Anomaly Detection in Financial Data

Behrouz Banitalebi

November 11, 2024

Contents

List of Figures

List of Tables

1 Introduction

In the realm of financial services, accurate calculation of risk statistics, daily return indices, and ESG metrics hinges significantly on the quality of input data. Despite investments in rule-based quality testing, anomalies often evade detection, impacting the reliability of our outputs. This challenge underscores the necessity for robust anomaly detection methods that can effectively pinpoint data quality issues. This report delves into leveraging statistical techniques and modern machine learning approaches to enhance anomaly detection in financial data. To begin addressing the challenge, `yfinance` library is used to download adjusted closing prices for a set of assets from Yahoo Finance. For this exercise, JPMorgan Chase & Co. (JPM), Goldman Sachs Group Inc. (GS), Morgan Stanley (MS), BlackRock Inc. (BLK), and Citigroup Inc. (C), spanning from January 1, 2018, to November 30, 2023 have been considered. The following sections address the case study questions.

2 Preprocessing Steps

2.1 Data Visualization and Initial Preprocessing

Summary statistics for the stock prices of five companies, including measures such as the mean, standard deviation, minimum, maximum, and percentiles is provided in Table 1. These statistics give a snapshot of the central tendency, variability, and range of the stock prices over the dataset's period. Descriptive analysis is crucial for anomaly detection in time series data as it helps establish a baseline for normal behavior. By understanding the typical range and distribution of data, one can identify deviations that may indicate anomalies. For example, if a stock's price suddenly falls outside the usual range or deviates significantly from its historical volatility, it could signal an unusual event or an anomaly. This baseline understanding enables more effective monitoring and alerts for unusual patterns in time series data.

	BLK	C	GS	JPM	MS
Count	1488.0	1488.0	1488.0	1488.0	1488.0
Mean	567.8	52.1	256.4	114.1	61.0
Std	152.8	8.7	72.2	22.9	21.6
Min	294.3	30.0	121.5	69.3	24.2
25%	429.8	44.4	188.6	93.0	40.6
50% (Median)	571.5	53.0	233.6	110.7	57.3
75%	677.8	58.9	321.0	136.4	81.4
Max	902.3	71.0	393.0	158.9	99.9

Table 1: Summary Statistics for Stock Prices

- **Count**: The number of observations for each stock is 1488, indicating a complete dataset for the given period with no missing values.
- **Mean**: The average stock price for BLK is \$567.8, for C is \$52.1, for GS is \$256.4, for JPM is \$114.1, and for MS is \$61.0.
- **Standard Deviation (Std)**: This measures the amount of variation in stock prices. BLK has the highest standard deviation (\$152.8), suggesting it has the most variability, while MS has the lowest (\$21.6).
- **Min**: The minimum observed price for BLK is \$294.3, and for MS is \$24.2, showing the range of stock prices.
- **25% (Q1)**: This is the 25th percentile, indicating that 25% of the prices are below this value. For example, 25% of BLK prices are below \$429.8.
- **50% (Median)**: The median stock price for each stock represents the middle value in the distribution. For instance, the median price for BLK is \$571.5.
- **75% (Q3)**: This is the 75th percentile, showing that 75% of the prices are below this value. For example, 75% of BLK prices are below \$677.8.

- **Max**: The maximum observed price for BLK is $902.3, indicating the highest price recorded for this stock.

Visualizing the data also helps to understand its characteristics. Figure 1 illustrates the price movements of five assets over time. An initial observation reveals the presence of trends and seasonality in the time series data, which can complicate the detection of anomalies. While decomposition methods can be employed to eliminate trends and seasonality, a simpler approach is to compute the percentage change in the daily closing price of each stock, defined as follows:

$$\delta_i = 100 \times \frac{x_i - x_{i-1}}{x_{i-1}} \tag{1}$$

where x_i represents the multivariate vector price of p stocks on day i and x_{i-1} denotes the price on the

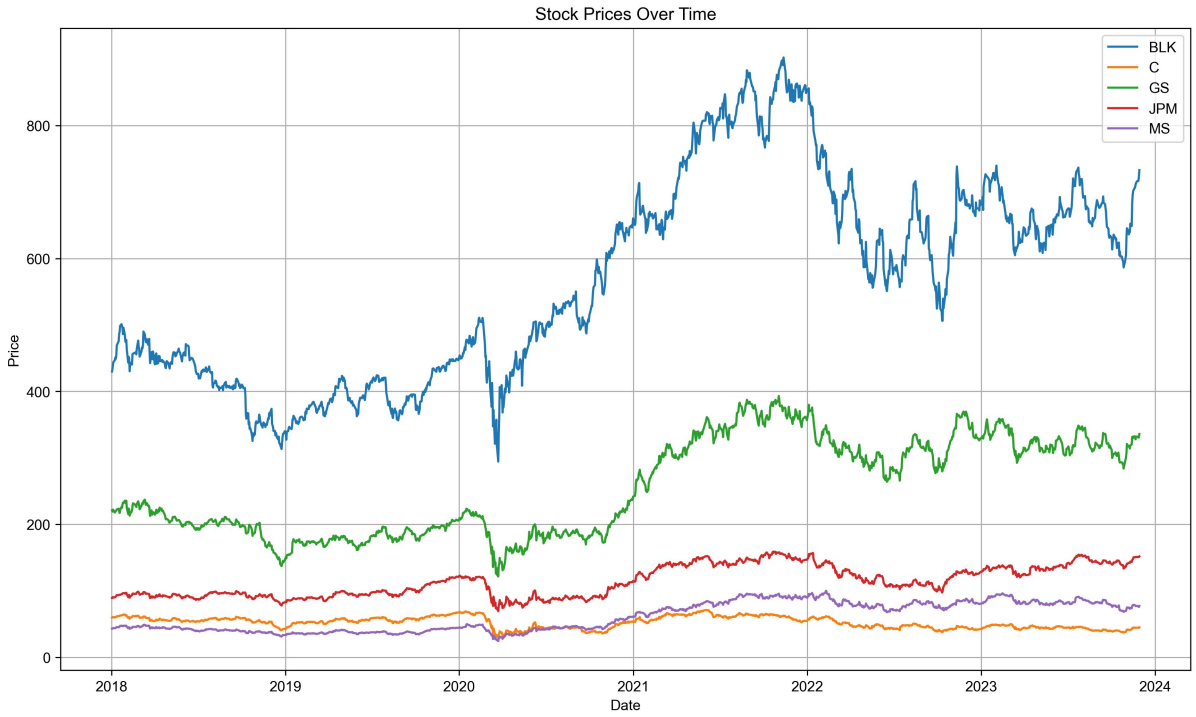

Figure 1: Stock prices timeseries data

previous day, $i-1$. The next step is to address any missing values in δ_i. The Carry Forward method is applied to fill in missing values; specifically, any missing value in each time series is replaced with the most recent non-missing value.

Collection of n samples of δ_i from n days creates an $n \times p$ matrix Δ. As Δ is a multivariate time series dataset, it is necessary to rescale each time series to enhance the accuracy of the analysis. Both scaling and normalization can be advantageous for anomaly detection in time series data, depending on the data characteristics and the specific requirements of the anomaly detection algorithm.

Scaling involves transforming data to a specific range, such as [0, 1] or [-1, 1]. It is particularly useful when data needs to be within a bounded range and is often employed for algorithms that are sensitive to the data scale, such as neural networks or those relying on distance metrics.

Normalization typically transforms data to have a mean of 0 and a standard deviation of 1, effectively standardizing it. This method is advantageous when data needs to be centered and scaled, which is important for algorithms that assume normally distributed data or where the scale of different features is significant, such as linear regression, support vector machines, or clustering algorithms.

Given that both distance-based algorithms and statistical methods are used in this analysis, normalizing $\Delta(t)$ is preferred.

2.2 Pair Plot and Q-Q Plot Analysis

Figure 2 presents the pair plot for the percentage changes in the daily closing prices of the five assets. Each scatter plot within the matrix shows the relationship between two different assets, while the diagonal

elements display the histograms of each asset's percentage changes. Supported by Table 2, the pair plot

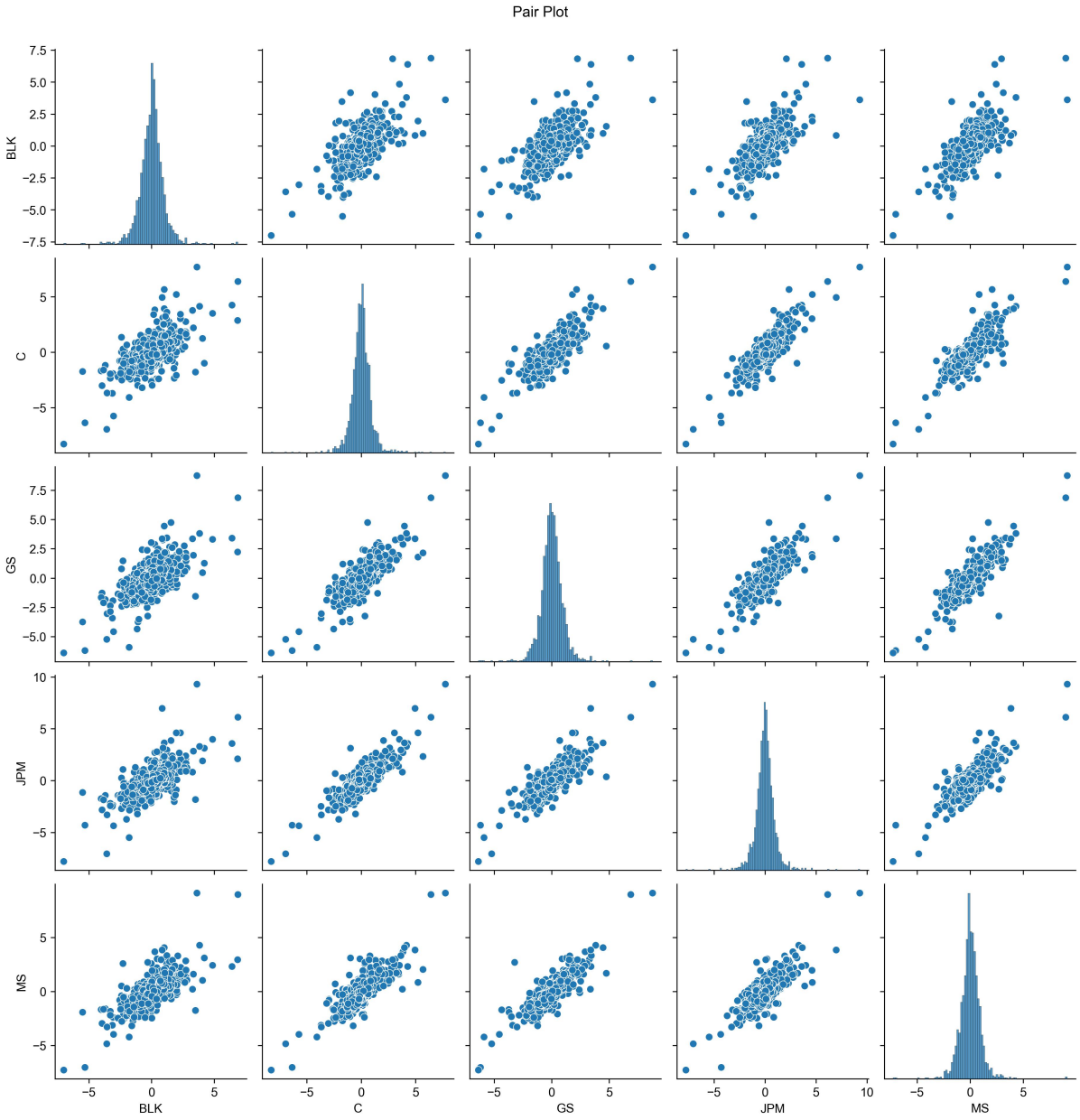

Figure 2: Stock prices pair plot.

indicates that there are strong linear relationships between the percentage changes of different assets, as evidenced by the tight clustering of points along the diagonals in the scatter plots. This suggests that the assets' price movements are correlated. The histograms along the diagonal reveal the distribution of each asset's percentage changes, which appear to be roughly normal but with some deviations, particularly in the tails.

Figure 3 shows the Q-Q plots for the percentage changes in the daily closing prices of the 5 assets. These plots compare the quantiles of the observed data to the quantiles of a normal distribution. The red line represents the theoretical quantiles from a standard normal distribution. The Q-Q plots for all five assets show a similar pattern: most of the data points lie along the red line, indicating that the percentage changes in daily closing prices are approximately normally distributed in the central region of the distribution. However, deviations from the red line are observed at the tails, suggesting the presence of outliers and heavy tails in the distribution. These deviations are especially pronounced for extreme values, indicating that the data might have more extreme changes than what is expected under a normal distribution. This observation is critical for anomaly detection as it highlights the potential for rare but

	BLK	C	GS	JPM	MS
BLK	1.000000	0.664580	0.686719	0.691056	0.722920
C	0.664580	1.000000	0.823280	0.869111	0.824168
GS	0.686719	0.823280	1.000000	0.828263	0.870732
JPM	0.691056	0.869111	0.828263	1.000000	0.834985
MS	0.722920	0.824168	0.870732	0.834985	1.000000

Table 2: Correlation Matrix of Asset Prices

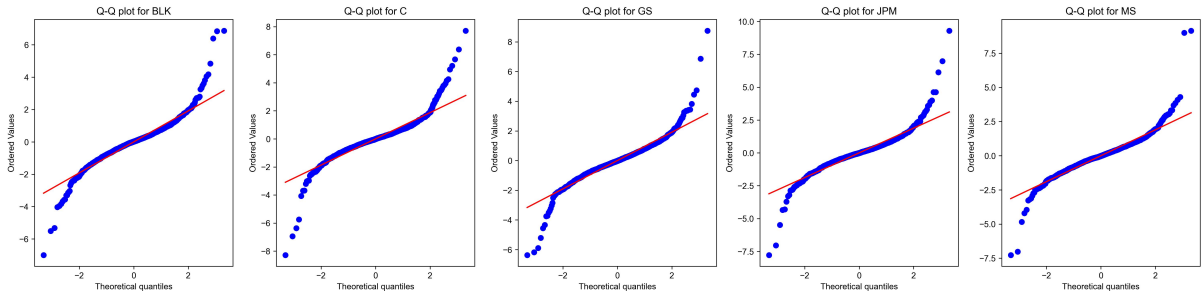

Figure 3: Q-Q plots for the percentage changes in daily closing prices of BLK, C, GS, JPM, and MS.

significant deviations that need to be accounted for in the analysis.

When combined with the pair plots from Figure 2, it can concluded that the percentage changes in the daily closing prices of the assets exhibit approximate normality in the central regions but display heavy tails and potential outliers. This indicates that while most of the data points are normally distributed, there are extreme values that deviate significantly from the normal distribution. These findings are critical for anomaly detection, as they highlight the presence of rare but significant deviations that may represent anomalies in the time series data. Properly accounting for these characteristics is essential for accurate anomaly detection.

2.3 Skewness and Kurtosis Analysis

Table 3 displays the skewness and kurtosis values for the percentage changes in the daily closing prices of the five assets. Skewness measures the asymmetry of the distribution, while kurtosis indicates the presence of outliers and the peakedness of the distribution. From the table, it can be observed that all assets exhibit positive skewness, with MS having the highest skewness value, suggesting a longer tail on the right side of the distribution. In terms of kurtosis, all assets show high values, significantly greater than 3 (which is the kurtosis of a normal distribution), indicating the presence of heavy tails or outliers. Particularly, JPM and MS have the highest kurtosis values, which suggests that their distributions are more prone to extreme values compared to the other assets.

Asset	Skewness	Kurtosis
BLK	0.1621	8.0480
C	0.0649	12.2953
GS	0.1918	9.8612
JPM	0.3786	13.2770
MS	0.5436	13.4952

Table 3: Skewness and Kurtosis of Asset Returns

3 Anomaly Detection Algorithms

3.1 IQR Method for Anomaly Detection

This section describes the Interquartile Range (IQR) method for anomaly detection. The IQR is a measure of statistical dispersion and is calculated as the difference between the third quartile ($Q3$) and

the first quartile ($Q1$) of the data:

$$\text{IQR} = Q3 - Q1$$

where:

- $Q1$ is the first quartile (25th percentile) of the data.
- $Q3$ is the third quartile (75th percentile) of the data.

Anomalies are detected by identifying data points that lie outside a specified range determined by the IQR. This range is typically set as:

$$\text{Lower Bound} = Q1 - k \cdot \text{IQR}$$
$$\text{Upper Bound} = Q3 + k \cdot \text{IQR}$$

where k is a constant that determines the sensitivity of the anomaly detection. In this implementation, k is set to 4, which means that any data point outside four times the IQR from the quartiles is considered an anomaly. Table 4 and red vertical dash lines in Figure 4 show the anomalies detected by IQR method.

Date	**BLK**	**C**	**GS**	**JPM**	**MS**
2019-01-16	357.6	50.9	172.7	87.3	37.4
2020-03-09	376.7	43.5	155.5	81.9	32.7
2020-03-10	412.6	47.1	165.9	88.2	34.4
2020-03-12	347.2	36.7	135.6	77.2	27.3
2020-03-13	372.0	43.2	159.4	91.1	32.7
2020-03-16	321.2	34.9	139.2	77.4	27.6
2020-03-18	335.6	30.9	126.0	73.5	26.8
2020-03-20	318.9	32.2	124.6	73.2	25.8
2020-03-24	334.2	34.4	138.2	77.5	28.9
2020-03-26	406.2	39.0	149.2	86.0	31.1
2020-04-06	404.1	34.8	142.4	79.2	32.2
2020-04-09	423.8	40.2	165.8	91.0	35.8
2020-04-17	428.7	38.5	165.1	84.3	34.0
2020-06-11	475.3	41.4	175.8	86.1	39.2
2020-11-09	606.4	41.7	195.8	105.5	49.9
2022-07-15	568.0	45.9	277.2	106.8	72.3
2022-11-10	724.2	45.4	359.6	128.9	83.9

Table 4: Anomalies Detected by the IQR Method

3.2 Rolling Median and MAD for Anomaly Detection

The Z-score method calculates the number of standard deviations a data point is from the mean:

$$Z = \frac{X - \mu}{\sigma} \tag{2}$$

where X is the data point, μ is the mean, and σ is the standard deviation of the data. This method assumes that the data follows a normal distribution. When the data is not normally distributed, the mean and standard deviation become unreliable, leading to incorrect Z-scores and poor anomaly detection.

Non-parametric methods like the rolling median and MAD do not assume a specific distribution for the data, making them more robust for anomaly detection in financial time series. Given an input matrix Δ of dimensions $n \times p$, where the i-th row is δ_i representing the percentage price changes for p stocks at time i, following metrics can be defined as:
Rolling median for a window size w :

$$\text{Rolling Median}(\delta_i) = \text{median}(\delta_{i-w+1}, \delta_{i-w+2}, \ldots, \delta_i) \tag{3}$$

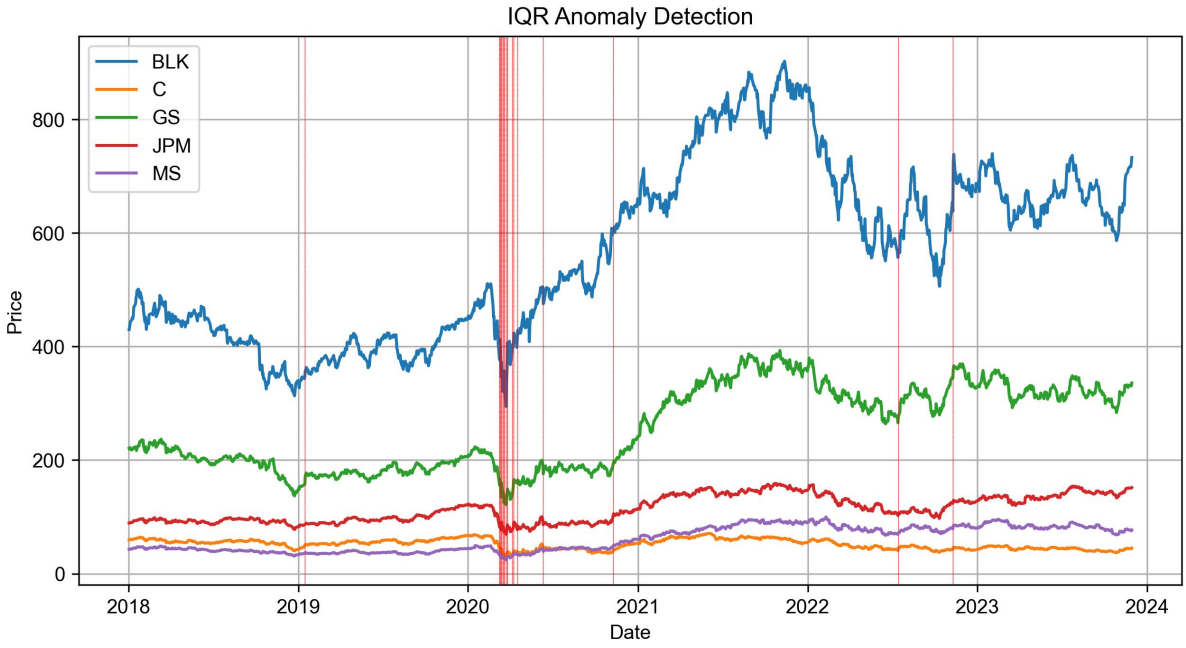

Figure 4: Anomaly Detection with IQR Method.

where δ_i is the i-th row of Δ.
Median Absolute Deviation:

$$\text{MAD} = \text{median}(|\delta_i - \text{median}(\delta)|) \tag{4}$$

where δ is the set of percentage price changes for a stock within a specified window and δ_i is an individual data point in that set. The modified Z-score for a data point δ_i is calculated using the rolling median and MAD:

$$\text{Modified Z-Score}(\delta_i) = \frac{1}{Q_3} \cdot \frac{\delta_i - \text{Rolling Median}(\delta_i)}{\text{MAD}(\delta_i)} \tag{5}$$

A threshold on the absolute value of the modified Z-score is used to identify anomalies. The algorithm to detect anomalies using rolling median and MAD is as follows:

1. Calculate the rolling median for each column of the data.
2. Calculate the rolling MAD for each column of the data.
3. Compute the modified Z-score for each data point.
4. Mark data points with an absolute modified Z-score greater than a specified threshold as anomalies.

The anomalies detected by the rolling statistics method are presented in Table 5 and Figure 5.

3.3 A Parametric Anomaly Detection Model

To detect anomalies in the multivariate time series dataset Δ the Mahalanobis distance, which accounts for correlations between variables, can be used. Given a dataset Δ with n observations and p variables:

$$\Delta = [\delta_1, \delta_2, \ldots, \delta_n]^T$$

where each δ_i is a p-dimensional vector. The covariance matrix $\mathbf{\Sigma}$ is calculated as:

$$\mathbf{\Sigma} = \frac{1}{n-1} \sum_{i=1}^{n} (\delta_i - \mu)(\delta_i - \mu)^T$$

where μ is the mean vector:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} \delta_i$$

Date	BLK	C	GS	JPM	MS
2019-01-16	357.6	50.9	172.7	87.3	37.4
2020-03-09	376.7	43.5	155.5	81.9	32.7
2020-03-10	412.6	47.1	165.9	88.2	34.4
2020-03-12	347.2	36.7	135.6	77.2	27.3
2020-03-13	372.0	43.2	159.4	91.1	32.7
2020-03-16	321.2	34.9	139.2	77.4	27.6
2020-03-18	335.6	30.9	126.0	73.5	26.8
2020-03-20	318.9	32.2	124.6	73.2	25.8
2020-03-24	334.2	34.4	138.2	77.5	28.9
2020-03-26	406.2	39.0	149.2	86.0	31.1
2020-04-09	423.8	40.2	165.8	91.0	35.8
2020-04-17	428.7	38.5	165.1	84.3	34.0
2022-07-15	568.0	45.9	277.2	106.8	72.3

Table 5: Anomalies Detected by the Rolling Statistics Method

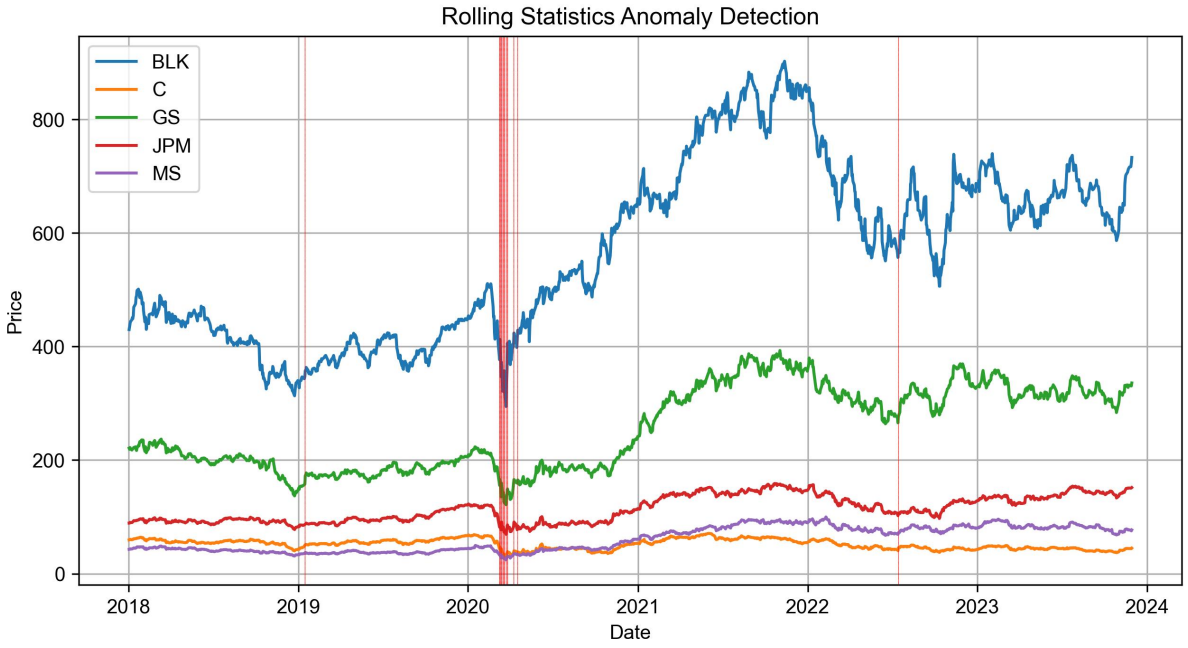

Figure 5: Anomaly Detection with Rolling Statistics Method.

The inverse of the covariance matrix $\mathbf{\Sigma}^{-1}$ is computed. This matrix will be used in calculating the Mahalanobis distance. The Mahalanobis distance for an observation $\delta_\mathbf{i}$ from the mean vector μ is defined as:

$$D_M(\delta_\mathbf{i}) = \sqrt{(\delta_\mathbf{i} - \mu)^T \mathbf{\Sigma}^{-1} (\delta_\mathbf{i} - \mu)}$$

This distance measures how many standard deviations away $\delta_\mathbf{i}$ is from the mean μ of the distribution, considering the correlations between variables. To determine the threshold for identifying anomalies, we use the chi-square distribution. For a p-dimensional dataset, the Mahalanobis distance squared follows a chi-square distribution with p degrees of freedom:

$$D_M^2 \sim \chi^2(p)$$

The 99th percentile of the chi-square distribution is used as the threshold:

$$\text{threshold} = \chi^2_{0.99}(p)$$

Observations with a Mahalanobis distance squared greater than this threshold are considered anomalies. Table 6 and Figure 6 show the anomalies in the stock prices based on the 99th percentile threshold in Mahalanobis distance method.

Date	BLK	C	GS	JPM	MS
2019-01-16	357.6	50.9	172.7	87.3	37.4
2020-03-09	376.7	43.5	155.5	81.9	32.7
2020-03-12	347.2	36.7	135.6	77.2	27.3
2020-03-13	372.0	43.2	159.4	91.1	32.7
2020-03-16	321.2	34.9	139.2	77.4	27.6
2020-03-17	347.7	34.1	142.8	82.2	29.4
2020-03-19	357.2	33.6	134.5	74.8	26.9
2020-03-20	318.9	32.2	124.6	73.2	25.8
2020-03-24	334.2	34.4	138.2	77.5	28.9
2020-03-26	406.2	39.0	149.2	86.0	31.1
2020-04-17	428.7	38.5	165.1	84.3	34.0
2020-11-09	606.4	41.7	195.8	105.5	49.9
2022-07-15	568.0	45.9	277.2	106.8	72.3
2022-11-10	724.2	45.4	359.6	128.9	83.9
2023-01-17	719.5	47.2	334.8	135.3	91.7

Table 6: Anomalies in stock prices based on Mahalanobis distance

3.4 VAR Anomaly Detection Model

Vector Autoregression (VAR) is a statistical model used to capture the linear interdependencies among multiple time series. The model can be used for forecasting and for anomaly detection by analyzing the forecast errors (residuals). The following steps outline the mathematical process behind the VAR anomaly detection model.

First, data preparation step ensures that data is stationary. In this case study, stationarity has already been achieved by calculating the Δ matrix, which is an $n \times p$ dimensional matrix where n is the number of historical data points and p is the number of stock prices. Then the VAR model is fitted to the stationary data. Let Y_t be a k-dimensional vector of time series at time t. The VAR model of order q is defined as:

$$Y_t = c + A_1 Y_{t-1} + A_2 Y_{t-2} + \ldots + AqY_{t-q} + \epsilon_t$$

where:

- c is a k-dimensional vector of constants.
- $A_1, A_2, \ldots, A_q$ are $k \times k$ coefficient matrices.
- ϵ_t is a k-dimensional vector of white noise error terms.

The optimal lag length q is selected using an information criterion such as the Akaike Information Criterion (AIC). The fitted VAR model is used to forecast future values:

$$\hat{Y}_{t+h} = c + A_1 \hat{Y}_{t+h-1} + A_2 \hat{Y}_{t+h-2} + \ldots + A_q \hat{Y}_{t+h-q}$$

where $\hat{Y}_{t+h}$ is the forecasted value at time $t+h$. The residuals, or forecast errors, are calculated as the difference between the actual and forecasted values:

$$\epsilon_t = Y_t - \hat{Y}_t$$

To detect anomalies, the 99th percentile threshold for each column of residuals is calculated:

$$\text{Threshold}_j = \text{Percentile}_{99}(|\epsilon_{t,j}|)$$

where $\epsilon_{t,j}$ represents the residuals of the j-th time series. Anomalies are detected by comparing the absolute residuals to the calculated thresholds:

$$\text{Anomaly}_{t,j} = \begin{cases} 1 & \text{if } |\epsilon_{t,j}| > \text{Threshold}_j \\ 0 & \text{otherwise} \end{cases}$$

Table 7 and 7 show the anomalies in the provided stock dataset detected by VAR model.

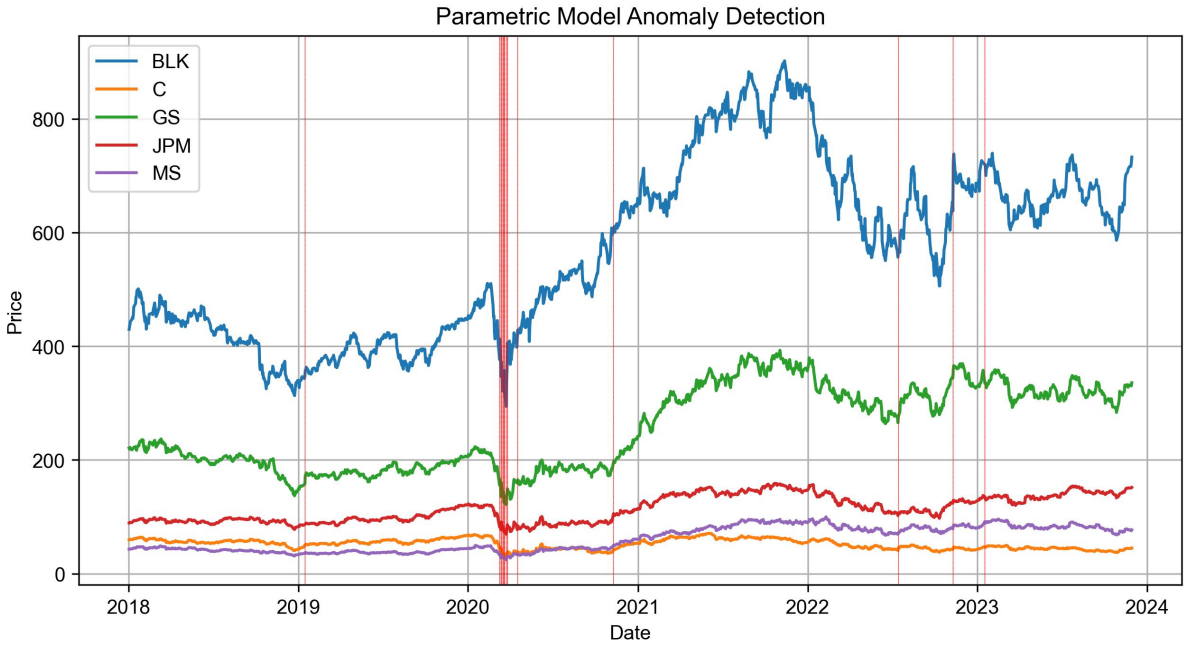

Figure 6: Anomaly Detection with a parametric model.

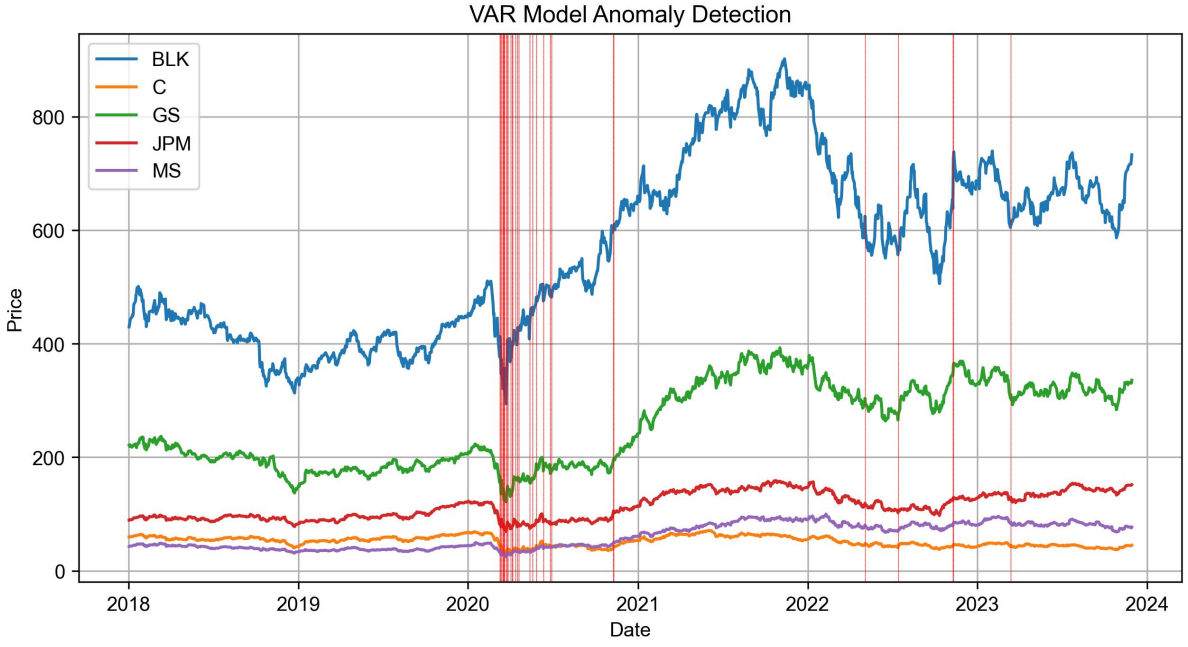

Figure 7: Anomaly Detection with VAR Method.

Date	Stock Value					Anomaly Status				
	BLK	C	GS	JPM	MS	BLK	C	GS	JPM	MS
2020-03-10	412.6	47.1	165.9	88.2	34.4	True	True	True	True	True
2020-03-11	387.4	43.0	154.7	84.1	32.1	True	True	True	True	True
2020-03-13	372.0	43.2	159.4	91.1	32.7	True	True	True	True	True
2020-03-16	321.2	34.9	139.2	77.4	27.6	True	True	True	True	True
2020-03-17	347.7	34.1	142.8	82.2	29.4	True	True	True	True	True
2020-03-18	335.6	30.9	126.0	73.5	26.8	True	False	True	True	True
2020-03-19	357.2	33.6	134.5	74.8	26.9	True	True	True	True	True
2020-03-20	318.9	32.2	124.6	73.2	25.8	True	False	True	False	False
2020-03-24	334.2	34.4	138.2	77.5	28.9	True	True	True	True	True
2020-03-25	360.8	35.5	139.6	80.4	29.6	False	False	True	False	True
2020-03-27	390.5	37.1	142.5	79.9	29.6	True	True	True	True	True
2020-04-02	383.4	33.2	134.9	76.7	29.5	True	False	False	False	True
2020-04-06	404.1	34.8	142.4	79.2	32.2	False	True	False	False	False
2020-04-07	398.4	34.9	149.4	80.3	32.2	False	False	False	False	True
2020-04-13	406.7	39.6	161.2	86.9	34.5	False	False	False	True	False
2020-04-17	428.7	38.5	165.1	84.3	34.0	False	True	False	True	False
2020-04-20	423.2	37.3	162.3	81.2	33.4	False	True	False	True	False
2020-05-13	436.7	34.8	154.6	74.4	32.5	True	False	False	False	False
2020-05-19	450.6	38.0	160.0	78.5	34.6	False	False	False	False	True
2020-05-28	482.6	42.1	181.1	88.4	39.0	False	True	True	False	True
2020-06-12	477.0	44.7	182.7	88.4	40.7	False	True	True	True	True
2020-06-26	484.0	42.5	171.3	82.0	41.3	False	False	True	False	False
2020-06-29	482.2	43.1	175.2	82.3	41.5	False	False	True	False	False
2020-11-09	606.4	41.7	195.8	105.5	49.9	False	False	False	True	False
2020-11-10	601.4	41.7	198.1	105.1	49.7	False	False	False	True	False
2022-05-05	595.3	47.1	293.5	116.2	78.7	True	False	False	False	False
2022-07-15	568.0	45.9	277.2	106.8	72.3	False	True	False	False	False
2022-11-10	724.2	45.4	359.6	128.9	83.9	True	False	False	False	False
2022-11-11	738.4	47.1	366.1	129.1	85.8	True	False	False	False	False
2023-03-14	616.5	44.9	310.4	129.4	85.7	False	True	False	False	False

Table 7: Anomaly Detection using VAR model

3.5 Kalman Filter Anomaly Detection

This section describes the mathematical foundation and implementation of the Kalman filter for anomaly detection, using the input matrix Δ, which is an $n \times p$ matrix where each column represents the percentage price change for a stock. There are p stocks in the matrix. The Kalman filter is used to estimate the state of a linear dynamic system in the presence of noise. The prediction step involves estimating the next state and its uncertainty based on the current state estimate and the process model:

$$\begin{aligned} \text{State prediction:} \quad & \mathbf{x}_{k|k-1} = \mathbf{F}\mathbf{x}_{k-1|k-1} \\ \text{Covariance prediction:} \quad & \mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^\top + \mathbf{Q} \end{aligned}$$

The update step involves correcting the prediction with the new measurement:

$$\begin{aligned} \text{Innovation (residual):} \quad & \mathbf{y}_k = \mathbf{z}_k - \mathbf{H}\mathbf{x}_{k|k-1} \\ \text{Innovation covariance:} \quad & \mathbf{S}_k = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\top + \mathbf{R} \\ \text{Kalman gain:} \quad & \mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^\top\mathbf{S}_k^{-1} \\ \text{State update:} \quad & \mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{K}_k\mathbf{y}_k \\ \text{Covariance update:} \quad & \mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\,\mathbf{P}_{k|k-1} \end{aligned}$$

The Mahalanobis distance is used to determine how far each residual is from the mean residuals, considering the covariance structure of the residuals. The Mahalanobis distance is defined as:

$$D_M(\mathbf{r}) = \sqrt{(\mathbf{r} - \boldsymbol{\mu}_r)^\top \mathbf{C}_r^{-1} (\mathbf{r} - \boldsymbol{\mu}_r)}$$

where:

- $\mathbf{r}$ is the residual.
- $\boldsymbol{\mu}_r$ is the mean of the residuals.
- $\mathbf{C}_r$ is the covariance matrix of the residuals.
- $\mathbf{C}_r^{-1}$ is the inverse of the covariance matrix.

Anomalies are detected by comparing the Mahalanobis distance to a threshold, which can be set based on the 99th percentile:

$$\text{Anomaly} = \begin{cases} 1 & \text{if } D_M > \text{threshold} \\ 0 & \text{otherwise} \end{cases}$$

Price anomalies detected by the Kalman Filter are presented in Table 8 and Figure 8.

Date	**BLK**	**C**	**GS**	**JPM**	**MS**
2019-01-16	357.6	50.9	172.7	87.3	37.4
2020-03-10	412.6	47.1	165.9	88.2	34.4
2020-03-13	372.0	43.2	159.4	91.1	32.7
2020-03-16	321.2	34.9	139.2	77.4	27.6
2020-03-17	347.7	34.1	142.8	82.2	29.4
2020-03-18	335.6	30.9	126.0	73.5	26.8
2020-03-19	357.2	33.6	134.5	74.8	26.9
2020-03-20	318.9	32.2	124.6	73.2	25.8
2020-03-24	334.2	34.4	138.2	77.5	28.9
2020-04-17	428.7	38.5	165.1	84.3	34.0
2020-05-13	436.7	34.8	154.6	74.4	32.5
2020-06-12	477.0	44.7	182.7	88.4	40.7
2020-11-09	606.4	41.7	195.8	105.5	49.9
2022-07-15	568.0	45.9	277.2	106.8	72.3
2023-01-17	719.5	47.2	334.8	135.3	91.7

Table 8: Anomalies in stock prices based on Kalman Filter

3.6 K-Nearest Neighbors (KNN) for Anomaly Detection

K-Nearest Neighbors (KNN) is a simple, yet powerful algorithm used for various tasks in machine learning, including classification, regression, and anomaly detection. In the context of anomaly detection, KNN identifies outliers by examining the distances between data points in a multi-dimensional space. KNN is a non-parametric method that classifies a data point based on how its neighbors are classified. The core idea is that data points that are close to each other in the feature space are likely to have similar properties. Given a dataset, KNN computes the distance between each data point and its k-nearest neighbors. The choice of k can affect the performance of the algorithm. For anomaly detection, the distance to the k-th nearest neighbor can be used as an anomaly score: larger distances indicate potential anomalies.

Let δ_i denote the i-th row of the data matrix Δ, representing the feature vector of the i-th data point. The anomaly score for δ_i is computed as follows:

$$\text{Anomaly score}(\delta_i) = \max_{j \in \{1,\ldots,k\}} d(\delta_i, \delta_j) \tag{6}$$

where $d(\delta_i, \delta_j)$ is the Euclidean distance between δ_i and its j-th nearest neighbor. This score is then used to rank the data points, with higher scores indicating potential anomalies. Table 9 and Figure 9 show anomalies based on the 99th percentile threshold.

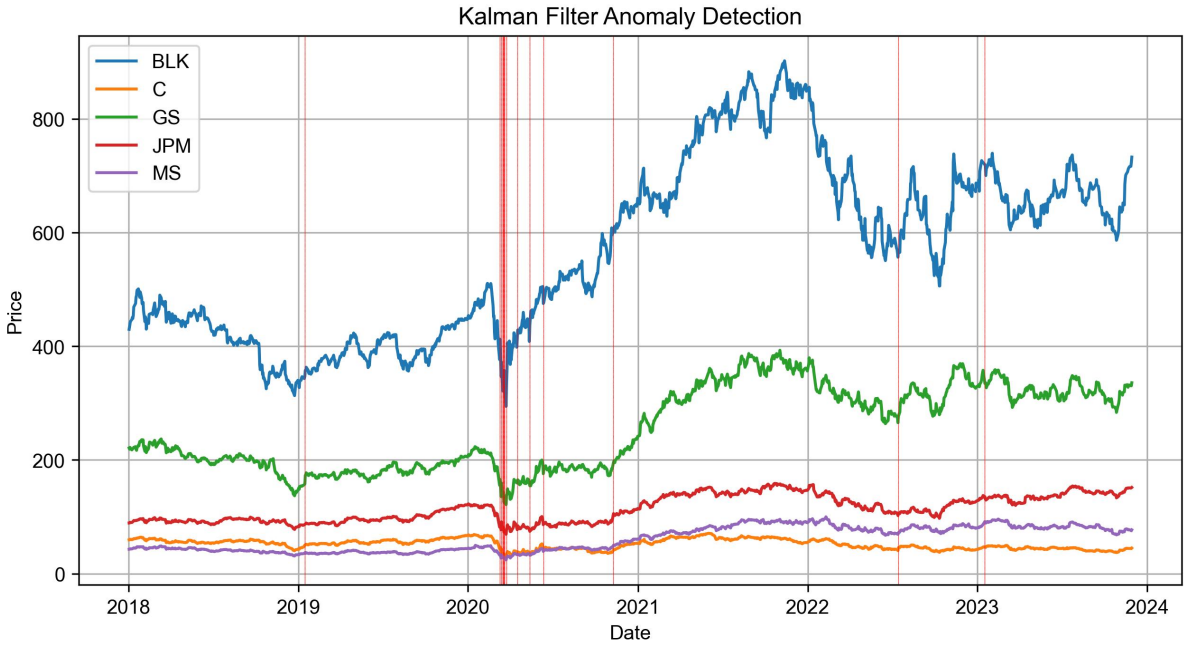

Figure 8: Anomaly Detection with Kalman Filter.

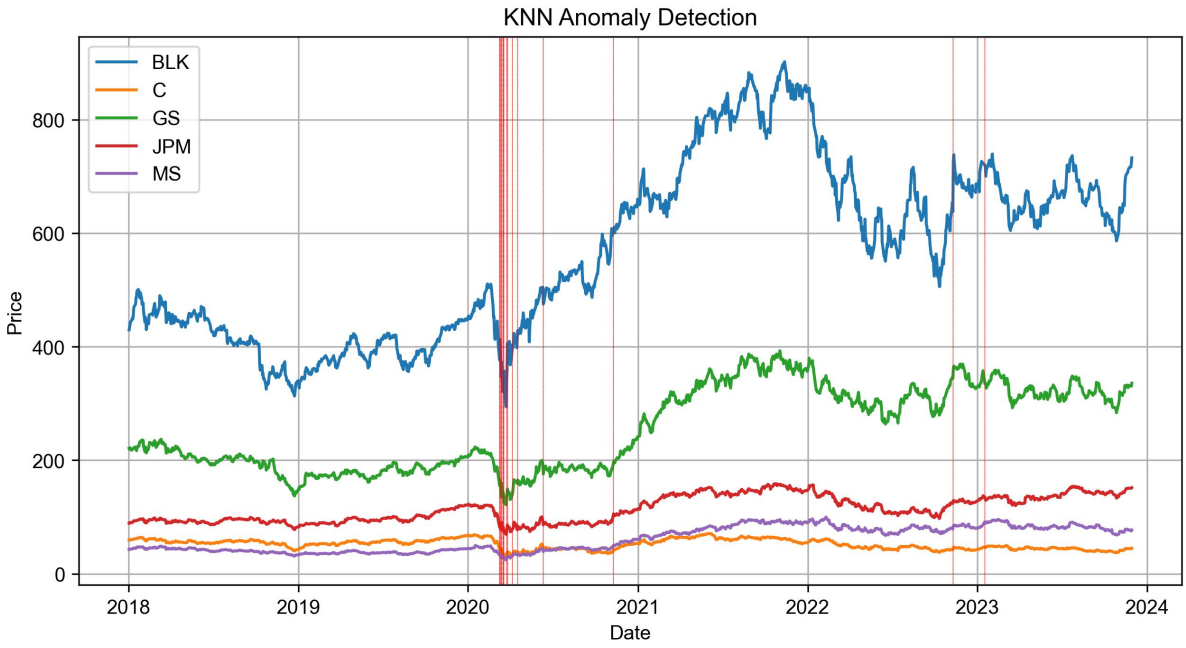

Figure 9: Anomaly Detection with KNN Method.

Date	BLK	C	GS	JPM	MS
2020-03-09	376.7	43.5	155.5	81.9	32.7
2020-03-10	412.6	47.1	165.9	88.2	34.4
2020-03-12	347.2	36.7	135.6	77.2	27.3
2020-03-13	372.0	43.2	159.4	91.1	32.7
2020-03-16	321.2	34.9	139.2	77.4	27.6
2020-03-17	347.7	34.1	142.8	82.2	29.4
2020-03-18	335.6	30.9	126.0	73.5	26.8
2020-03-24	334.2	34.4	138.2	77.5	28.9
2020-03-26	406.2	39.0	149.2	86.0	31.1
2020-04-06	404.1	34.8	142.4	79.2	32.2
2020-04-17	428.7	38.5	165.1	84.3	34.0
2020-06-11	475.3	41.4	175.8	86.1	39.2
2020-11-09	606.4	41.7	195.8	105.5	49.9
2022-11-10	724.2	45.4	359.6	128.9	83.9
2023-01-17	719.5	47.2	334.8	135.3	91.7

Table 9: Results of KNN Anomaly Detection with $k = 4$ Nearest Neighbors

3.7 Anomaly Detection Using Isolation Forest

Isolation Forest is an unsupervised learning algorithm specifically designed for anomaly detection. The basic principle of the Isolation Forest algorithm is that anomalies are few and different from the majority of the data points, making them easier to isolate. The algorithm isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. The process is repeated recursively until all data points are isolated. Isolation Forest does not require a normal distribution in the data which makes it a robust method for anomaly detection. The Isolation Forest algorithm builds an ensemble of isolation trees (iTrees), each constructed by recursively partitioning the data. The key steps in the algorithm are as follows:

1. **Random Subsampling:** A random subset of the data is selected without replacement.

2. **Recursive Partitioning:** For each subset, an isolation tree is built by randomly selecting a feature and a split value until:

- Each data point is isolated.
- A predefined maximum tree depth is reached.

3. **Path Length Calculation:** The number of splits required to isolate a data point corresponds to the path length from the root node to the terminating node. The path length for a data point x is denoted as $h(x)$.

The anomaly score for a data point is based on the path length. Since anomalies are isolated closer to the root of the tree, they tend to have shorter path lengths. The anomaly score is calculated as follows:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{7}$$

where:

- $s(x, n)$ is the anomaly score of a data point x given n data points.
- $E(h(x))$ is the average path length of x across the isolation trees.
- $c(n)$ is the average path length of unsuccessful searches in a Binary Search Tree, given by:

$$c(n) = 2H(n-1) - \left(\frac{2(n-1)}{n}\right) \tag{8}$$

where $H(i)$ is the i-th harmonic number, approximated as $H(i) \approx \ln(i) + 0.5772156649$ (Euler's constant).

The anomaly score ranges between 0 and 1. A score close to 1 indicates that the data point is more likely to be an anomaly, while a score close to 0 indicates that it is more likely to be normal. Table 10 and Figure 10 show the anomalies detected by the Isolation Forest method.

Date	BLK	C	GS	JPM	MS
2020-03-09	376.7	43.5	155.5	81.9	32.7
2020-03-10	412.6	47.1	165.9	88.2	34.4
2020-03-11	387.4	43.0	154.7	84.1	32.1
2020-03-12	347.2	36.7	135.6	77.2	27.3
2020-03-13	372.0	43.2	159.4	91.1	32.7
2020-03-16	321.2	34.9	139.2	77.4	27.6
2020-03-18	335.6	30.9	126.0	73.5	26.8
2020-03-23	294.3	30.0	121.5	69.3	24.2
2020-03-24	334.2	34.4	138.2	77.5	28.9
2020-03-26	406.2	39.0	149.2	86.0	31.1
2020-04-01	368.2	32.6	130.8	73.9	27.5
2020-04-06	404.1	34.8	142.4	79.2	32.2
2020-05-26	470.7	41.2	176.4	84.8	38.4
2020-06-11	475.3	41.4	175.8	86.1	39.2
2020-11-09	606.4	41.7	195.8	105.5	49.9

Table 10: Anomalies Detected by Isolation Forest Method

4 Supervised and Unsupervised Learning

Supervised learning is a type of machine learning where the algorithm is trained on a labeled dataset, meaning each training example is paired with an output label. The goal is to learn a mapping from inputs to outputs to predict the output for new, unseen inputs. Common supervised learning tasks include classification, where inputs are assigned to discrete classes, and regression, where a continuous output value is predicted. Algorithms such as linear regression, logistic regression, decision trees, and support vector machines are widely used in supervised learning. Performance is typically evaluated using metrics like accuracy, precision, recall, and mean squared error.

Unsupervised learning involves training algorithms on data without labeled responses. The main objective is to identify patterns, structures, or relationships in the data. This is useful for tasks such as clustering, where similar data points are grouped together, and dimensionality reduction, which reduces the number of features while retaining essential characteristics. Common algorithms include k-means clustering, hierarchical clustering, principal component analysis (PCA), and t-distributed stochastic neighbor embedding (t-SNE). The performance of unsupervised learning algorithms is often more subjective and may require domain knowledge to interpret effectively.

The fundamental difference between supervised and unsupervised learning lies in the presence of labeled data. Supervised learning uses labeled data to train algorithms for predictions or classification, whereas unsupervised learning works with unlabeled data to uncover hidden patterns or structures without predefined labels. Consequently, supervised learning is generally more straightforward to evaluate due to the availability of ground truth labels, while unsupervised learning often requires more interpretation and domain expertise to understand the discovered patterns and their relevance to the problem at hand.

5 Using AutoEncoder for Anomaly Detection

5.1 Dense Model

An autoencoder is a type of artificial neural network used for learning efficient codings of input data. It consists of two main parts: the encoder, which compresses the input data into a lower-dimensional representation, and the decoder, which reconstructs the input data from the compressed representation. Autoencoders are particularly useful for anomaly detection because they can learn the normal patterns of the data during training and identify deviations from these patterns as anomalies.

A simple autoencoder can be defined by an architecture with an input layer, one or more hidden layers (encoder), and an output layer (decoder). Let X be the input data matrix with n samples and p features. The autoencoder aims to minimize the reconstruction error, which is the difference between

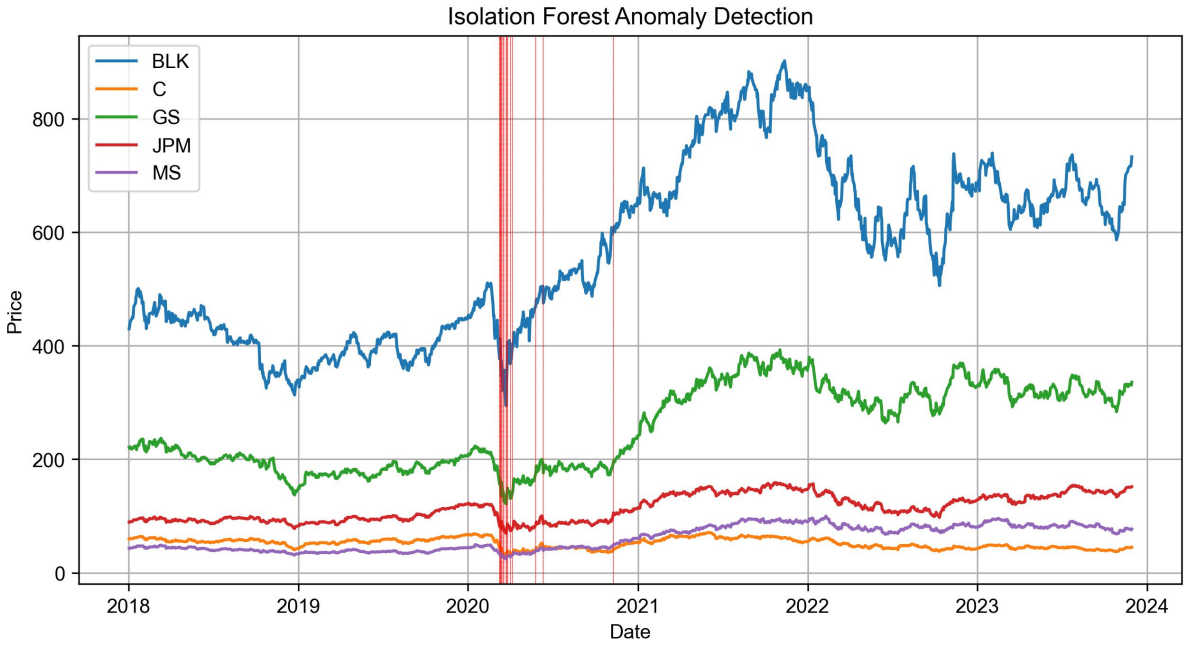

Figure 10: Anomaly Detection with the Isolation Forest Method.

the input data and the reconstructed data. Mathematically, this can be expressed as:

$$\hat{\Delta} = f(g(\Delta)), \tag{9}$$

where g is the encoder function and f is the decoder function. The objective is to minimize the mean squared error (MSE) between Δ and $\hat{\Delta}$:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} \|\delta_i - \hat{\delta}_i\|^2. \tag{10}$$

In this case study, the first autoencoder is defined with an input layer, a hidden layer with 10 neurons and ReLU activation function, and an output layer with linear activation function. The model is trained using the Adam optimizer to minimize the mean squared error between the input and reconstructed data. The training process involves adjusting the weights of the network to reduce the reconstruction error over multiple epochs.

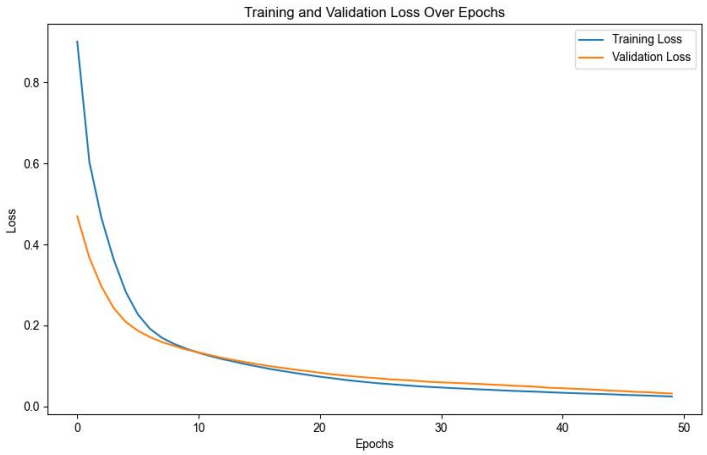

Figure 11: Training and Validation Loss Over Epochs

The plot of training and validation loss over epochs, as shown in Figure 11, provides valuable insights into the learning process of the autoencoder model. Initially, both the training and validation losses decrease rapidly, indicating that the model is effectively learning to reconstruct the input data. As the training progresses, the rate of decrease in loss slows down and eventually plateaus, suggesting that the model is converging towards an optimal solution. The close proximity of the training and validation loss curves throughout the training process implies that the model is not overfitting, as there is no significant divergence between the two. This behavior demonstrates that the autoencoder is generalizing well to unseen data, effectively capturing the underlying patterns without memorizing the training data. After training the autoencoder, the model is used to reconstruct the input data. The reconstruction error for each sample is calculated as the mean squared error between the original and reconstructed data:

$$\text{Reconstruction Error}_i = \|\delta_i - \hat{\delta}_i\|^2. \tag{11}$$

To detect anomalies, a threshold is set based on the 99th percentile of the reconstruction errors. Any sample with a reconstruction error greater than this threshold is classified as an anomaly as provided in Table 11 and Figure 12.

Date	BLK	C	GS	JPM	MS
2018-11-12	348.4	52.6	179.9	92.0	37.1
2019-01-16	357.6	50.9	172.7	87.3	37.4
2020-03-09	376.7	43.5	155.5	81.9	32.7
2020-03-17	347.7	34.1	142.8	82.2	29.4
2020-03-18	335.6	30.9	126.0	73.5	26.8
2020-03-23	294.3	30.0	121.5	69.3	24.2
2020-04-15	398.3	36.3	160.7	80.4	33.4
2020-05-13	436.7	34.8	154.6	74.4	32.5
2020-06-11	475.3	41.4	175.8	86.1	39.2
2020-06-26	484.0	42.5	171.3	82.0	41.3
2020-11-09	606.4	41.7	195.8	105.5	49.9
2022-01-14	791.8	60.4	355.1	146.9	90.2
2023-01-17	719.5	47.2	334.8	135.3	91.7
2023-01-20	711.8	48.0	327.0	129.8	90.9
2023-10-18	604.8	39.2	295.6	143.4	72.6

Table 11: Anomalies detected by the Dense model.

5.2 LSTM Autoencoder Anomaly Detection Algorithm

The LSTM autoencoder anomaly detection algorithm is a method used to identify unusual patterns or anomalies in time series data by leveraging the reconstructive capabilities of LSTM networks. LSTM requires an additional data preparation in which the input time series data is reshaped into a 3D array to fit the LSTM network requirements, where the dimensions represent samples, timesteps, and features. The encoder part of the autoencoder reduces the input time series data to a fixed-size latent space representation. Mathematically, if the input data is $\Delta \in \mathbb{R}^{T \times F}$, where T is the number of timesteps and F is the number of features, the encoder maps Δ to $Z \in \mathbb{R}^d$ (latent space) using an LSTM network, where d is the dimensionality of the latent space:

$$Z = \text{LSTM}_{\text{encoder}}(\Delta)$$

The decoder reconstructs the input data from the latent space representation. It takes Z and maps it back to $\hat{\Delta} \in \mathbb{R}^{T \times F}$ using another LSTM network:

$$\hat{\Delta} = \text{LSTM}_{\text{decoder}}(Z)$$

The autoencoder is trained to minimize the reconstruction loss, which is the mean squared error (MSE) between the input Δ and the reconstructed output $\hat{\Delta}$:

$$\text{Loss} = \frac{1}{T \times F} \sum_{t=1}^{T} \sum_{f=1}^{F} (\Delta_{t,f} - \hat{\Delta}_{t,f})^2$$

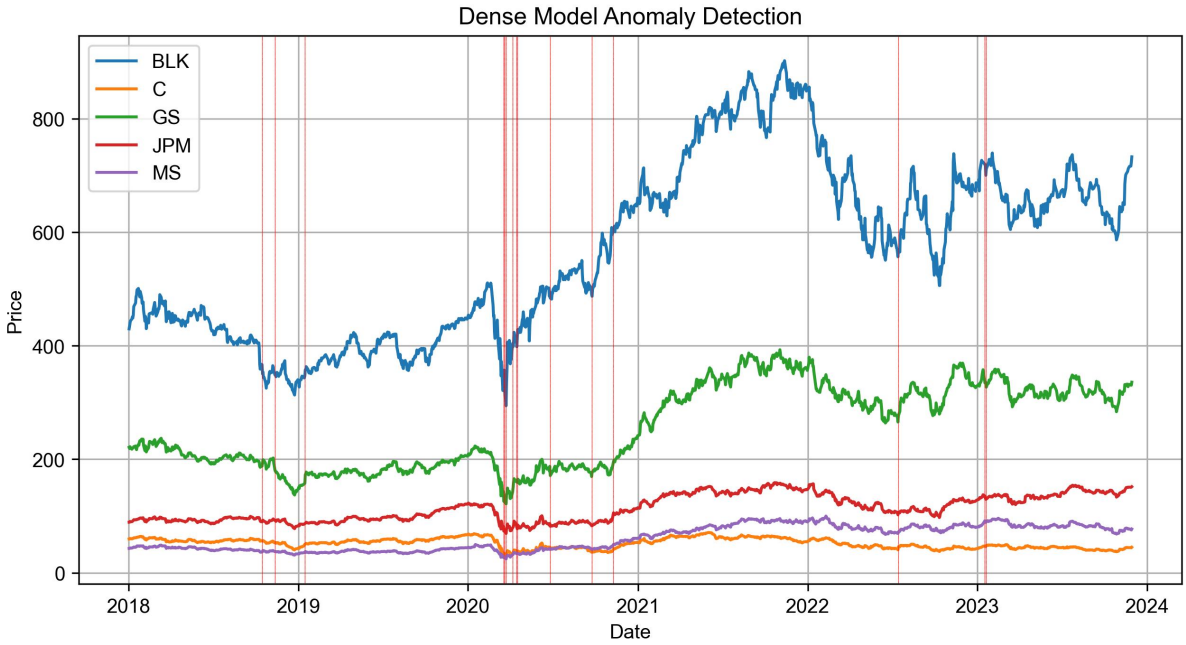

Figure 12: Anomaly Detection with Autoencoder Dense Model.

The training and validation loss over epochs is illustrated in the plot saved as 13. This plot shows the convergence of the model during training After training, the reconstruction error is calculated for each

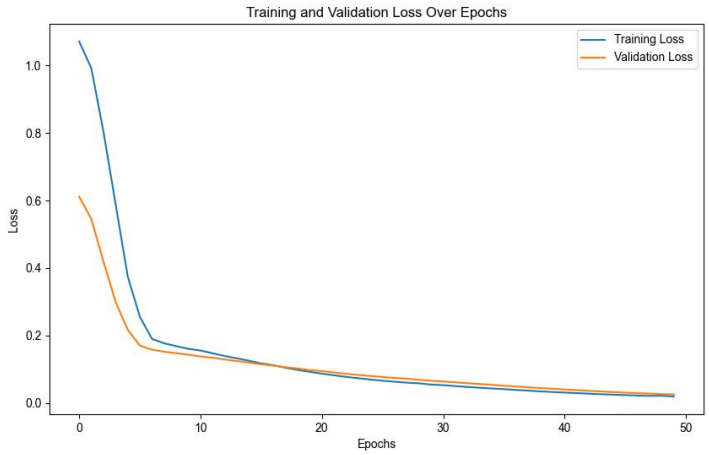

Figure 13: Training and Validation Loss Over Epochs for LSTM model

sample in the dataset. Anomalies are detected by comparing the reconstruction error to a predefined threshold. If the error exceeds the threshold, the sample is classified as an anomaly.

Given a sequence of input data Δ, the LSTM autoencoder model is trained to reconstruct Δ as accurately as possible. The reconstruction process can be broken down into the following steps:

Encoding: The input sequence Δ is processed by the LSTM encoder to produce the latent representation Z:

$$Z = \text{LSTM}_{\text{encoder}}(\Delta)$$

The encoder LSTM transforms the input sequence through a series of hidden states, capturing temporal dependencies and reducing dimensionality.

Decoding: The latent representation Z is fed into the LSTM decoder, which reconstructs the original sequence:

$$\hat{\Delta} = \text{LSTM}_{\text{decoder}}(Z)$$

The decoder LSTM uses the latent representation to generate an output sequence that approximates the input sequence.

Reconstruction Error: The reconstruction error for each sample is calculated as the MSE between the original and reconstructed sequences:

$$\text{MSE}(\Delta, \hat{\Delta}) = \frac{1}{T \times F} \sum_{t=1}^{T} \sum_{f=1}^{F} (\Delta_{t,f} - \hat{\Delta}_{t,f})^2$$

Thresholding: A threshold is set based on the distribution of reconstruction errors. Samples with reconstruction errors exceeding the threshold are flagged as anomalies:

$$\text{Anomaly} = \begin{cases} 1 & \text{if MSE}(\Delta, \hat{\Delta}) > \text{threshold} \\ 0 & \text{otherwise} \end{cases}$$

The anomalies are listed in Table 12 and presented in 14, showing the dates and corresponding stock prices that were flagged as unusual.

Date	BLK	C	GS	JPM	MS
2018-10-16	350.6	56.7	193.5	91.7	38.4
2019-01-16	357.6	50.9	172.7	87.3	37.4
2020-01-16	477.3	68.2	223.4	120.3	48.8
2020-03-13	372.0	43.2	159.4	91.1	32.7
2020-03-17	347.7	34.1	142.8	82.2	29.4
2020-03-19	357.2	33.6	134.5	74.8	26.9
2020-03-20	318.9	32.2	124.6	73.2	25.8
2020-03-24	334.2	34.4	138.2	77.5	28.9
2020-09-14	498.3	41.6	183.4	91.6	45.3
2021-01-15	665.9	56.2	275.7	126.0	67.1
2022-01-14	791.8	60.4	355.1	146.9	90.2
2023-01-17	719.5	47.2	334.8	135.3	91.7
2023-03-10	613.8	45.8	315.7	128.5	85.8
2023-03-13	604.9	42.4	304.0	126.2	83.8
2023-04-14	668.2	47.0	324.6	134.4	82.6

Table 12: Detected anomalies in stock prices detected by LSTM autoencoder

The two autoencoders used initially have a very simple architecture with a single layer containing 10 neurons. These models provided relatively small validation losses, ranging between 0.03 and 0.05. However, a more complex model can better capture the trends in the time series data. By adding more neurons and layers to the model, the non-linear relationships between the stock prices and their lagged values can be captured more effectively. To validate this statement, a complex autoencoder with two LSTM layers, each containing 128 units, has been implemented. The Figure 15 shows that the validation loss decreased to 0.0007. This model is used to predict the price changes in the dataset, and the residuals above the 99th percentile are identified as outliers which are shown in Figure 16 and listed in Table 13.

6 Conclusion

When using any time series data for daily, weekly, or monthly production, it is crucial to ensure there are no significant increases or drops in the input data. Such irregularities can impact the output data and lead to lower-quality products. To maintain consistent historical output data, anomalies in the input data must be detected and appropriately treated. The notebooks in this repository implement various statistical and machine learning methods to identify the best approach for each use case.

The results indicate that both statistical and machine learning methods are effective for anomaly detection. The key advantage of ML-based models in data production is their ability to be trained to capture anomalies and then reused for multiple data products. This eliminates the need for repeated

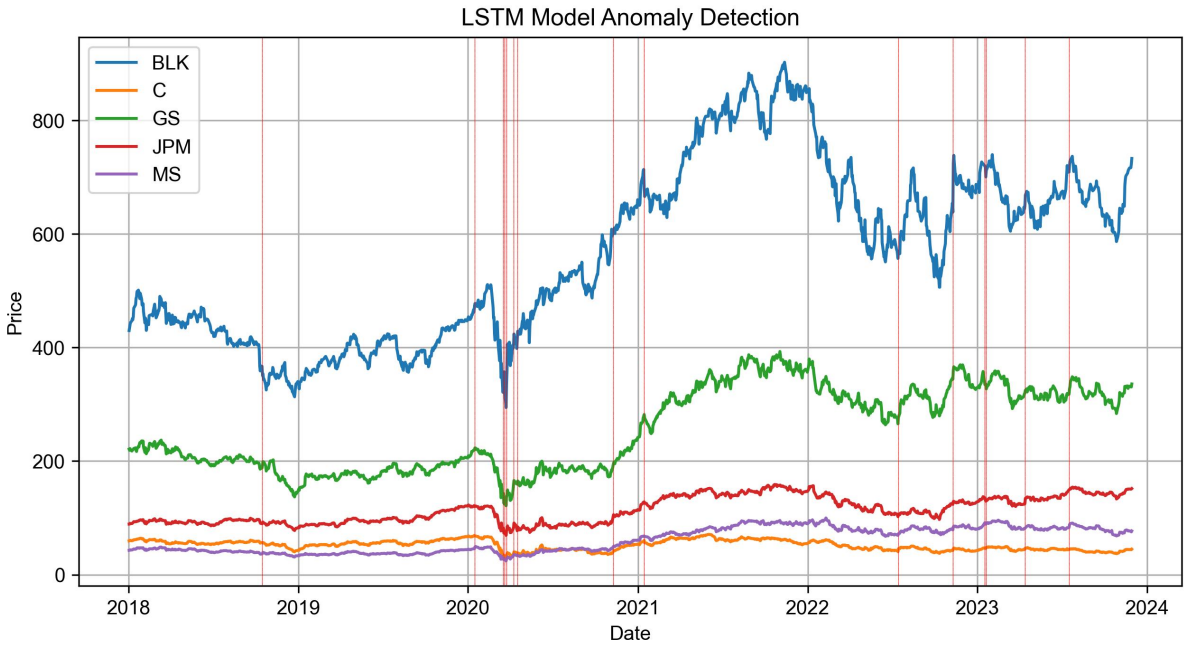

Figure 14: Anomaly Detection with LSTM Autoencoder.

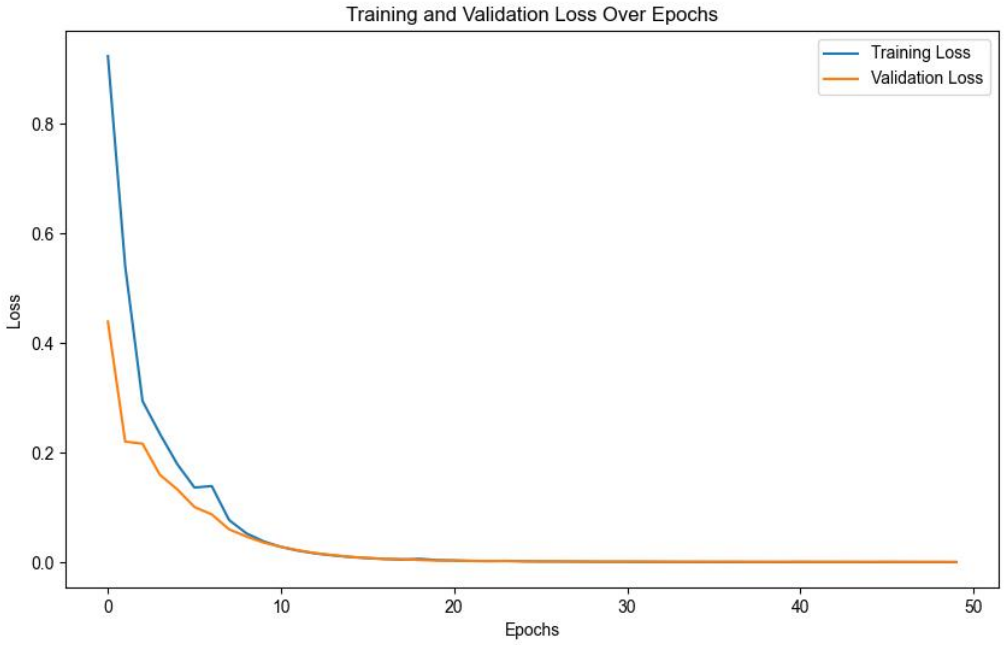

Figure 15: Training and Validation Loss in the Complex LSTM Model.

Date	BLK	C	GS	JPM	MS
2020-03-09	376.7	43.5	155.5	81.9	32.7
2020-03-10	412.6	47.1	165.9	88.2	34.4
2020-03-12	347.2	36.7	135.6	77.2	27.3
2020-03-13	372.0	43.2	159.4	91.1	32.7
2020-03-16	321.2	34.9	139.2	77.4	27.6
2020-03-18	335.6	30.9	126.0	73.5	26.8
2020-03-24	334.2	34.4	138.2	77.5	28.9
2020-03-26	406.2	39.0	149.2	86.0	31.1
2020-04-01	368.2	32.6	130.8	73.9	27.5
2020-04-06	404.1	34.8	142.4	79.2	32.2
2020-04-17	428.7	38.5	165.1	84.3	34.0
2020-05-26	470.7	41.2	176.4	84.8	38.4
2020-06-11	475.3	41.4	175.8	86.1	39.2
2020-11-09	606.4	41.7	195.8	105.5	49.9
2022-11-10	724.2	45.4	359.6	128.9	83.9

Table 13: Detected anomalies in stock prices detected by the Complex LSTM Model

data analysis, allowing practitioners to deploy the model immediately. However, ML-based models often operate as black boxes, making it difficult to fully understand how they detect anomalies and why certain data points are flagged. This lack of transparency can present challenges when explaining anomalies to data providers, as solid justifications are necessary. Additionally, while ML-based methods offer flexibility in adjusting thresholds for anomaly detection, such adjustments are often challenging to justify and may require trial and error.

Conversely, statistical methods are less generalizable compared to ML-based models. A statistical approach that works for one use case may not be effective for another. For example, using a Z-score to detect anomalies may work well in one dataset but perform poorly in another with heavy tails, as Z-scores are not optimal for such distributions. Nevertheless, statistical methods offer easier threshold adjustment and clearer justification of results, making them advantageous in certain scenarios.

In summary, the choice between statistical and machine learning methods for anomaly detection depends on the specific use case, the nature of the data, and the need for transparency and flexibility.

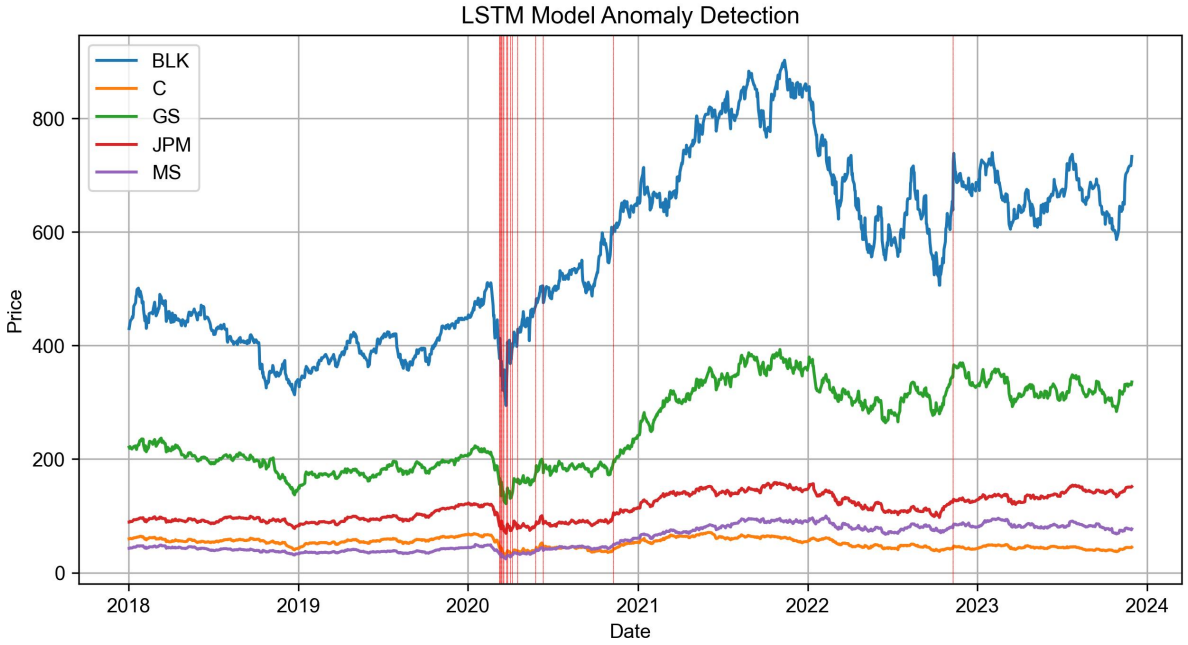

Figure 16: Anomaly Detection with the Complex LSTM Model.