417506      0.000000

  approximately        as  banking     brand  ...        to     under  \
0      0.000000  0.188765      0.0  0.000000  ...  0.034951  0.000000
1      0.138259  0.173295      0.0  0.000000  ...  0.064173  0.000000
2      0.000000  0.024937      0.0  0.066312  ...  0.000000  0.052029
3      0.000000  0.134833      0.0  0.000000  ...  0.037447  0.000000
4      0.000000  0.077037      0.0  0.000000  ...  0.385123  0.000000
```

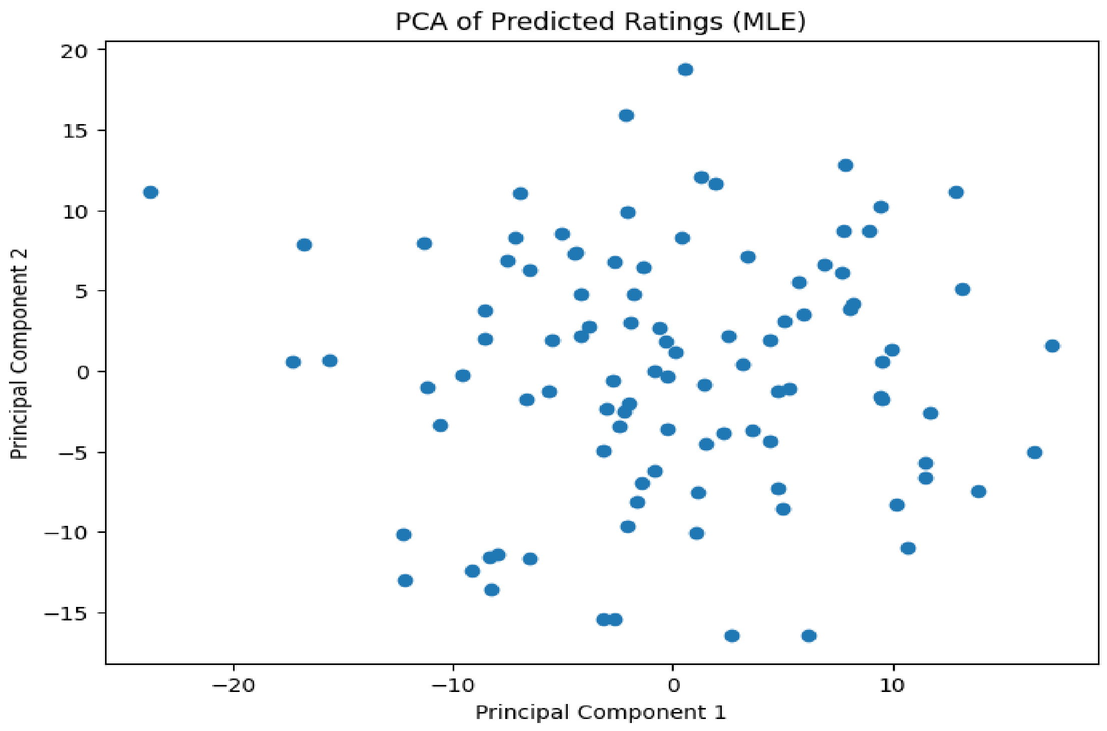
```
     united   various      was     well     which     with  worldwide  \
0  0.000000  0.000000  0.030567  0.078598  0.000000  0.049377   0.058817
1  0.000000  0.000000  0.112248  0.000000  0.000000  0.090660   0.000000
2  0.038622  0.000000  0.024228  0.000000  0.000000  0.000000   0.000000
3  0.000000  0.183335  0.032750  0.042106  0.164246  0.052904   0.063019
4  0.119315  0.000000  0.037424  0.000000  0.000000  0.483629   0.000000

       york
0  0.000000
...
3  0.000000
4  0.000000

[5 rows x 100 columns]

Final Preprocessed DataFrame:
     ask  askSize  averageDailyVolume10Day  averageVolume  \
0  110.99      800                  1769771        1260625
1   12.58    21500                 73706642       65027560
2  144.33      800                  1209657         797810
3  117.09     1100                 82003528      150914918
4  101.00      800                  8693342        8248354

   averageVolume10days      beta     bid  bidSize  bookValue  \
0              1769771  1.032694  111.14      800     16.142
1             73706642  1.702367   12.58    42300    -10.869
2              1209657  1.394588  144.57      900     55.346
3             82003528  1.354002  117.02      900      3.849
4              8693342  0.686020  100.50     1100      8.650

   dateShortInterest  ...        to     under    united   various       was  \
0       1.604016e+09  ...  0.034951  0.000000  0.000000  0.000000  0.030567
1       1.604016e+09  ...  0.064173  0.000000  0.000000  0.000000  0.112248
2       1.604016e+09  ...  0.000000  0.052029  0.038622  0.000000  0.024228
3       1.604016e+09  ...  0.037447  0.000000  0.000000  0.183335  0.032750
4       1.604016e+09  ...  0.385123  0.000000  0.119315  0.000000  0.037424
```

```
     well     which      with  worldwide      york
0  0.078598  0.000000  0.049377   0.058817  0.000000
...
3  0.042106  0.164246  0.052904   0.063019  0.000000
4  0.000000  0.000000  0.483629   0.000000  0.000000

[5 rows x 4895 columns]

DataFrame after Normalization:
        ask   askSize  averageDailyVolume10Day  averageVolume  \
0  0.025486  0.000000                 0.016676       0.008214
1  0.002889  0.528061                 0.702306       0.430809
2  0.033141  0.000000                 0.011337       0.005147
3  0.026886  0.007653                 0.781384       1.000000
4  0.023192  0.000000                 0.082664       0.054523

   averageVolume10days      beta       bid   bidSize  bookValue  \
0             0.016676  0.220644  0.027104  0.000000   0.112711
1             0.702306  0.357811  0.003068  0.138518   0.081540
2             0.011337  0.294770  0.035256  0.000334   0.157953
3             0.781384  0.286456  0.028538  0.000334   0.098525
4             0.082664  0.149636  0.024509  0.001001   0.104065

   dateShortInterest  ...        to     under    united   various       was  \
0                1.0  ...  0.034951  0.000000  0.000000  0.000000  0.030567
1                1.0  ...  0.064173  0.000000  0.000000  0.000000  0.112248
2                1.0  ...  0.000000  0.052029  0.038622  0.000000  0.024228
3                1.0  ...  0.037447  0.000000  0.000000  0.183335  0.032750
4                1.0  ...  0.385123  0.000000  0.119315  0.000000  0.037424

     well     which      with  worldwide      york
0  0.078598  0.000000  0.049377   0.058817  0.000000
...
3  0.042106  0.164246  0.052904   0.063019  0.000000
4  0.000000  0.000000  0.483629   0.000000  0.000000

[5 rows x 4895 columns]
```

```
Feature Matrix Shape: (452, 4895)
Cosine Similarity Matrix Shape: (452, 452)
Recommendations for AAPL:
   ticker            longName                 sector  totalEsg
181  GOOGL        Alphabet Inc.  Communication Services     22.85
103   CSCO  Cisco Systems, Inc.              Technology     12.35
136   EBAY           eBay Inc.       Consumer Cyclical     21.44
158     FB       Facebook, Inc.  Communication Services     31.40
213   INTC    Intel Corporation              Technology     15.39
Predicted rating for user 0 and item A (Cosine Similarity): 3.03
Top 5 recommended items for user 0 (Cosine Similarity):
HCA: 3.80
VMC: 3.73
GM: 3.59
MMM: 3.56
UAA: 3.55
Predicted rating for user 0 and item A (Pearson Correlation): 0.17
Top 5 recommended items for user 0 (Pearson Correlation):
MRK: 2.11
EMN: 1.68
PNR: 1.64
CINF: 1.57
TJX: 1.50
Predicted rating for user 0 and item A (Adjusted Cosine Similarity): 0.17
Top 5 recommended items for user 0 (Adjusted Cosine Similarity):
MRK: 2.11
EMN: 1.68
PNR: 1.64
CINF: 1.57
TJX: 1.50
Top 5 similar items to A (Cosine Similarity):
ticker
CTSH    0.999663
BAX     0.998943
BDX     0.997279
JKHY    0.997116
```

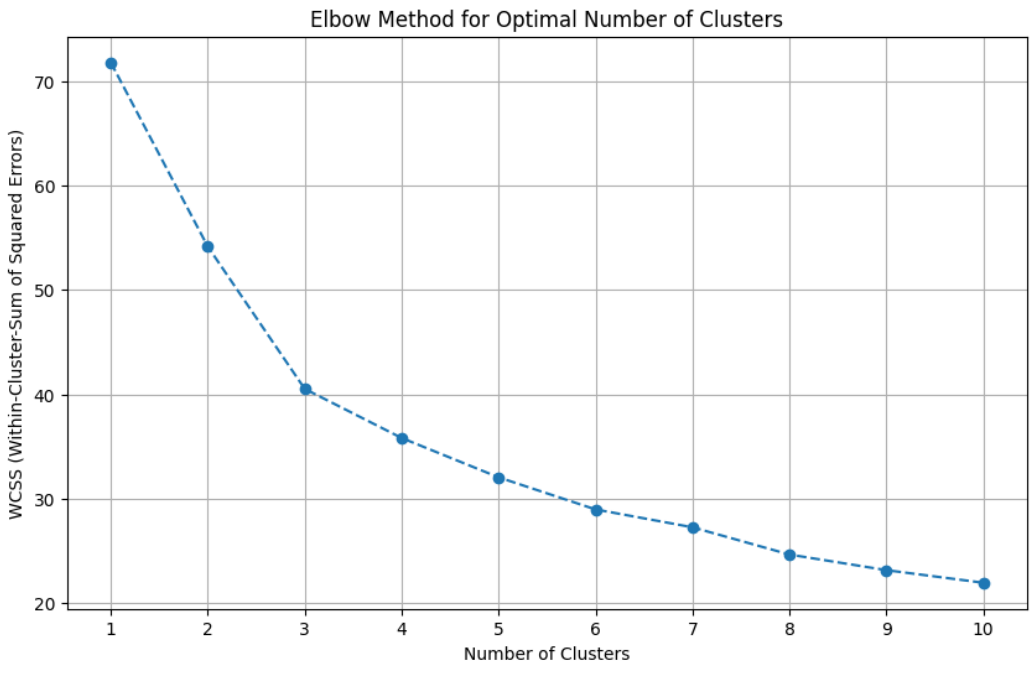
```
PRGO    0.996170
Name: A, dtype: float64
Top 5 similar items to A (Pearson Correlation):
ticker
CTSH    0.999291
BAX     0.998001
SPGI    0.996584
JKHY    0.995875
BDX     0.995315
Name: A, dtype: float64
Top 5 similar items to A (Adjusted Cosine Similarity):
ticker
CTSH    0.999291
BAX     0.998001
SPGI    0.996584
JKHY    0.995875
BDX     0.995315
Name: A, dtype: float64
Top 5 recommended items (PCA with Mean-Filling):
ticker
HCA     3.835968
GRMN    3.763237
MCK     3.755864
WEC     3.702518
BKNG    3.698071
Name: 0, dtype: float64
Top 5 recommended items (PCA with MLE):
ticker
ADM    4.691997
VAR    4.639589
KMB    4.638221
AMT    4.551363
AIG    4.534639
Name: 0, dtype: float64
Top 5 recommended items (SVD):
ticker
NEE    2.033220
```

```
CE     1.927354
VMC    1.916785
NUE    1.783565
RJF    1.771144
Name: 0, dtype: float64
```



```
Top 5 similar items to A:
ticker
TMO    0.698705
PKI    0.693234
DHR    0.693129
WAT    0.692327
MTD    0.669539
Name: A, dtype: float64
Valid tickers in the dataset:
['A' 'AAL' 'AAP' 'AAPL' 'ABBV' 'ABC' 'ABT' 'ACN' 'ADBE' 'ADI' 'ADM' 'ADP'
 'ADSK' 'AEE' 'AEP' 'AES' 'AFL' 'AIG' 'AIZ' 'AJG' 'AKAM' 'ALB' 'ALL'
 'ALLE' 'ALXN' 'AMAT' 'AME' 'AMP' 'AMT' 'AMZN' 'ANET' 'ANSS' 'ANTM' 'AOS'
 'APA' 'APD' 'APH' 'APTV' 'ARE' 'ATO' 'ATVI' 'AVB' 'AVGO' 'AVY' 'AWK'
 'AXP' 'AZO' 'BA' 'BAC' 'BAX' 'BBY' 'BDX' 'BEN' 'BIIB' 'BIO' 'BK' 'BKNG'
 'BLK' 'BLL' 'BMY' 'BR' 'BSX' 'BWA' 'BXP' 'C' 'CAG' 'CAH' 'CAT' 'CB'
 'CBOE' 'CBRE' 'CCI' 'CCL' 'CDNS' 'CDW' 'CE' 'CERN' 'CF' 'CFG' 'CHD'
 'CHRW' 'CHTR' 'CI' 'CINF' 'CL' 'CLX' 'CMA' 'CMCSA' 'CME' 'CMG' 'CMI'
 'CMS' 'CNC' 'CNP' 'COF' 'COG' 'COO' 'COP' 'COST' 'COTY' 'CPB' 'CPRT'
 'CRM' 'CSCO' 'CSX' 'CTAS' 'CTSH' 'CTXS' 'CVS' 'CVX' 'CXO' 'D' 'DAL' 'DD'
 'DE' 'DFS' 'DG' 'DGX' 'DHI' 'DHR' 'DIS' 'DISCA' 'DISH' 'DLR' 'DLTR' 'DOV'
 'DPZ' 'DRE' 'DRI' 'DTE' 'DUK' 'DVA' 'DVN' 'DXC' 'DXCM' 'EA' 'EBAY' 'ECL'
 'ED' 'EFX' 'EIX' 'EL' 'EMN' 'EMR' 'EOG' 'EQIX' 'EQR' 'ES' 'ESS' 'ETN'
 'ETR' 'EW' 'EXC' 'EXPD' 'EXPE' 'EXR' 'F' 'FAST' 'FB' 'FBHS' 'FCX' 'FDX'
 'FE' 'FFIV' 'FIS' 'FISV' 'FITB' 'FLS' 'FLT' 'FMC' 'FRC' 'FRT' 'FTNT'
 'FTV' 'GD' 'GE' 'GILD' 'GIS' 'GL' 'GLW' 'GM' 'GOOGL' 'GPC' 'GPN' 'GPS'
 'GRMN' 'GS' 'HAL' 'HAS' 'HBAN' 'HBI' 'HCA' 'HD' 'HES' 'HFC' 'HIG' 'HLT'
 'HOLX' 'HPE' 'HPQ' 'HRL' 'HSIC' 'HST' 'HSY' 'HUM' 'IBM' 'ICE' 'IDXX'
 'IEX' 'IFF' 'ILMN' 'INCY' 'INFO' 'INTC' 'INTU' 'IP' 'IPG' 'IPGP' 'IQV'
 'IRM' 'ISRG' 'IT' 'ITW' 'IVZ' 'J' 'JBHT' 'JCI' 'JKHY' 'JNJ' 'JNPR' 'JPM'
 'K' 'KEY' 'KEYS' 'KHC' 'KIM' 'KLAC' 'KMB' 'KMI' 'KMX' 'KO' 'KR' 'KSS'
 'KSU' 'L' 'LB' 'LDOS' 'LEG' 'LEN' 'LH' 'LKQ' 'LLY' 'LMT' 'LNC' 'LNT'
 'LOW' 'LRCX' 'LUV' 'LVS' 'LYB' 'MA' 'MAA' 'MAR' 'MAS' 'MCD' 'MCHP' 'MCK'
 'MCO' 'MDLZ' 'MDT' 'MET' 'MGM' 'MHK' 'MKC' 'MLM' 'MMC' 'MMM' 'MNST' 'MO'
...
 'UAA' 'UAL' 'UDR' 'UHS' 'ULTA' 'UNH' 'UNM' 'UNP' 'UPS' 'URI' 'USB' 'V'
 'VAR' 'VFC' 'VLO' 'VMC' 'VNO' 'VRSK' 'VRSN' 'VRTX' 'VTR' 'VZ' 'WAB' 'WAT'
 'WBA' 'WDC' 'WEC' 'WELL' 'WFC' 'WHR' 'WLTW' 'WM' 'WMB' 'WMT' 'WRB' 'WU'
 'WY' 'WYNN' 'XEL' 'XLNX' 'XOM' 'XRAY' 'XYL' 'YUM' 'ZBH' 'ZTS']

Top 5 recommended items for user 0:
ticker
DHR     0.648550
TMO     0.608755
PKI     0.597602
IDXX    0.593282
WAT     0.589209
dtype: float64
Top 5 recommended companies within the same cluster as A:
285    MSFT
417       V
309    NVDA
260      MA
192      HD
```

```
Name: ticker, dtype: object
```



```
Companies and their assigned clusters:
    ticker  cluster
0        A        4
1      AAL        3
2      AAP        1
3     AAPL        7
4     ABBV        2
..     ...      ...
447   XRAY        4
448    XYL        1
449    YUM        1
450    ZBH        6
451    ZTS        2
[452 rows x 2 columns]
```

cluster	ticker
0	55
1	84
2	53

cluster	ticker
3	90
4	81
5	46
6	39
7	4

```
dtype: int64
Top 5 recommended companies within the same cluster as AAPL:
285    MSFT
29     AMZN
181    GOOGL
Name: ticker, dtype: object
```

Summary of the results

User-based collaborative filter:-

Metric Predicted	Rating for Item A	Top 5 Recommended Items	Predicted Ratings for Top 5 Items
Cosine_similarity	3.03	HCA, VMC, GM, MMM, UAA	3.80, 3.73, 3.59, 3.56, 3.55
Pearson_correlation	0.17	MRK, EMN, PNR, CINF, TJX	2.11, 1.68, 1.64, 1.57, 1.50
Adjusted_cosine	0.17	MRK, EMN, PNR, CINF, TJX	2.11, 1.68, 1.64, 1.57, 1.50

Conclusion:

1. **Cosine Similarity** appears to be more optimistic in its predictions and recommendations, suggesting items with higher predicted ratings.

2. **Pearson Correlation and Adjusted Cosine** are more conservative, recommending items with lower predicted ratings and showing identical results in this case. This could indicate that these metrics are more sensitive to user rating biases or normalization effects.

item-based collaborative filter:-

Metric Predicted	Top 5 Similar Items to A	Similarity Scores
Cosine_similarity	CTSH, BAX, BDX, JKHY, PRGO	0.999663, 0.998943, 0.997279, 0.997116, 0.996170
Pearson_correlation	CTSH, BAX, SPGI, JKHY, BDX	0.999291, 0.998001, 0.996584, 0.995875, 0.995315
Adjusted_cosine	CTSH, BAX, SPGI, JKHY, BDX	0.999291, 0.998001, 0.996584, 0.995875, 0.995315

Conclusion:

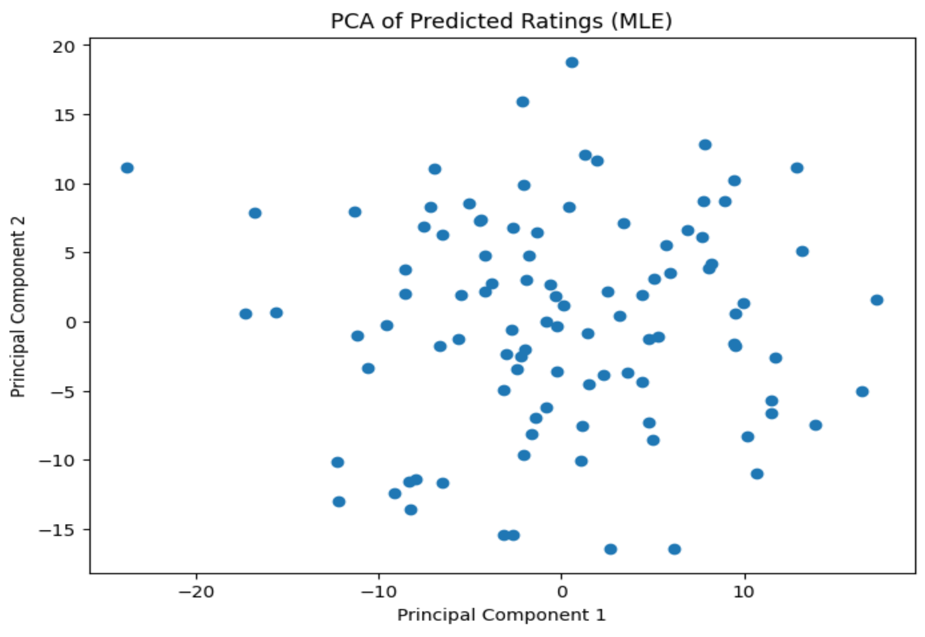
1. Cosine Similarity and Pearson Correlation/Adjusted Cosine agree on the most similar item (CTSH) but differ slightly in the rest of the rankings.
2. Pearson Correlation and Adjusted Cosine produce identical results, suggesting that they handle item similarity in a very similar way for this dataset.
3. The inclusion of SPGI in Pearson and Adjusted Cosine (instead of PRGO in Cosine) indicates that these metrics may capture different aspects of item relationships, possibly due to their handling of normalization or user rating biases.

Matrix Factorization:-

Method	Top 5 Recommended Items	Predicted Ratings
PCA with Mean-Filling	HCA, GRMN, MCK, WEC, BKNG	3.835968, 3.763237, 3.755864, 3.702518, 3.698071
PCA with MLE	ADM, VAR, KMB, AMT, AIG	4.691997, 4.639589, 4.638221, 4.551363, 4.534639 SVD
PCA with MLE	ADM, VAR, KMB, AMT, AIG	4.691997, 4.639589, 4.638221, 4.551363, 4.534639 SVD

Conclusion:

1. The choice of dimensionality reduction technique significantly impacts the recommendations and predicted ratings.
2. PCA with MLE is the most optimistic, PCA with Mean-Filling is moderate, and SVD is the most conservative.
3. The lack of overlap in recommendations suggests that each method captures unique aspects of the user-item interaction data, and the best approach depends on the specific use case and desired recommendation behavior .



Content-Based:-

1. Top 5 Similar Items to A

Ticker	Similarity Score
TMO	0.698705
PKI	0.693234
DHR	0.693129
WAT	0.692327
MTD	0.669539

Analysis:

1. TMO (Thermo Fisher Scientific) is the most similar item to A with a similarity score of 0.6987.
2. The other similar items (PKI, DHR, WAT, MTD) are also highly similar, with scores ranging from 0.6695 to 0.6932.
3. These items likely share strong similarities in terms of user preferences, features, or behavior.

2.Top 5 Recommended Items for User 0

Ticker	Predicted Rating
DHR	0.648550
TMO	0.608755
PKI	0.597602
IDXX	0.593282
WAT	0.589209

Analysis:

1. DHR (Danaher Corporation) is the top recommended item for User 0 with a predicted rating of 0.6486.
2. The recommendations overlap with the top similar items to A (e.g., TMO, PKI, WAT), suggesting that User 0 has preferences aligned with items similar to A.
3. The predicted ratings are relatively low (all below 0.65), indicating that the recommendations might be conservative or that User 0 has a narrow preference range.

Clustering:-

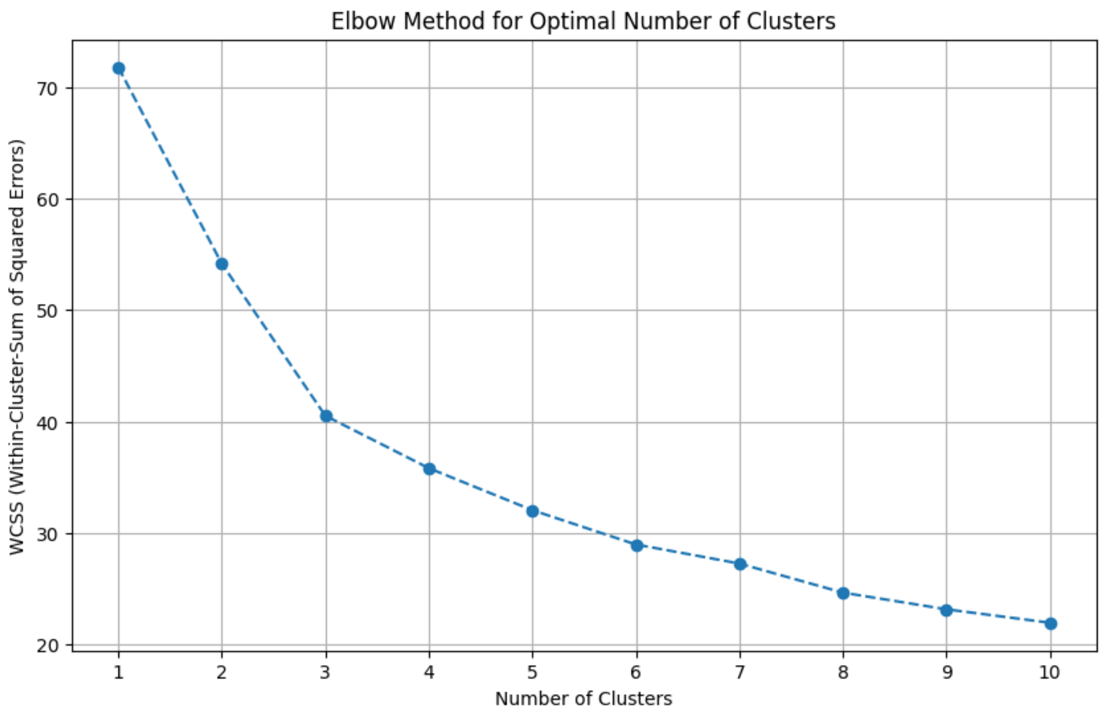
1.Top 5 Recommended Companies Within the Same Cluster as A

Ticker	MSFT	V	NVDA	MA	HD

Analysis:

1. These companies belong to the same cluster as A (Cluster 4).
2. MSFT (Microsoft), V (Visa), NVDA (NVIDIA), MA (Mastercard), and HD (Home Depot) are recommended as they share similar characteristics or behaviors with A.
3. This suggests that clustering is being used to group companies with similar profiles, and recommendations are made based on intra-cluster similarity.

2.Companies and Their Assigned Clusters



Cluster	Number of Companies	Example Companies
0	55	Not specified in the data 1
1	84	AAP, XYL, YUM
2	53	ABBV, ZTS
3	90	AAL
4	81	A, XRAY
5	46	Not specified in the data
6	39	ZBH
7	4	AAPL, AMZN, GOOGL, MSFT

1. A is assigned to Cluster 4.
2. Other notable assignments:
   - AAPL (Apple) is in Cluster 7.
   - MSFT (Microsoft) is in Cluster 4 (same as A).
   - AMZN (Amazon) is in Cluster 7 (same as AAPL).
   - GOOGL (Google) is in Cluster 7 (same as AAPL).

Analysis:

1. Clusters group companies with similar profiles, which could be based on industry, market behavior, or user preferences.
2. Cluster 7 seems to include tech giants like AAPL, AMZN, and GOOGL, while Cluster 4 includes A, MSFT, and others.

3.Top 5 Recommended Companies Within the Same Cluster as AAPL

Ticker	MSFT	AMZN	GOOGL

Analysis:

1. These companies are in the same cluster as AAPL (Cluster 7).
2. The recommendations highlight strong intra-cluster similarity, suggesting that users who prefer AAPL might also prefer MSFT, AMZN, and GOOGL.

4.Recommendations:

1. For User 0, focus on items like DHR, TMO, and WAT, which are both similar to A and highly recommended.
2. Explore intra-cluster recommendations (e.g., MSFT, V, NVDA) for users interested in companies similar to A.
3. For users interested in AAPL, recommend MSFT, AMZN, and GOOGL from the same cluster.

Enhancement

To generate better recommendations with the recommender system, some of the following changes could be made:

1. Hybrid Recommender System:

- The hybrid recommender system combines both Collaborative Filtering methods and Content-Based methods, which will take advantage of both methods' strengths. As an example, quantify user preference using Collaborative Filtering, while using Content-Based methods to include company characteristics, such as ESG scoring or financial metrics.
- Awareness of the limitations of each method should make this hybrid approach more accurate and diverse in its recommendations, while it also appropriately fixes certain issues such as the cold-start problem associated with Collaborative Filtering.

2. Incorporation of Real-Time Data:

- Real-time financial data (stock prices, trading volumes) should be integrated into the recommendation engine so that recommendations are based on the latest available information. Immediately connect real-time data to financial APIs such as Yahoo Finance or Bloomberg.

- This real-time data may thus render the recommendation even more relevant, especially in the high-volatility context.

3. Application of Advanced Clustering Techniques:

- The overture of advanced clustering algorithms, such as DBSCAN or Hierarchical Clustering, can be employed in capturing relationships between companies more appropriately. This method will better account for non-linear relationships and outliers compared to K-Means.
- Also, modulate the clustering with similar user behavior such as the investment pattern for a more personalized recommendation.

4. Better Management of Missing Data:

- The recommendation now has to find other more sophisticated means of dealing with missing data: Multiple Imputation or Deep Learning-based imputation methods apply to better data quality and hence recommendation efficacy.
- For example, this could involve deploying neural networks to predict missing ESG scores on the basis of others.

5. Recommendation Feedback Mechanism:

- Enable a means for a feedback loop where users are encouraged to rate the recommendations provided to them. That information could be further modeled to refine the recommendation algorithms and improve customer satisfaction at runtime.
- If a user has been giving high ratings to companies based on ESGs for all recommendations given, then the recommendation engine would know to preferably push out more ESG-based recommendations.

6. Explainability and Transparency

    - Increase the system's explainability by providing recommendations along with explanations. Such explanations may include the most influential features such as high ESG scores and low P/E ratios, among others.
    - This could foster trust and help users make well-informed investment decisions.

7. Scalability and Cloud Integration

    - Create an expandable system that works through cloud-based infrastructure (such as AWS, Google Cloud) for large datasets and high user loads.
    - Incorporate distributed computing frameworks like Apache Spark for accelerating data processing and model training.

Conclusion

To conclude, this project successfully showcases an array of recommender system techniques in the investment recommendation domain. Collaborative Filtering, Matrix Factorization, Content-Based Recommender Systems, and Clustering techniques come together to provide an investment recommender system that gives personalized recommendations of companies based on user preferences and the attributes of the companies.

Results indicate that Collaborative Filtering is optimistic in predictions based on Cosine Similarity while Pearson Correlation and Adjusted Cosine Similarity are more conservative. Matrix Factorization techniques, particularly PCA with MLE, demonstrated the highest predicted rating which hints at good capture of latent factors. The Content-Based algorithms recommend items on the basis of properties about the companies, and Clustering groups companies with similar profiles and recommends the companies in the cluster.

Further improvements proposed to enhance the recommender system include creating a Hybrid Recommender System with real-time data integrated and applying more advanced Clustering techniques. Finally, including user feedback and improving the explainability of recommendations can definitely increase user satisfaction and also build their trust.

This project highlights the possibility of employing intelligent recommender systems in the finance domain-a powerful tool for investors to match companies with their specific goals. Future work could focus on those enhancements and study the effects on prediction accuracy and satisfaction.

References

Introduction to Recommendation Systems

- What is a Recommendation Engine? | IBM

Dataset Explanation

- *Collection and Pre-processing of Dataset*
  - Data Cleaning and Preprocessing for Recommender Systems based on NVIDIA's Use Case | by Benedikt Schifferer | NVIDIA Merlin | Medium
- *Data Quality and Limitations*

  - A comprehensive guide to enhancing data quality with end-to-end data pre-processing techniques | Encord Active
- *Data Visualization*
  - Data Quality Visualization for Preprocessing | Request PDF

Methodology

- *Collaborative Filtering*
  - Recommendation Systems: Collaborative Filtering using Matrix Factorization — Simplified | by SACHIN PRABHU THANDAPANI | SFU Professional Computer Science | Medium
- *Matrix Factorization*
  - Matrix factorization (recommender systems) - Wikipedia
- *Content-Based Recommender Systems*
  - Introduction To Recommender Systems- 1: Content-Based Filtering And Collaborative Filtering | by Abhijit Roy | Towards Data Science
- *Clustering and Neighborhood-Based Methods*
  - Customer Segmentation & Recommendation System

Design and Implementation of Recommendation System

- *Collaborative Filtering*
  - Item-to-Item Based Collaborative Filtering - GeeksforGeeks
- *Matrix Factorization*
  - [2308.04661] Unified Matrix Factorization with Dynamic Multi-view Clustering
- *Content-Based Recommender Systems*
  - [2206.14133] Item Recommendation Using User Feedback Data and Item Profile