Hunter Fernandez, Dexter Schincke

Bellevue University

DSC 630-T301

**Milestone 4\_Fernandez\_Schincke**

October 30, 2024

# NVIDIA Stock Movement Prediction - Milestone 4

[1]: *# Imports all the necessary packages* **import polars as pl import numpy as np**

**from sklearn.preprocessing import** MinMaxScaler **from sklearn.linear\_model import** LogisticRegression **from sklearn.ensemble import** RandomForestClassifier **from sklearn.model\_selection import** train\_test\_split **from sklearn.metrics import** accuracy\_score, precision\_score, recall\_score,␣

↪f1\_score, roc\_auc\_score **from sklearn.model\_selection import** GridSearchCV **from tensorflow.keras.models import** Sequential **from tensorflow.keras.layers import** LSTM, Dense **import warnings**

*# Disables the warnings* warnings.filterwarnings('ignore')

## 1.0.1 Importing the data

[2]:

*# Reads in the csv file*

df

=

pl

.

read\_csv(

"

data/main/NVDA.csv

"

, try\_parse\_dates

=

**True**

)

*# Sorts by date*

df

=

df

.

sort(by

=

[

"

Date

"

])

df

.

head()

[2]:

shape: (5, 7)

Date Open High Low Close Adj Close Volume

--- --- --- --- --- --- ---

date f64 f64 f64 f64 f64 i64

|  |  |
| --- | --- |
|  |  |
| 2000-01-03 0.984375 0.992188 0.919271 0.97526 0.894608 | 30091200 |
| 2000-01-04 0.958333 0.960938 0.901042 0.949219 0.870721 | 30048000 |
| 2000-01-05 0.921875 0.9375 0.904948 0.917969 0.842055 | 18835200 |
| 2000-01-06 0.917969 0.917969 0.822917 0.858073 0.787112 | 12048000 |
| 2000-01-07 0.854167 0.88151 0.841146 0.872396 0.800251 | 7118400 |

## 1.0.2 Checking for missing values

[3]:

*# Check for missing values*

df

.

describe()

[3]: shape: (9, 8)

|  |  |  |  |
| --- | --- | --- | --- |
| statistic Date Open High  Close Volume | Low | Close | Adj |
| --- --- --- ---  --- | --- | --- | --- |
| str str f64 f64  f64 | f64 | f64 | f64 |
| count 6116 6116.0 6116.0  6116.0 6116.0 | 6116.0 | 6116.0 |  |
| null\_count 0 0.0 0.0  0.0 | 0.0 | 0.0 | 0.0 |
| mean 2012-02-28 53.052266 54.017201 52.0317  52.794253 6.2219e7 | | 53.064741 | |
| 11:27:16.3 | |  | |
| 63000 | |  | |

std null 121.267334 123.42398 118.83511 121.18323

121.21486 4.3167e7

2 4 4 4

min 2000-01-03 0.608333 0.656667 0.6 0.614167

0.563377 4.5644e6

|  |  |  |
| --- | --- | --- |
| 25% 2006-02-02 2.96  2.708334 3.61608e7 | 3.0275 | 2.875 2.950521 |
| 50% 2012-02-29 4.685  4.389289 5.20639e7 | 4.7475 | 4.61 4.6825 |
| 75% 2018-03-27 42.099998  41.730057 7.46548e7 | 42.645 | 41.4925 42.099998 |
| max 2024-04-24 958.51001  950.02002 9.230856e | 974.0 | 935.09997 950.02002 |
| 8 |  | 6 |
|  | |

There are no missing values in our dataset.

## 1.0.3 Data Preparation/Feature Engineering

[4]:

*# Create a lag shift column to show the previous day's closing price.*

df

=

df

.

with\_columns(prev\_close

=

pl

.

col(

"

Close

"

)

.

shift(

1

))

*# Removes records without a previous close price*

df

=

df

.

drop\_nulls()

*# Normalizing the Close and Volume columns.*

scaler

=

MinMaxScaler()

df[[

'

Close

'

,

'

Volume

'

,

'

prev\_close

'

]]

=

scaler

.

fit\_transform(df[[

'

Close

'

,

␣

↪

'

Volume

'

,

'

prev\_close

'

]])

*# Creates a target column that shows if the stock price increased or decreased*

␣

↪

*from the previous day.*

df

=

df

.

with\_columns(target

=

(

pl

.

col(

"

Close

"

)

>

pl

.

col(

"

prev\_close

"

))

.

cast(pl

.

↪

Int8))

*# Selects the features and target columns*

X

=

df

.

select(

"

prev\_close

"

,

"

Volume

"

)

y

=

df

.

select(

"

target

"

)

*# Splits the data into training and testing sets*

X\_train, X\_test, y\_train, y\_test

=

train\_test\_split(X, y, test\_size

=

0.2

,

␣

↪

random\_state

=

42

)

df

.

head()

[4]:

shape: (5, 9)

Date Open High Low … Adj Close Volume

|  |  |
| --- | --- |
| prev\_close target |  |
| --- --- --- --- ---  --- | --- --- |
| date f64 f64 f64 f64  i8 | f64 f64 |
| 2000-01-04 0.958333 0.960938 0.901042 … 0.870721  0.00038 0 | 0.027744 |
| 2000-01-05 0.921875 0.9375 0.904948 … 0.842055  0.000353 0 | 0.015537 |
| 2000-01-06 0.917969 0.917969 0.822917 … 0.787112  0.00032 0 | 0.008147 |
| 2000-01-07 0.854167 0.88151 0.841146 … 0.800251  0.000257 1 | 0.002781 |
| 2000-01-10 0.875 0.9375 0.859375 … 0.826528  0.000272 1 | 0.021144 |

## 1.0.4 Logistic Regression

[5]:

*# Creates a logistic regression model*

model

=

LogisticRegression()

*# Fits the model*

model

.

fit(X\_train, y\_train)

*# Predicts the target values*

y\_pred

=

model

.

predict(X\_test)

## 1.0.5 Model Evaluation

Use common metrics to evaluate model performance

[6]:

*# Calculate metrics*

accuracy

=

accuracy\_score(y\_test, y\_pred)

precision

=

precision\_score(y\_test, y\_pred)

recall

=

recall\_score(y\_test, y\_pred)

f1

=

f1\_score(y\_test, y\_pred)

roc\_auc

=

roc\_auc\_score(y\_test, y\_pred)

*# Print the evaluation metrics*

print

(

f

'

Accuracy:

**{**

accuracy

**:**

.2

f

**}**

'

)

print

(

f

'

Precision:

**{**

precision

**:**

.2

f

**}**

'

)

print

(

f

'

Recall:

**{**

recall

**:**

.2

f

**}**

'

)

print

(

f

'

F1 Score:

**{**

f1

**:**

.2

f

**}**

'

)

print

(

f

'

ROC-AUC:

**{**

roc\_auc

**:**

.2

f

**}**

'

)

Accuracy: 0.52

Precision: 0.52

Recall: 1.00

F1 Score: 0.68

ROC-AUC: 0.50

The accuracy of 0.52 suggests that it performs slightly better than random. Precision is also 0.52, indicating moderate reliability when predicting positives. A recall of 1.00 reflects that the model successfully identifies all actual positives, likely at the cost of increased false positives. The F1 score of 0.68 represents a reasonable balance between precision and recall but could be improved. However, the ROC-AUC score of 0.50 suggests the model is currently no better than random in distinguishing between classes.

## 1.0.6 Random Forest Model

[7]: *# This is the grid search for the Random Forest Classifier to select the best*␣

↪*n\_estimators* param\_grid = {'n\_estimators': [50, 100, 150, 200, 250]} grid\_search = GridSearchCV(RandomForestClassifier(random\_state=42), param\_grid,␣

↪cv=5) grid\_search.fit(X\_train, y\_train) best\_n\_estimators = grid\_search.best\_params\_['n\_estimators'] best\_n\_estimators

[7]: 200

[8]: *# Trains the Random Forest model* rf\_model = RandomForestClassifier(n\_estimators=200, random\_state=42) rf\_model.fit(X\_train, y\_train)

*# Make predictions*

y\_rf\_pred = rf\_model.predict(X\_test)

*# Evaluate Random Forest model*

rf\_accuracy = accuracy\_score(y\_test, y\_rf\_pred) rf\_precision = precision\_score(y\_test, y\_rf\_pred) rf\_recall = recall\_score(y\_test, y\_rf\_pred) rf\_f1 = f1\_score(y\_test, y\_rf\_pred) rf\_roc\_auc = roc\_auc\_score(y\_test, y\_rf\_pred)

*# Print the evaluation metrics for Random Forest* print(f'Random Forest Accuracy: **{**rf\_accuracy**:**.2f**}**') print(f'Random Forest Precision: **{**rf\_precision**:**.2f**}**') print(f'Random Forest Recall: **{**rf\_recall**:**.2f**}**') print(f'Random Forest F1 Score: **{**rf\_f1**:**.2f**}**') print(f'Random Forest ROC-AUC: **{**rf\_roc\_auc**:**.2f**}**') Random Forest Accuracy: 0.51

Random Forest Precision: 0.52

Random Forest Recall: 0.53

Random Forest F1 Score: 0.53

Random Forest ROC-AUC: 0.51

With an accuracy of 0.51, the model is performing just above random chance, classifying only about half of the cases correctly. Precision and recall are similarly low at 0.52 and 0.53, suggesting that while it captures some true positives, it also misclassifies a comparable number of false positives. The F1 score of 0.53 reflects this imbalance, indicating limited trade-off between precision and recall. The ROC-AUC score, close to 0.51, shows minimal class separation, underscoring the model’s difficulty in distinguishing between classes.

## 1.0.7 LSTM model

[9]: *# Reshape the data to be compatible with LSTM input requirements.* X\_lstm = np.array(X).reshape((X.shape[0], 1, X.shape[1]))

*# Split the dataset into training and testing sets for LSTM.*

X\_train\_lstm, X\_test\_lstm, y\_train\_lstm, y\_test\_lstm = train\_test\_split(X\_lstm,␣ ↪y, test\_size=0.2, random\_state=42)

*# Build the LSTM model using the Sequential API.* lstm\_model = Sequential()

*# Add an LSTM layer with 50 units and define the input shape.* lstm\_model.add(LSTM(50, input\_shape=(X\_train\_lstm.shape[1], X\_train\_lstm. ↪shape[2])))

*# Add a Dense layer with a sigmoid activation function for binary*␣ ↪*classification.*

lstm\_model.add(Dense(1, activation='sigmoid'))

*# Compile the model.*

*# Using the Adam optimizer for its adaptive learning rate capabilities, which*␣ ↪*helps achieve faster convergence and stability during training.*

*# Binary cross-entropy loss is chosen for binary classification, as it*␣ ↪*effectively measures the difference between the actual and predicted class*␣ ↪*probabilities.*

*# It works well with the sigmoid activation function used in the output layer.*

lstm\_model.compile(optimizer='adam', loss='binary\_crossentropy',␣ ↪metrics=['accuracy'])

*# Train the LSTM model on the training dataset.*

*# The model will be trained for 10 epochs with a batch size of 32 as a good*␣ ↪*starting point without overfitting.*

lstm\_model.fit(X\_train\_lstm, y\_train\_lstm, epochs=10, batch\_size=32)

*# The predictions are thresholded at 0.5 to convert probabilities to binary*␣ ↪*class labels.* y\_lstm\_pred = (lstm\_model.predict(X\_test\_lstm) > 0.5).astype("int32")

*# Calculate evaluation metrics for the LSTM model's performance.*

lstm\_accuracy = accuracy\_score(y\_test\_lstm, y\_lstm\_pred) lstm\_precision = precision\_score(y\_test\_lstm, y\_lstm\_pred) lstm\_recall = recall\_score(y\_test\_lstm, y\_lstm\_pred) lstm\_f1 = f1\_score(y\_test\_lstm, y\_lstm\_pred) lstm\_roc\_auc = roc\_auc\_score(y\_test\_lstm, y\_lstm\_pred)

*# Print the evaluation metrics for the LSTM model.* print(f'LSTM Accuracy: **{**lstm\_accuracy**:**.2f**}**') print(f'LSTM Precision: **{**lstm\_precision**:**.2f**}**') print(f'LSTM Recall: **{**lstm\_recall**:**.2f**}**') print(f'LSTM F1 Score: **{**lstm\_f1**:**.2f**}**') print(f'LSTM ROC-AUC: **{**lstm\_roc\_auc**:**.2f**}**')

Epoch 1/10

**153/153**  **1s** 487us/step accuracy: 0.5094 - loss: 0.6929

Epoch 2/10

**153/153**  **0s** 385us/step accuracy: 0.5124 - loss: 0.6928

Epoch 3/10

**153/153**  **0s** 376us/step accuracy: 0.5256 - loss: 0.6919

Epoch 4/10

**153/153**  **0s** 377us/step accuracy: 0.5186 - loss: 0.6923

Epoch 5/10

**153/153**  **0s** 378us/step accuracy: 0.5048 - loss: 0.6931

Epoch 6/10

**153/153**  **0s** 395us/step accuracy: 0.5114 - loss: 0.6926

Epoch 7/10

**153/153**  **0s** 475us/step accuracy: 0.5132 - loss: 0.6926

Epoch 8/10

**153/153**  **0s** 380us/step accuracy: 0.5249 - loss: 0.6922

Epoch 9/10

**153/153**  **0s** 375us/step accuracy: 0.5186 - loss: 0.6920

Epoch 10/10

**153/153**  **0s** 381us/step accuracy: 0.5051 - loss: 0.6929

**39/39**  **0s** 1ms/step

LSTM Accuracy: 0.52

LSTM Precision: 0.52

LSTM Recall: 1.00

LSTM F1 Score: 0.68

LSTM ROC-AUC: 0.50

The LSTM model shows an accuracy of 0.52, which is only slightly better than random guessing. Both precision and F1 score are at 0.52 and 0.68, indicating some success in identifying positive predictions but also a struggle with false positives. The recall is perfect at 1.00, suggesting the model identifies all true positives, though this may indicate overfitting. Finally, the ROC-AUC score of 0.50 shows limited class distinction.

**Conclusion**

The performance analysis of the three models—Logistic Regression, Random Forest, and LSTM— provides valuable insights into their ability to predict NVIDIA’s stock price movements. The Logistic Regression model shows a perfect recall of 1.00, which means it captures every actual stock price increase. However, it falls short in accuracy and precision, both sitting at 0.52. This indicates that while the model identifies positive cases effectively, it misclassifies a significant number of negative instances, leading to a high rate of false positives. This kind of misclassification can be a real headache for investors, as it might give them the wrong impression that a stock price increase is likely when it isn’t.

On the other hand, the Random Forest model records an accuracy of 0.51, a precision of 0.52, and a recall of 0.53. These numbers suggest that it struggles to accurately classify stock price movements, failing to recognize both actual price increases and avoiding false positives. Its relatively low scores in precision and recall indicate that it’s not performing efficiently when it comes to identifying true positives.

The LSTM model mirrors the performance of Logistic Regression, boasting a recall of 1.00 but also showing the same low accuracy and precision scores. Like Logistic Regression, it effectively identifies all stock price increases but misclassifies many downward movements, as reflected in its F1 score. While LSTM models are usually great at capturing sequential patterns in time-series data, in this case, it seems like it’s not fully utilizing that capability.

There’s plenty of room for improvement across all models. A major area to focus on is feature engineering. Adding more features, such as market sentiment, macroeconomic indicators, or company news, could really boost model accuracy by providing deeper insights into stock price movements. We could also experiment with techniques like feature selection or dimensionality reduction to help clean up the data and focus on the most impactful predictors.

Additionally, utilizing hyperparameter tuning and optimizing model parameters could significantly enhance the Random Forest and LSTM models. Techniques like grid search or random search could help pinpoint the most effective configurations for these models, making them more powerful.

From a business perspective, it’s crucial to prioritize a higher precision score, especially if the goal is to make financial decisions based on predicted stock price increases. A model with high precision means that when it forecasts a price increase, it’s more likely to be correct, which helps minimize the risk of overly optimistic predictions that could lead to poor investment choices. However, if the focus shifts to minimizing missed opportunities, we should aim to improve recall to ensure that the model captures a wider range of stock price increases.

In conclusion, by striking a careful balance between precision, recall, and overall model performance, I believe we can enhance decision-making in stock market predictions. Continuing to explore ways to improve our models, while strategically incorporating additional data and refining existing features, will lead us to a more robust and reliable predictive framework for NVIDIA’s stock movements.

This will ultimately empower stakeholders to make more informed investment decisions.

**Milestone 3**

**Introduction**

The semiconductor industry is crucial for pushing technology forward, with NVIDIA standing out as a major company in GPUs and AI hardware. Their innovations, especially with AI-focused chips, have significantly impacted everything from gaming to AI research. Launching new GPU generations, like the GTX and RTX series, has caused noticeable shifts in NVIDIA's stock price.

In this project, we will investigate how technological breakthroughs, particularly NVIDIA's AI-powered RTX chips, have influenced their stock price movement. Rather than predicting exact future prices, we will focus on whether the stock price will likely go up or down based on past data. We will also look at how big events, such as the release of the RTX series compared to the earlier GTX series, affect stock trends. By exploring these connections, our study will shed light on how tech advancements impact investor behavior and market reactions, offering valuable insights for financial analysts and those following the tech sector.

**Data**

The dataset for this project is centered on historical stock data for NVIDIA (NVDA). The data includes key stock market variables such as:

-          Date: The trading date.

-          Open: NVIDIA stock's opening price.

-          High: The highest price recorded during the trading day.

-          Low: The lowest price recorded during the trading day.

-          Close: The final price at the close of the trading day.

-          Adjusted Close: The closing price is adjusted for corporate actions, such as stock splits and dividends.

-          Volume: The number of shares traded on that day.

**Model Selection**

To predict NVIDIA's stock price movement, we will evaluate the following models:

-          Logistic Regression: This will classify price movements as increases or decreases based on historical data, serving as a baseline for binary classification.

-          LSTM (Long Short-Term Memory): This recurrent neural network will analyze historical trends to forecast future price direction, considering short-term and long-term patterns.

-          Random Forest Classification: This algorithm will handle complex interactions and non-linear relationships to predict stock price movements using features like historical prices, trading volume, and price volatility.

These models will be tested to assess their effectiveness in predicting NVIDIA's stock price direction, offering a range of traditional and advanced methods.

**Why these Models?**

We chose these models because of their unique strength in predicting stock price movements. Logistic Regression gives us a simple baseline for classifying price changes. LSTM is excellent at capturing long-term patterns and time-based trends through deep learning. Random Forest handles complex, non-linear relationships and interactions, like price volatility and trading volume. By testing these models, we can compare their performance and determine which approach works best for forecasting stock prices.

**Evaluation of Results**

We will use several metrics to evaluate and ensure accurate predictions of NVIDIA's stock price movements. Accuracy will measure the proportion of correct predictions for classification models like Logistic Regression and Random Forest. Precision and recall will assess how well the models predict upward or downward movements, with precision focusing on the correctness of predicted movements and recall on identifying actual movements. The F1 score will balance precision and recall, particularly for imbalanced classes. AUC-ROC will evaluate how well the models distinguish between price increases and decreases. Lastly, cross-validation will help prevent overfitting by assessing model performance across different data splits. These metrics will guide us in selecting the most effective model for predicting stock price directions.

**Learning Objectives**

This project aims to deepen our understanding of the factors influencing NVIDIA's stock price movements by analyzing fundamental variables such as trading volume, volatility, and major product releases. Through this analysis, we seek to identify the elements that significantly drive stock price increases or decreases. Additionally, by comparing models like Logistic Regression, LSTM, and Random Forest, we aim to determine which algorithms are most effective for predicting stock price movements. The project will also help refine our skills in classification-based modeling and machine learning techniques, enabling us to enhance our predictive capabilities within financial markets. Ultimately, we expect to develop a model that accurately predicts stock price movements while providing insights into the behavior of NVIDIA's stock about technological advancements.

**Risks**

This project faces several risks that must be carefully managed. One critical risk is overfitting, where the model may perform well on historical data but needs to generalize to new, unseen data. To address this, we will employ cross-validation techniques and avoid using overly complex models. Another concern is data quality; inaccuracies or missing data in the historical stock dataset could lead to flawed predictions. We will use thorough data preprocessing and validation steps to mitigate this risk and improve the model's reliability.

**Ethical Concerns**

There are also ethical considerations to keep in mind for this project. One significant concern is market manipulation. The developed predictive models could be used for unethical purposes, such as manipulating stock prices. While this project is for educational purposes, it is essential to consider the broader implications of sharing predictive models. Additionally, data privacy becomes another concern if we decide to integrate sentiment analysis from social media or financial reports. We must ensure that any external data sources comply with privacy regulations to avoid misuse of personal information.

**Contingency Plan**

Suppose the initial project fails to deliver the expected results. In that case, we will pivot to a predictive analytics project using an NCAA dataset containing information on sports participation, revenues, expenses, and enrollment at various institutions. The alternative project will focus on predicting the financial performance of sports programs based on participation numbers and other relevant factors. Similar preprocessing steps will be applied, such as handling missing data and feature engineering and testing models like linear Regression and random forest to forecast revenue and expenses. This pivot will allow us to apply the same machine-learning techniques in a different domain.

**Exploratory Data Analysis**

After diving into the NVIDIA stock data, we believe the data will be insufficient to answer our main questions by itself. The dataset gives us ample coverage of key variables, like daily prices and trading volumes, which should provide insights into how the stock fluctuates around major product launches. However, incorporating external factors, like major technological releases (e.g., RTX or GTX series) and market sentiment, could give us a clearer picture of how outside influences impact stock prices. Adding event markers for significant product releases and analyzing how prices and volumes behave before and after these launches will help us capture the direct effects of those events. Market sentiment data sourced from news articles or social media could further enhance our analysis by offering a deeper look into how public perception impacts price movements. Sentiment analysis using NLP techniques could quantify this data and give us a more nuanced understanding.

For visualizations, we still think line charts and bar charts will be useful for showing trends over time, but candlestick charts could offer a more detailed view of stock price movements, particularly around key dates like product launches. Additionally, we plan to generate a correlation heatmap to analyze the relationships between variables like volume, volatility, and price movements against the new sentiment data. We could also include metrics like daily price volatility (high vs. low) to better understand stock fluctuations and how they relate to major events.

While the data covers most of our needs, marking specific event dates for product launches will be a key adjustment to help us analyze how the stock behaves before and after these milestones. Along with this, we are considering creating lagged features (e.g., previous day’s close or volume) to account for time-dependent relationships in the data. This will help capture the typical lag effects seen in stock market behavior.

The models we picked (Logistic Regression, LSTM, and Random Forest) still feel like the right choice for predicting stock movements. However, we may tweak our evaluation metrics depending on how the data behaves. If we find a heavy imbalance (like mostly upward price movements), precision, recall, and F1 score might provide a more accurate reflection of model performance, rather than relying solely on accuracy.

Overall, we are still aligned with our original expectations. NVIDIA’s tech breakthroughs, especially the AI-driven RTX chips, seem to have a noticeable impact on stock prices. However, we need to ensure these events are truly driving price movements and not just coinciding with broader market or economic trends.

**Conclusion**

This project aims to predict NVIDIA's stock price movements by leveraging predictive analytics models, such as Logistic Regression, LSTM, and Random Forest Classification. By focusing on how technological innovations, like the release of NVIDIA's AI-powered RTX chips, have influenced investor behavior, we seek to uncover patterns in stock market reactions. The project will evaluate short- and long-term trends using key metrics such as accuracy, precision, recall, F1 score, and AUC-ROC. These evaluations will guide us toward identifying the most effective model for predicting stock price movements.

We have carefully considered risks such as overfitting and data quality, with plans for cross-validation and robust data preprocessing. Ethical concerns, especially around market manipulation and data privacy, are also addressed, ensuring the models remain responsibly used. Suppose the stock price prediction project does not yield satisfactory results. In that case, we are prepared to pivot to a backup project focused on predicting the financial performance of NCAA sports programs, applying similar machine learning techniques.

Overall, this project will enhance our understanding of stock market dynamics and technological advancements, refining our forecasting techniques and providing actionable insights for financial analysts and investors.

**Milestone 2**

**Introduction**

The semiconductor industry is crucial for pushing technology forward, with NVIDIA standing out as a major company in GPUs and AI hardware. Their innovations, especially with AI-focused chips, have significantly impacted everything from gaming to AI research. Launching new GPU generations, like the GTX and RTX series, has caused noticeable shifts in NVIDIA's stock price.

In this project, we will investigate how technological breakthroughs, particularly NVIDIA's AI-powered RTX chips, have influenced their stock price movement. Rather than predicting exact future prices, we will focus on whether the stock price will likely go up or down based on past data. We will also look at how big events, such as the release of the RTX series compared to the earlier GTX series, affect stock trends. By exploring these connections, our study will shed light on how tech advancements impact investor behavior and market reactions, offering valuable insights for financial analysts and those following the tech sector.

**Data**

The dataset for this project is centered on historical stock data for NVIDIA (NVDA). The data includes key stock market variables such as:

-          Date: The trading date.

-          Open: NVIDIA stock's opening price.

-          High: The highest price recorded during the trading day.

-          Low: The lowest price recorded during the trading day.

-          Close: The final price at the close of the trading day.

-          Adjusted Close: The closing price is adjusted for corporate actions, such as stock splits and dividends.

-          Volume: The number of shares traded on that day.

**Model Selection**

To predict NVIDIA's stock price movement, we will evaluate the following models:

-          Logistic Regression: This will classify price movements as increases or decreases based on historical data, serving as a baseline for binary classification.

-          LSTM (Long Short-Term Memory): This recurrent neural network will analyze historical trends to forecast future price direction, considering short-term and long-term patterns.

-          Random Forest Classification: This algorithm will handle complex interactions and non-linear relationships to predict stock price movements using features like historical prices, trading volume, and price volatility.

These models will be tested to assess their effectiveness in predicting NVIDIA's stock price direction, offering a range of traditional and advanced methods.

**Why these Models?**

We chose these models because of their unique strength in predicting stock price movements. Logistic Regression gives us a simple baseline for classifying price changes. LSTM is excellent at capturing long-term patterns and time-based trends through deep learning. Random Forest handles complex, non-linear relationships and interactions, like price volatility and trading volume. By testing these models, we can compare their performance and determine which approach works best for forecasting stock prices.

**Evaluation of Results**

We will use several metrics to evaluate and ensure accurate predictions of NVIDIA's stock price movements. Accuracy will measure the proportion of correct predictions for classification models like Logistic Regression and Random Forest. Precision and recall will assess how well the models predict upward or downward movements, with precision focusing on the correctness of predicted movements and recall on identifying actual movements. The F1 score will balance precision and recall, particularly for imbalanced classes. AUC-ROC will evaluate how well the models distinguish between price increases and decreases. Lastly, cross-validation will help prevent overfitting by assessing model performance across different data splits. These metrics will guide us in selecting the most effective model for predicting stock price directions.

**Learning Objectives**

This project aims to deepen our understanding of the factors influencing NVIDIA's stock price movements by analyzing fundamental variables such as trading volume, volatility, and major product releases. Through this analysis, we seek to identify the elements that significantly drive stock price increases or decreases. Additionally, by comparing models like Logistic Regression, LSTM, and Random Forest, we aim to determine which algorithms are most effective for predicting stock price movements. The project will also help refine our skills in classification-based modeling and machine learning techniques, enabling us to enhance our predictive capabilities within financial markets. Ultimately, we expect to develop a model that accurately predicts stock price movements while providing insights into the behavior of NVIDIA's stock about technological advancements.

**Risks**

This project faces several risks that must be carefully managed. One critical risk is overfitting, where the model may perform well on historical data but needs to generalize to new, unseen data. To address this, we will employ cross-validation techniques and avoid using overly complex models. Another concern is data quality; inaccuracies or missing data in the historical stock dataset could lead to flawed predictions. We will use thorough data preprocessing and validation steps to mitigate this risk and improve the model's reliability.

**Ethical Concerns**

There are also ethical considerations to keep in mind for this project. One significant concern is market manipulation. The developed predictive models could be used for unethical purposes, such as manipulating stock prices. While this project is for educational purposes, it is essential to consider the broader implications of sharing predictive models. Additionally, data privacy becomes another concern if we decide to integrate sentiment analysis from social media or financial reports. We must ensure that any external data sources comply with privacy regulations to avoid misuse of personal information.

**Contingency Plan**

Suppose the initial project fails to deliver the expected results. In that case, we will pivot to a predictive analytics project using an NCAA dataset containing information on sports participation, revenues, expenses, and enrollment at various institutions. The alternative project will focus on predicting the financial performance of sports programs based on participation numbers and other relevant factors. Similar preprocessing steps will be applied, such as handling missing data and feature engineering and testing models like linear Regression and random forest to forecast revenue and expenses. This pivot will allow us to apply the same machine-learning techniques in a different domain.

**Conclusion**

This project aims to predict NVIDIA's stock price movements by leveraging predictive analytics models, such as Logistic Regression, LSTM, and Random Forest Classification. By focusing on how technological innovations, like the release of NVIDIA's AI-powered RTX chips, have influenced investor behavior, we seek to uncover patterns in stock market reactions. The project will evaluate short- and long-term trends using key metrics such as accuracy, precision, recall, F1 score, and AUC-ROC. These evaluations will guide us toward identifying the most effective model for predicting stock price movements.

We have carefully considered risks such as overfitting and data quality, with plans for cross-validation and robust data preprocessing. Ethical concerns, especially around market manipulation and data privacy, are also addressed, ensuring the models remain responsibly used. Suppose the stock price prediction project does not yield satisfactory results. In that case, we are prepared to pivot to a backup project focused on predicting the financial performance of NCAA sports programs, applying similar machine learning techniques.

Overall, this project will enhance our understanding of stock market dynamics and technological advancements, refining our forecasting techniques and providing actionable insights for financial analysts and investors.

Sources

HRTERHRTER. (2024). \*Nvidia stock historical data\* [Data set]. Kaggle.

<https://www.kaggle.com/datasets/programmerrdai/nvidia-stock-historical-data>

*Nvidia announces financial results for second quarter fiscal 2025.* NVIDIA Newsroom. (2024,

August 29). <https://nvidianews.nvidia.com/news/nvidia-announces-financial-results-for->second-quarter-fiscal-2025