CMPT459 Project Report: Predicting Stocks Performance using Data Science Algorithms

Duy Anh Vu[301386587]
Parham Khadem[301451192]
Simon Fraser University
Burnaby, BC, Canada

1 INTRODUCTION

1.1 Motivations

Stock market trading is an essential and dominant financial market activity, significantly impacting society and individuals economically. Professional investors and enthusiastic stock traders are vested in finding techniques to predict upcoming market trends to maximize their returns and minimize their losses. The uncertainty and volatility of stock prices depend on several variables, such as historical performance, periodic patterns, and economic changes, as well as sentimental variables, such as new product releases, wars, or political changes. Many researchers have developed prediction models to address this complexity using state-of-the-art algorithms such as deep neural networks [1]. At the same time, stock prediction models are often designed for short-term prediction of daily or weekly results, using time series of individual stocks as input. As such, we are interested in designing stock prediction models that appeal to causal stock traders interested in investing in a few stocks for long-term returns.

1.2 Project Goals

Our goals for this project are to develop hybrid stock prediction models that consider not only the stock market's historical performance but also incorporate information about the stock and sentiment analysis through quarterly reports to predict changes in the stock market. Our models will predict the performance of the stock in the next quarter compared to the S&P 500 stock, which is one of the most commonly followed equity indices and used by organizations such as the Conference Board to forecast the direction of the United States economy [2]. This project also generated a large stock dataset of around 8000 stocks from January to June 2023. We reported 1000 stocks in quarter 1 of 2023, allowing others to use them to improve our model and other data mining projects.

1.3 Methodologies Outline

The overarching method that our project followed is as follows:

(1) We created the historical stock data and stock information using Python library's *yfinance*. For quarterly data, we acquire the stock quarterly report using US Securities and Exchange Commission (SEC) companies fillings. Then, we extracted features using natural language processing (NLP) from these quarterly reports to create tabular data.
(2) We organized the stock data by stock and combined it with the feature-extracted report data to create the final raw data for this project. We then cleaned the dataset by imputing or removing missing samples and features, normalizing the numerical features. We also created the label column for the classification task
(3) We performed EDA to understand the structure and distribution of the dataset. We then analyzed the dataset further, including clustering and outlier detection.
(4) We applied feature selection algorithms on the dataset to reduce dimensionality. We then implemented classification and regression algorithms on the dataset with and without feature selection and evaluated the model's result. We also perform hyperparameter tuning on the models to improve the results.

2 DATASET SELECTION

2.1 Stock Data

To build a comprehensive dataset for our analysis, we collected historical stock price data for a wide range of companies over the period from January to June 2023. We utilized the *yfinance* library [3], a popular Python package that provides a convenient interface to download historical market data from Yahoo Finance.

2.2 Stock Information

The *yfinance* library[3]also allows us to extract more information from a particular stock as a Python dictionary that indicates the various information about the company owning it and the stock's technical indicator. In particular, we will extract the following information to use as features for our data:

- Number of full-time employees
- The industry and sector that the stock belongs to
- The company's location (city, state, country) and ZIP code
- The company's phone number
- Stock exchange where it is traded
- *Yahoo Finance* analysis recommendations
- Stock financial metrics' like revenue growths, profit margins and Cash per share (CPS)

We have created 13 new information features that are most relevant to our project.

2.3 Quarterly Report

To handle more complexities of our dataset, we utilized the U.S. Securities and Exchange Commission's (SEC) Electronic Data Gathering, Analysis, and Retrieval (EDGAR) system [4]. We developed a Python script that interfaces with the EDGAR Application Programming Interface (API) to programmatically download 10-Q filings for the companies in our dataset. By mapping each company's ticker symbol to its Central Index Key (CIK), we retrieved the relevant

filings for the first quarter of 2023. The downloaded HTML content of these filings was parsed using the *BeautifulSoup* library to extract textual data. This textual data was then processed using natural language processing techniques to generate features for our analysis.

3 DATA PREPROCESSING

After acquiring and creating data as outlined in Section 2, we begin to perform data preprocessing on our raw data to shape them into a format usable for later analysis and machine learning tasks.

3.1 Dimensionality reduction of NLP features

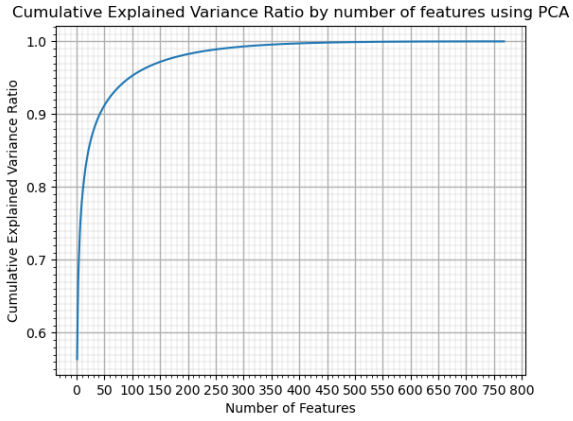

Figure 1: Culminate Explained Variances Ratio by number of features using PCA

After processing the reports into NLP features in Section 2.3, we still end up with 600 NLP features. The large number of features here would cause many problem in analysis and machine learning tasks, as per the curse of dimensionality. To address this problem, we reduced the NLP features using Principal component analysis (PCA). PCA projects the data to a lower dimensional space using Singular Value Decomposition, while ensures that the reduced dataset still retain most of the explained variances of the old dataset. From Figure 1, we noticed that:

- 80% of the original data variances is explained by the 12 first features
- 90% of the original data variances is explained by the 42 first features
- 95% of the original data variances is explained by the 95 first features

To determine the dimension of the reduced NLP data, we aimed to strike a balance between reducing as much dimension as possible, while maintaining a majority of variance explained from the original NLP features. As such, we choose to reduced number the NLP features from 600 to 42, which strike a good balance as we have explained.

3.2 Creating compressed data

The current stock data format is daily records, so we need to organize the data around the stock instead and create aggregated features from the time-series data for each stock. This would simplify the problem by focusing on the most relevant summary statistics, which makes the modelling process more efficient and reduces noise. Furthermore, since the stock information and NLP features are organized by stock, we would simplify the merging operation to create a final raw dataset.

We created the following aggregated features for each stock:

- The open value, close value and volume on the last day of Q1 2023
- The close value on the first day of Q1 2023
- The average of daily volume in Q1 2023
- The highest price among the *high* feature in Q1 2023
- The lowest price among the *low* feature in Q1 2023
- The quarterly stock performance in Q1 2023, which is calculated by:

$$Performance_{Q1} = \frac{\overline{Close_{Mar}} - \overline{Close_{Jan}}}{\overline{Close_{Jan}}} * 100\%$$

We ended up with 11 aggregated features, combined with 13 information features and 42 NLP features, resulting in 66 total features. On the other hand, our sample size has decreased significantly from 50, 000 to 939 samples. We have to accept this unfortunate cost since we organized the data by stock instead of day.

3.3 Compute the target feature

We computed the quarterly stock (including the index stock S&P) performance in Q2 2023 using the following equation:

$$Performance_{Q2} = \frac{\overline{Close_{Q1}} - \overline{Close_{June}}}{\overline{Close_{Q1}}} * 100\%$$

Afterward, we compute the relative quarterly stock performance in Q2 2023 as follows:

$$Relative\ Performance = Performance_{Q2} - Performance_{S\&P}$$

We will use this result as our target feature and later create a label feature from it as well.

3.4 Handling Missing Values

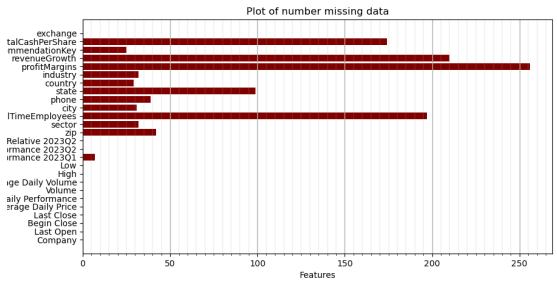

Figure 2: The number of missing data for each feature

We plot the missing data for each feature in Figure 2 to handle missing values. We will handle the missing values as follows:

- For features *Performance 2023Q1* and *country*, we consider them as importance. As such, we will remove samples that are missing these features.
- For features *profitMargins*, *revenueGrowth* and *totalCashPerShare*, we perceive as being less important. Since these features are also missing more than 150 values, we remove these features.
- For features *fullTimeEmployees* and *recommendationKey*, we still consider them as somewhat important. So, we will impute the missing values for each feature as follows:
  - *recommendationKey*: We fill the missing value with *none*, which is the default value for this feature
  - *fullTimeEmployees*: We impute the missing values using K-Nearest Neighbors, where the samples with missing values are imputed using the mean value from k nearest neighbours. To perform this imputation, we also need to encode categorical features, which will be discussed in Section 3.6

We also drop features *Performance 2023Q2*, *phone*, *state*, *city*, *zip* and *industry*, as we feel that they are made redundant by other features. This also allows us to reduce the number of input features from 66 to 57.

3.5 Normalizing Numerical Features

Looking at the numerical distribution of the data in Figure 3, we notice that all numerical features besides NLP features do not follow normally distributed shapes with high right-skewness and have different distances. As such, we performed normalization numerical features to transform the data to follow a normal distribution with equal distance. We did not perform this step on the NLP features, as they all seem to follow a Gaussian distribution.

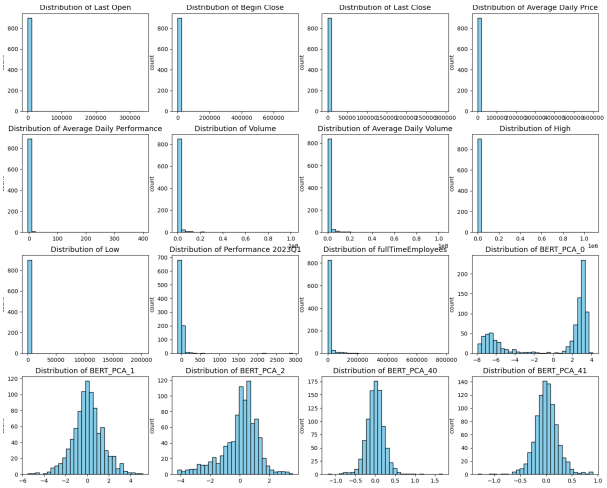

Figure 3: Distribution of Numerical Features Pre-Normalization

We perform two different normalization methods for different numerical features, as follows:

- For features like *fullTimeEmployees*, *Volume* and *Average Daily Volume*, we transformed the data using $log(x+1)$ due to the data have high right-skewness.
- For the remaining numerical features, we used quantile transforms, a non-linear transformation that maps the data to a uniform distribution of $\mu = 0$ and $\sigma^2 = 1$. While this transformation distorts linear correlations between variables and features, it renders features' ranges more comparable to each other.

The result of the normalization process can be seen in Figure 5.

3.6 Encoding Categorical Features

While many algorithms in *scikit-learn* support categorical features, k-nearest neighbours imputation (in Section 3.4), clustering and outlier detection algorithms require us the encode categorical features to numerical to compute distance function. For this reason, we provide an encoding function for this task, but we will keep our categorical features intact in the dataset.

We only performed an ordinal encoding for most categorical features to convert the value into non-negative integer values. There are two exception however, *recommendationKey* and *country*, whose methods we cover bellow:

3.6.1 Encoding feature recommendationKey. There are 5 different categories for this feature: *none*, *buy*, *hold*, *strong_buy*, and *underperform*. We want to highlight that *underperform* is a negative value, so our encoding will be as follows:

- *underperform* is encoded as -1
- *none* is encoded as 0
- *hold* is encoded as 1
- *buy* is encoded as 2
- *strong_buy* is encoded as 3

3.6.2 Encoding feature country. There are 21 distinct categories in *country*. However, most of these categories only contain a few samples, besides the United States, which presents roughly 800 samples (as seen in Figure 6). As such, will we compress the *country* feature into only five categories, which are: *United States*, *Asia and Pacific*, *North America*, *Europe* and *Africa*. This reduces the number of categories while increasing the number of samples in each category. For encoding, we performed ordinal encoding on the compressed feature as with other features.

3.7 Building the Label Column

From visualizing the distribution of *Relative Performance 2023Q2* target in Figure 4, we notice that while the target follows a normal distribution shape, there are more samples with $< 0\%$ target than positive. To balance the number of samples in each Label, we decided to create the label target as follows:

(1) *Exceptionally Bad Performance*: $(-\infty, -50\%]$
(2) *Bad Performance*: $(-50\%, -10\%]$
(3) *Neutral*: $(-10\%, 10\%]$
(4) *Good Performance*: $(-10\%, +\infty)$

The distribution of our label target *Relative 2023Q2 Label* can be seen in Figure 6.

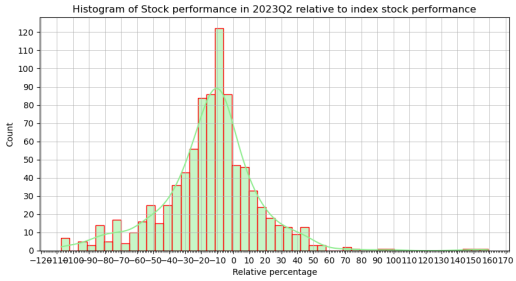

Figure 4: Distribution of *Relative Performance 2023Q2*. Note that the values above 200% are cut off for visualization purposes

4 EXPLORATORY DATA ANALYSIS (EDA)

After the data preprocessing step, we will perform EDA to visualize the data and understand the structure and distribution of the dataset. EDA includes plot distributions of key features using a histogram (Figures 5 and 6), plot 2D Scatter of the data (Figure **??**), and visualize relationships between features and identify correlations using heatmap plot (Figure 8).

4.1 EDA results

4.1.1 Distribution of Numerical Features. Looking at the distribution of numerical features, we notice that most non-NLP features are mapped to a normal distribution, with equal range to each other. Even for features normalized using a logarithmic algorithm, they still follow a somewhat normal distribution shape, with a value range from 0 to 15. The exception here is *Volume*, which has an extremely large number of samples close to 0 despite the remaining samples following normal distribution. This motivates us to drop *Volume* features from the dataset.

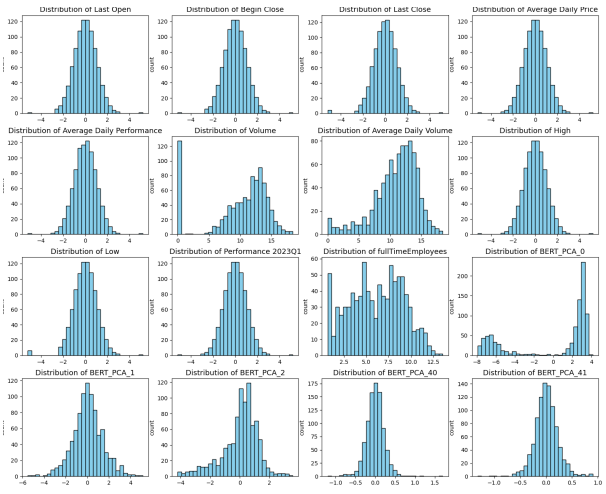

Figure 5: Distribution of Numerical Features Post-Normalization

4.1.2 Distribution of Categorical Features. For categorical features, we notice category imbalance in all features, especially *country*, where *United States* has close to 800 samples. The target label is also somewhat imbalanced, despite our attempt to split the bad category into *Bad Performance* and *Exceptionally Bad Performance*.

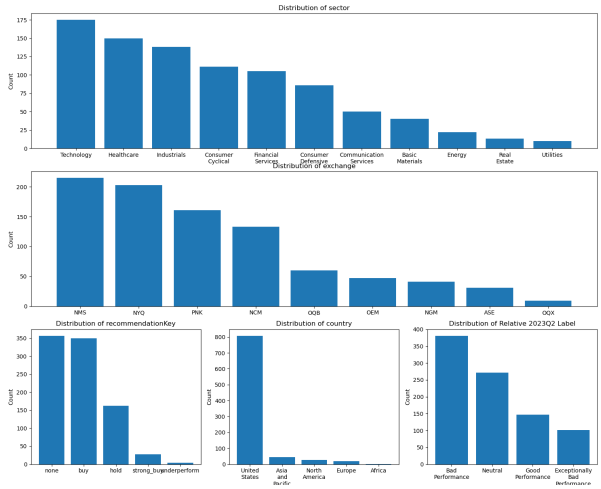

Figure 6: Distribution of Categorical Features

4.1.3 2D scatter of data. While we can see that the 2D scatter result has a good delineation of 2 different clusters, our labels did not fit into these clusters but instead distributed all over the Scatter result with no clear separation. Unfortunately, this will make our prediction tasks much harder.

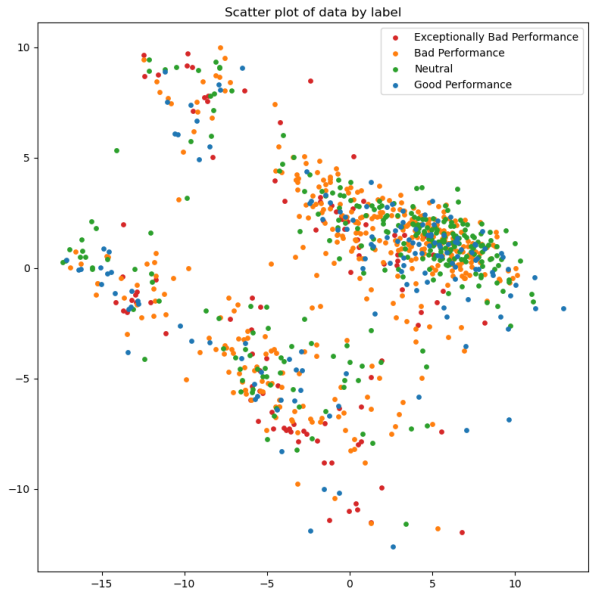

Figure 7: 2D Scatter of the dataset with Label

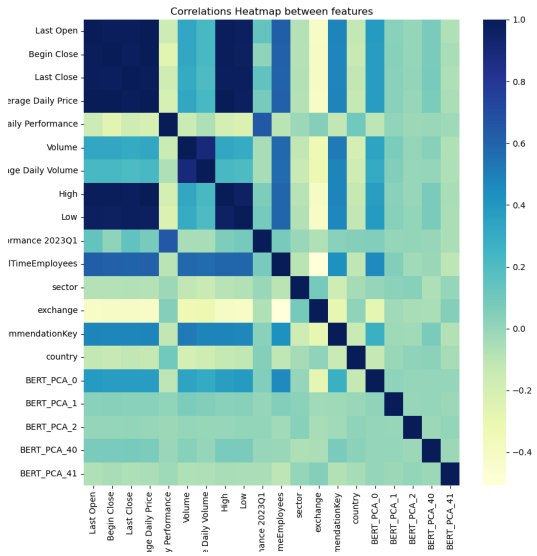

Figure 8: Correlation heatmap between features

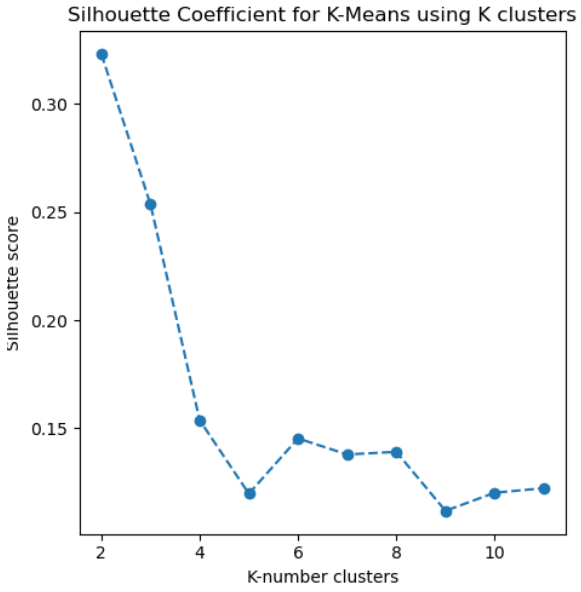

Figure 9: Silhouette-Coefficient result from K-Means clustering using different K hyper-parameter

4.1.4 Correlation Heatmap. From the heatmap result, we notice that the aggregated features have an extremely high correlation to each other. While this is likely due to the step we take in compressing the time-series data to organize by stock, it is still very concerning. On the other hand, the remaining features, including stock information and NLP, only have a medium to low correlation to each other.

4.2 Discussion of EDA Result

Our dataset has several challenges, including unbalanced categorical features and target labels, highly correlated aggregated features and complex data, which result in difficult-to-separate labels. Some of these challenges result from our decision to compress the dataset, while the others are due to the complexity of the prediction task we are dealing with. As such, we do not expect our models to predict accurately due to these challenges in our dataset.

5 CLUSTER ANALYSIS

We perform clustering as a continuation of EDA to identify and visualize possible groupings of the dataset. The two algorithms we used are K-Means clustering and Density-Based Spatial Clustering of Applications with Noise (DBSCAN).

5.1 K-Means clustering

K-means clustering is an unsupervised algorithm that groups the sample into k clusters based on its distance from the cluster's center following several iterations. While the algorithm is simple to implement and compute, the difficulty lies in the choice of initial clustering since simply drawing a sample of k objects from the dataset can result in outliers and slow convergence. A better approach we used is K-Means++, which samples the initial cluster centers based on an empirical probability distribution of the points' contribution to the overall inertia.

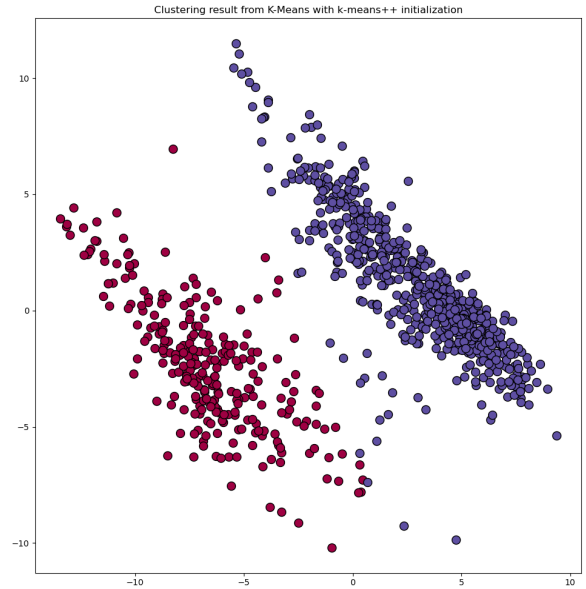

Figure 10: Visualization of clustering using K-Means ($K = 2$)

Another issue is regarding selecting hyper-parameter K clusters. For this, we will run K-Means clustering several times with different K values from 2 to 12 and select the K value that produces the cluster with the highest silhouette-coefficient score. From Figure 9, we can see that $K = 2$ produce the highest silhouette-coefficient score. We

also visualize the K-Means clustering result with $K = 2$ in Figure 10.

5.2 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

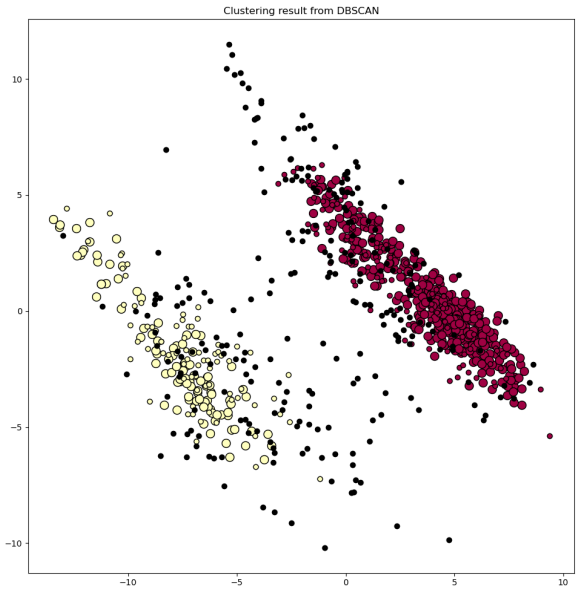

Figure 11: Visualization of clustering using DBSCAN ($\epsilon = 5.0$, *MinPoints*= 8)

DBSCAN is a different unsupervised algorithm that groups together closely packed points while labelling points in the low-density region as noise. Compared to K-Means, DBSCAN has several strengths, such as its robustness to outliers and ability to find arbitrarily shaped clusters. However, DBSCAN also has problems with datasets with large differences in densities, which can result in points being mislabeled as outliers.

While DBSCAN can automatically determine the number of clusters, unlike K-Means, we will need to determine the hyper-parameter ϵ, the maximum distance between two samples for one to be considered as in the neighbourhood of the other. Another hyper-parameter to consider is *MinPoints*, which is the minimum number of samples in an object's neighbourhood for it to be considered the core object. Based on our 2D Scatter of the data and some initial testing, we determine our hyper-parameters are $\epsilon = 5.0$ and *MinPoints*= 8. We also visualize the clustering result of DBSCAN in 11.

5.3 Discussion of clustering results

The clustering results of K-Means and DBSCAN are very similar, mainly due to the Scatter of the data being easy to separate into two groups. However, DBSCAN produced a lot of noise, which may be due to the high dimensionality of the data and differences in densities in the dataset. For further evaluation, we can use internal or external validation using the Label as ground truth. However,

Table 1: Evalution of Clustering Algorithm

Algorithm	K-Means	DBSCAN
K clusters	2	2
Noise points	N/A	247
Silhouette Coefficient	0.323079	0.194831
Calinski-Harabasz Index	412.087807	184.843933
Davies-Bouldin Index	1.291942	2.549656

as seen in Figure **??**, the labelled result does not correlate to the distribution of the dataset. As such, we use internal validation algorithms to evaluate our clustering result:

- Silhouette Coefficient: Compute based on the cluster tightness and separation from other clusters. Higher scores correspond to more defined clusters.
- Calinski-Harabasz Index: the ratio of the sum of between-clusters dispersion and within-cluster dispersion for all clusters. Higher scores correspond to more defined clusters.
- Davies-Bouldin Index: signifies the average 'similarity' between clusters. Lower scores correspond to better separation between the clusters.

As seen in table 1, in all metrics, K-Means perform better than BDSCAN. BDSCAN also label 247 samples as noise, a significant number out of the 900 samples we have. While the clustering suggests a well-defined clustering, it is inappropriate for our dataset since these clusters do not correlate to our Label, as shown in Figure **??**. However, it is still an interesting result that is worth more analysis.

6 OUTLIER DETECTION

For the final step in our analysis task, we perform outlier detection, another unsupervised learning task. Unlike clustering, outlier detection seeks samples that are different from all clusters. We use two algorithms for outlier analysis: Isolation Forest and Local Outlier Factor.

6.1 Local Outlier Factor (LOF)

Local Outlier Factor (LOF) is an outlier detection which computes the anomaly score of each sample by measuring the local deviation of the density of that sample with respect to its k-nearest neighbours. For an outlier sample, the anomaly score will be high because the density of the sample is lower than its neighbours. The hyper-parameter K neighbours need to be carefully chosen; with a small k value can cause the model to not be robust enough, whereas a large k value can cause the model to not be local enough. The result of LOF detection can be seen in Figure 12.

6.2 Isolation Forest (IF)

Isolation Forest (IF) is an outlier detection algorithm that 'isolates' outlier by constructing multiple binary trees to partition the data on a random number of features, similar to a Random Forest algorithm. Unlike Random Forest, IF randomly chooses a feature to partition on and chooses a random partition value within the range of that feature. Anomaly score is inversely associated with the path length due to the algorithm assumption that anomalies are few and can

be isolated from other clusters with only a few partitions, with the final outlier points being chosen from the aggregated decision of these binary trees. The result of IF detection can be seen in Figure 12.

6.3 Discussion of Outlier Detection results

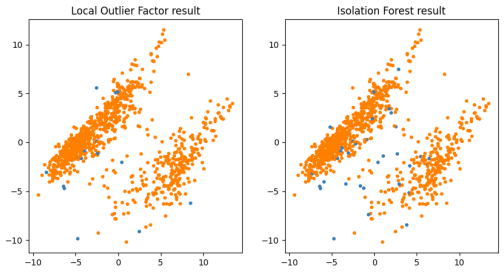

Figure 12: Visualization of outlier detection using Local Outlier Factor (Left) and Isolation Forest (Right)

Looking at the result of both outlier detection algorithms in Figure 12, we see that IF identifies more objects as outliers than LOF, perhaps due to LOF working locally, whereas IF detects outliers globally. In terms of number, LOF identifies 15 samples as outliers, while IF identifies 34 samples as outliers, with nine samples being identified by both algorithms as outliers.

Since our project goal is to detect the stock's quarterly relative performance, these outlier points may contain useful data for our regression tasks. We also have a small number of samples, 900, so removing further samples, even if they are outliers, would not be wise. As such, we decided to keep the outlier in the dataset.

7 FEATURE SELECTION

To enhance the performance of our classification and regression models, we employed several feature selection techniques to identify the most informative features among the 78 available in our dataset. Feature selection reduces the dimensionality of the data, thus mitigating the risk of overfitting and improving computational efficiency and model interpretability.

7.1 Feature Selection Methods

We utilized the following feature selection methods to ensure the effectiveness of our models:

- **Variance Thresholding**: This method removes features with low variance, assuming that such features may not contribute significantly to the model's predictive power. We applied variance thresholds of 0.3 and 0.9 to identify and eliminate features with insufficient variability.

- **Mutual Information (MI)**: MI measures the dependency between each feature and the target variable. Features with higher mutual information scores are considered more informative. We computed mutual information for both classification and regression tasks to select features most relevant to our target variables.

- **Recursive Feature Elimination (RFE)**: RFE is an iterative method that fits a model and removes the least essential feature (or features) at each step until the desired number of features is reached. It ranks the features according to their importance in the model, allowing us to select the most significant ones.

7.2 Feature Selection Results

We evaluated the performance of our models using different numbers of features selected by each method. The results are summarized in Figure 13, which shows the model performance metrics as a function of the number of features for each feature selection method.

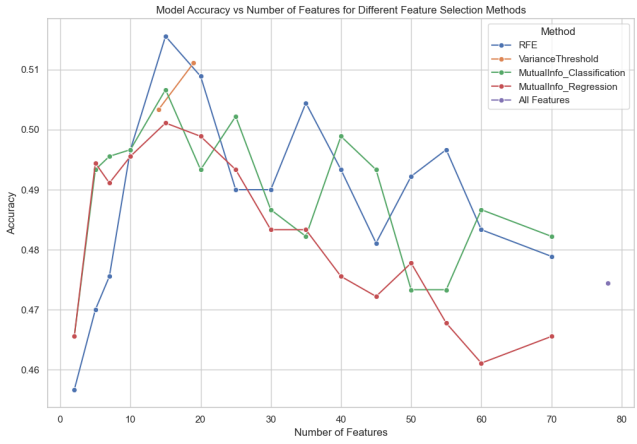

Figure 13: Feature selection results for different methods.

As shown, the optimal performance for the classification model was achieved when using 15 features selected by Recursive Feature Elimination. For the regression model, the best results were obtained with 15 features selected based on mutual information scores. These findings suggest that a subset of approximately 15 features is sufficient to capture the most relevant information in the data set for both tasks.

7.3 Discussion

By reducing the number of features from 78 to 15, we simplified the models and mitigated potential issues associated with high-dimensional data, such as overfitting and increased computational complexity.

8 CLASSIFICATION AND REGRESSION

In this section, we explore both classification and regression models to predict stock performance. By predicting the actual numerical performance of the stocks and categorizing their performance levels, we aim to provide a comprehensive analysis that can be useful for different investment strategies.

8.1 Classification Models

Classification algorithms that we use to predict the performance label are **Random Forest Classifier**, and **Support Vector Classifier (SVC)**. We also compared them against a **Baseline Random Classifier**, which predicts the Label randomly based on the distribution of the classes in the training data.

8.1.1 Results. The results for each classifier are presented below with their respective confusion matrix figure.

Random Forest Classifier.

- **Accuracy**: 46.67%
- **Precision**: 46.76%
- **Recall**: 46.67%
- **F1-Score**: 45.43%

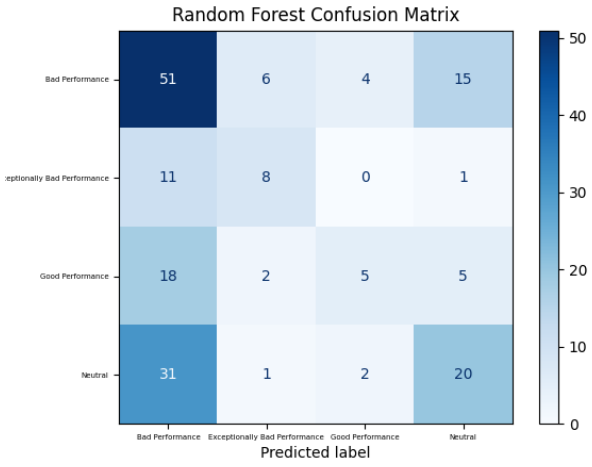

Figure 14: Random Forest Classifier confusion matrix

Support Vector Classifier (SVC).

- **Accuracy**: 44.44%
- **Precision**: 44.91%
- **Recall**: 44.44%
- **F1-Score**: 42.57%

Baseline Random Classifier.

- **Accuracy**: 26.11%
- **Precision**: 32.85%
- **Recall**: 26.11%
- **F1-Score**: 27.50%

8.1.2 Analysis. The results show that the Random Forest Classifier slightly outperformed the SVC in terms of accuracy and F1-score. Both models significantly outperformed the Baseline Random Classifier, indicating that they have learned patterns from the data useful for predicting the performance labels.

However, the overall accuracy and F1-scores are relatively low, suggesting that the models have difficulty distinguishing between the classes.

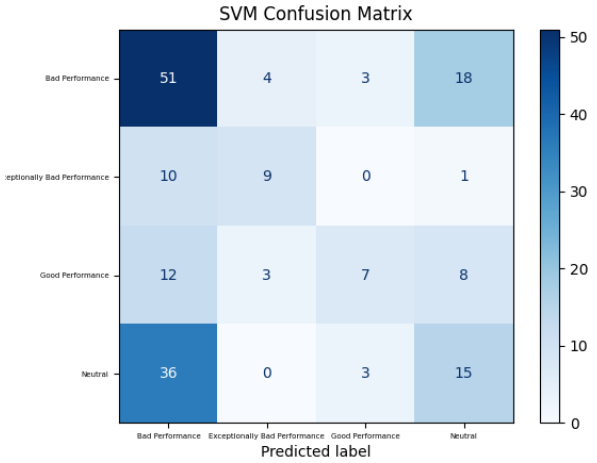

Figure 15: Support Vector Classifier (SVC) confusion matrix

8.2 Regression Models

For predicting the actual numerical percentage performance of the stocks, we utilized regression models: **Random Forest Regressor**, and **Support Vector Regressor (SVR)**. We compared these models against a **Baseline Random Regressor**, which predicts random values between -200% and +200%.

8.2.1 Results.

Random Forest Regressor.

- **Mean Squared Error (MSE)**: 580,756.70
- **Mean Absolute Error (MAE)**: 77.79
- **Root Mean Squared Error (RMSE)**: 762.07
- **R-squared Score (R^2)**: 0.0244

Support Vector Regressor (SVR).

- **Mean Squared Error (MSE)**: 598,179.91
- **Mean Absolute Error (MAE)**: 77.86
- **Root Mean Squared Error (RMSE)**: 773.42
- **R-squared Score (R^2)**: -0.0049

Baseline Random Regressor.

- **Mean Squared Error (MSE)**: 619,135.67
- **Mean Absolute Error (MAE)**: 160.84
- **Root Mean Squared Error (RMSE)**: 786.85
- **R-squared Score (R^2)**: -0.0401

8.2.2 Analysis. The Random Forest Regressor performed slightly better than the SVR and the Baseline Random Regressor, as indicated by lower MSE and higher R^2 scores. However, the R^2 values for all models are close to zero, suggesting that the models do not explain much of the variance in the target variable.

The high error metrics indicate that the regression models are incapable of producing exact values for stock performance with the current data mining pipeline.

8.3 Discussion

Overall, the classification models performed better than the regression models regarding relative improvement over the baseline. The Random Forest Classifier, in particular, showed a modest improvement in accuracy and F1-score. The results suggest that while our models can capture some patterns in the data, there is significant room for improvement.

9 HYPER-PARAMETER TUNING

To improve the model's performance, we conducted hyper-parameter tuning using grid search for the Random Forest and Support Vector Machine (SVM) algorithms.

9.1 Random Forest Hyper-parameter Tuning

9.1.1 Methodology. We defined a grid of hyper-parameters for the Random Forest models to explore. The parameters and their respective values were as follows:

- `n_estimators`: [100, 200, 500, 1000]
- `max_depth`: [None, 10, 25, 100]
- `min_samples_split`: [2, 5, 10, 25]

We used Grid Search Cross-Validation to evaluate all possible combinations of these parameters, selecting the best combination on the validation set.

9.1.2 Results.

Classification Model. After tuning, the Random Forest Classifier achieved the following performance metrics:

- **Accuracy**: 46.67%
- **Precision**: 46.84%
- **Recall**: 46.67%
- **F1-Score**: 44.53%

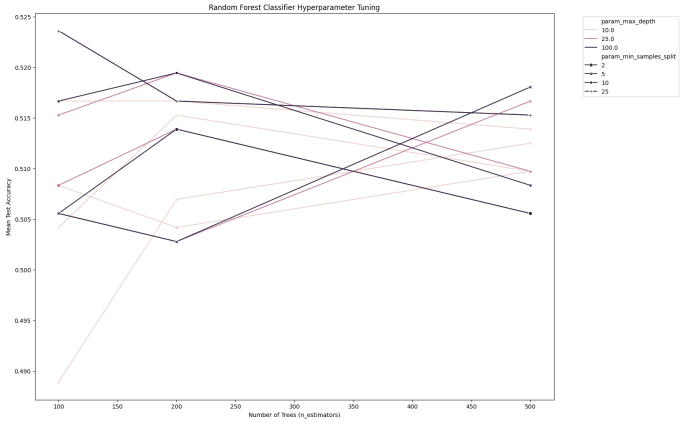

Figure 16: Random Forest Classifier tuning results vs. validation accuracy

Regression Model. For the Random Forest Regressor, the tuning process yielded the following results:

- **Mean Squared Error (MSE)**: 580,756.70
- **Mean Absolute Error (MAE)**: 77.79
- **Root Mean Squared Error (RMSE)**: 762.07
- **R-squared Score (R^2)**: 0.0244

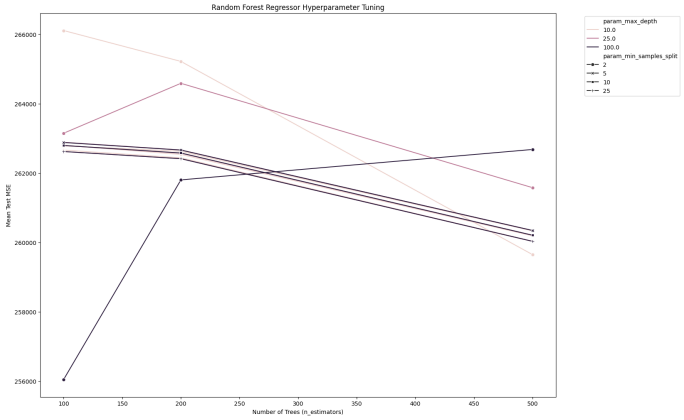

Figure 17: Random Forest Regressor tuning results vs. validation accuracy

9.1.3 Analysis. The hyper-parameter tuning for the Random Forest models resulted in slightly improved performance metrics compared to the default settings. The classifier's accuracy remained at 46.67%, but there was a marginal increase in precision and F1-score. The confusion matrix indicates a better identification of the "Bad Performance" class, with an increased true positive rate.

For the regression model, the tuning did not change the error metrics or the R^2 score. This suggests that the model's performance plateaued, and further tuning these parameters did not yield substantial gains.

9.2 Support Vector Machine Hyper-parameter Tuning

9.2.1 Methodology. For the Support Vector Machine models, we defined the following grid of hyper-parameters:

- `C`: [1, 5, 10, 20]
- `gamma`: ['scale', 'auto']
- `kernel`: ['rbf', 'linear', 'sigmoid', 'poly']

We utilized Grid Search Cross-Validation to explore the combinations of these hyper-parameters, aiming to find the optimal settings for both the classification and regression tasks.

9.2.2 Results.

Classification Model. The tuned Support Vector Classifier achieved the following performance:

- **Accuracy**: 45.56%
- **Precision**: 45.69%
- **Recall**: 45.56%
- **F1-Score**: 43.64%

Regression Model. For the Support Vector Regressor, the tuning process did not yield improved results, and the performance metrics remained the same:

- **Mean Squared Error (MSE)**: 598,179.91

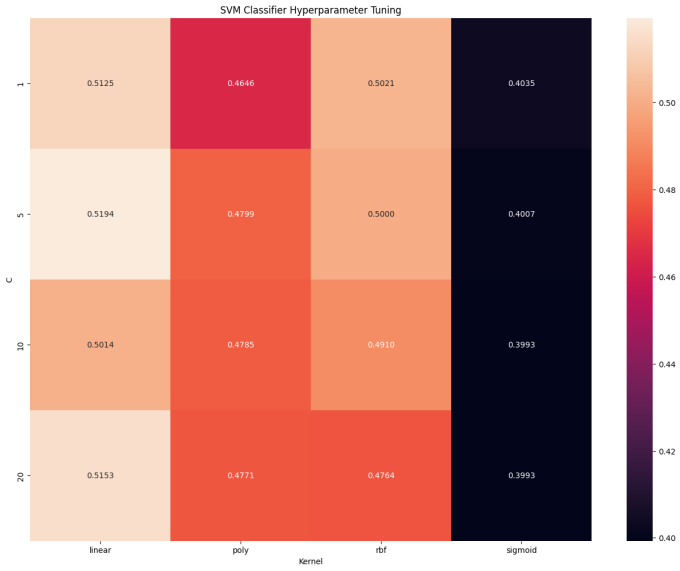

Figure 18: SVC tuning results vs validation accuracy

- **Mean Absolute Error (MAE)**: 77.86
- **Root Mean Squared Error (RMSE)**: 773.42
- **R-squared Score (R^2)**: -0.0049

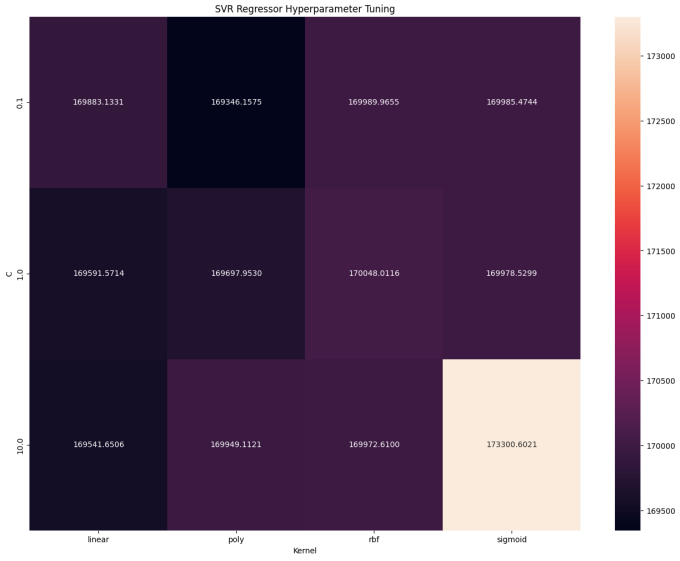

Figure 19: SVC tuning results vs. validation accuracy

9.2.3 Analysis. The hyper-parameter tuning for the SVM classifier resulted in a slight improvement in precision and F1-score, particularly for the "Exceptionally Bad Performance" and "Good Performance" classes. The accuracy increased marginally to 45.56%. The confusion matrix shows better classification of the "Exceptionally Bad Performance" class, indicating that the tuned model is more effective in identifying this category.

However, the regression model did not improve after tuning. The error metrics and the R^2 score remained virtually unchanged, suggesting that the SVM regression model may not be well-suited for this particular task or that additional features and data preprocessing are required.

9.3 Discussion

While hyper-parameter tuning efforts led to modest improvements in the classification models, the improvements were not significant enough to affect their predictive power.

10 RESULTS

Our experiments demonstrated that the classification models achieved modest accuracy in predicting stock performance labels for various companies in 2nd quarter of 2023. The **Random Forest Classifier** attained an accuracy of **46.67%**, slightly outperforming the **Support Vector Classifier (SVC)** at **44.44%**, with both models significantly surpassing the **Baseline Random Classifier** 's accuracy of **26.11%**. Feature selection using **Recursive Feature Elimination** effectively reduced the feature set to 15 key features without compromising performance, enhancing model efficiency.

Hyperparameter tuning yielded minimal improvements, indicating that model performance was more constrained by data limitations than parameter settings. The regression models struggled to predict exact stock performance, evidenced by low **R-squared** scores close to zero, suggesting they explained slight variance in stock returns.

These findings indicate that while our models captured some underlying patterns, accurately predicting stock performance remains challenging due to the complexity and unpredictability of financial markets. The results underscore the necessity for more advanced modelling techniques and richer datasets to enhance predictive accuracy in stock market analysis.

11 CONCLUSION

In this project, we developed hybrid stock prediction models that combine historical stock data, company information, and sentiment analysis from quarterly reports to forecast stock performance relative to the S&P 500 index. Through comprehensive data preprocessing and effective feature selection, we reduced the dataset to the most informative 15 features, enhancing model efficiency and interpretability.

Our classification models achieved modest accuracy, with the Random Forest Classifier reaching approximately 46.67%, outperforming the baseline but indicating limited predictive power. Regression models struggled to predict exact stock performance, evidenced by low R-squared scores near zero, highlighting minimal explanatory capability.

While our models captured some patterns in the data, predicting stock performance remains challenging. This project highlights the need for continued research and methodological innovation in financial data science to improve predictive outcomes.

11.1 Challenges and Limitations

Throughout this project, we faced several challenges and limitations that impacted the effectiveness of our models and the overall outcomes.

11.1.1 Complexity of Stock Market Data. The stock market's multi-factor complexity posed a significant challenge. Many factors influence stock prices, including economic indicators, company performance, investor sentiment, geopolitical events, and unforeseen circumstances. Capturing all these variables in predictive models is inherently tricky. Our dataset, while comprehensive in certain aspects, could not encompass the full spectrum of factors affecting stock performance. This limitation contributed to the modest accuracy of our models, as they could only account for external influences within the scope of our data.

11.1.2 Limited Computational Resources. Our computational power constraints limited the complexity and scale of models we could implement. Processing large volumes of high-dimensional financial data requires substantial resources, especially when employing advanced algorithms and extensive hyperparameter tuning. These limitations prevented us from exploring more sophisticated models.

11.1.3 Insufficient Quantity and Variability of Quarterly Reports. The limited number of quarterly reports restricted our ability to perform robust sentiment analysis and feature extraction. More documents would have provided higher variance features and a richer dataset. This limitation may have hindered the models' ability to generalize across different stocks and sectors, affecting predictive performance.

11.1.4 Aggregation of Data and Lack of Time Series Modeling. By aggregating daily stock data into quarterly summaries, we simplified the dataset and removed temporal volatility crucial for capturing stock price dynamics. This approach neglected short-term fluctuations and trends that could predict future performance. Additionally, not incorporating time series analysis meant our models could not leverage temporal dependencies and patterns inherent in financial data, potentially limiting their predictive power.

11.1.5 Summary. These challenges highlight the complexities involved in stock market prediction. Addressing them would require more advanced modelling techniques, richer and more varied datasets, incorporation of time series analysis, and potentially greater computational resources. Recognizing these limitations is crucial for guiding future research efforts to improve predictive models in financial markets.

11.2 Future Work

Future research can focus on several key areas to enhance predictive accuracy and address this project's limitations. Gathering more refined data about each company's hierarchical structure, such as detailed corporate governance information and management profiles, could provide deeper insights into factors influencing stock performance. Understanding the distribution of investors, including institutional versus retail ownership and their trading behaviours, can shed light on market dynamics and investor sentiment.

Integrating financial news articles and sentiment analysis can capture real-time market perceptions and reactions to events, improving the models' ability to predict short-term fluctuations. Employing natural language processing techniques on news data can help identify trends and shifts in sentiment that are not reflected in historical price data alone.

Incorporating alternative data sources like social media sentiment, macroeconomic indicators, and industry trends can enrich the dataset and provide a more holistic view of the factors affecting stock performance. Implementing advanced modelling techniques, such as deep learning architectures and time-series analysis, can better capture the data's complex patterns and temporal dependencies.

Addressing class imbalance through resampling methods or specialized algorithms and enhancing data quality by expanding the dataset across different periods and markets can improve model performance. Collaboration with financial experts and adopting interdisciplinary approaches combining finance and data science can contribute to more robust and insightful predictive models.

Finally, future work can analyze the dataset further, particularly address the 2 distinct clusters that we identified in Section 5 which was not captured by our Label.

In summary, future work should focus on enriching data sources, integrating advanced analytical methods, and addressing current limitations to improve the predictive accuracy of stock performance models in complex and dynamic financial markets.

REFERENCES

[1] A. Thakkar and K. Chaudhari, "A comprehensive survey on deep neural networks for stock market: The need, challenges, and future directions," *Expert Systems with Applications*, vol. 177, p. 114800, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421002414

[2] T. C. Board. (2024, September) The conference board leading economic index® (lei) for the u.s. inched down further in august. [Accessed: 11 October 2024]. [Online]. Available: https://www.conference-board.org/topics/us-leading-indicators

[3] R. Aroussi, "*yfinance*: Yahoo! finance market data downloader," https://github.com/ranaroussi/yfinance, 2023, [Accessed: 10 October 2023].

[4] U. Securities and E. Commission. (2023) Edgar—how do i use edgar? [Accessed: 10 October 2023]. [Online]. Available: https://www.sec.gov/edgar/about