

Applying Machine Learning Algorithms to Stock Market Data for Future Stock Price Prediction

Submitted by:

Eugene Lowe

Alonzia Stephens

Master of Business Analytics and Data Science

Wayne State University

Detroit, MI 48201, USA

Submitted to Dr. Suzan Arslanturk

in partial fulfillment of the requirements for

CSC 5800: Intelligent Systems: Algorithms and Tools

Stock Market Analysis & Prediction

Overview

The stock market plays a crucial role in the economy, influencing and reflecting the financial state and growth of businesses and countries. Investors, analysts, and financial institutions continuously seek to understand and predict stock price movements to make informed investment decisions, minimize risks, and maximize returns.

Stock market prediction involves using historical data, statistical techniques, and machine learning models to forecast future prices or trends. However, the market is affected by a multitude of factors including economic indicators, political events, investor sentiment, and company performance, making accurate prediction a challenging task.

The NASDAQ-100 Index is one of the most closely watched stock indices in the world, representing 100 of the largest non-financial companies listed on the NASDAQ stock exchange. It includes industry giants such as Apple, Microsoft, Amazon, and Alphabet, making it a key indicator of the performance of the technology and innovation-driven sectors of the U.S. economy.

This project aims to analyze historical price trends of some of the top portfolio holding companies in the NASDAQ-100 and build a predictive model that estimates future stock movements. By leveraging historical data and applying modern analytical techniques such as machine learning, we hope to identify patterns that can improve the accuracy of stock market forecasts. We will also look at correlation between different stocks in the NASDAQ-100.

Understanding stock price trends and being able to predict them, even in the short term, can provide significant advantages for traders and investors. It also contributes to the broader field of financial analytics and helps bridge the gap between traditional financial theory and modern data science techniques.

Dataset Overview

For this project, we are using historical stock market data obtained from Yahoo Finance and its yfinance API for Python. The dataset includes daily stock prices for NASDAQ-100 companies. The key features in the dataset include:

- Financial data such as dividends, splits, earnings, actions, financials, balance sheets / cash flow, price (open/close), volume, market cap, etc.
- Forecast data such as analyst price targets, growth estimates, revenue estimates, etc.
- Ownership and Insider data such as insider purchases / transactions, institutional and major holders, etc.

- News and filing data such as recent news headlines about the company and SEC filings.

There are also callable functions / methods to fetch or update data about stocks.

Data Pre-Processing

Data cleaning

```
stocks = ['PLTR', 'AAPL', 'MSFT', 'NVDA', 'AMZN', 'META', 'AVGO', 'NFLX', 'GOOGL', 'TSLA']
ndx_stocks = yf.Tickers(stocks)

ndx_stocks = ndx_stocks.history(start='2020-12-1', group_by='ticker')
columns_to_drop = [(ticker, col) for ticker in stocks for col in ['Dividends', 'Stock Splits']]
ndx_stocks = ndx_stocks.drop(columns=columns_to_drop)
```

First, 10 of the top stocks in the NASDAQ-100 were identified and stored in a variable named “stocks.” These stocks include Palantir(PLTR), Apple(AAPL), Microsoft(MSFT), Nvidia(NVDA), Amazon(AMZN), Meta(META), Broadcom Inc.(AVGO), Netflix(NFLX), Google(GOOGL), and Tesla(TSLA). These are the 10 stocks that will be used in this stock market analysis & prediction study. A dataframe was created, named “ndx_stocks,” grouping each of the 10 stocks into one dataframe. It was decided to focus on historical data dating back to December of 2020 (12-01-2020) because the newest company, Palantir, became public on the stock market in October of 2020.

Feature Subset Selection - Irrelevant Features

There were columns in the data that did not provide any useful information for the data mining task at hand, such as “Dividends” and “Stock Splits,” so those were removed.

[7]: `ndx_stocks.head()`

Ticker	GOOGL						AMZN						...		
	Price	Open	High	Low	Close	Volume	Open	High	Low	Close	Volume	...	Open	High	Low
Date															
2020-12-01	87.912704	90.652603	87.732064	89.340874	37350000	159.425003	162.447495	157.858994	161.003998	90740000	...	199.196671	199.283340	190.683334	194
2020-12-02	89.340861	91.200971	88.833787	90.814316	29424000	161.082504	161.600006	158.662994	160.176498	62586000	...	185.479996	190.513336	180.403336	189
2020-12-03	90.593884	91.752840	90.417724	90.658577	24728000	160.272995	161.432007	159.065506	159.336502	57840000	...	196.673340	199.656662	194.143326	197
2020-12-04	90.577953	91.039742	90.248025	90.754112	20544000	159.910507	159.910507	157.938004	158.128998	58272000	...	197.003326	199.679993	195.166672	199
2020-12-07	90.577953	91.039742	90.248025	90.754112	20544000	159.910507	159.910507	157.938004	158.128998	58272000	...	197.003326	199.679993	195.166672	199

Upon checking the contents contained in the dataframe previously defined, it shows each stock with its corresponding Open, High, Low, and Close prices along with the Volume for each day dating back to December 1, 2020. These are the attributes for each stock that will be used to predict the target variable, which will be defined later.

```
[9]: ndx_stocks.describe()
```

	GOOGL					AMZN					...				
Ticker	Price	Open	High	Low	Close	Volume	Open	High	Low	Close	Volume	...	Open	High	Low
count	1100.000000	1100.000000	1100.000000	1100.000000	1.100000e+03	1100.000000	1100.000000	1100.000000	1100.000000	1100.000000	1.100000e+03	...	1100.000000	1100.000000	1100.000000
mean	132.038867	133.559634	130.613665	132.101108	3.141098e+07	154.157905	156.026545	152.160440	154.127430	6.008819e+07	...	247.420867	252.960100	247.420867	252.960100
std	28.149205	28.332312	27.908616	28.104736	1.304106e+07	34.803277	34.916654	34.593495	34.782281	2.752445e+07	...	63.547599	65.050400	63.547599	65.050400
min	84.608485	86.108321	82.943452	83.033028	9.312000e+06	82.800003	83.480003	81.430000	81.820000	1.500750e+07	...	103.000000	111.750000	103.000000	111.750000
25%	109.676023	111.570209	107.856598	109.854668	2.298805e+07	128.104996	129.834999	126.327501	128.087503	4.233858e+07	...	202.572498	207.947400	202.572498	207.947400
50%	131.471445	132.919514	130.048239	131.894417	2.816295e+07	159.308998	161.319244	158.000000	159.438248	5.438605e+07	...	237.951668	243.621600	237.951668	243.621600
75%	148.475521	149.985806	146.719796	148.160286	3.615635e+07	176.942501	178.954998	174.787621	176.774998	7.047295e+07	...	278.184998	283.910000	278.184998	283.910000
max	203.156027	206.811821	202.576693	206.142593	1.232000e+08	239.020004	242.520004	238.029999	242.059998	2.726620e+08	...	475.899994	488.540000	475.899994	488.540000

8 rows x 50 columns

The above image provides some basic statistics such as the mean(average) stock prices and volumes of each stock. This gives a general idea of the average stock price and volume of each stock over the time period previously specified. Also listed is the standard deviation.

```
[103]: print(f'Duplicate Rows: {ndx_stocks.duplicated().sum()}')
```

Duplicate Rows: 0

```
[107]: print(f'Missing values per column:\n\n {ndx_stocks.isnull().sum()}')
```

Missing values per column:

```

Ticker Price
NFLX    Open    0
        High    0
        Low     0
        Close   0
        Volume  0
PLTR    Open    0
        High    0
        Low     0
        Close   0
        Volume  0
META    Open    0
        High    0
        Low     0
        Close   0
        Volume  0

```

```

--- PLTR ---
No Missing Data

--- AAPL ---
No Missing Data

--- MSFT ---
No Missing Data

--- NVDA ---
No Missing Data

--- AMZN ---
No Missing Data

--- META ---
No Missing Data

--- AVGO ---
No Missing Data

--- NFLX ---
No Missing Data

--- GOOGL ---
No Missing Data

--- TSLA ---
No Missing Data

```

Simple code to verify there aren't any missing data or duplicates.

Feature Creation

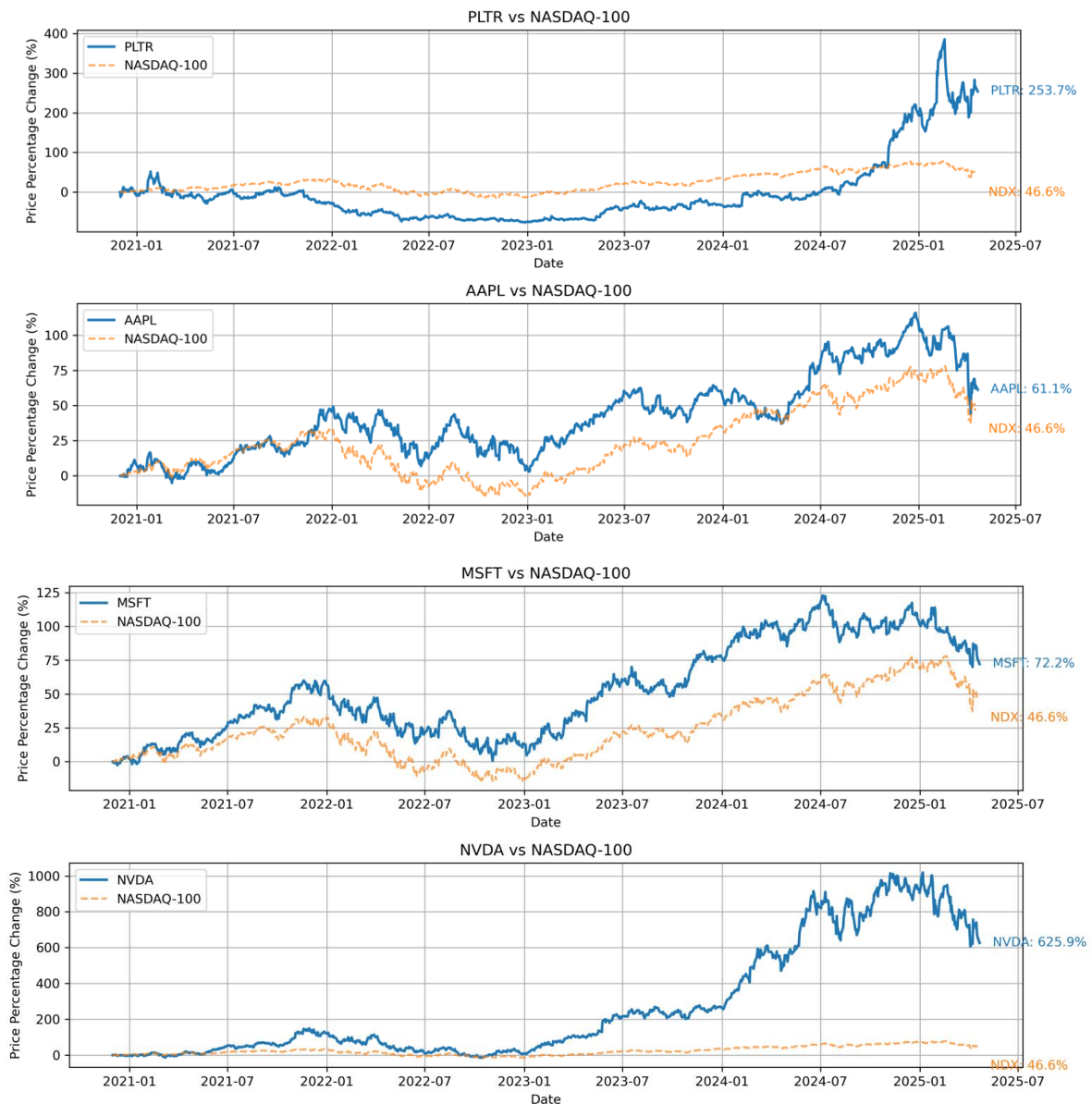
Symbol	PLTR							AAPL				GOOGL			
Feature	Open	High	Low	Close	Volume	Tomorrow	Target	Open	High	Low	...	Volume	Tomorrow	Target	Open
Date															
2020-12-01	28.090000	28.139999	24.450001	25.670000	84050000	22.510000	0	118.200280	120.603161	117.223499	...	37350000	90.814316	1	199.196671
2020-12-02	22.240000	23.400000	21.150000	22.510000	149062100	24.030001	1	119.186821	120.505481	118.083061	...	29424000	90.658577	0	185.479996
2020-12-03	23.549999	25.620001	23.500000	24.030001	85634300	23.850000	0	120.651984	120.905949	119.372403	...	24728000	90.754112	1	196.673340
2020-12-04	24.879999	25.240000	23.510000	23.850000	56044100	28.940001	1	119.753348	120.007313	118.698423	...	20544000	90.419220	0	197.003326
2020-12-07	24.440001	29.000000	24.440001	28.940001	118463700	28.590000	0	119.470088	121.677616	119.411484	...	22288000	90.135567	0	201.639999
...
2025-04-11	87.980003	89.550003	85.470001	88.550003	95130700	92.620003	1	186.100006	199.539993	186.059998	...	33636200	159.070007	1	251.839996
2025-04-14	95.800003	97.330002	91.459999	92.620003	122836900	98.400002	1	211.440002	212.940002	201.160004	...	30333000	156.309998	0	258.359985
2025-04-15	93.529999	98.989998	93.050003	98.400002	118457700	92.709999	0	201.860001	203.509995	199.800003	...	27551500	153.330002	0	249.910004
2025-04-16	96.279999	97.300003	89.620003	92.709999	121997600	93.779999	1	198.360001	200.699997	192.369995	...	28187400	151.160004	0	247.610001
2025-04-17	94.709999	95.110001	92.269997	93.779999	83991800	NaN	0	197.199997	198.830002	194.419998	...	33046600	NaN	0	243.470001

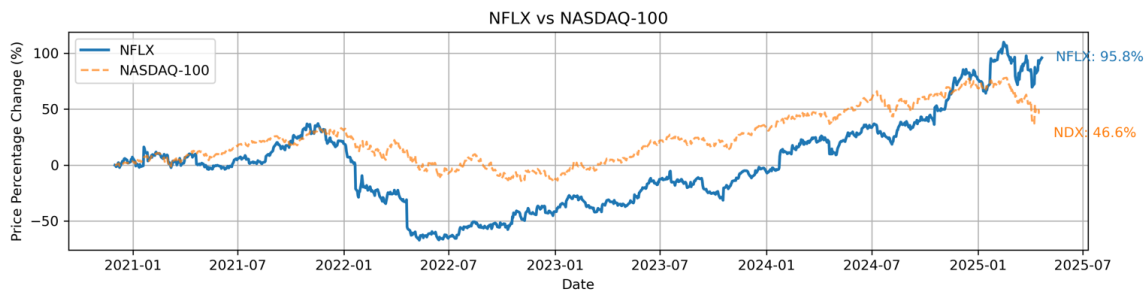
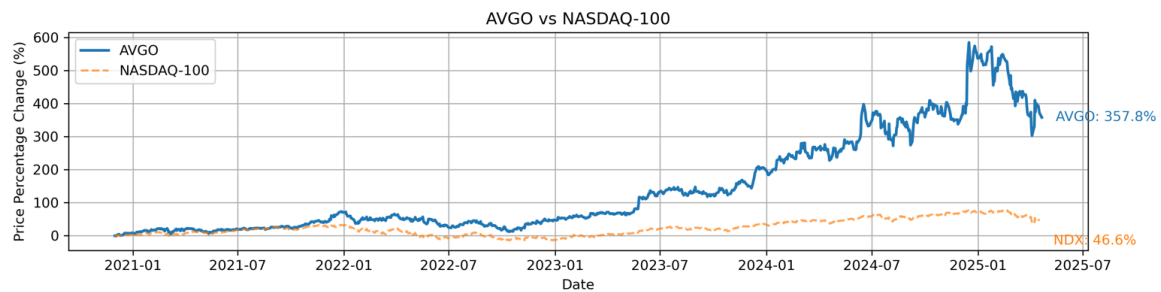
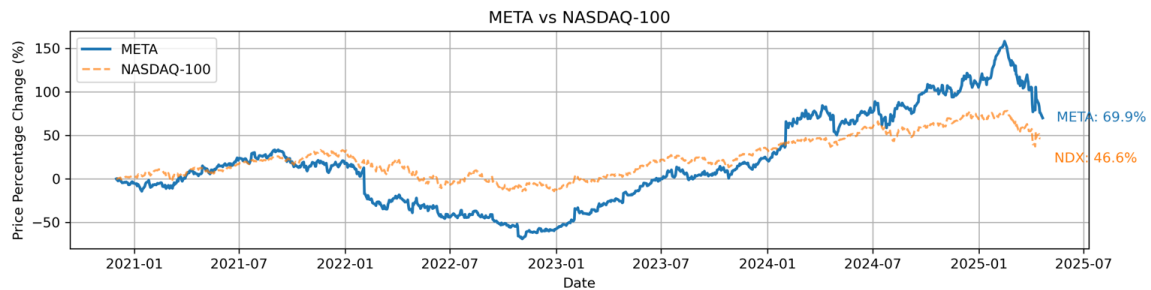
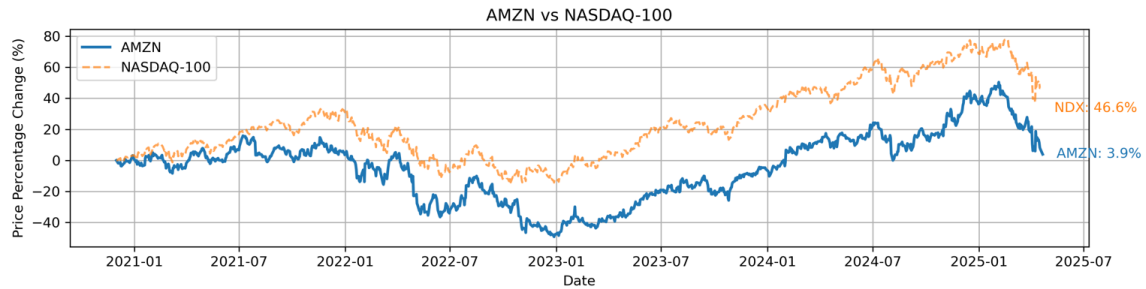
New attributes were created, labeled "Tomorrow" and "Target." The Tomorrow attribute is used to predict the closing price of a stock for the next day. The Target variable is also defined. More on the target variable below.

Binarization of Data - Closing Price

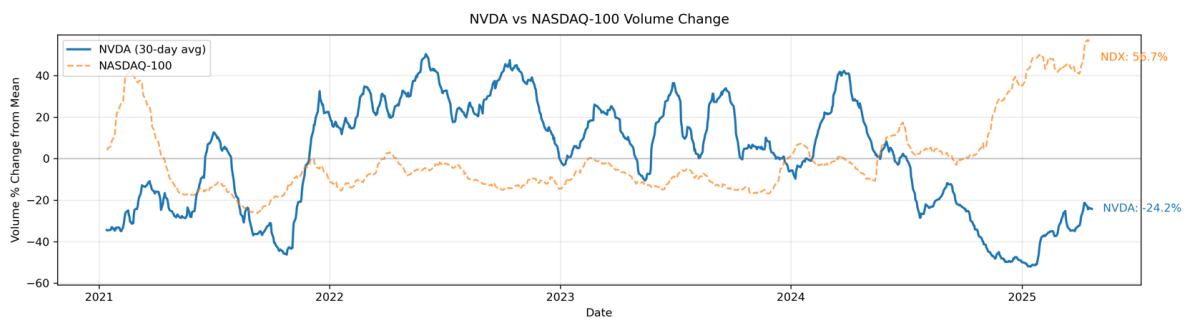
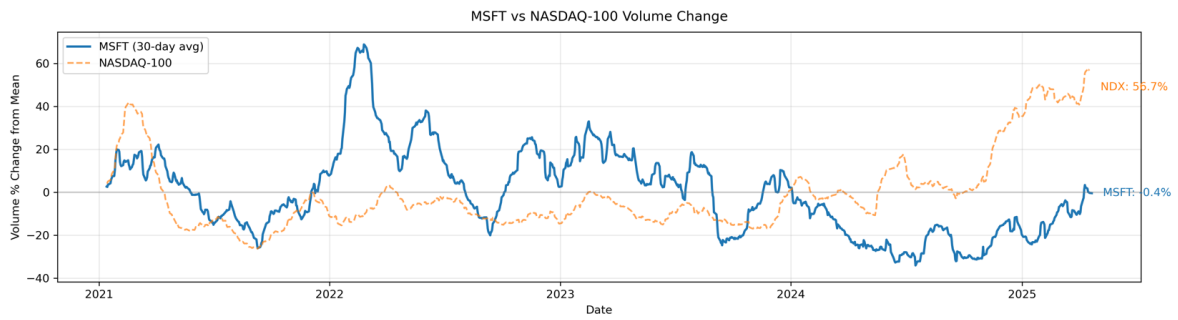
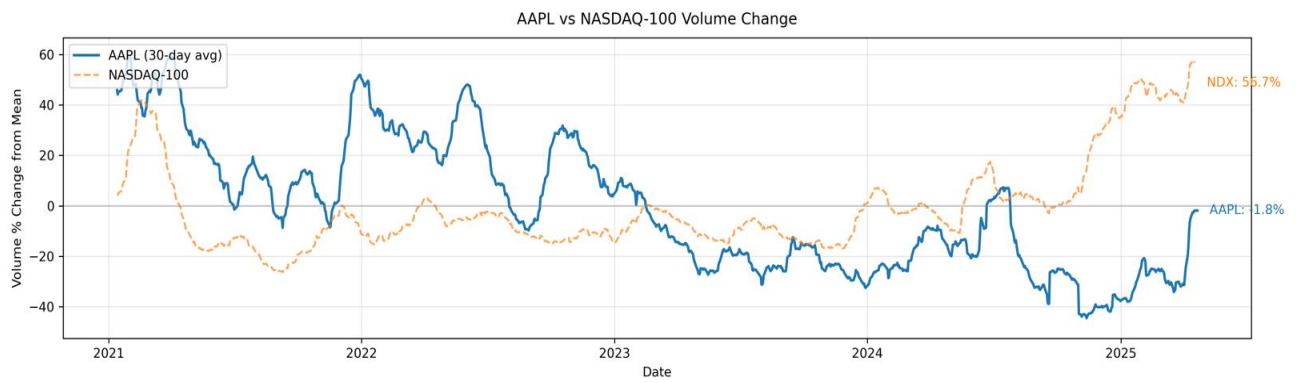
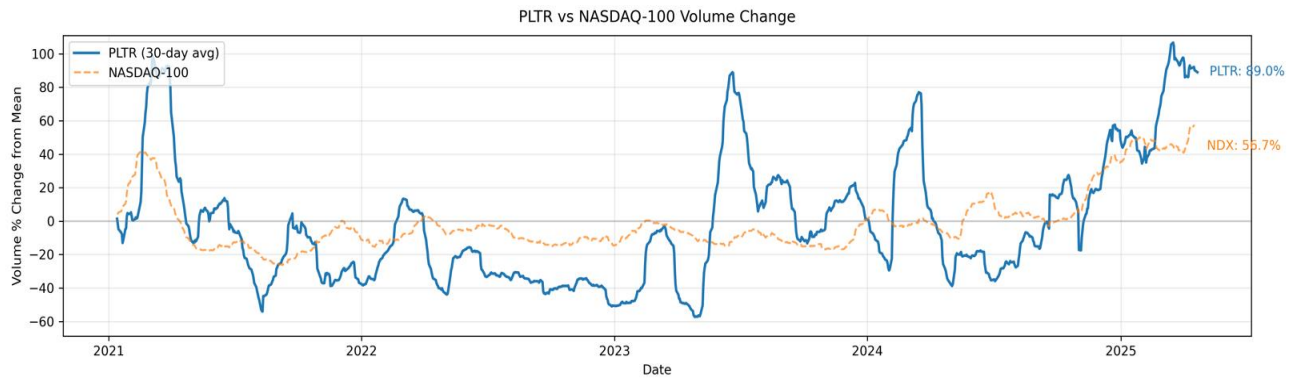
The previous image shows a new attribute labeled “Target.” This will be the target variable for the modeling task. The target variable in this case represents the percentage change of the closing price on each day for each stock. If the closing price increased from the previous day, it is represented with a value of 1, otherwise (the closing price decreases or doesn’t change) the value will be set to 0. Creating this binary target variable will help in the classification modeling tasks explained later.

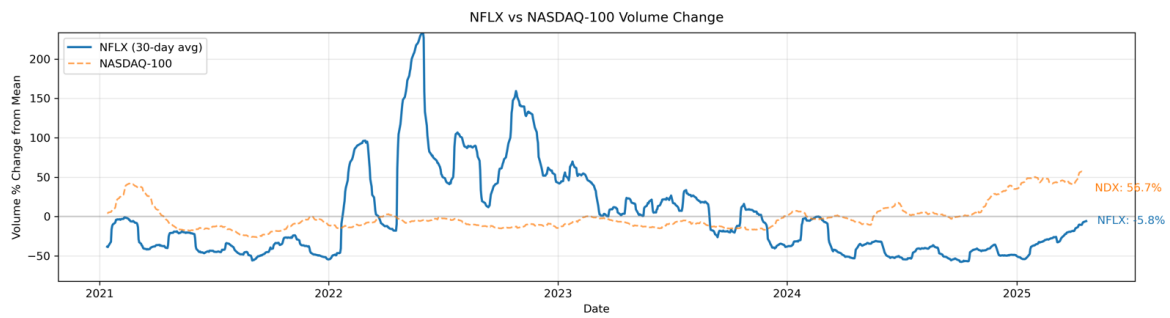
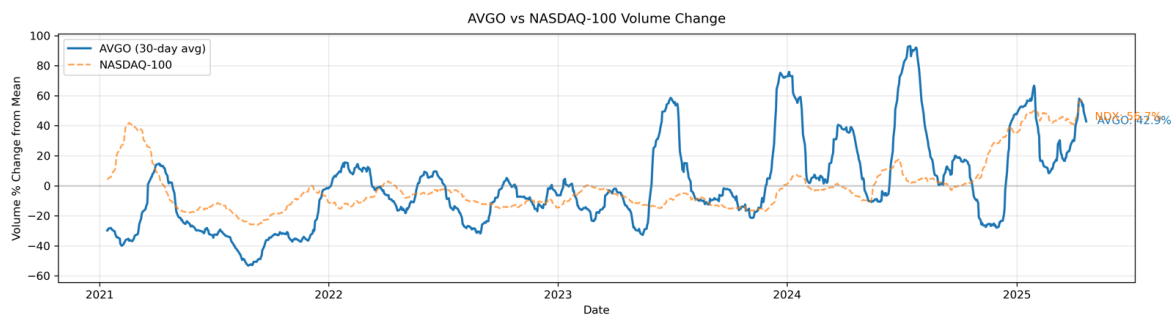
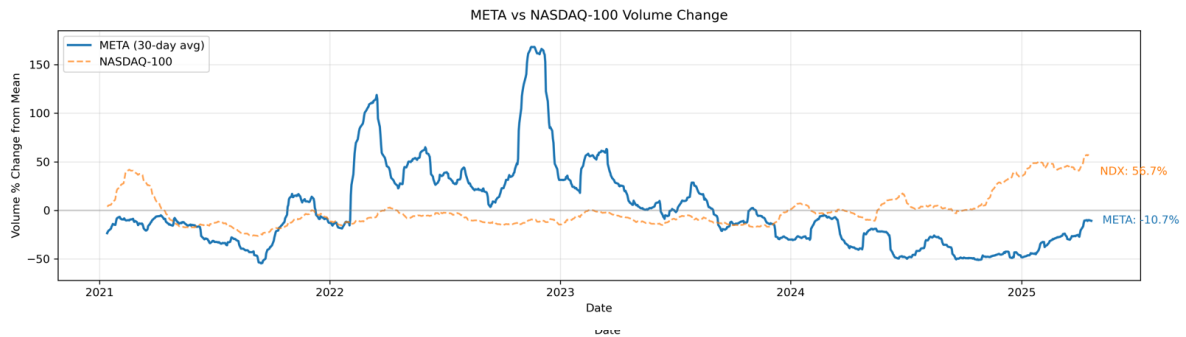
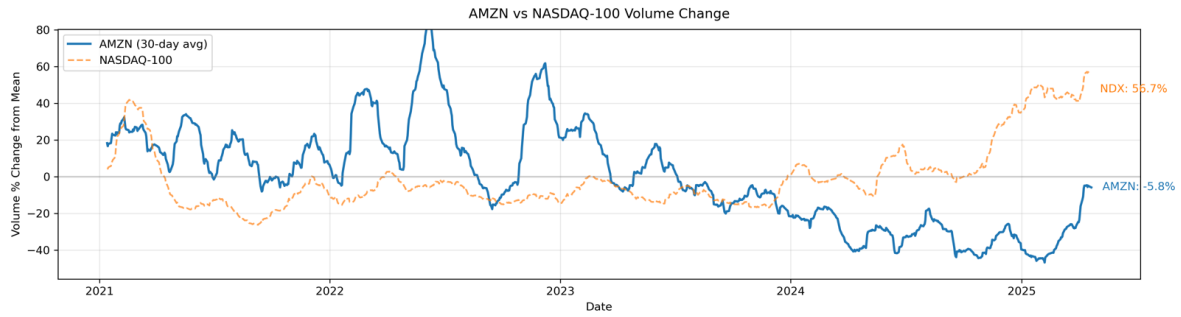
Visualizations

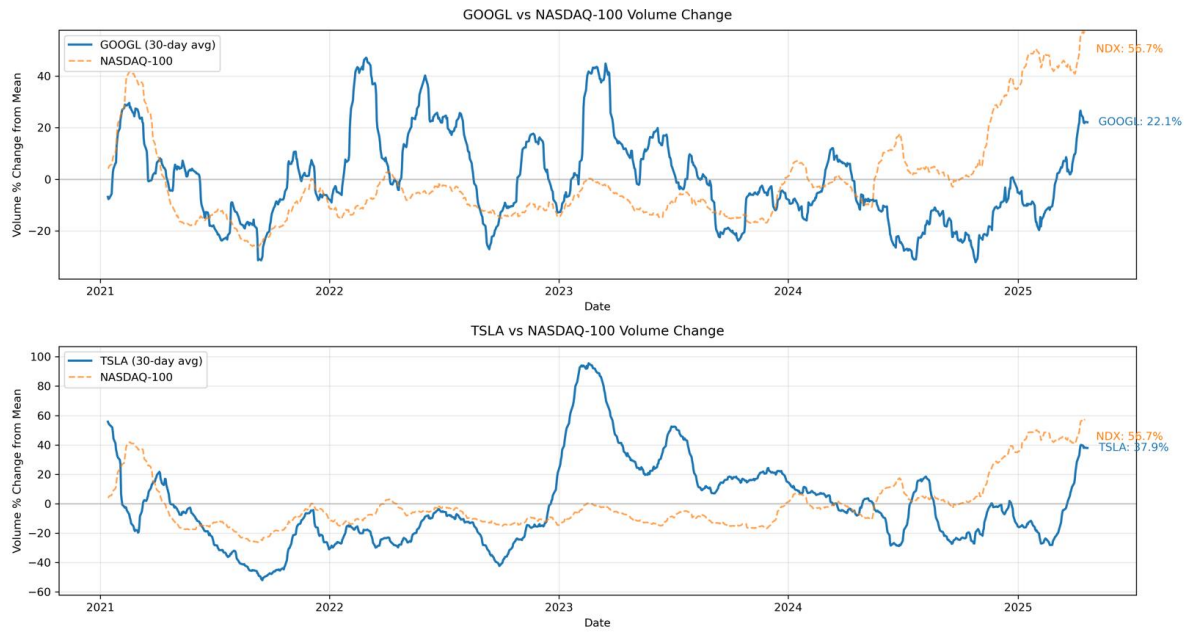




The charts above represent the price movements of each of the 10 stocks against the entire Nasdaq-100 index. As shown, most these stocks have outperformed the Nasdaq-100 in terms of percentage change of price. Netflix, Palantir, Tesla, and Amazon, however, show periods of not performing above the Nasdaq-100. Nvidia, Palantir, and Broadcom show the most exponential growth while Google, Microsoft, and Apple show more consistency and gradual growth overtime.



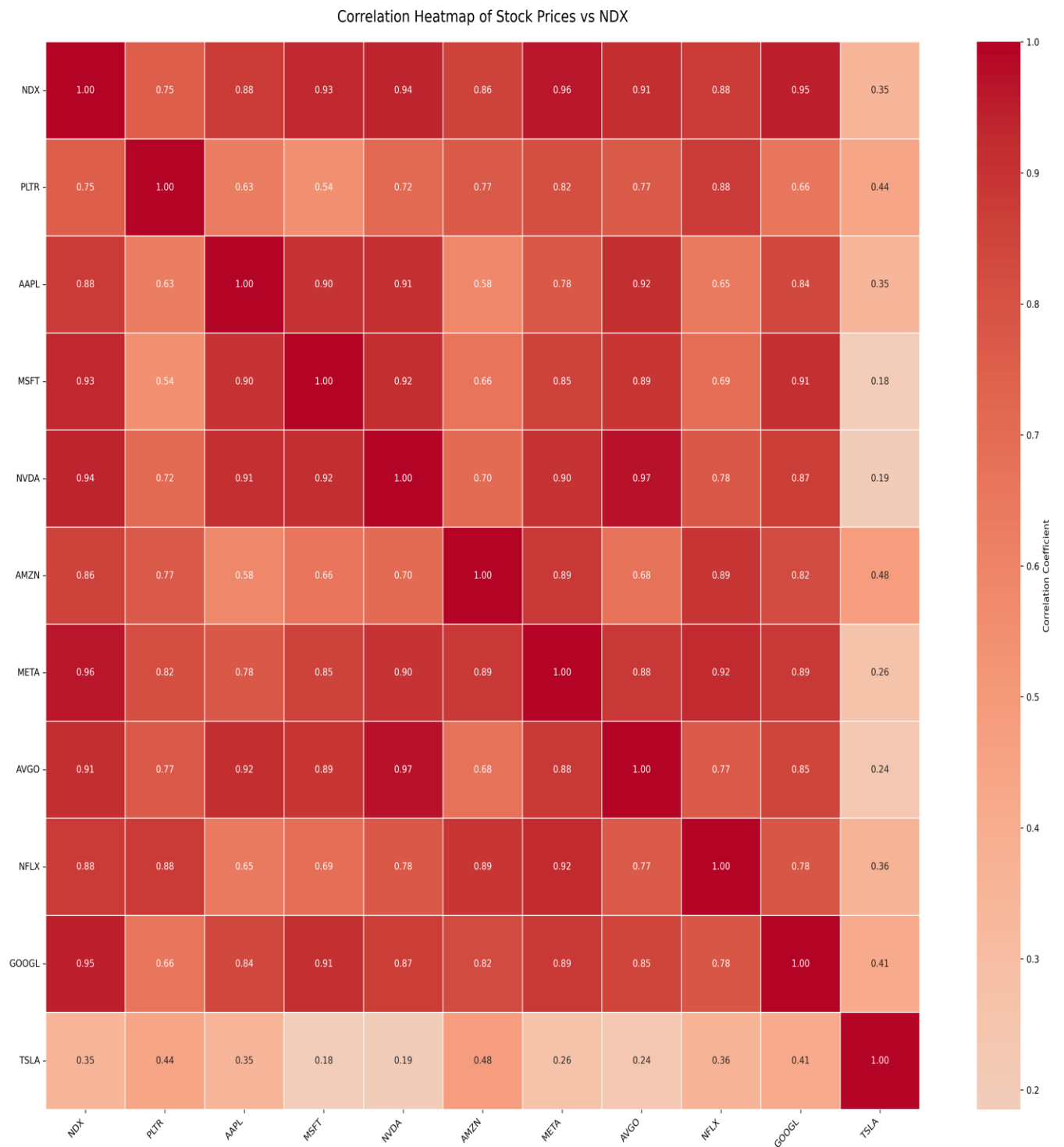




The charts above show the volume change on a 30-day average for each stock against the Nasdaq-100 index. As seen in most of charts, the Nasdaq-100 index has more shares traded than the stocks themselves within the last year



The above scatter plot represents each stock's closing price as the y-axis and the trading volume as the x-axis. Here shows a clear influence of stock price on volume. That is, the higher the stock price the lower trading volume is. It could be implied that investors aren't purchasing shares of stocks when they're at all-time highs. Instead, they wait until the stock price falls for a better buying opportunity.



A correlation heatmap showing the correlation coefficients between each stock and the Nasdaq-100 index. The below image shows the stocks with the highest/lowest price

correlations with the Nasdaq-100 index. Meta and Google rank amongst the highest correlations with the Nasdaq while Tesla and Palantir ranks among the lowest.

Highest Price Correlations with NDX:

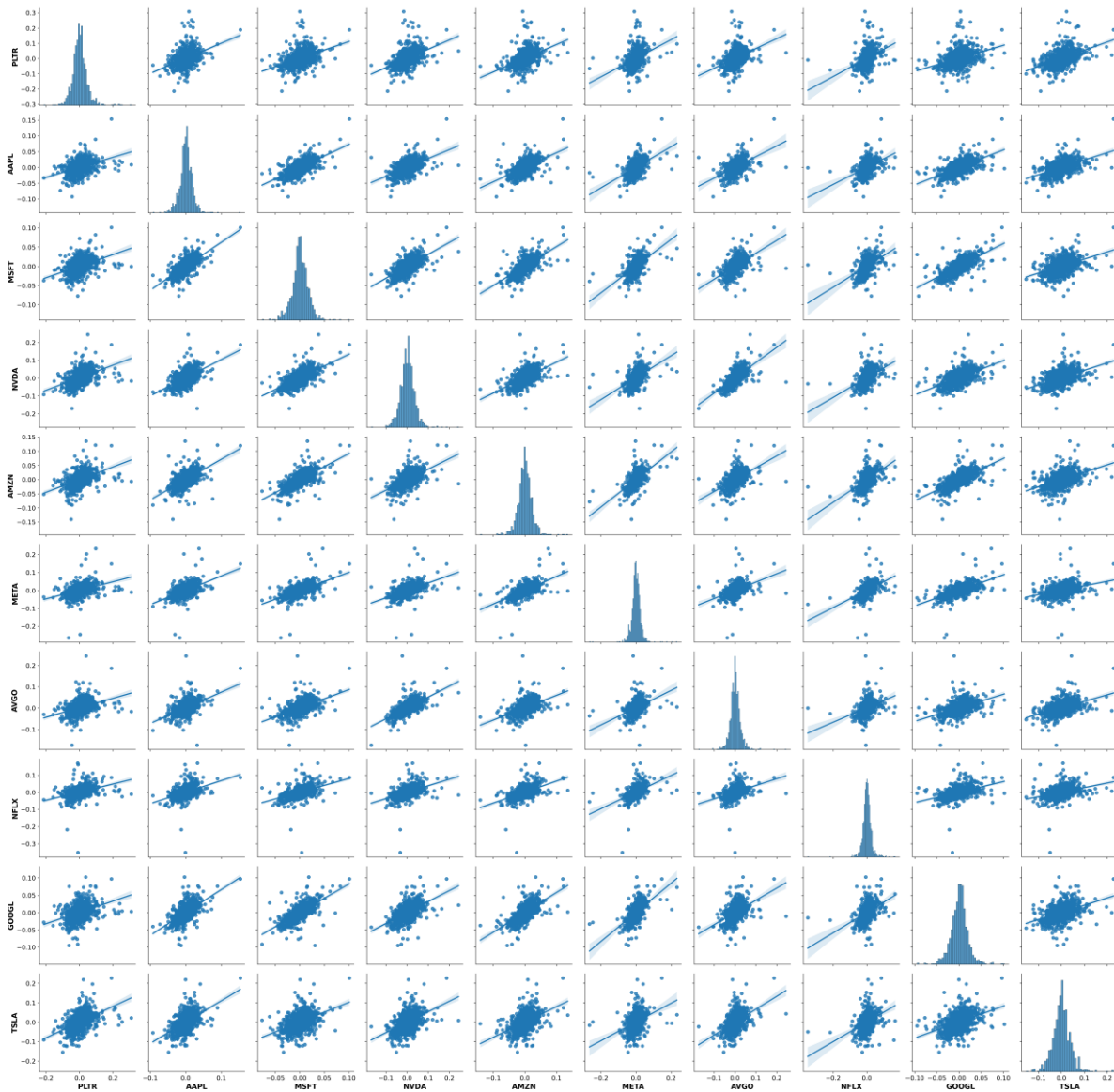
NDX	1.000000
META	0.963615
GOOGL	0.948494
NVDA	0.936465
MSFT	0.934715

Name: NDX, dtype: float64

Lowest Price Correlations with NDX:

TSLA	0.349267
PLTR	0.748714
AMZN	0.855580
NFLX	0.877488
AAPL	0.880254

Pairwise Analysis of Daily Returns: Tech Stock Correlation Matrix



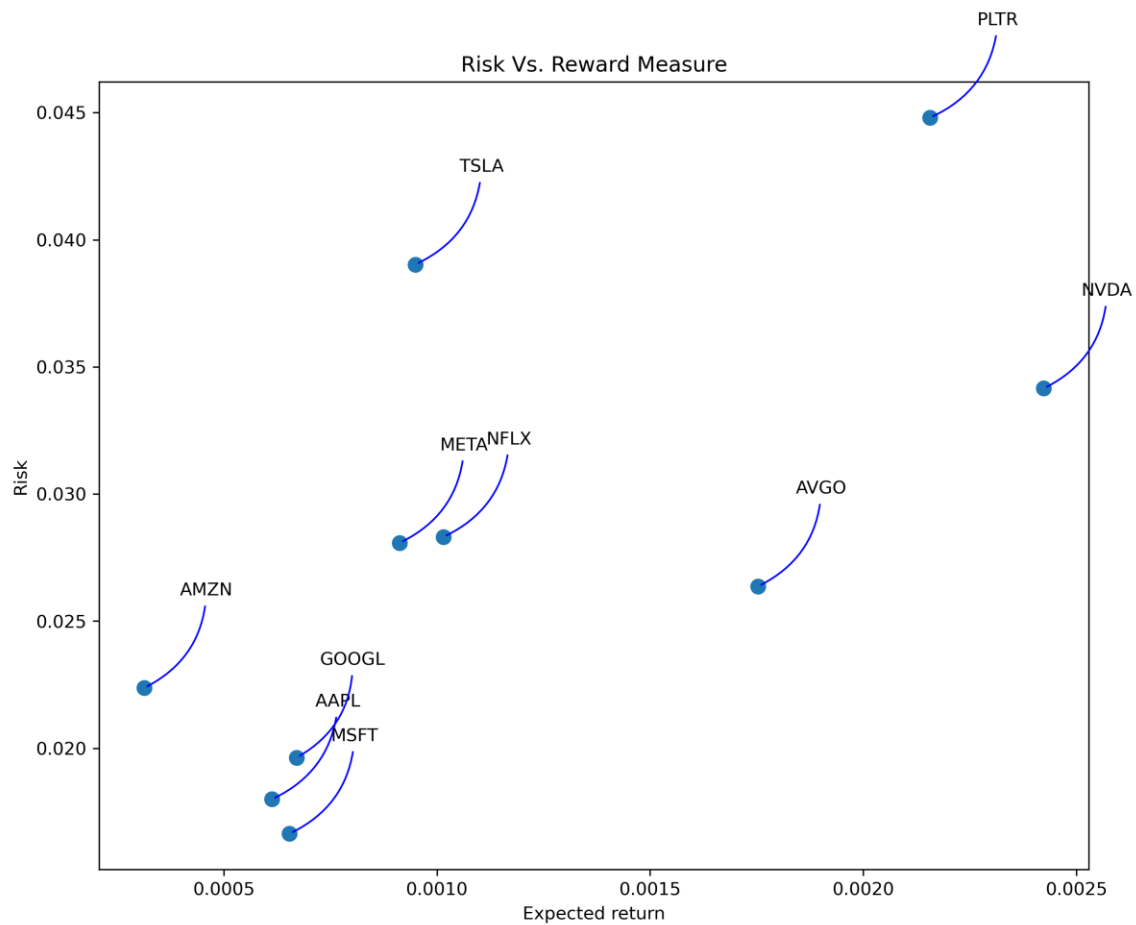
A pairwise plot of stock daily returns is shown above. From this plot, the top 5 most/least correlated pairs between stocks was calculated (see calculations below).

Top 5 Most Correlated Pairs:

	Stock 1	Stock 2	Correlation
22	MSFT	GOOGL	0.694288
18	MSFT	AMZN	0.682075
9	AAPL	MSFT	0.673739
26	NVDA	AVGO	0.660678
33	AMZN	GOOGL	0.653194

Top 5 Least Correlated Pairs:

	Stock 1	Stock 2	Correlation
38	META	TSLA	0.348691
39	AVGO	NFLX	0.364179
43	NFLX	TSLA	0.365764
7	PLTR	GOOGL	0.373507
6	PLTR	NFLX	0.377181



The above plot shows the risk at which investing in a particular stock is versus its expected return. The risk factor was calculated by the standard deviation of the daily returns. This helps show which stocks are a high or low risk investment and also whether or not the stock has a high expected return. As shown, Nvidia has the highest expected return with moderate risk, while Amazon has the lowest expected return with a low risk. Amazon appears to give slow returns, but at least it is a low-risk investment, which may be seen as a good stock to invest in for long term gains. Palantir, on the other hand, is a very high risk investment, but also has a high expected return. This high risk / high reward nature can imply that this stock is more volatile than the others.

Algorithm Selection & Analysis Results with Comparison

The goal for modeling and prediction is to predict future stock price movement (up or down) based off historical stock market data. Stock market analysis and prediction was performed using the 3 supervised classification algorithms: Decision Trees Classifier, Boosting, and Neural Networks. In our 3 models, the following features were selected as predictors: Open, High, Low, Close, and Volume. The training and test splits for each model were 70% training and 30% testing on the data. The following performance metrics were used to assess the effectiveness of each model: Accuracy, normalized accuracy, error rate, precision, recall, and F1-measure

Decision Tree Classifier

The following performance measurements were processed without added parameters such as max_depth and min_sample_size. As shown below, the best overall performance is for predicting the stock price movement of Microsoft, (MSFT) having a solid precision of 50.77%, high recall of 66.28%, and the best F1-measure of 0.5714. These performance measures suggest that the decision tree classifier works well with the Microsoft stock, but not so much with the remaining 9 stocks. Tesla (TSLA) is next best, with a normalized accuracy of slightly above randomly picking whether or not the stock price will go up or down. Although Nvidia (NVDA) has a high precision, recall is very low, suggesting that it's missing nearly all the real "up" days in its stock price movement prediction. Meta (META) has this same issue.


Decision Tree Classifier Performance by Stock (Without Parameters):

Stock	Accuracy	Normalized Accuracy	Error Rate	Precision	Recall	F1-Score
PLTR	0.4804	0.4941	0.5196	0.5077	0.1908	0.2773
AAPL	0.4592	0.4873	0.5408	0.5085	0.1667	0.2510
MSFT	0.4834	0.4760	0.5166	0.5022	0.6628	0.5714
NVDA	0.4653	0.5078	0.5347	0.8000	0.0222	0.0432
AMZN	0.5196	0.5250	0.4804	0.5610	0.2722	0.3665
META	0.4713	0.5021	0.5287	0.6000	0.0170	0.0331
AVGO	0.4924	0.5019	0.5076	0.5294	0.0533	0.0968
NFLX	0.4653	0.4777	0.5347	0.4824	0.2356	0.3166
GOOGL	0.4471	0.4925	0.5529	0.5161	0.0870	0.1488
TSLA	0.5317	0.5285	0.4683	0.5145	0.4465	0.4781

The performance metrics below show calculations with added parameters to the decision tree classifier. The added parameters are as follows:

Parameters	Max_depth=5, min_samples_split=100
------------	------------------------------------

Adding these parameters helped improve model performance, most notably the recall and F1-measure for Broadcom(AVGO). With a near-perfect 100% recall measure, predicting the real “up” days for Broadcom stock price has massively improved from its previous recall of 5.33%. Same for the F1-measure, it has significantly improved. Model performance for other stocks such as Tesla has improved as well. Adding parameters helped improved overall model performance, but there are still some stocks, like Google, with low performance that may improve with using a different method.

 Decision Tree Classifier Performance by Stock (With Added Parameters):


Stock	Accuracy	Normalized Accuracy	Error Rate	Precision	Recall	F1-Score
PLTR	0.4834	0.5050	0.5166	0.6250	0.0289	0.0552
AAPL	0.4562	0.4701	0.5438	0.5000	0.3111	0.3836
MSFT	0.5136	0.4978	0.4864	0.5184	0.9012	0.6582
NVDA	0.4653	0.5078	0.5347	0.8000	0.0222	0.0432
AMZN	0.5287	0.5362	0.4713	0.6327	0.1834	0.2844
META	0.4804	0.5091	0.5196	0.6250	0.0568	0.1042
AVGO	0.5106	0.5000	0.4894	0.5106	1.0000	0.6760
NFLX	0.4773	0.4774	0.5227	0.5030	0.4770	0.4897
GOOGL	0.4713	0.5149	0.5287	0.6216	0.1250	0.2081
TSLA	0.4864	0.4973	0.5136	0.4786	0.7736	0.5913

Handling class imbalance:

The target variable may have many more down days (0) than up days (1) presented. In order to handle this class imbalance, the following parameter was added to the decision tree classifier:

Parameter	Class_weight='balanced'
-----------	-------------------------

Below are the updated performance metrics with the added class_weight parameter. Here, stocks such as Nvidia and Google still have a poor recall and F1-score. However, model performance for Palantir improved, with a higher recall and F1-score. This suggests an improvement in predicting the real “up” days in the market for this stock. Tesla and Broadcom, on the other hand, took a big hit in recall and F1-score, making the model perform much worse for these stocks.

 Decision Tree Classifier Performance by Stock (With Added Parameters):

Stock	Accuracy	Normalized Accuracy	Error Rate	Precision	Recall	F1-Score
PLTR	0.4683	0.4727	0.5317	0.4887	0.3757	0.4248
AAPL	0.4562	0.4701	0.5438	0.5000	0.3111	0.3836
MSFT	0.5136	0.4978	0.4864	0.5184	0.9012	0.6582
NVDA	0.4653	0.5078	0.5347	0.8000	0.0222	0.0432
AMZN	0.5287	0.5362	0.4713	0.6327	0.1834	0.2844
META	0.4804	0.5091	0.5196	0.6250	0.0568	0.1042
AVGO	0.5045	0.5119	0.4955	0.5490	0.1657	0.2545
NFLX	0.4773	0.4774	0.5227	0.5030	0.4770	0.4897
GOOGL	0.4713	0.5149	0.5287	0.6216	0.1250	0.2081
TSLA	0.5287	0.5139	0.4713	0.5366	0.1384	0.2200

Resampling the training set using over/under sampling was another method used to try to help with the class imbalance. Here are the updated metrics with over vs. under sampling .

 Oversampling vs Undersampling Comparison:

Method	Accuracy		Precision		Recall		F1-Score		\
	Over	Under	Over	Under	Over	Under	Over	Under	
Stock									
AAPL	0.4562	0.4683	0.5000	0.5200	0.3111	0.2889	0.3836	0.3714	
AMZN	0.5257	0.5076	0.6000	0.7143	0.2130	0.0592	0.3144	0.1093	
AVGO	0.5076	0.4864	0.5093	0.4928	0.9704	0.2012	0.6680	0.2857	
GOOGL	0.4502	0.4713	0.5625	0.6216	0.0489	0.1250	0.0900	0.2081	
META	0.4713	0.5227	0.5122	0.5288	0.1193	0.9375	0.1935	0.6762	
MSFT	0.5136	0.5166	0.5184	0.5201	0.9012	0.9012	0.6582	0.6596	
NFLX	0.4773	0.4773	0.5030	0.5030	0.4770	0.4770	0.4897	0.4897	
NVDA	0.4622	0.4653	1.0000	0.8000	0.0111	0.0222	0.0220	0.0432	
PLTR	0.4683	0.4713	0.4887	0.4923	0.3757	0.3699	0.4248	0.4224	
TSLA	0.5136	0.5347	0.4783	0.5641	0.1384	0.1384	0.2146	0.2222	

Method	Normalized Accuracy	
	Over	Under
Stock		
AAPL	0.4701	0.4855
AMZN	0.5324	0.5172
AVGO	0.4976	0.4926
GOOGL	0.5006	0.5149
META	0.4951	0.4946
MSFT	0.4978	0.5009
NFLX	0.4774	0.4774
NVDA	0.5056	0.5078
PLTR	0.4727	0.4761
TSLA	0.4994	0.5198

As shown, recall and F1-measure significantly improved for Meta with undersampling, on the contrary, these metrics improved significantly for Broadcom(AVGO) with oversampling. Generally, Oversampling is better when wanting to preserve data and improve recall.

Undersampling is better when wanting a faster model with less risk of overfitting, but it may miss rare events.

Ensemble Method – Boosting

Boosting is an iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records. Initially, all N records are assigned equal weights. Records that are wrongly classified will have their weights increased Records that are classified correctly will have their weights decreased.

Boosting was used to combine multiple weak learners (small trees) into a strong model. It automatically handles nonlinear patterns and interactions better than a basic tree. Boosting method often improves recall and F1-score, especially on noisy financial data.

Below is a side-by-side comparison of performance metrics of the boosting method vs. the basic decision tree from the previous model.

Side-by-Side Performance Comparison: Decision Tree vs XGBoost

Stock	DT_Accuracy	DT_NormAcc	DT_Error	DT_Precision	DT_Recall	DT_F1	XGB_Accuracy	XGB_NormAcc	XGB_Error	XGB_Precision	XGB_Recall	XGB_F1
PLTR	0.4834	0.5050	0.5166	0.6250	0.0289	0.0552	0.5136	0.5221	0.4864	0.5577	0.3353	0.4188
AAPL	0.4562	0.4701	0.5438	0.5000	0.3111	0.3836	0.4562	0.4803	0.5438	0.5000	0.2056	0.2913
MSFT	0.5136	0.4978	0.4864	0.5184	0.9012	0.6582	0.4864	0.4771	0.5136	0.5041	0.7151	0.5913
NVDA	0.4653	0.5078	0.5347	0.8000	0.0222	0.0432	0.4592	0.5022	0.5408	0.6667	0.0111	0.0219
AMZN	0.5287	0.5362	0.4713	0.6327	0.1834	0.2844	0.5106	0.5151	0.4894	0.5368	0.3018	0.3864
META	0.4804	0.5091	0.5196	0.6250	0.0568	0.1042	0.4743	0.4587	0.5257	0.5041	0.7045	0.5877
AVGO	0.5106	0.5000	0.4894	0.5106	1.0000	0.6760	0.4924	0.4978	0.5076	0.5062	0.2426	0.3280
NFLX	0.4773	0.4774	0.5227	0.5030	0.4770	0.4897	0.5045	0.5175	0.4955	0.5610	0.2644	0.3594
GOOGL	0.4713	0.5149	0.5287	0.6216	0.1250	0.2081	0.4562	0.4951	0.5438	0.5400	0.1467	0.2308
TSLA	0.4864	0.4973	0.5136	0.4786	0.7736	0.5913	0.5196	0.5193	0.4804	0.5000	0.5094	0.5047

The Boosting method helped improve the recall and f-measure for Palantir and Amazon, but saw a decrease of performance with many of the other stocks such as Microsoft, Tesla, and most notably Broadcom(AVGO). These finding go against the intended goal of the boosting method, which is to improve recall and F1-measures, it resulted in the opposite for many of the stocks compared to the decision tree.

Artificial Neural Networks (ANN)

Traditional models like linear regression or ARIMA focus on linear relationships, which can often oversimplify the complexity of the stock market. Neural networks, on the other hand, can detect non-linear patterns. This gives them a distinct edge in stock market predictions where volatility and rapid changes are the norm. A 3-hidden layer model was applied in attempt to capture any complex patterns in the stock price movements. A max_iteration of 1000 was also assigned to the model in order to give it enough time to converge.

Neural Network Performance by Stock:

Stock	Accuracy	Normalized Accuracy	Error Rate	Precision	Recall	F1-Score
PLTR	0.5378	0.5372	0.4622	0.5588	0.5491	0.5539
AAPL	0.4562	0.4829	0.5438	0.5000	0.1778	0.2623
MSFT	0.5076	0.5062	0.4924	0.5254	0.5407	0.5330
NVDA	0.4562	0.4989	0.5438	0.5000	0.0111	0.0217
AMZN	0.5317	0.5391	0.4683	0.6400	0.1893	0.2922
META	0.4713	0.5028	0.5287	1.0000	0.0057	0.0113
AVGO	0.5015	0.5115	0.4985	0.7000	0.0414	0.0782
NFLX	0.4924	0.5120	0.5076	0.5750	0.1322	0.2150
GOOGL	0.4743	0.5210	0.5257	0.6786	0.1033	0.1792
TSLA	0.5076	0.5079	0.4924	0.4881	0.5157	0.5015

The above metrics show that the ANN performed generally well on predicting stock price movements, with normalized accuracies slightly above random. However, this model does not outperform the previous decision tree or boosting methods in many areas, such as recall and F1

measures. With the accuracies high and recalls low, the model may be favoring the majority class (e.g. predicting too many “down” days). The high precision measures suggests the ANN model usually predicts the “up” days correctly. The low recall values, however, suggests that the model is missing out on some profitable opportunities for investors, but many actual “up” days were missed in the model.

Conclusion

This project investigated the use of machine learning techniques to analyze and predict stock price movements among 10 leading NASDAQ-100 companies: PLTR, AAPL, MSFT, NVDA, AMZN, META, AVGO, NFLX, GOOGL, and TSLA. The goal was to assess how well various models could forecast next-day price direction (up or down) using historical stock data sourced from Yahoo Finance’s API (yfinance).

- **Decision Trees** provided decent performance for some stocks like **Microsoft (MSFT)** and **Tesla (TSLA)**, especially after parameter tuning. However, they struggled with recall on more volatile or less correlated stocks like NVDA and META.
 - Adding **class_weight='balanced'** helped improve model performance on some underperforming stocks (e.g., PLTR) but hurt others (e.g., TSLA and AVGO), showing that class imbalance handling must be stock-specific.
 - **Oversampling** generally improved **recall and F1** (especially for Broadcom), while **undersampling** helped Meta but reduced overall data size and may miss rare but important events.
- **Boosting**, despite its theoretical advantages, showed **mixed results**. It improved performance for **Palantir and Amazon**, but decreased it for **Microsoft, Tesla**, and **Broadcom**, highlighting that boosting may not always generalize better for stock prediction.
- The **Artificial Neural Network (ANN)**, while capable of modeling non-linear relationships, yielded **only moderate improvements**. It showed high **precision** but **low recall**, suggesting it was overly conservative — catching “up” days when confident but missing many others. This makes it less effective for aggressive trading strategies seeking to capitalize on every opportunity.

No single model consistently outperformed the others across all stocks. Model performance varies significantly by stock, likely due to different volatility levels, investor behavior, and sensitivity to market news.

- Future work could include:
 - Incorporating technical indicators (RSI, MACD, moving averages)
 - Exploring LSTM networks for sequential modeling

- Adding sentiment analysis from financial news or social media
- Using ensemble blending of all models for improved generalization

This project demonstrates the challenges and potential of applying machine learning to financial forecasting and highlights the importance of model selection, feature engineering, and data handling in achieving meaningful predictive performance in the stock market.

Literature Review

For nearly 45 years, there have been numerous methodologies that predicted the stock market forecasting models; according to Jimenez, K. C., Significant improvements that closed the leverage on the net loss income within a volatile market by applying nonlinear models such as AutoRegressive Integrated Moving Average (ARIMA), which is a statistical model used for analyzing and forecasting time series data. In addition, the Markov chains were another method to predict markets better, soon after neural networks, including Long Short-Term Memory (LSTM), Ruh & B.B. These methodologies have been facilitated with other innovative machine-learning capabilities.

Neural network strategies outperformed the generalized regression networks and the probabilistic neural network. These networks enhanced the support vector machines (SVMS) due to their strong predictability performance metrics used in predicting the movement of stock market indices, as stated by Ayyildiz, N., & Iskenderoglu, O. However, Artificial Neural Networks (ANNs) had the best overall average accuracy ratio. There were attempts to improve upon the accuracy when combining algorithms. Anjani et al. researched the significant contributions of the K-Nearest Neighbors (KNN) algorithm to predict outcomes based on the datasets. Still, there were drawbacks with this algorithm because it could not accurately account for non-centric data points, which led to underfitting and a reduction in accuracy.

Anjani et al. supplemented KNN by adding probabilistic techniques like the Bayes theorem to reduce the error from the non-centric data points. So, this hybrid KNN-Probabilistic model had a higher accuracy than the ordinary KNN. They contend that switching models from binary to multiclass would warrant better results. Further advancement of combining algorithms to mitigate the unpredictability of the stock market focused on ANNs and random forest techniques to better predict closing stock prices, according to Mehar Vijh. These techniques showed lower RMSE and MAPE values and increased accuracy. At the same time, Singh et al. found that if they continued applying the ANNs iterations a reasonable number of times by running several valuation tests, changing the percentages for both the training sets and test sets.

The authors acknowledge that machine learning is the new normative way of approaching stock market predictions, as Eslamieh et al. affirm. She repeatedly revealed how LSTM, CNNs, and ANNs are the most robust tools. CNNs can manage complex computational sequences that can

build an initial training set with high proficiency, while CNNs and ANNs will provide better 3D images and financial prediction tools. Many authors used pre-processing steps, training sets, and test sets like ours to ensure higher accuracy in validating the models. Anjani et al. and Eslamieh et al. both implemented binary classification models like ours, whereas we will test our model for accuracy and

F-measure, unlike Eslamieh et al., who performed all four measurements.

Works Cited

Anjani, C., Sushma, Kasoju., Bhavani, Kethu.

“Stock Market Prediction Using K-Nearest Neighbors (KNN) Algorithm

AES Volume 54 Issue 02, July, (2022) 977-985.

Ayyildiz, Nazif., Iskenderoglu, Omar.

“How effective is machine learning in stock market predictions?”

Science Direct Heliyon 10 (2024) es24123 1-10.

Eslamieh, Peyah., Shajari, Mehdi., Nickabadi, Ahmad.

“User 2Vec: A Novel Representation for the Information of the Social Networks for Stock Market Prediction Using Convolutional and Recurrent Neural Networks”

Mathematics 1 July (2023) 1-26.

Jimenez, C. K.

“Neural Network-Based Predictive Models for Stock Market Index Forecasting”

Journal of Risk and Financial Management 11 June, (2024) 1-18.

Kumawat, K. S., Bansal, Alok., Saini, S. S.

“Design Analysis and Implementation of Stock Market Forecasting System Using Improved Soft Computing Technique”

International Journal on Future Revolution in Computer Science & Communication Engineering (IJFRCSCE) Volume 8, Issue 4 (2022) 9-15.

