Unsupervised Learning Stock Classifier

SEG 4300 - Applied Machine Learning

Winter 2025

School of Electrical Engineering and Computer Science

University of Ottawa

Course Instructor: Daniel Shapiro

Group 07

Josué Dazogbo – 300258449

Racine Kane - 300273077

Submission Date: Mars 24 2025

1. Introduction

This project aims to collect and analyze financial data from S&P 500 companies to uncover meaningful patterns and group similar firms together. By pulling data from both Wikipedia (for company tickers) and Yahoo Finance (for multi-year financial metrics), we create a clean, validated dataset through systematic EDA. We then apply k-Means clustering to group companies based on key financial attributes (e.g., revenue, expenses, net income), and use dimensionality reduction to visualize these clusters. Finally, we build a supervised model that can quickly predict the cluster membership of any new company data, making the entire process more efficient and scalable for real-world financial analysis.

This report showcases a three-stage pipeline for analyzing S&P 500 companies: (2) data ingestion and cleaning (data_analysis.py), (3) unsupervised clustering to group firms by financial similarity (clustering.py), and (4) supervised classification for predicting those cluster labels (clusterPrediction.py).

Note: All plots and visualizations referenced in this report (heatmaps, correlation matrices, cluster plots, model performance graphs, etc.) can be viewed in the project's GitHub repository.

2. Data Collection & Exploratory Data Analysis (data_analysis.py)

2.1 Data Collection

1. Wikipedia S&P 500 List

- We scrape the List of S&P 500 companies table using pandas.read_html(url), storing the symbols, names, and industries in sp500_df.
- We confirm the DataFrame's shape and schema (sp500_df.info()) to ensure correct parsing.

2. Yahoo Finance Data

- Using the yfinance library, each ticker in sp500_df["Symbol"] is queried to retrieve multiple years of financial statements: total revenue, gross profit, net income, operating expense, and more. These appear as columns like 2024-Gross Profit, 2023-Net Income, etc.
- Data for 2020–2024 is combined into a single DataFrame (financials_df), with column naming convention year-metric.

3. Great Expectations Setup

- A Great Expectations data context manages validations to check for unique tickers, required columns, and correct data types. This ensures the raw data conforms to expected standards prior to cleaning.

2.2 Initial Data Validation with Great Expectations

Expectations:

- Unique Ticker: Ensuring no duplicates in financials_df["Ticker"].
- Non-Null Key Columns: Verifying that 2024-Net Income or 2024-Operating Expense are not completely missing.
- Industry in Set: Confirming each industry is among recognized GICS sectors.
- Row/Column Counts: With about 503 S&P 500 firms, we expect row counts in a similar range (470–503 if some are dropped).
- Data Types: Important numeric columns should be floats.

Any warnings prompt further inspection to maintain data integrity.

2.3 Data Cleaning

1. Assessing Missing Data

- A missing-values heatmap (see MissingHeatmapNotCleaned.png) illustrates columns/rows with many NaNs. Rows with ≥ 30% missing cells are dropped to remove highly incomplete entries.
- Each cleaning step is tracked with financials_df.info() to monitor changes in shape and null counts.

2. Filling Strategy

- Backward Fill (Year-by-Year): For time-series columns, older years are filled from more recent data if available.
- Median Imputation: If gaps persist, the global median for each metric is applied.

3. Dropping Sparse Columns

- Any column with ≥ 30% missing data is dropped. For instance, rarely reported metrics or older historical columns might be removed if they are too incomplete.

4. Removing Highly Correlated Features

- A correlation matrix (see CorrelationMatrix.png) flags feature pairs with correlation > 0.90. For each pair, the newer metric is typically preserved (assuming more recent data is more relevant), and the older is removed to avoid multicollinearity.

5. Post-Cleaning Verification

- A second heatmap (see MissingHeatmapCleaned.png) confirms essential columns remain populated. The final cleaned DataFrame is saved to data.csv for downstream clustering.

2.4 Exploratory Analysis

1. Industry Frequency

- A bar chart (see IndustryFrequency.png) breaks down the number of companies per GICS sector, identifying heavily or lightly represented industries.

2. Distribution of Financial Metrics

- Histograms (see FiniancialMetricsDistrubution.png) help reveal skewness and outliers across numeric columns (e.g., revenue or expense). Large-cap firms can often create heavy right tails in distributions.

3. Correlation Matrix

- A heatmap again referencing CorrelationMatrix.png highlights relationships among metrics (e.g., strong correlation of revenue with operating expenses). Uncorrelated features may add unique insights in the clustering stage.

3. Unsupervised Clustering (clustering.py)

Objective: Group companies with similar financial profiles.

3.1 Data Preparation

1. Loading data.csv

- The StockClusterer class loads the cleaned CSV. If FEATURES_TO_USE is specified, the script only scales and uses those columns, providing flexibility.

2. Standard Scaling

- StandardScaler transforms each feature to mean=0 and std=1, ensuring that high-magnitude metrics (like total revenue) don't dominate distances in the clustering step.

3. Dimensionality Reduction

- TruncatedSVD (n_components=2) condenses the data into two principal axes for plotting and quick cluster visualization. The script prints out each component's explained variance ratio.

3.2 Determining Optimal Clusters (Elbow Method)

We vary k from 1 to max_k (e.g., 10 or 20) and track the distortion (inertia) for each.

A plot (see ElbowGraphMethod.png) of distortion vs. k helps identify the "elbow," where increasing k yields diminishing returns.

3.3 k-Means Clustering

1. Fitting k-Means

- Based on the chosen k, KMeans(n_clusters=k, init='k-means++', random_state=42) is run, assigning a cluster label (0, 1, 2, …) to each company.

2. Saving Clusters

- We save a “preprocessed” version of the DataFrame with scaled features and the cluster labels as preprocessedData.csv.

3.4 Plotting Clusters in 2D

The script produces a scatter plot (see Cluster2D.png) of the two SVD components, coloring companies by cluster.

Hover Annotations offer interactive details (ticker, industry) when a point is hovered over, enhancing interpretability.

3.5 Cluster Analysis

Each cluster is analyzed by printing descriptive statistics for its member firms. Below are example insights from a 7-cluster solution, incorporating your additional descriptive details:

1. Cluster 0

- Mix of technology, defense, and manufacturing firms (e.g., NVIDIA, Raytheon Technologies, Texas Instruments, Oracle, and General Electric).
- Characterized by high R&D costs and cyclical revenue trends, driven by innovation and long-term contracts that foster profitability despite diverse end markets.

2. Cluster 1

- Broad mix of financial services, logistics, and industrial companies (e.g., Goldman Sachs, FedEx, Lockheed Martin, Starbucks, and UPS).
- Share capital-intensive operations and sensitivity to economic fluctuations. Their revenue structures center on large infrastructure investment or consumer-driven demand.

3. Cluster 2

- Dominant tech and telecommunications giants (e.g., Alphabet/Google, Amazon, Apple, Meta/Facebook, Microsoft, Verizon).
- Known for strong cash flows, high valuation multiples, and substantial market influence. Heavy reliance on digital services, advertising, and subscriptions puts them in a unique category.

4. Cluster 3

- Composed entirely of financial institutions (e.g., Bank of America, JPMorgan Chase, Wells Fargo, Citigroup, Capital One).
- Commonality arises from similar regulatory environments, revenue tied to interest rates, and lending/investment portfolios. Their business models inherently cluster them together.

5. Cluster 4

- Diverse group of industrial, technology, and consumer goods companies (e.g., Adobe, AMD, Boeing, Caterpillar, Tesla).
- Likely share stable revenue streams and operational costs, balancing growth-oriented strategies (tech/manufacturing) with market stability. Strong demand in these sectors contributes to reliable performance.

6. Cluster 5

- Major healthcare, telecommunications, and retail giants (e.g., Abbott Laboratories, Johnson & Johnson, Pfizer, Disney, Home Depot).
- Often in defensive sectors where demand remains more stable, even during downturns. Trusted brand recognition and entrenched market positions lend them similar financial characteristics.

7. Cluster 6

- Real estate investment trusts (REITs) and utilities (e.g., American Tower, AvalonBay Communities, Duke Energy, Public Storage, Welltower).
- Long-term contracts and high capital expenditures underscore steady income streams. They often rely on property leasing, infrastructure investments, or regulated utility pricing, forming a distinct cluster.

4. Supervised Classification of Clusters (clusterPrediction.py)

Objective: Predict cluster labels for any new or unseen company data, without having to re-run k-Means.

4.1 Data Loading and Splitting

1. preprocessedData.csv

- Contains scaled numeric columns plus the cluster label.

2. Feature/Target Separation

- X includes all columns except cluster.
- y is the cluster label (0, 1, 2, …).

3. train_test_split

- 80% for training, 20% for testing, ensuring reproducible results via random_state=11.

4.2 LazyPredict Evaluation

LazyClassifier trains multiple algorithms (e.g., RandomForest, SVM, Logistic Regression) and ranks them by accuracy and other metrics.

A summary table (see AutoMLResults.png) compares performance.

4.3 Best Model: NearestCentroid

1. Model Selection

- NearestCentroid with metric="euclidean" emerges as a top performer. It is then trained on the training set and tested on the test set.

2. Performance

- Accuracy Score: 0.92 (92%).
- Classification Report (see NearestCentroidReport.png): Lists precision, recall, and f1-score for each cluster.
- Confusion Matrix (see ConfusionMatrix.png): Showcases how each true cluster label compares to predicted labels.

4.4 Model Deployment

The trained NearestCentroid model can be saved using joblib.dump. This allows fast cluster prediction for any new data point without repeating the entire k-Means clustering process.

5. Conclusions and Future Directions

Key Achievements:

- Thorough Data Pipeline: Scraping, validating, and cleaning financial data ensures a robust foundation for clustering and classification.
- Meaningful Clusters: k-Means uncovered distinct groups (financial institutions, big tech, REITs/utilities, etc.), illustrating how certain industries or financial profiles cluster together.
- Efficient Classification: A NearestCentroid model provides immediate predictions for cluster membership, making the solution scalable and deployable.

Potential Enhancements:

- Expanded Feature Engineering: Include more holistic metrics (e.g., forward P/E, ESG scores, intangible assets) for richer clustering.

- Alternative Clustering Methods: Compare DBSCAN, hierarchical clustering, or Gaussian mixture models to see if a different algorithm better captures the data's structure.
- Longitudinal Analysis: Track how companies shift between clusters over multiple quarters or years, revealing time-based trends in financial performance.
- Hyperparameter Tuning: Further optimization (e.g., for k-Means or the NearestCentroid model) might improve cluster separation or classification accuracy.

By combining EDA, k-Means clustering, and a supervised classifier, this pipeline delivers both insights into how S&P 500 companies group financially and a quick method to categorize any new firm into an existing cluster.