IT3212 Assignment 1: Data Preprocessing

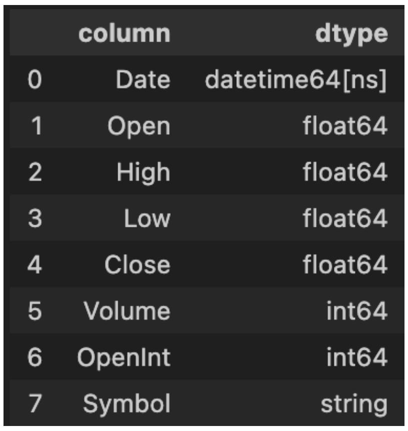
Table of Contents

1. Data Exploration

a. Explore the dataset by displaying the first few rows, summary statistics, and data types of each column.

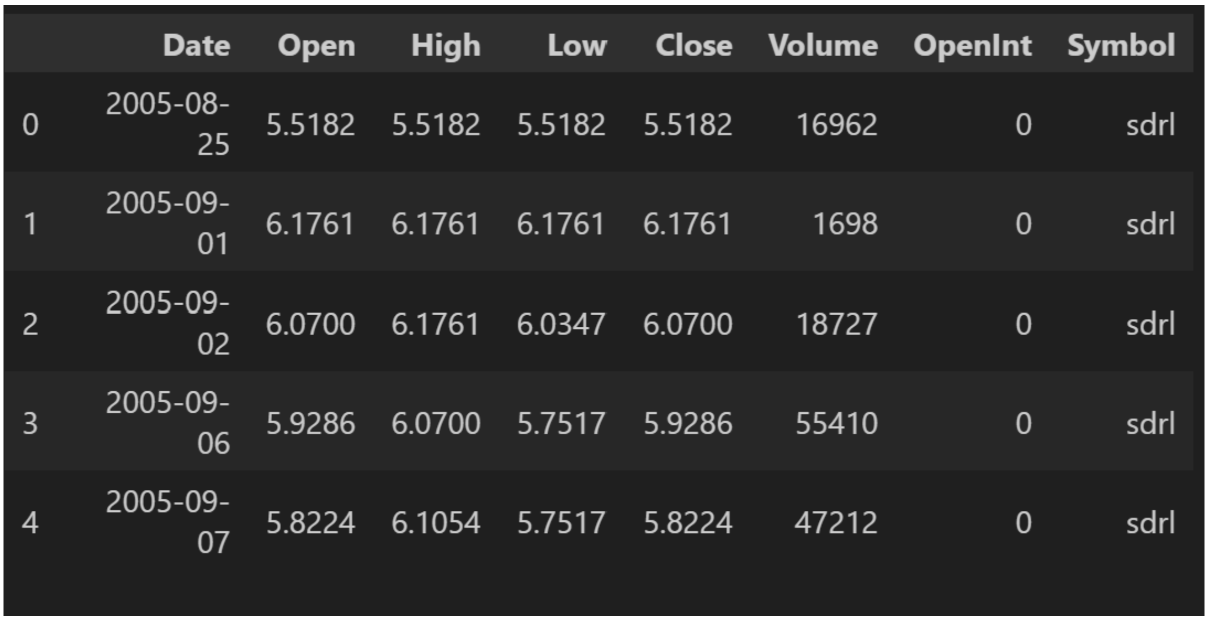
We have chosen the Stock market dataset. The dataset contains contains the following columns:

- `Date`: The date the stock was traded (datetime)
- `Open`: Price of the first stock that was traded on that date (float)
- `High`: Highest price of the stock was traded on that date (float)
- `Low`: Lowest price of the stock that was traded on that date (float)
- `Close`: Last price of the stock that was traded on that date (float)
- `Volume`: Number of traded stocks on that date (integer)
- `OpenInt`: Open contract, number of stocks that are still open to be traded on that date (integer)
- `Symbol`: Stock symbol, abbreviation used to identify a stock (string)



	column	dtype
0	Date	datetime64[ns]
1	Open	float64
2	High	float64
3	Low	float64
4	Close	float64
5	Volume	int64
6	OpenInt	int64
7	Symbol	string

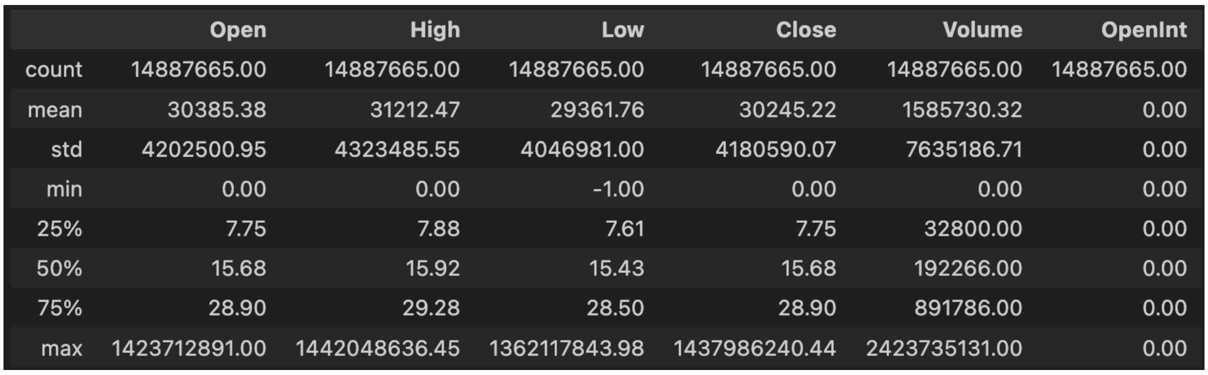
Figure 1: Data types for each column



	Date	Open	High	Low	Close	Volume	OpenInt	Symbol
0	2005-08-25	5.5182	5.5182	5.5182	5.5182	16962	0	sdrl
1	2005-09-01	6.1761	6.1761	6.1761	6.1761	1698	0	sdrl
2	2005-09-02	6.0700	6.1761	6.0347	6.0700	18727	0	sdrl
3	2005-09-06	5.9286	6.0700	5.7517	5.9286	55410	0	sdrl
4	2005-09-07	5.8224	6.1054	5.7517	5.8224	47212	0	sdrl

Figure 2: First few rows of the dataset

Below are a few figures describing some summary statistics of the dataset.



	Open	High	Low	Close	Volume	OpenInt
count	14887665.00	14887665.00	14887665.00	14887665.00	14887665.00	14887665.00
mean	30385.38	31212.47	29361.76	30245.22	1585730.32	0.00
std	4202500.95	4323485.55	4046981.00	4180590.07	7635186.71	0.00
min	0.00	0.00	-1.00	0.00	0.00	0.00
25%	7.75	7.88	7.61	7.75	32800.00	0.00
50%	15.68	15.92	15.43	15.68	192266.00	0.00
75%	28.90	29.28	28.50	28.90	891786.00	0.00
max	1423712891.00	1442048636.45	1362117843.98	1437986240.44	2423735131.00	0.00

Figure 3: Initial statistics of the dataset

The initial summary statistics also revealed several sudden significant changes in mean and median price.

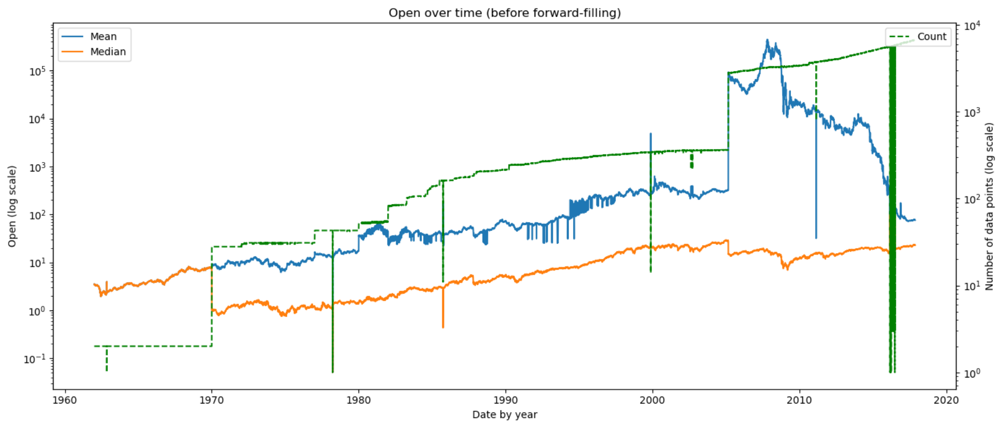


Figure 4: Open price over time

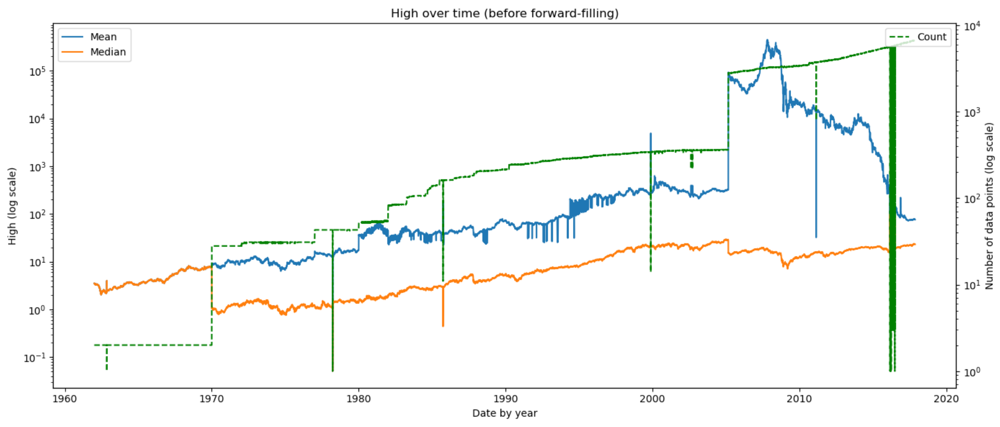


Figure 5: High price over time

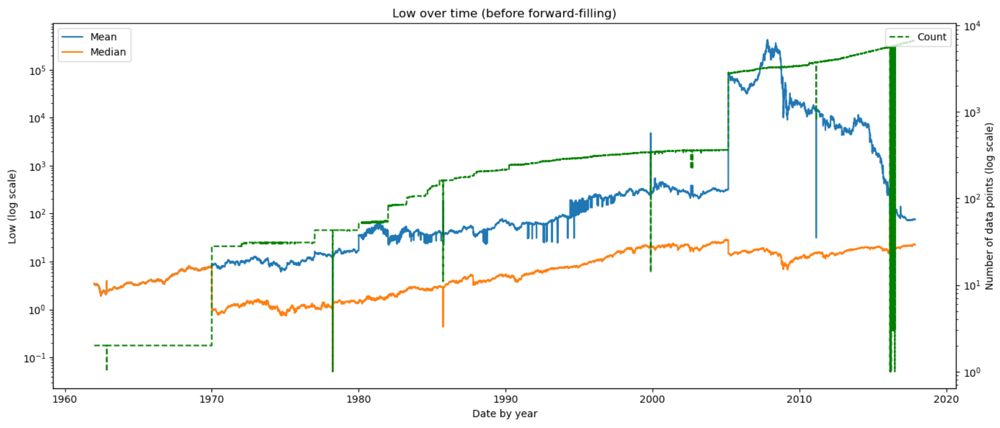


Figure 6: Low price over time

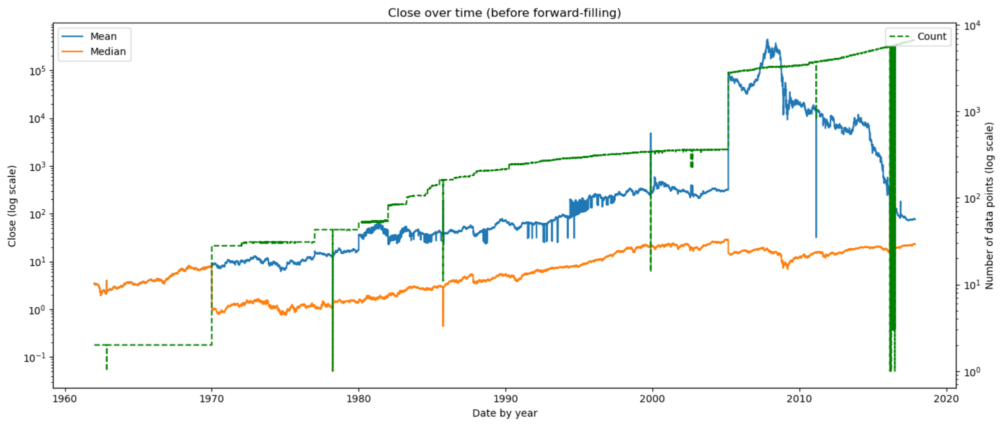


Figure 7: Close price over time

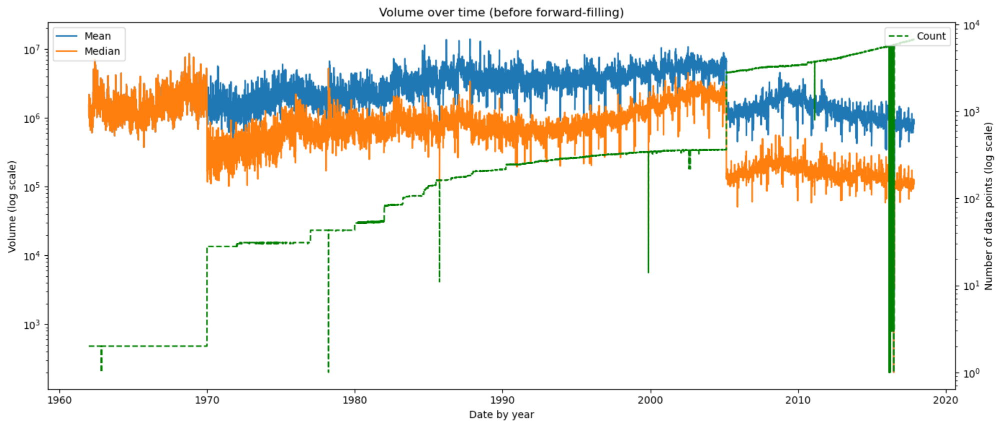


Figure 8: Volume over time

b. Identify missing values, outliers, and unique values in categorical columns.

When comparing the price trends with the number of recorded stock entries, we found that these drops were not caused by actual market movements but rather by missing data during those periods.

In the initial analysis, we observed that the open, high, low, and close prices of the stocks were relatively similar in terms of their mean and average values (see Figures 4–8).

We also observed a significant shift in stock prices around 1970 and 2005. As illustrated in Figure 9, this aligns with substantial increases in the number of recorded stocks.
The correlation between the surge in available stock data (Visualized by the dotted green count line) and the change in price statistics suggests that the shift was primarily driven by the expansion of the dataset rather than by underlying market dynamics.

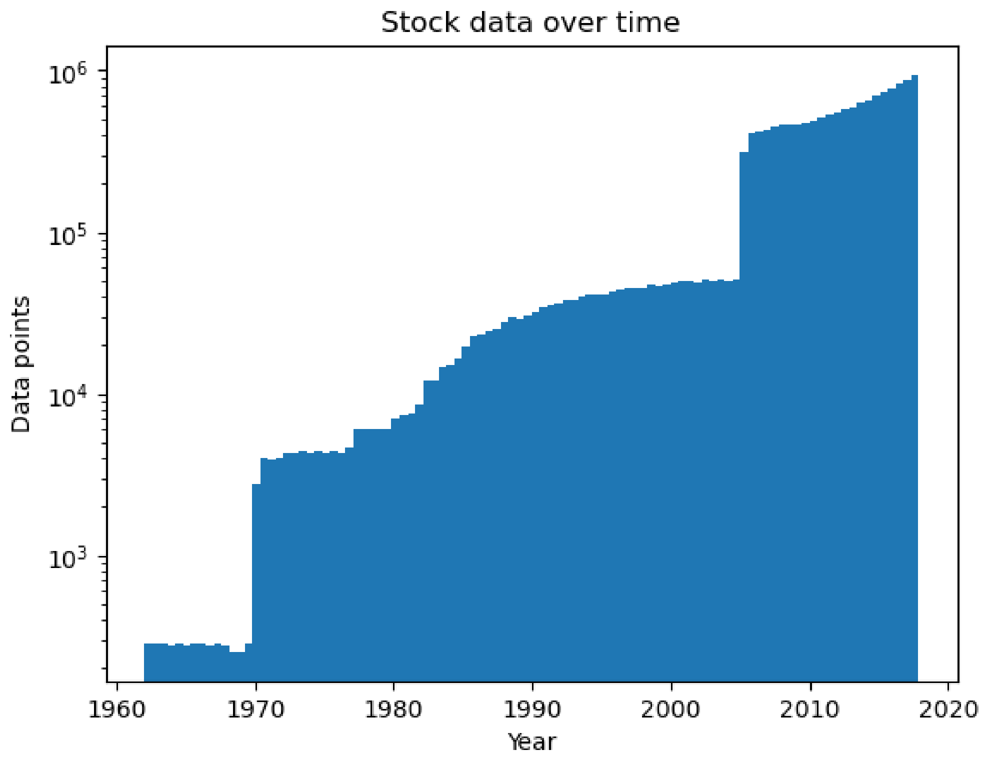


Figure 9: Number of stock data points over time

Boxplots for each column (Figure 10) reveal many high outliers, though they do not fully explain their causes. It is also important to note that market trends vary, and sharp drops do not always indicate unrealistic prices but can reflect real economic events, such as the 2008 financial crisis observed in Figures 4–8.

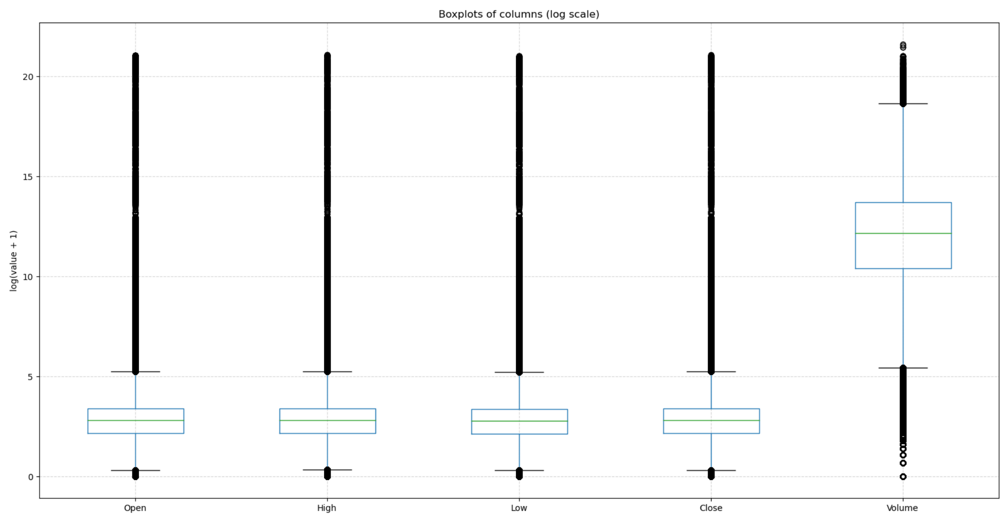
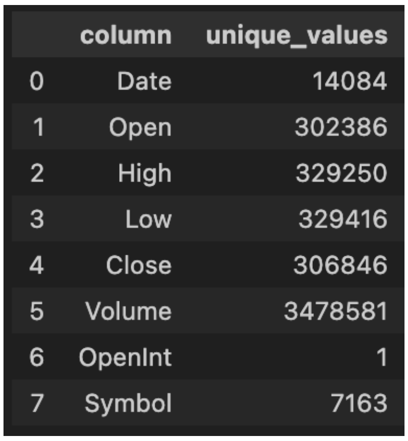


Figure 10: Boxplots for each column

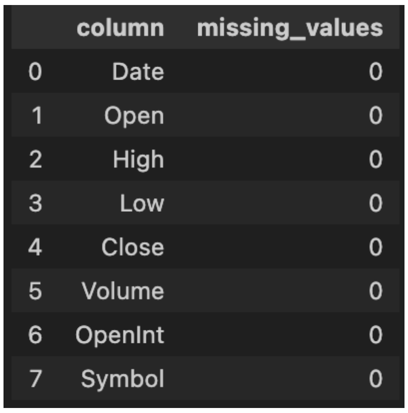
The unique values in categorial columns are the stock symbols, i.e., the identifiers for the different companies. See figure 11.
The OpenInt column has the value 0 for every row.



	column	unique_values
0	Date	14084
1	Open	302386
2	High	329250
3	Low	329416
4	Close	306846
5	Volume	3478581
6	OpenInt	1
7	Symbol	7163

Figure 11: Unique values for each column

In total, stock data was collected for 7195 companies. Of these, 32 files were empty and therefore unusable. For the retrieved companies, the collected rows (date with given stock prices for a company) contained no missing data as shown in figure 12, but figures 4-8 suggest that there might be some missing days of stock data.



	column	missing_values
0	Date	0
1	Open	0
2	High	0
3	Low	0
4	Close	0
5	Volume	0
6	OpenInt	0
7	Symbol	0

Figure 12: Missing values for each column

The dataset also contains some cells with negative values, this will be treated as invalid data as the price of a stock can't be below 0.

2. Data Cleaning

a. Handling Missing Values

Below are the same figures as in Task 1, but now with forward-filling applied to the dataset.

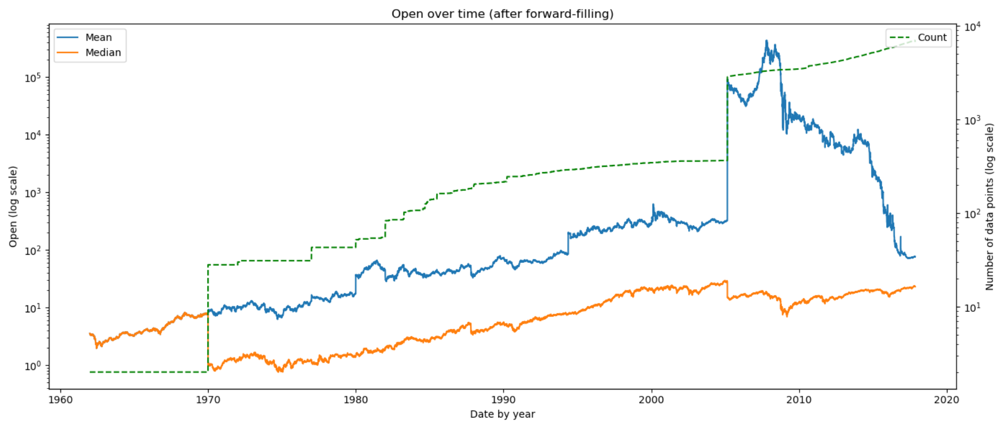


Figure 13: Open price over time after forward-filling

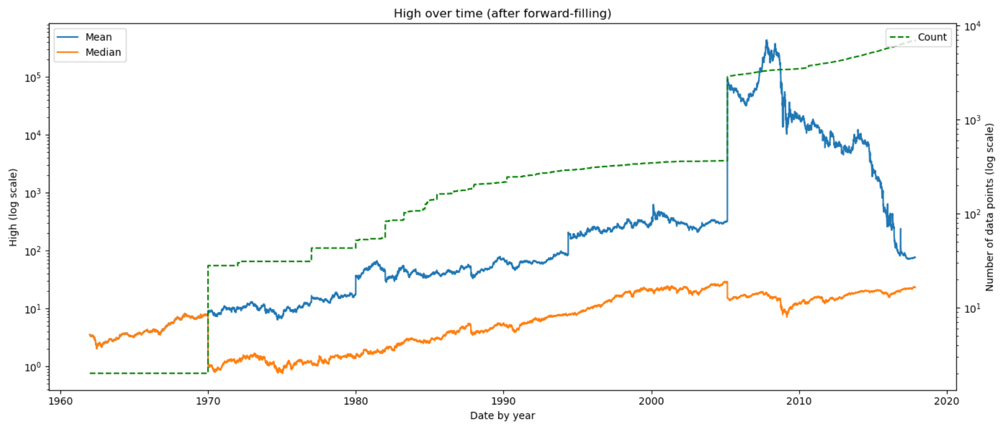


Figure 14: High price over time after forward-filling

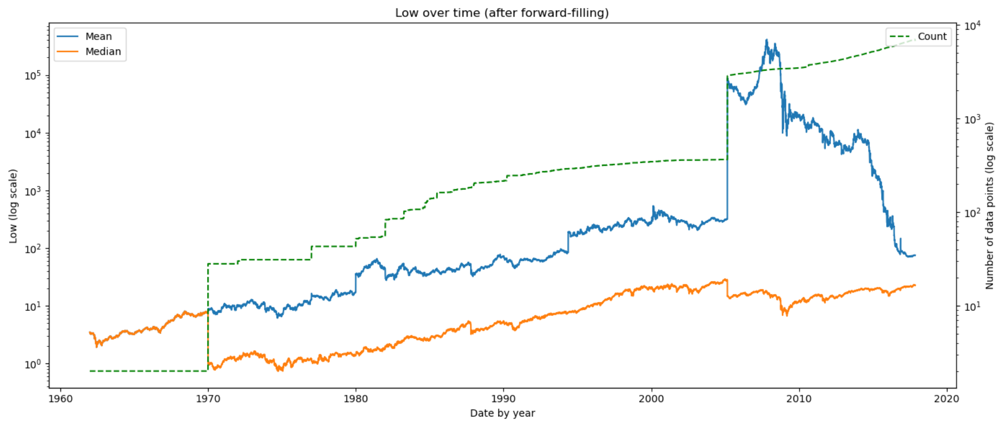


Figure 15: Low price over time after forward-filling

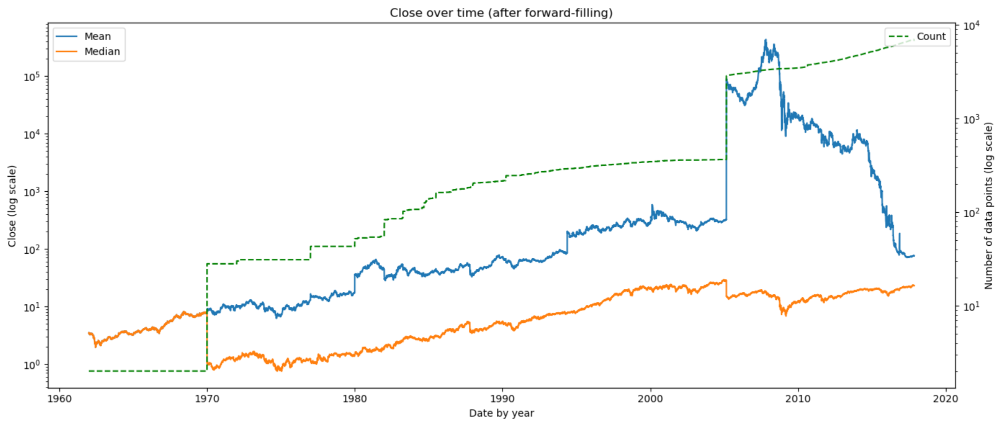


Figure 16: Close price over time after forward-filling

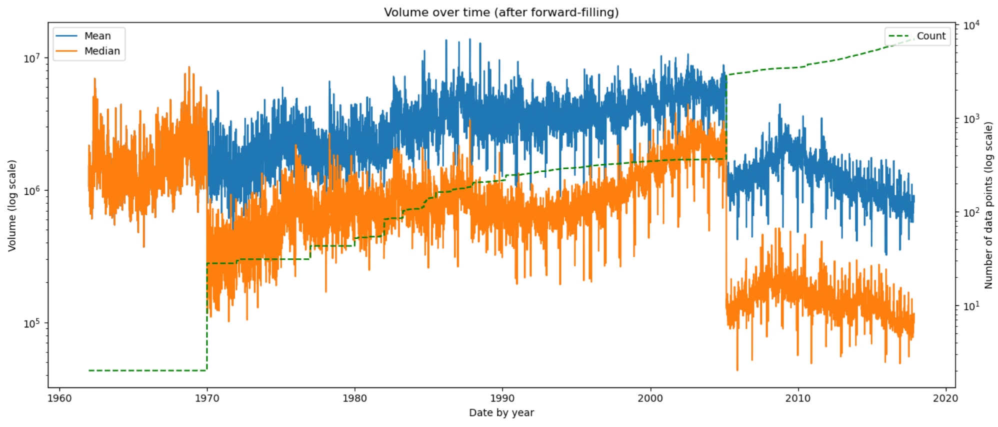
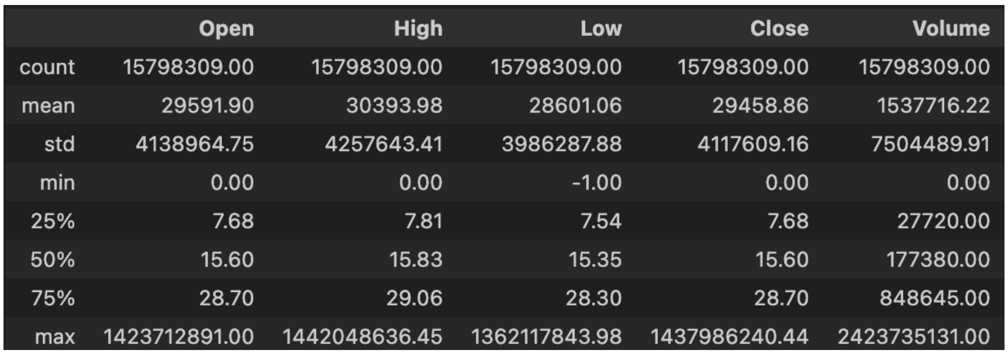


Figure 17: Volume over time after forward-filling

After handling missing values, the fluctuations seen in Figures 4–8 are resolved, leaving only the major shifts around 1970 and 2005, which correlate with the previously discussed increases in available stock data.



	Open	High	Low	Close	Volume
count	15798309.00	15798309.00	15798309.00	15798309.00	15798309.00
mean	29591.90	30393.98	28601.06	29458.86	1537716.22
std	4138964.75	4257643.41	3986287.88	4117609.16	7504489.91
min	0.00	0.00	-1.00	0.00	0.00
25%	7.68	7.81	7.54	7.68	27720.00
50%	15.60	15.83	15.35	15.60	177380.00
75%	28.70	29.06	28.30	28.70	848645.00
max	1423712891.00	1442048636.45	1362117843.98	1437986240.44	2423735131.00

Figure 18: Statistics of the dataset after forward-filling

Despite forward-filling resolving the price fluctuations, outliers remain visible (Figure 19).

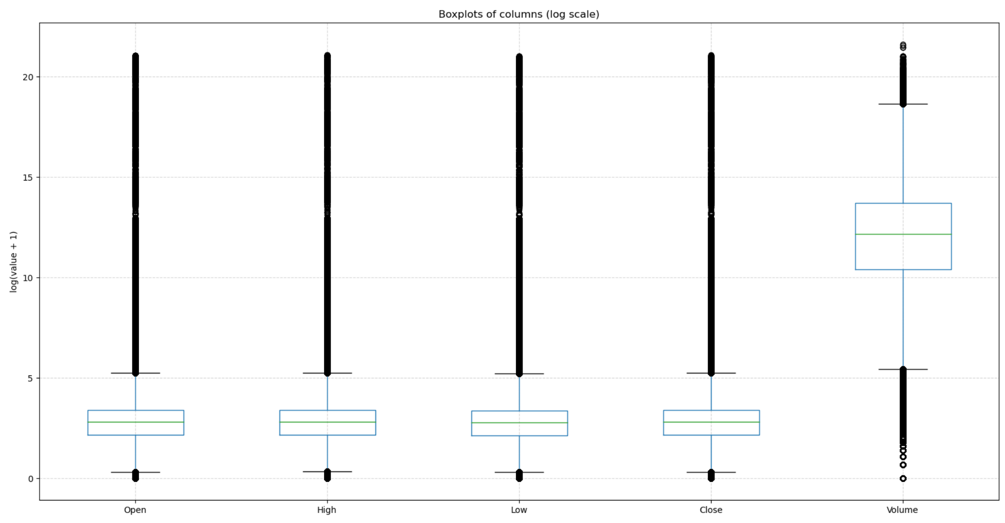


Figure 19: Boxplots for each column after forward-filling

b. Choose appropriate methods to handle missing values (e.g., mean/median imputation for numerical data, mode imputation for categorical data, or deletion of rows/columns).

As stated above, we used forward-fill to handle missing values.
Filling in with average values would be wrong since the market could vary, so it is reasonable to fill in with the previous stock data.
If there are missing stock data on Monday, we forward-fill with data from Friday.

We also decided to exclude weekends and market holidays.

The OpenInt column was completely dropped.

We ignore the 32 companies with empty stock data, dropped the open interest column, and apply forward-fill for missing values in price columns.

c. Justify your choices for handling missing data.

The 32 companies represent a very small share of the dataset. We chose to ignore these companies to avoid creating stock data with nothing to base it on.
The open interest column is excluded since it contains only zeros for all entries (see Figures 2 and 3), indicating no reported open contracts and adding no useful information for prediction while introducing unnecessary computation overhead.
Forward-fill is used because missing prices are more likely close to the previous day's data rather than to mean or median price.
We preferred filling with the previous day's data instead of the next day's data as otherwise that would fill our dataset using the knowledge of future prices.

3. Handling Outliers

a. Detect outliers using methods such as the IQR method or Z-score.

We have decided to use Z-score to detect outliers.
We use a per-stock rolling Z-score with a window of 1 month, filling all cells with a Z-score above 3 with the previous day's value for the same stock.

We also do this with all cells where the value is negative, as neither price nor volume can be negative.

Below are same figures as before, but with Z-score outliers removed. This doesnt have much of an effect on the graphs as we didn't find many outliers, around 5000 in each column in total.

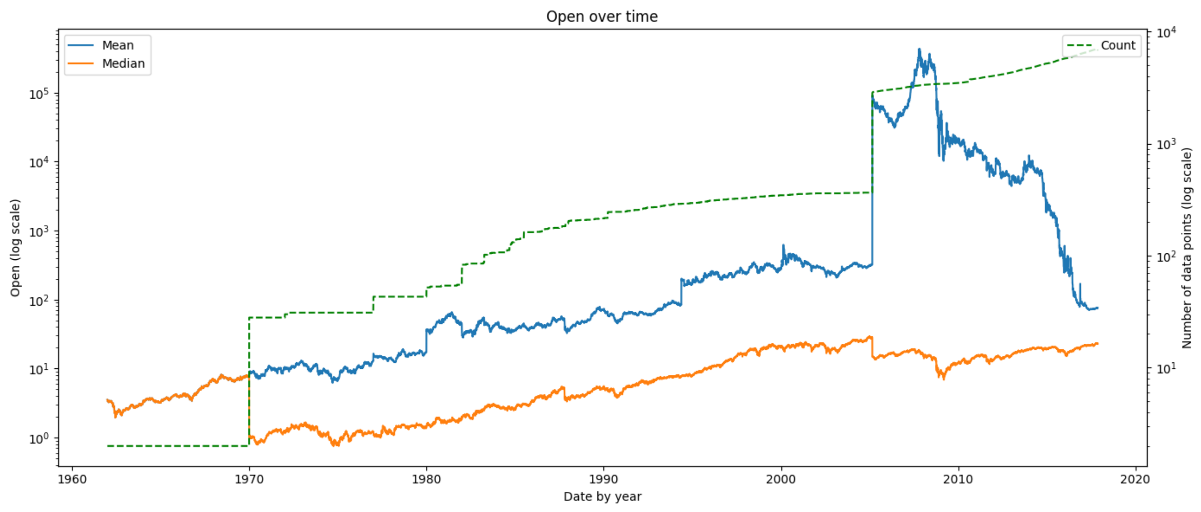


Figure 20: Open price over time after removing outliers

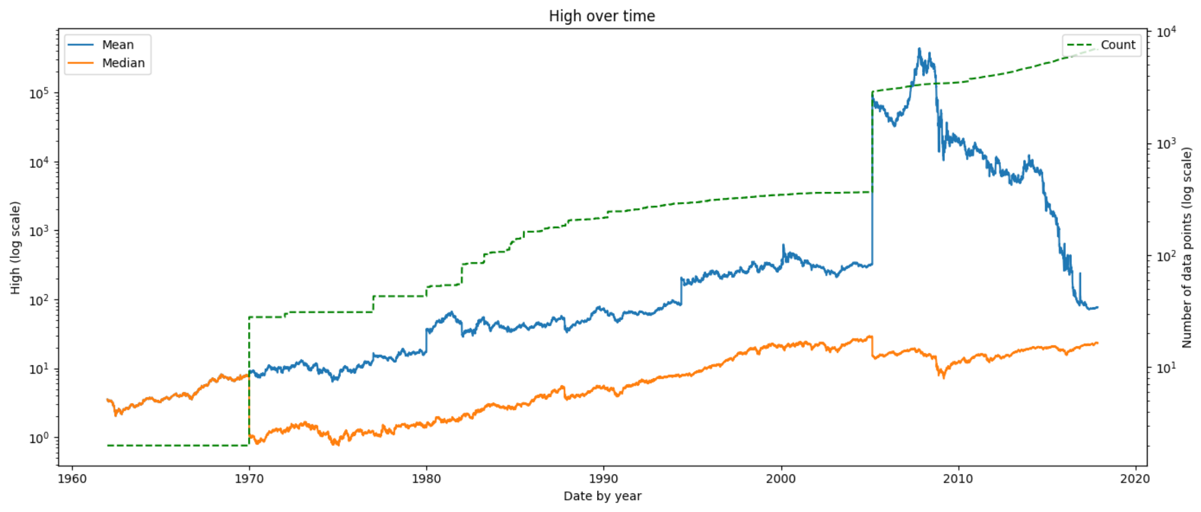


Figure 21: High price over time after removing outliers

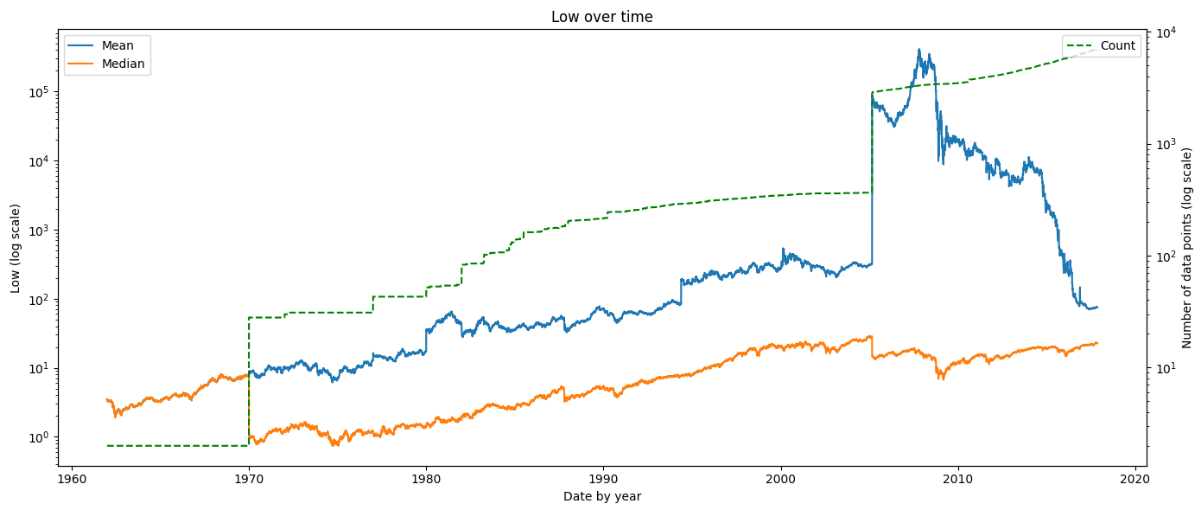


Figure 22: Low price over time after removing outliers

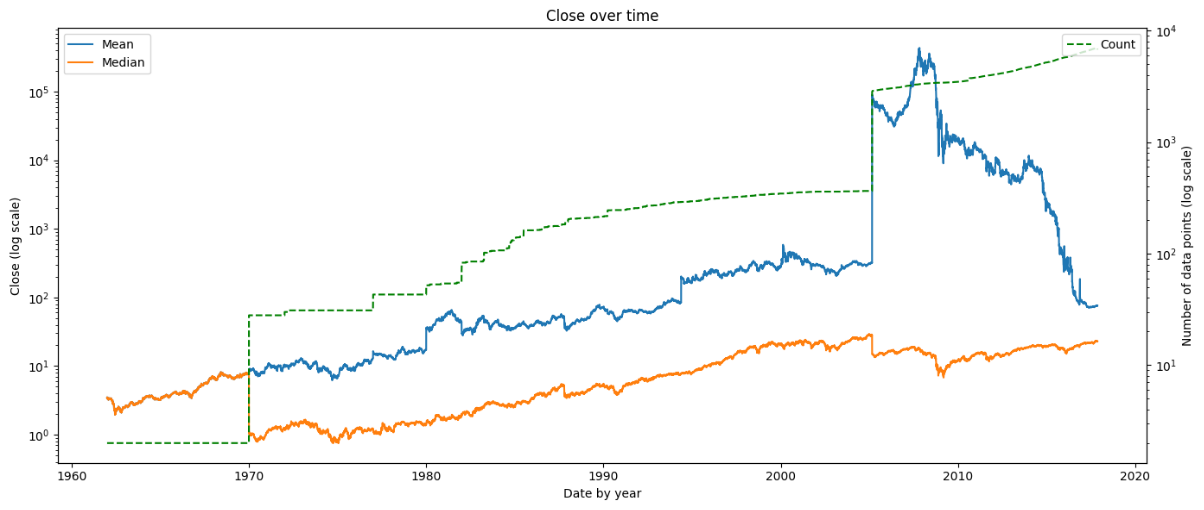


Figure 23: Close price over time after removing outliers

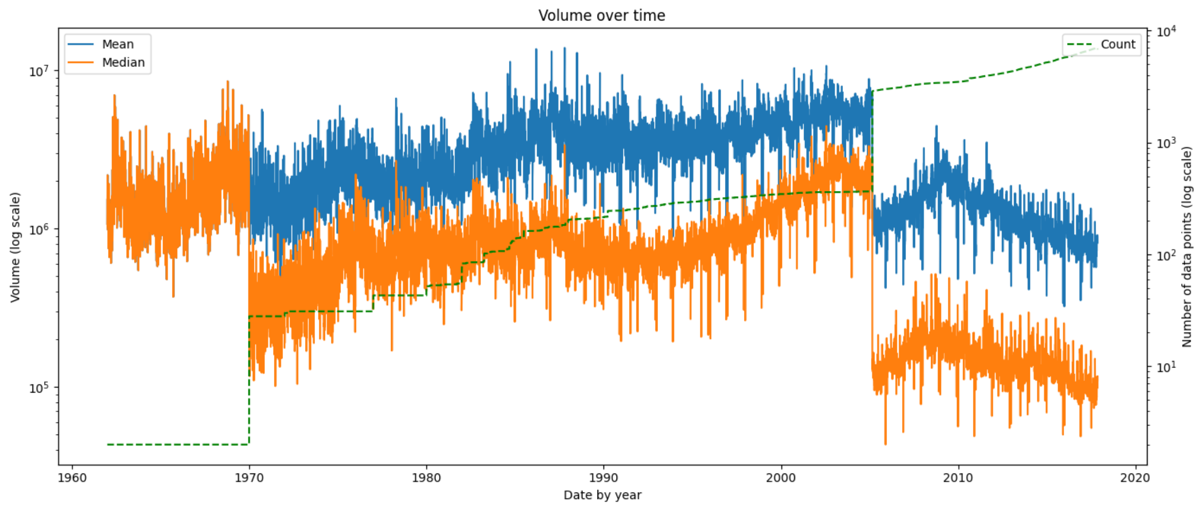


Figure 24: Volume over time after removing outliers

b. Decide whether to remove, cap, or transform the outliers. Justify your decisions.

All fields where we detected outliers were filled with the previous day's data.
We have chosen to do this to be consistent with our method of filling cells, as we do the same with missing rows.
Z-score is chosen to remove outliers, as it seems to be most used method for detecting outliers in stock data.

Our group also has more experience using Z-score, so we're more familiar with how the method works, meaning we understand our new dataset better.
Finally, rolling Z-score handles temporal fluctuations more effectively than global Z-score, which may flag small jumps after a normal price increase as an outlier.

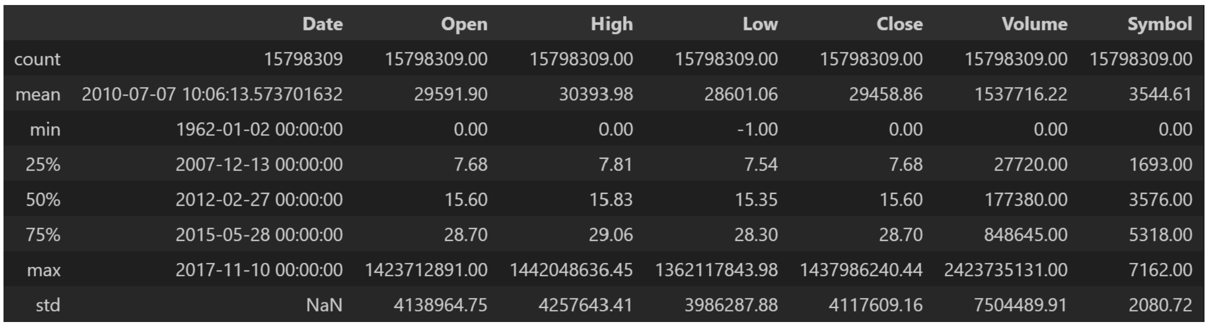
We know Z-score can remove values for very volatile stocks, but since we only detected around 5000 outliers per column, we have decided that it's okay to get rid of these data points since our dataset is so large.

4. Data Transformation

a. Encoding Categorical Data

i. Apply label encoding or one-hot encoding to transform categorical data into numerical form.

We have chosen to apply label encoding.
Below are some summary statistics of the dataset after encoding the `Symbol` column.



	Date	Open	High	Low	Close	Volume	Symbol
count	15798309	15798309.00	15798309.00	15798309.00	15798309.00	15798309.00	15798309.00
mean	2010-07-07 10:06:13.573701632	29591.90	30393.98	28601.06	29458.86	1537716.22	3544.61
min	1962-01-02 00:00:00	0.00	0.00	-1.00	0.00	0.00	0.00
25%	2007-12-13 00:00:00	7.68	7.81	7.54	7.68	27720.00	1693.00
50%	2012-02-27 00:00:00	15.60	15.83	15.35	15.60	177380.00	3576.00
75%	2015-05-28 00:00:00	28.70	29.06	28.30	28.70	848645.00	5318.00
max	2017-11-10 00:00:00	1423712891.00	1442048636.45	1362117843.98	1437986240.44	2423735131.00	7162.00
std	NaN	4138964.75	4257643.41	3986287.88	4117609.16	7504489.91	2080.72

Figure 25: First few rows of the dataset after label encoding

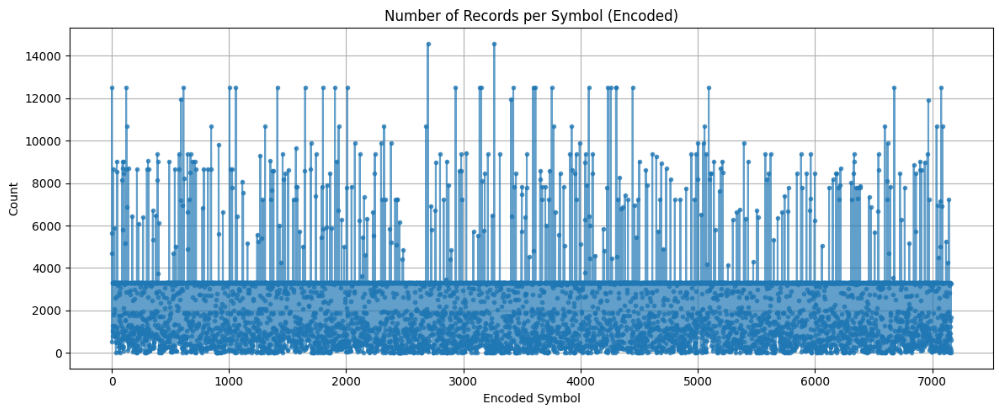


Figure 26: Number of entries per stock symbol after label encoding

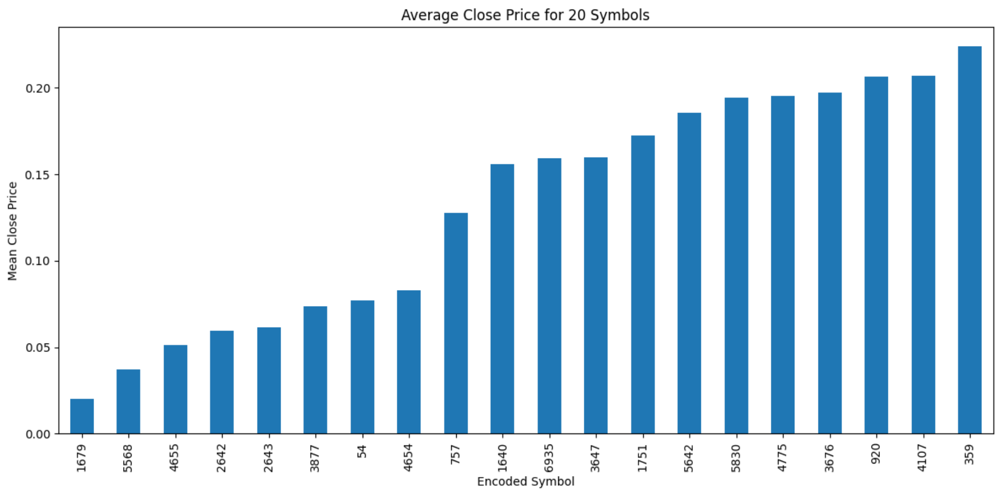


Figure 27: Average close price for 20 stock symbols after label encoding

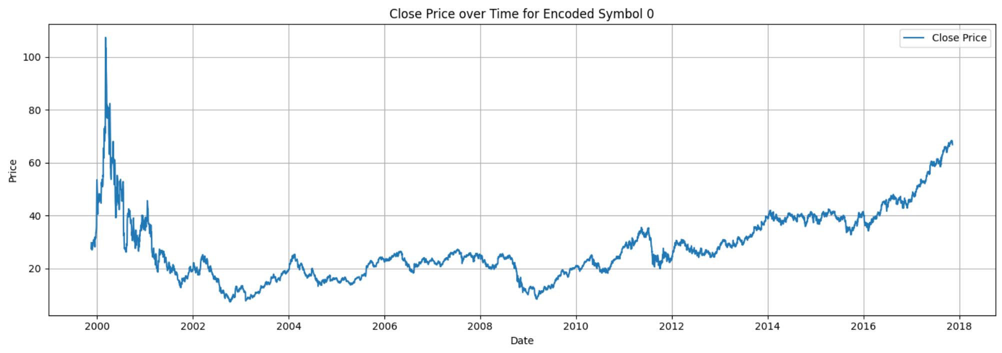


Figure 28: Close price over time for stock symbol 0 after label encoding

We chose to label encode the symbol column, setting an integer to replace every unique value of symbol in the dataset.
We applied label encoding to the symbol column. Since encoding starts from 0, the count in Figure 19 is 7162, despite there being 7163 unique companies.

ii. Justify your choice of encoding method.

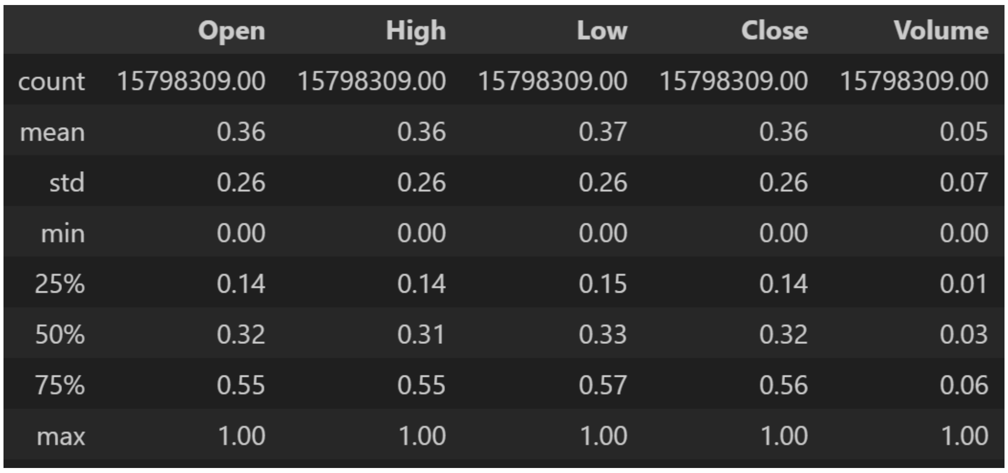
With 7,162 unique companies, one-hot encoding would introduce 7,162 extra features, making the model unnecessarily complex. Label encoding is thus more suitable in this case, as it assigns each unique company a numeric value, reducing dimensionality while preserving category distinction.

b. Feature Scaling

i. Apply feature scaling techniques such as normalization (Min-Max scaling) or standardization (Z-score normalization) to the dataset.

We decided to use Min-Max normalization to scale the dataset.
The scaling was only used on the columns that have ordinal data, meaning values have a meaningful order.
These columns are Open price, High price, Low price, Close price, and Volume.

Min-Max was chosen instead of other scaling techniques, as the result of Min-Max scaling gives an intuitive understanding of the output.
What you can tell from the result of Min-Max scaling is what percentage from the minimum value to the maximum value a give datapoint is.
This is preferred over Z-score normalization, as we felt we don't have an intuitive understanding of it in the context of stock data.



	Open	High	Low	Close	Volume
count	15798309.00	15798309.00	15798309.00	15798309.00	15798309.00
mean	0.36	0.36	0.37	0.36	0.05
std	0.26	0.26	0.26	0.26	0.07
min	0.00	0.00	0.00	0.00	0.00
25%	0.14	0.14	0.15	0.14	0.01
50%	0.32	0.31	0.33	0.32	0.03
75%	0.55	0.55	0.57	0.56	0.06
max	1.00	1.00	1.00	1.00	1.00

Figure 29: First few rows of the dataset after Min-Max scaling

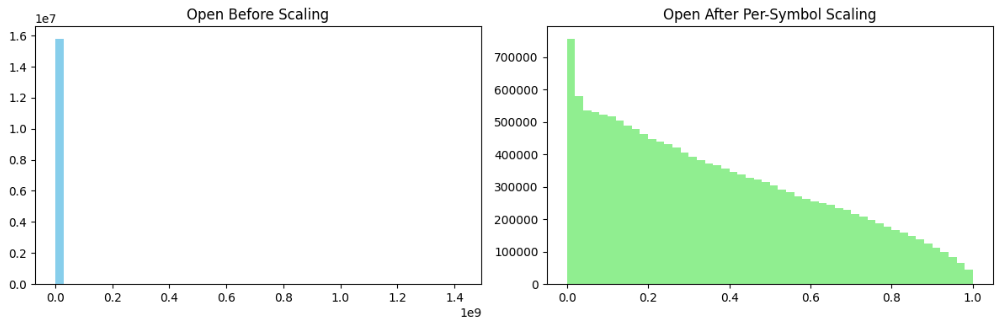


Figure 30: Open price before and after Min-Max scaling

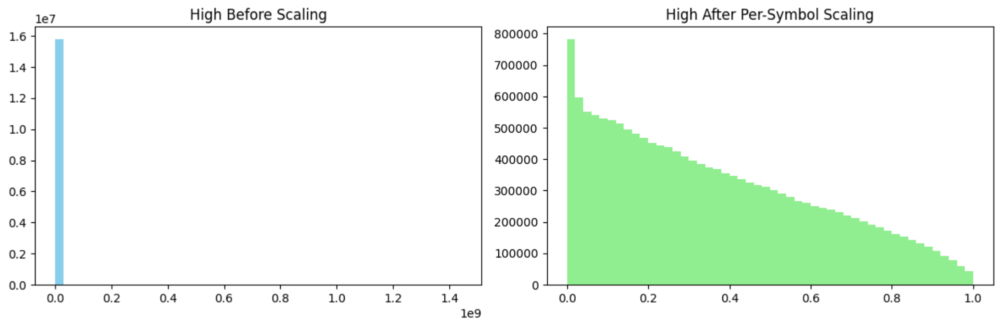


Figure 31: High price before and after Min-Max scaling

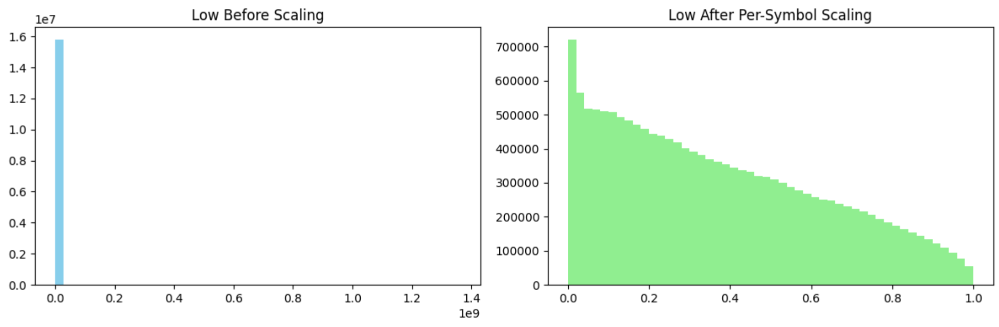


Figure 32: Low price before and after Min-Max scaling

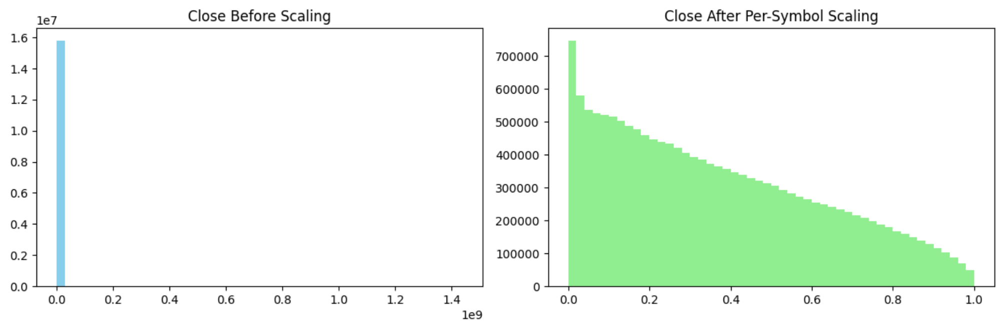


Figure 33: Close price before and after Min-Max scaling

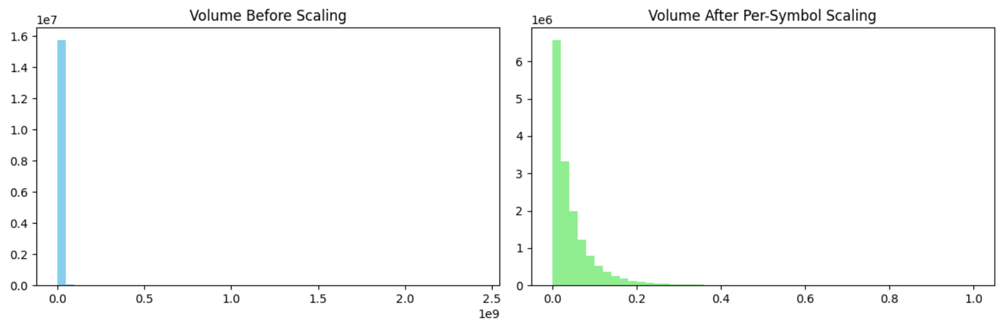


Figure 34: Volume before and after Min-Max scaling

We can see that after scaling we get a left-skewed distribution.
This means that the stock prices are densely packed at the lower end of their own range.

ii. Explain why feature scaling is necessary and how it impacts the model.

Feature scaling is important because raw features often have very different ranges, and this can cause models to give more weight to features with larger values.
By scaling, we ensure that all features contribute equally, which improves fairness and accuracy.

5. Data Splitting

a. Split the preprocessed dataset into training and testing sets. Typically, an 80-20 or 70-30 split is used.

We split the dataset into a training and testing dataset using an 80-20 split.
Instead of splitting the dataset randomly, we choose to use the chronologically last 20% of the dataset as testing data.
This avoids data leakage, as the model isn't trained on events and values that aren't available at the time of the prediction.

```
Training set shape: (12634695, 7)
Testing set shape: (3163614, 7)
```

When training the model, we would prefer using chronological splits with rolling validation, as this method would mean most of the training data could be used for both validation and training without data leakage.

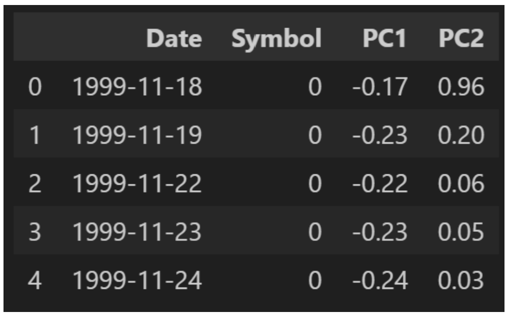
b. Explain the importance of splitting the data and how it prevents overfitting.

Splitting the data allows the model to be trained on one set and evaluated on another, ensuring that performance is measured on unseen data.
The training set adjusts model parameters, while the test set checks generalization. This prevents overfitting by forcing the model to learn patterns instead of memorizing the training data.
A validation set is often used during training to tune hyperparameters and monitor performance.

6. Apply dimensionality reduction techniques such as Principal Component Analysis (PCA) and discuss how it affects the dataset.

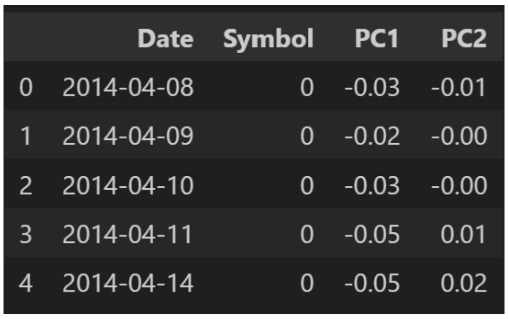
Applying Principal Component Analysis (PCA) to our stock dataset — which includes features such as Open, High, Low, Close, Volume, and OpenInt — helps reduce dimensionality by transforming the original correlated features into a smaller set of uncorrelated components. Since stock prices (Open, High, Low, Close) are often highly correlated, PCA captures most of their shared variance in the first few principal components.

This dimensionality reduction simplifies the dataset while retaining the majority of the information. As a result, the model becomes less complex, less prone to multicollinearity, and easier to visualize. However, interpretability decreases since the new principal components are linear combinations of the original features rather than directly meaningful attributes like “Open” or “Close” price.



	Date	Symbol	PC1	PC2
0	1999-11-18	0	-0.17	0.96
1	1999-11-19	0	-0.23	0.20
2	1999-11-22	0	-0.22	0.06
3	1999-11-23	0	-0.23	0.05
4	1999-11-24	0	-0.24	0.03

Figure 35: First few lines from the PCA train set



	Date	Symbol	PC1	PC2
0	2014-04-08	0	-0.03	-0.01
1	2014-04-09	0	-0.02	-0.00
2	2014-04-10	0	-0.03	-0.00
3	2014-04-11	0	-0.05	0.01
4	2014-04-14	0	-0.05	0.02

Figure 36: First few lines from the PCA test set

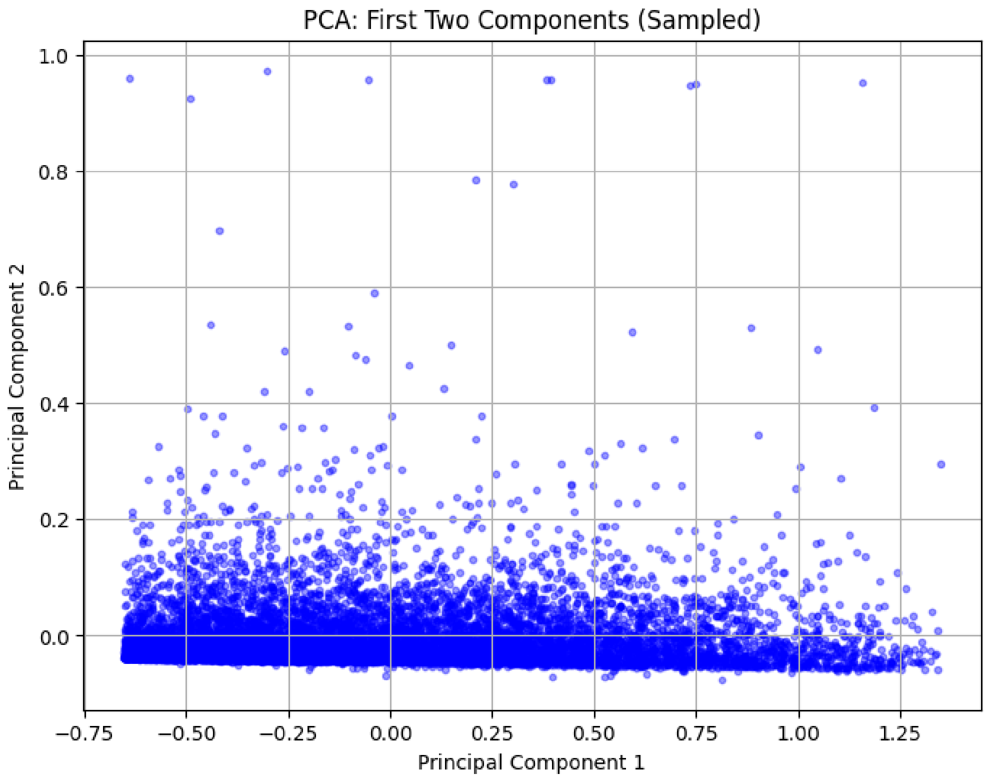


Figure 37: A random sample from the first two PCA components

Applying dimensionality reduction using PCA changes the dimensions of the dataset by replacing columns with a smaller number of new features. The effect is the reduction of dimensions, which can speed up model training and remove features that aren't valuable for training.

There are also problems with dimensionality reduction. One is that it makes the features in the data less intuitively understood, making it hard to understand the dataset and interpret predictions. Dimensionality reduction also loses some information, which might be valuable to training the model.