

Machine Learning to Predict Stock Prices

Aditi Sen, Charissa Zou, Di Cao,
Jae Hun Lee, William Li

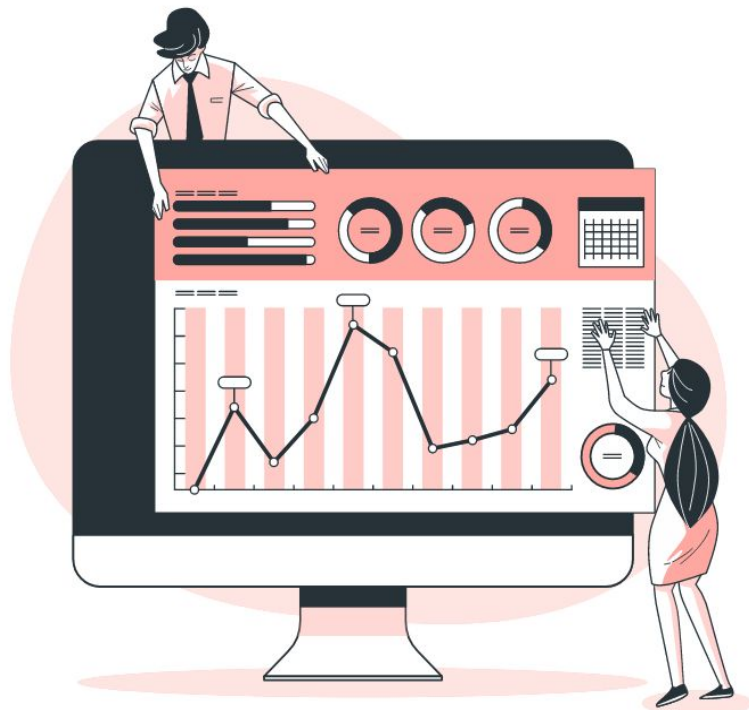


Table of contents

01 Introduction

02 Deliverables

03 Models



04 Challenges

05 Future

06 Conclusion


01



Introduction



Significance

- Determining the future value of a company stock or other financial instrument traded on an exchange has been an area of interest for both researchers and companies
 - The efficient-market hypothesis suggests that stock prices reflect all currently available information, making them inherently unpredictable
 - In spite of this, the research is ongoing as it can yield great profits for individuals
 - Moreover, the ability to predict stock prices would have an enormous impact on the economy
- 

Problem Statement



Given **time-series data** on the opening, closing, low, and high prices of a stock over the years, we want to be able to **predict the short-term pricing and trends** using machine learning algorithms and models.

Dataset and Features

- Company ticker symbols and stock information since their IPO
 - Source: yfinance API
- Historical prices of stocks with features including
 - Date, open price, high price, low price, and volume
- Included other popular stock market indicators
 - Moving Average Convergence Divergence (MACD)
 - On Balance Volume (OBV)
 - Relative Strength Index (RSI)
 - Bollinger Bands

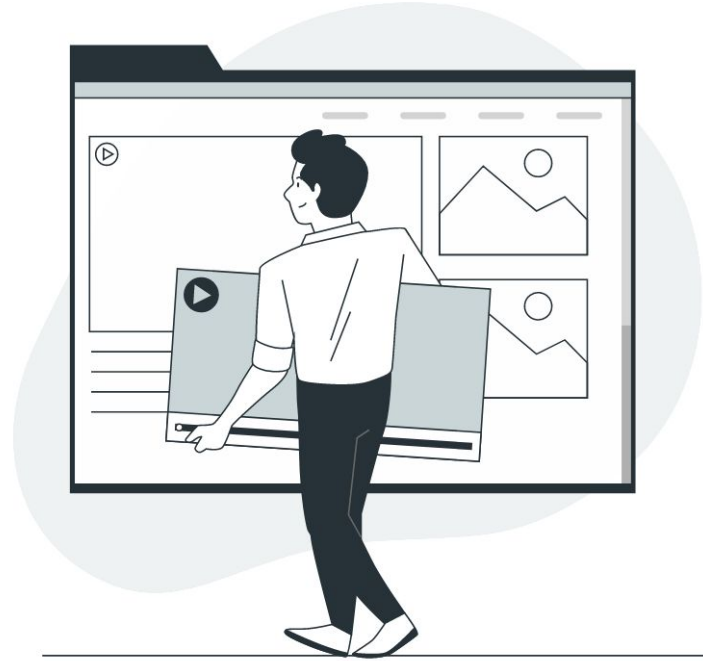
Our Aim

Predict whether the **stock price** will go up, down, or stay the same at the end of the week **on Friday** compared to that of previous **Friday**, using the stock data available from Monday to Thursday.

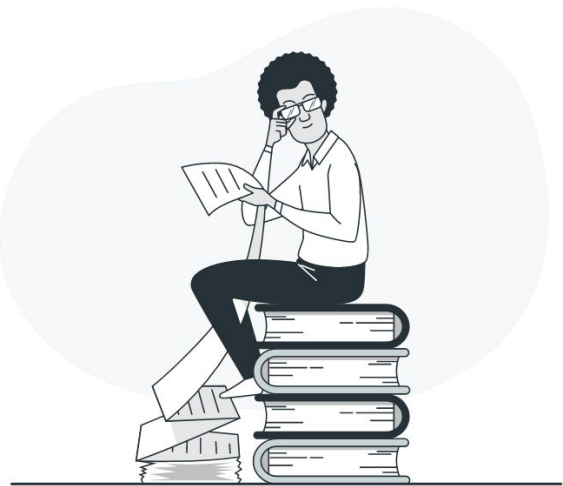


02

Deliverables



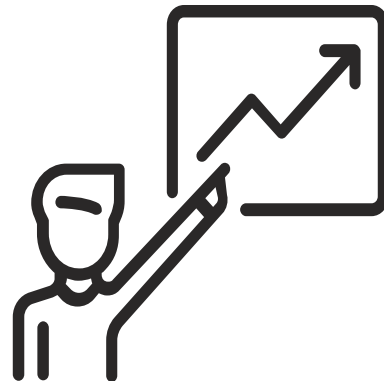
Must Accomplish



- Find and feature engineer at least 2-4 new features that will improve model performance
- Achieve > 55% accuracy on predicted trends on test data with our best model.
- Find features with high predictive powers through exploring correlation between features and target variable

Expect to Accomplish

- Train our models on a different stocks and compare how the model performances vary.
- Train at least 3 different models and compare their performances and determine the best approach.
- Create a backtester class after training to test if hypothetical users would be able to make money based on our model's predictions.



Would Like to Accomplish

- Compare stocks of various domains: established companies vs. unicorns vs. startups. We may be able to find particular industries that our model works well for (perhaps because of differing investor behaviors) and apply our model there to make money.
- Through profound analysis of our features and reliability of the model, we hope to find any currently undervalued companies with high potential.
- Convert our framework from data preprocessing, exploratory data analysis, training and testing into a pipeline that is transferable to other trading domains like Bitcoins, commodities, and etc.

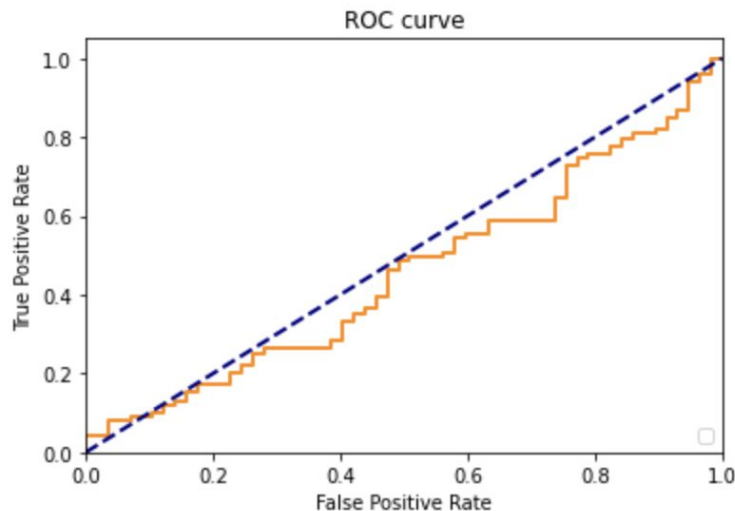


03

Models

Logistic Regression

- Scikit Learn's Logistic Regression package
- All solvers - *newton-cg*, *liblinear*, *sag*, *saga* - had comparable performance on the data
- Test accuracy:
 - Microsoft: 63%
 - McDonald's: 60%
 - Target: 60%
- Other metrics:
 - Precision: 63%
 - Recall: 100%
 - F-score: 77%
- ROC curve

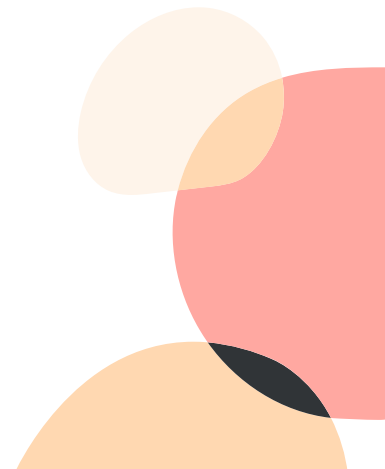


Random Forest

- Scikit Learn's Random Forest package
- Used Grid Search CV to find the best parameters:
 - max_depth: 2
 - min samples leaf: 2
 - min samples split: 3
- Test accuracy:
 - Microsoft: 58%
 - McDonald's: 43%
 - Target: 45%
- Benefit of n_estimator (number of decision trees) plateau after ~100

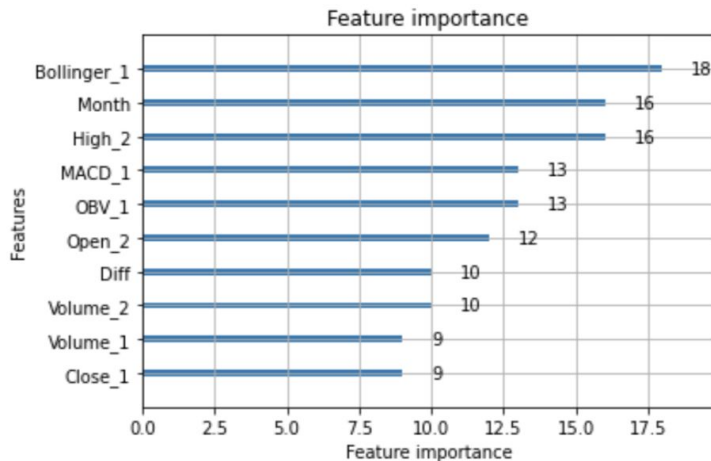
Neural Network

- Neural Network model with torch
- Input size: Number of the features (45)
 - Layer 1: Linear(input size, 30)
 - ReLU
 - Layer 2: Linear(30, 2)
- Binary Cross Entropy as a loss function
- Model trained with 50,000 iterations
- Test accuracy:
 - Microsoft: 63%
 - McDonald's: 60%
 - Target: 44%



Light GBM

- Gradient Boosting framework that uses tree based learning algorithms
- LGBM is much faster than conventional gradient boosting decision tree algorithm:
 - Gradient-based One-Side Sampling
 - Exclusive Feature Bundling
- Test accuracy:
 - Microsoft: 52%
 - McDonald's: 46%
 - Target: 47%
- feature importance analysis



Backtester

We created a Backtester class to simulate a user actually using our models' predictions to purchase stock and the resulting profit/loss.

```
class Backtester:
    def __init__(self, model):
        self.model = model

    # data: data used for prediction
    # price: actual closing prices on each Friday
    # num_stocks: number of stocks to purchase each time
    def performance(data, price, num_stocks = 10):
        predictions = self.model.predict(data)
```

```
# data: data used for prediction
# price: actual closing prices on each Friday
# num_stocks: number of stocks to purchase each time
def performance(data, price, num_stocks = 10):
    predictions = self.model.predict(data)

    profit = 0
    bought = False
    for i in range(len(predictions)):
        # Assumption: if the closing price of this Friday shows an increase, then
        # assume that next week will also increase, so will buy this Friday and
        # sell next Friday no matter what (to simplify things)
        if i == 0:
            profit = profit - num_stocks * price[i]
            bought = True
            continue
        else:
            if bought:
                profit = profit + num_stocks * price[i]
                bought = False
            if i == len(predictions):
                break
            if predictions[i] == 1:
                profit = profit - num_stocks * price[i]
                bought = True

    return profit
```

<Test Accuracies>

	Logistic Regression	Random Forest	Neural Network	Light GBM
Microsoft	63.5%	64.74%	63.4%	51.92%
McDonald's	60.4%	50.0%	60.8%	46.2%
Target	59.4%	45.2%	43.8%	47.6%

<Precision (Up)>

	Logistic Regression	Random Forest	Neural Network	Light GBM
Microsoft	63.5%	64.8%	63.4%	65.0%
McDonald's	60.4%	57.3%	60.6%	56.8%
Target	59.8%	61.0%	87.5%	55.4%

<Projected Net Profit>

	Logistic Regression	Random Forest	Neural Network	Light GBM
Microsoft	-\$977.00	-\$993.10	-\$977.00	-\$1,968.40
McDonald's	-\$1,228.60	-\$1,924.60	-\$1,224.40	\$359.30
Target	-\$452.10	\$346.60	\$45.20	-\$1,294.20

04

Challenges



Challenges

- Discovering correlation between our input features and target variable was difficult.
- The prices have been consistently increasing on a long term, and some of the relevant features have increased over time. Due to the difficulty of accounting for long-term trend, the recent test set has features that are skewed, making it challenging for the models to make predictions
- Even if the model displays relatively high precision, it does not necessarily guarantee profits - we need more fine-tuned models that can also predict probable range of prices
- Backtesting / simulation is difficult without a smart AI to make purchasing decisions




05

Future



Future Work

- Build Regression models
 - Our model only predicts the direction of the movement. Since it does not consider the magnitude of the divergence from the current price, we should work on regression models to be able to predict the actual range of price. It will allow us to make more discrete decision.
 - Improve Backtester
 - More flexibility in terms of decision-making. We have a simplified version that allows us to hold a long position at a “buy” signal. We hope to build a tester that can incorporate the models’ prediction to make more dynamic decisions.
- 

06



Conclusion

Conclusion

- Completed all our Must Accomplish and Expect to Accomplish goals
- Did not have time to finish Would Like to Accomplish
- Unable to accurately predict short term trends
- Profit / loss very stock-dependent
- Monetary loss might be due to simplifying assumptions made in our Backtester



“I figured something out. The future is unpredictable.”

—John Green



Thank you!