Stock Price Forecasting using Machine Learning Methods

LEVENT GÜNER I.ERDEM DEMIR
leventg | iedemir @kth.se

January 16, 2022

Abstract

Stock price prediction is a complex real-life machine learning problem with nontrivial solutions to suggest a well-performing predictive model. This project aims to research if it is possible to construct a combined technical analysis indicator by using various indicator signals in virtue of Random Forest (RF) Classifier and Long Short Term Memory (LSTM) Regressor. The project also covers evaluations of this newly created indicator's performance using backtesting and the results' comparisons with the buy and hold returns. The goal of this project is solely for research purposes, not to make a profit or to use with any other financial instruments.

Contents

List of Acronyms and Abbreviations

AD	Accumulation/Distribution
CCI	Commodity Channel Index
CNN	Convolutional Neural Network
DT	Decision Tree
EMA	Exponential Moving Average
LSTM	Long Short Term Memory
MACD	Moving Average Convergence Divergence
OHLCV	Open, High, Low, Close, Volume
RF	Random Forest
RNN	Recurrent Neural Network
RSI	Relative Strength Index
SDG	Sustainable Development Goals
SMA	Simple Moving Average
SSE	Sustainable Stock Exchanges Initiative

1 Introduction

Novel approaches utilize machine learning and deep learning models in financial analysis, and have successful findings. In this project, the main objective is to create a combined stock price indicator by training a Random Forest Classifier and an LSTM Regressor Network that is optimized for each industry and different cases such as long-term or short-term investments. Furthermore, the goal of the project is to give academic research about the combination of several indicator signals' performance on stock price data and discuss its results.

Certain objectives are determined to complete throughout the development process of the project. Firstly, data is gathered from Yahoo Finance, using its Python API. For stocks and prices, NASDAQ Stock Exchange stocks are used. Afterward, the technical analysis signals are generated from the stocks data, for each stock. Then, these signals are used to build the new indicator, in the best performing way by using RF Classifier and LSTM Regressor. After building and optimizing the indicator, predictions are made on the test data and an exchange simulator is run to perform the backtesting process. After building all the models, the evaluations are discussed.

1.1 Hypothesis

The combination of several different indicator signals using tree-based classifiers and sequential neural network regressors will give a well-performing, single-output indicator as a decision-supporting system which can be used with other technical analysis indicator signals.

1.2 Related Work

Stock price prediction is a subject having plentiful scientific works done using machine learning applications and algorithms. There have been several scientific relevant works observed during developing this project. In [1], deep learning techniques for financial analysis are used. [2] compares deep learning and classical machine learning approaches. In [3], Random Forests, Neural Networks, and standard strategic indicators are used. [4] shows an example of an independent approach where the data is not fitted to a specific model, rather the latent dynamics existing in the data are identified using deep learning architectures. Here, LSTM, Recurrent Neural Network (RNN), and Convolutional Neural Network (CNN)-sliding window model is used for financial analysis. [5] uses RNN and LSTM for stock price prediction.

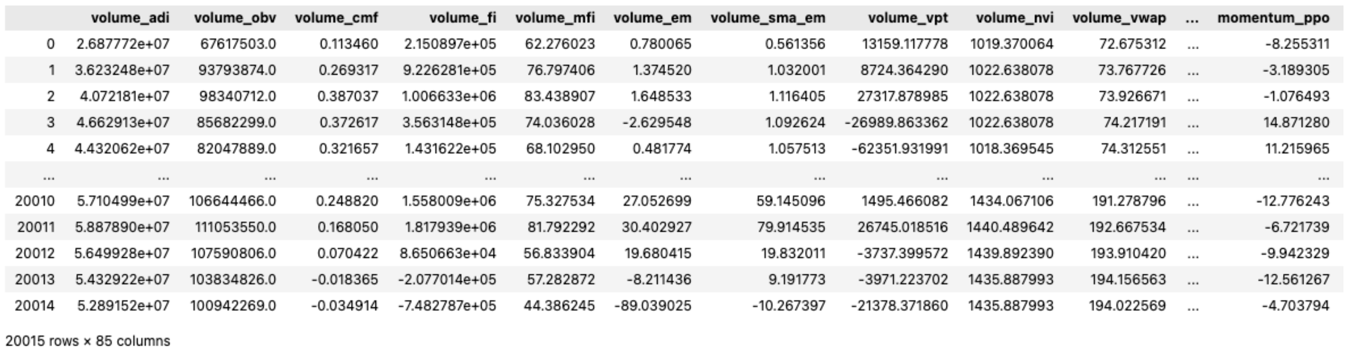
1.3 Ethics & Sustainability

Using high-efficiency computing resources, this project aims to achieve a low-energy consumption model. Moreover, according to the Sustainable Stock Exchanges Initiative (SSE), six Sustainable Development Goals (SDG) are defined as most relevant to stock exchanges: gender equality (SDG 5), decent work and economic growth (SDG 8), reduced inequalities (SDG 10), responsible consumption and production (SDG 12), climate action (SDG 13), and partnership for the goals (SDG 17). It seeks to contribute to other SDGs through this initiative such as sustainable energy (SDG 7) and sustainable communities (SDG 11). SSE explains these SDGs and their relation with the stock market exchange in their website [6].

Apart from reaching the sustainability objective, ethical considerations such as using the project to capture misleading signals and detect the market manipulations are indeed critical. The methods that are introduced in this project can be used for the detection of market manipulations. More about these are covered in Section 5.

1.4 Financial Background

In our tree-based models, there are several indicators used by extracting indicator signals from different technical analysis indicators. Technical indicators have two different kinds: overlays and oscillators. Overlays use a similar scale and plot over the stock price values, such as Bollinger Bands. Oscillators are the indicators that oscillate among local minimum and maximum and plotted below or above the stock price chart, such as Moving Average Convergence Divergence (MACD) [7]. To generate a well-performing technical indicator, indicator signals such as Bollinger Bands, Moving Average Convergence Divergence (MACD), Relative Strength Index (RSI), William's %R, Stochastic, Commodity Channel Index (CCI), Simple Moving Average (SMA), Accumulation/Distribution (AD), and other indicators are extracted and trained on several ticker prices from the NASDAQ stock market. These companies are listed in Table 1.



	volume_adi	volume_obv	volume_cmf	volume_fi	volume_mfi	volume_em	volume_sma_em	volume_vpt	volume_nvi	volume_vwap	...	momentum_ppo
0	2.687772e+07	67617503.0	0.113460	2.150897e+05	62.276023	0.780065	0.561356	13159.117778	1019.370064	72.675312	...	-8.255311
1	3.623248e+07	93793874.0	0.269317	9.226281e+05	76.797406	1.374520	1.032001	8724.364290	1022.638078	73.767726	...	-3.189305
2	4.072181e+07	98340712.0	0.387037	1.006633e+06	83.438907	1.648533	1.116405	27317.878985	1022.638078	73.926671	...	-1.076493
3	4.662913e+07	85682299.0	0.372617	3.563148e+05	74.036028	-2.629548	1.092624	-26989.863362	1022.638078	74.217191	...	14.871280
4	4.432062e+07	82047889.0	0.321657	1.431622e+05	68.102950	0.481774	1.057513	-62351.931991	1018.369545	74.312551	...	11.215965
...	...	...	...	...	...	...	...	...	...	...	...	...
20010	5.710499e+07	106644466.0	0.248820	1.558009e+06	75.327534	27.052699	59.145096	1495.466082	1434.067106	191.278796	...	-12.776243
20011	5.887890e+07	111053550.0	0.168050	1.817939e+06	81.792292	30.402927	79.914535	26745.018516	1440.489642	192.667534	...	-6.721739
20012	5.649928e+07	107590806.0	0.070422	8.650663e+04	56.833904	19.680415	19.832011	-3737.399572	1439.892390	193.910420	...	-9.942329
20013	5.432922e+07	103834826.0	-0.018365	-2.077014e+05	57.282872	-8.211436	9.191773	-3971.223702	1435.887993	194.156563	...	-12.561267
20014	5.289152e+07	100942269.0	-0.034914	-7.482787e+05	44.386245	-89.039025	-10.267397	-21378.371860	1435.887993	194.022569	...	-4.703794

20015 rows × 85 columns

Figure 1: Some of the features for combined data

Figure 1 shows the technical analysis indicators. For instance, the RSI in technical analysis is considered as a momentum indicator that has a value from 0 to 100. The common thought on RSI is that any value below 30 represents oversold, which means that the shares are more likely to increase, while a value over 70 represents overbought, which means that the shares are more likely to decrease. Bollinger Bands is an overlaying indicator that generates a margin around the market price by having an upper band that is higher than the market price and the lower band that is lower than the market prices values with $2 \times standard\ deviation$ interval for every time frame of the market price. Market price tends to go between the upper and lower band values and if exceeds those boundaries, it needs to go back to the $2 \times standard\ deviation$ interval. MACD indicator is an oscillating trend-following momentum indicator derived by subtracting the 26-period Exponential Moving Average (EMA) from the 12-period EMA. Whenever MACD crosses above (buy) or just below (sell) its signal line, technical signals are generated [8]. CCI is the comparison of the current price and the average price over a certain time [9]. For the Williams %R, like the RSI indicator, by denoting whether the ticker is overbought or oversold. Unlike RSI, boundaries of below 30 are considered as oversold, and above 70 are considered as overbought. Now on Williams %R the boundaries are considered as above -20 is oversold and below -80 is oversold indications. Stochastic indicator also works similar to RSI and Williams %R and denotes whether a ticker is overbought or oversold. Finally, AD indicator is a cumulative volume indicator that is being used to denote whether a ticker is accumulated or distributed. AD indicator helps to recognize patterns such as confirming a price raise upon an upward trend and a price decrease upon a downward trend.

2 Methods

The pipeline for the process is shown in Figure 2. The technical analysis features from Open, High, Low, Close, Volume (OHLCV) data are extracted. Even though there are data for different companies here, the first 90% of the data from each company (from 2020-01-03 to 2021-10-05) is separated as the train data and the last 10% (from 2021-10-05 to 2021-12-14) is taken as the test set. Both the combined indicator (RF model) as classification and the LSTM model as regression are trained with this train data. Afterward, the model predictions are gathered with the test data, and these predictions are combined with the indicator signals, which are explained in Section 2.1. With all this combination, a backtesting algorithm is run to see the overall trading algorithm's performance.

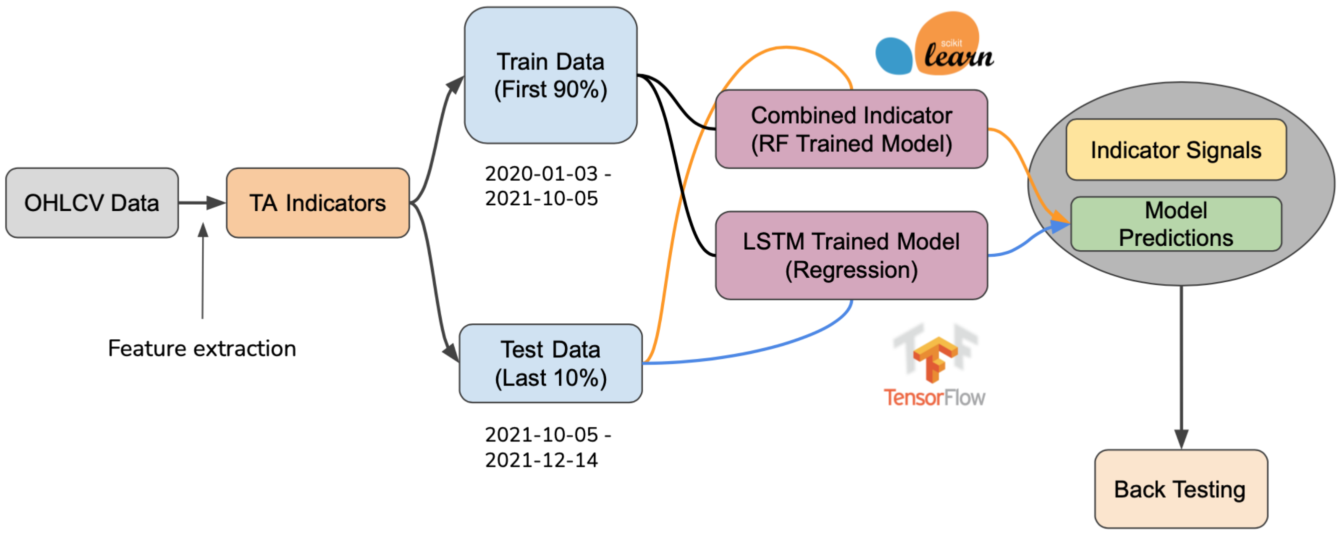


Figure 2: Process pipeline

2.1 Data Collection and Preparation

The data is gathered using Yahoo Finance's Python API. Yahoo Finance provides a free and easy-to-use API that permits gathering any company's data, as well as the currency exchanges. The time interval, starting dates, and end dates can be selected with the API. There are different time intervals such as 1 minute, 5 minutes, 15 minutes, and 1 hour are tried and decided to move with 1 hour of interval. The reason for that is to get the maximum amount of data in intraday trading. The API permits gathering data 2 years back when 1 hour interval is used and 2 months when 15 minutes interval is used. Afterward, the technical analysis features are extracted. These technical analysis features mainly include momentum, volume, volatility, and trend indicators [10]. The features that are used are selected by running different feature selection algorithms such as correlation tables, chi-squared, and tree-based feature selection models [11]. Since some of the technical analysis signals are based on rolling order, there are some empty values in the data. The rows which are having empty values are dropped.

The data that used in machine learning predictions are prepared as single-ticker data and multi-ticker data. In single-ticker data, only one ticker's data are trained and tested. Since there are not much many data points for a single ticker (around 3500 data points per ticker), multiple tickers' data has been combined by concatenating them. A threshold t based on the absolute mean of the hourly percentage change is determined. Any price change value p in the interval $-t < p < t$ is excluded, in order not to confuse the model with the stable prices. In this part, companies in the same industry have been used, since they would be more likely to be sharing similar properties in their technical analysis indicator signals. Table 1 shows the tickers that are used. The code for this process can be found in Appendix A.1.

Ticker Name	Company Name
AAPL	Apple Inc.
GOOGL	Alphabet Inc.
TSLA	Tesla Inc.
MSFT	Microsoft Corporation
AMZN	Amazon.com Inc.
FB	Meta Platforms Inc.
NVDA	NVIDIA Corporation
ADI	Analog Devices Inc.
ADBE	Adobe Inc.
NFLX	Netflix Inc.
CMCSA	Comcast Corporation
CSCO	Cisco Systems Inc.
AVGO	Broadcom Inc.
PYPL	PayPal Holdings Inc.

Table 1: Companies that are used for creating the model

2.2 Decision Tree

The Decision Tree (DT) algorithm can be considered as the closest machine learning algorithm to a rule-based system. Since the commonly known indicators are based on rules, the DT algorithm splits on each decision to attain the best decision rule on its leaf nodes by the definition. As a result, using a tree based algorithm to build a new indicator is a theoretically correct approach to execute.

The aim of using DTs is to catch these rules among different signal indicators and to create a combined single indicator from them. In this part of the project, a new indicator is created by using the selected signal indicators with a DT algorithm. This new indicator is expected to catch the patterns between the indicator signals and the upcoming price changes. The indicator is expected to give two signals: possible increase or possible decrease. First 90% of all the ticker data are set as train data, and the upcoming price change is given as a label in order to train the DT model. Scikit-Learn library is used for the analyses.

2.3 Random Forest

Random Forest is defined as an ensemble learning method with the combination of Decision Trees. In RF, the aim is to aggregate numerous DTs to decide the final result instead of depending on an individual tree. Same processes in Section 2.2 are applied for data processing and modeling. An RF classifier is generated and trained over all the data. Different depths are tried to tune the model. The code of this process can be seen in the Appendix A.2. The model is overfitted on the test data to catch most features. The best performing model is found with depth 18, and the results are discussed in Section 3.2.

2.4 Neural Network (LSTM)

The upcoming price changes are predicted by using neural networks, with the help of the signal indicators and the newly created indicator. Unlike the previous sections, this problem is a regression problem, not a classification problem. There are several studies in this area that have used Random Forests and Neural Networks, especially LSTM networks [3] [2] [4]. During the experiments, TensorFlow's Keras API is used. The closing prices and the other technical analysis indicators are given to an LSTM model. The last 7 data points are given as the features (independent variables) and the upcoming closing price that follows these 7 data points are given as the label (dependent variable).

Different neural network architectures are tried, and hyper-parameter optimizations are made. Related code for LSTM modeling can be found in the Appendix A.3. The regression results are converted to classes -1, 0, 1, where -1 indicates potential price drop, 1 indicates potential price rise, and 0 indicates potential stability for the next time frame. Here, the threshold t mentioned in Section 2.1 is used to determine the labels.

3 Results and Analysis

3.1 Decision Tree Results

Figure 3 shows the train and test confusion matrices and ROC curves for the Decision Tree model. The test accuracy score of the DT model is 0.52. It is seen that the DT algorithm is underperforming in the test set, even though it is performing well in the train set. The main reasons behind this result are having data with too many features and the complexity of the data. There is no simple relation between the features and the upcoming return, therefore a more complex model is needed for the predictions. After getting the results for RFs, which are discussed in Section 3.2, the RF model is used instead of the DT.

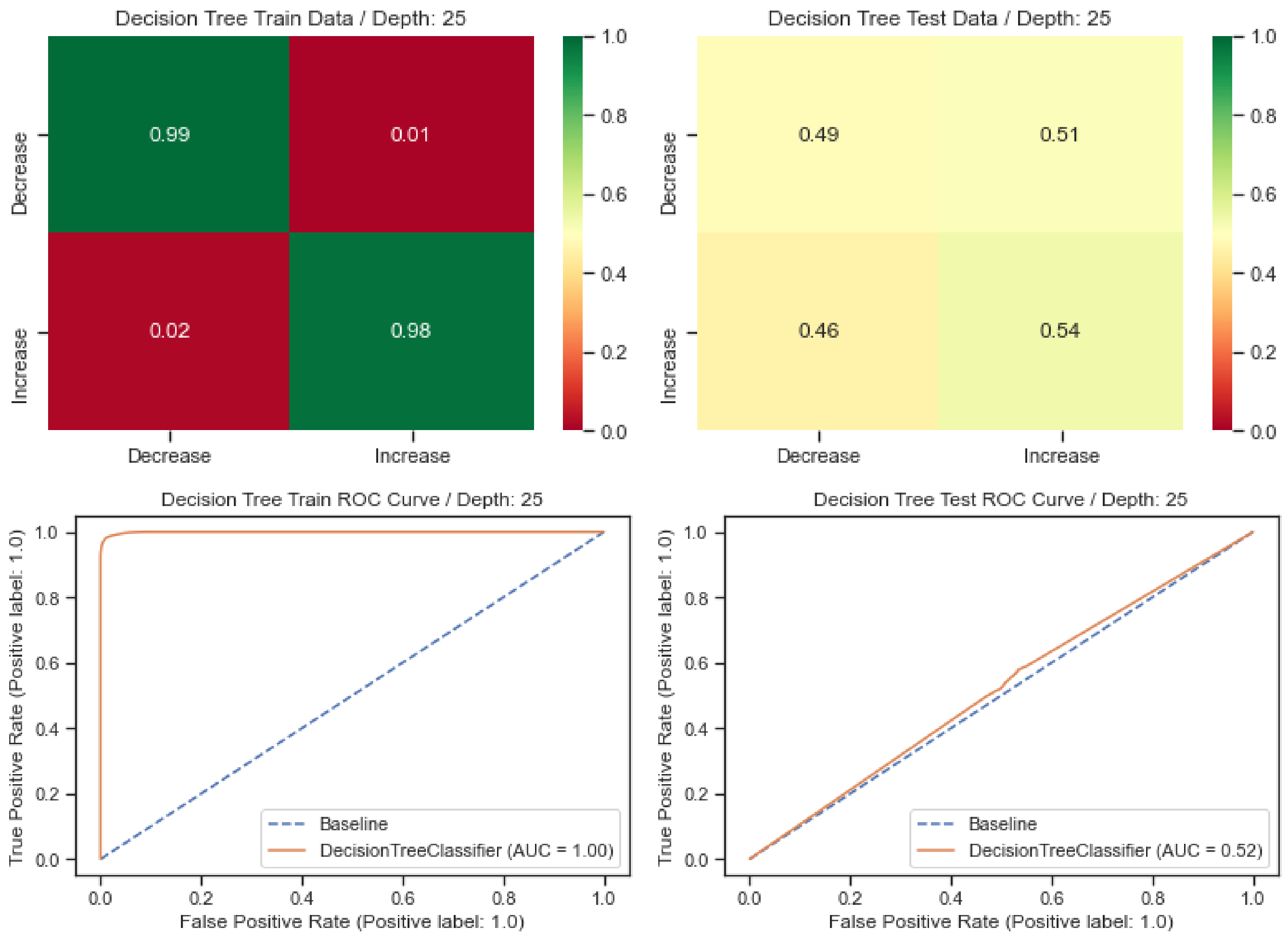


Figure 3: Multiple stock outcome prediction with DT for tech companies

3.2 Random Forest Results

Slightly better results have been observed in Random Forest compared to Decision Tree. Figure 3 shows the train and test confusion matrices of the RF model, as well as the ROC curve for test data. The accuracy score of RF is 0.56, which is slightly better than DT. It is seen in the confusion matrix that the model predicted 69% of the increments correctly, and 42% of the decrements correctly. Even though the data is equally distributed with the classes *increase* and *decrease*, there is some bias on the *increase* class. The data is still too complex for the RF algorithm, and the relations that the model caught are not seen in test data, as the complexity is too high. However, the results are better than a coin toss, so still, a positive impact on backtesting is observed.

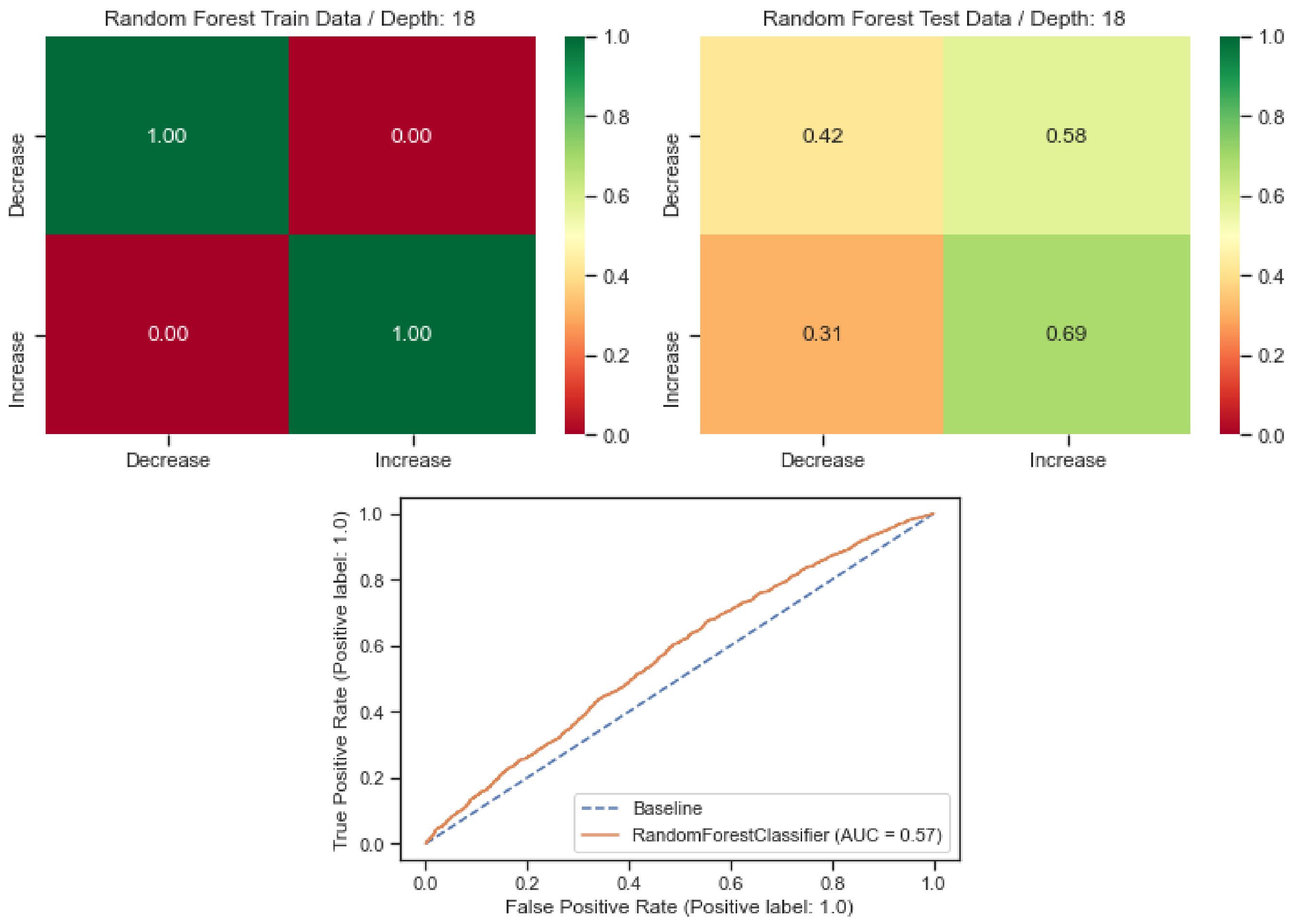


Figure 4: Multiple stock outcome prediction with RF for tech companies

3.3 LSTM Results

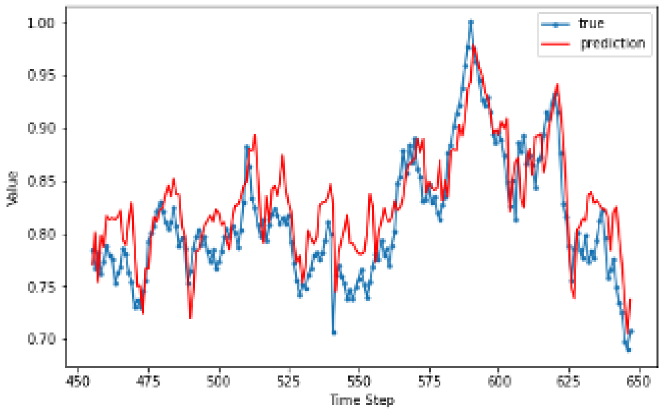
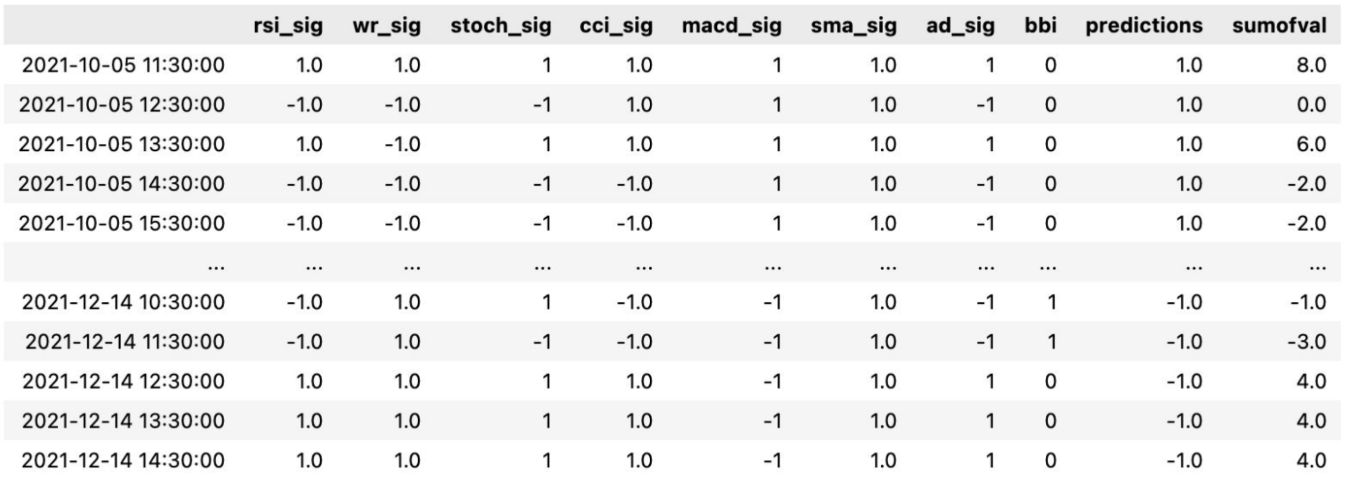


Figure 5: LSTM prediction results

The LSTM model is trained by looking at the past 7 data points and predicting the next data point. The model seems to perform well in Figure 5, however, the predictions are generally shifted values of the previous value. This problem is frequently encountered in LSTM networks while predicting complex data like stock prices. As there is no periodically repeating value in the data, and the next value is generally not related to past data, the neural network model has difficulties predicting proper values. Nevertheless, the combination of these predictions and RF predictions result better together, which is discussed in Section 3.4.

3.4 Backtesting Results

The backtesting is performed by creating a simulation by combining the technical analysis indicator signals, previously predicted prices that are coming from the neural network, and signals coming from the newly created indicator. Relevant code can be found in Appendix A.4



	rsi_sig	wr_sig	stoch_sig	cci_sig	macd_sig	sma_sig	ad_sig	bbi	predictions	sumofval
2021-10-05 11:30:00	1.0	1.0	1	1.0	1	1.0	1	0	1.0	8.0
2021-10-05 12:30:00	-1.0	-1.0	-1	1.0	1	1.0	-1	0	1.0	0.0
2021-10-05 13:30:00	1.0	-1.0	1	1.0	1	1.0	1	0	1.0	6.0
2021-10-05 14:30:00	-1.0	-1.0	-1	-1.0	1	1.0	-1	0	1.0	-2.0
2021-10-05 15:30:00	-1.0	-1.0	-1	-1.0	1	1.0	-1	0	1.0	-2.0
...	...	...	...	...	...	...	...	...	...	...
2021-12-14 10:30:00	-1.0	1.0	1	-1.0	-1	1.0	-1	1	-1.0	-1.0
2021-12-14 11:30:00	-1.0	1.0	-1	-1.0	-1	1.0	-1	1	-1.0	-3.0
2021-12-14 12:30:00	1.0	1.0	1	1.0	-1	1.0	1	0	-1.0	4.0
2021-12-14 13:30:00	1.0	1.0	1	1.0	-1	1.0	1	0	-1.0	4.0
2021-12-14 14:30:00	1.0	1.0	1	1.0	-1	1.0	1	0	-1.0	4.0

Figure 6: Indicators used for backtesting

Figure 6 shows the values used for the backtesting. Here, the technical analysis indicator signals and the predictions coming from the machine learning models are combined in a table. For each time point, their summations are taken and a threshold t is determined for the trading process. If the sum of the values is greater than this threshold t, the system performs a buy and if the sum of the values is smaller than t, the system performs a sell.

Name	Volatility	Buy&Hold % Return	Threshold	Our Model's % Return (With ML)
MSFT	0.0078	12.71	5 / -5	38.13
			7 / -7	10.41
AAPL	0.0086	22.84	5 / -5	67.67
			7 / -7	24.03
NVDA	0.0117	36.66	5 / -5	20.73
			7 / -7	33.85
TSLA	0.0169	20.58	5 / -5	-9.41
			7 / -7	18.42

Table 2: Backtesting results for different companies

Table 2 shows the returns for the selected timeline (from 2021-10-05 to 2021-12-14) for different threshold values. Here, one can interpret that a higher threshold value performs a safer trading process, as more indicators are showing the same direction. This theory can be verified by looking at the table. Microsoft and Apple have lower volatility, therefore lower threshold values have worked better. On the contrary, Nvidia and Tesla have higher volatility and a higher threshold value worked better in these companies' simulations.

In Figure 7, the upper part shows the equity over time. Under that, the profits and losses for each trading process are shown. Then, the price change over time is shown followed by volume and the sum of the indicators. The library provides a complete report for the backtesting process. It is seen that even though the stock prices are dropping, the model achieved to make a gain at the same time.

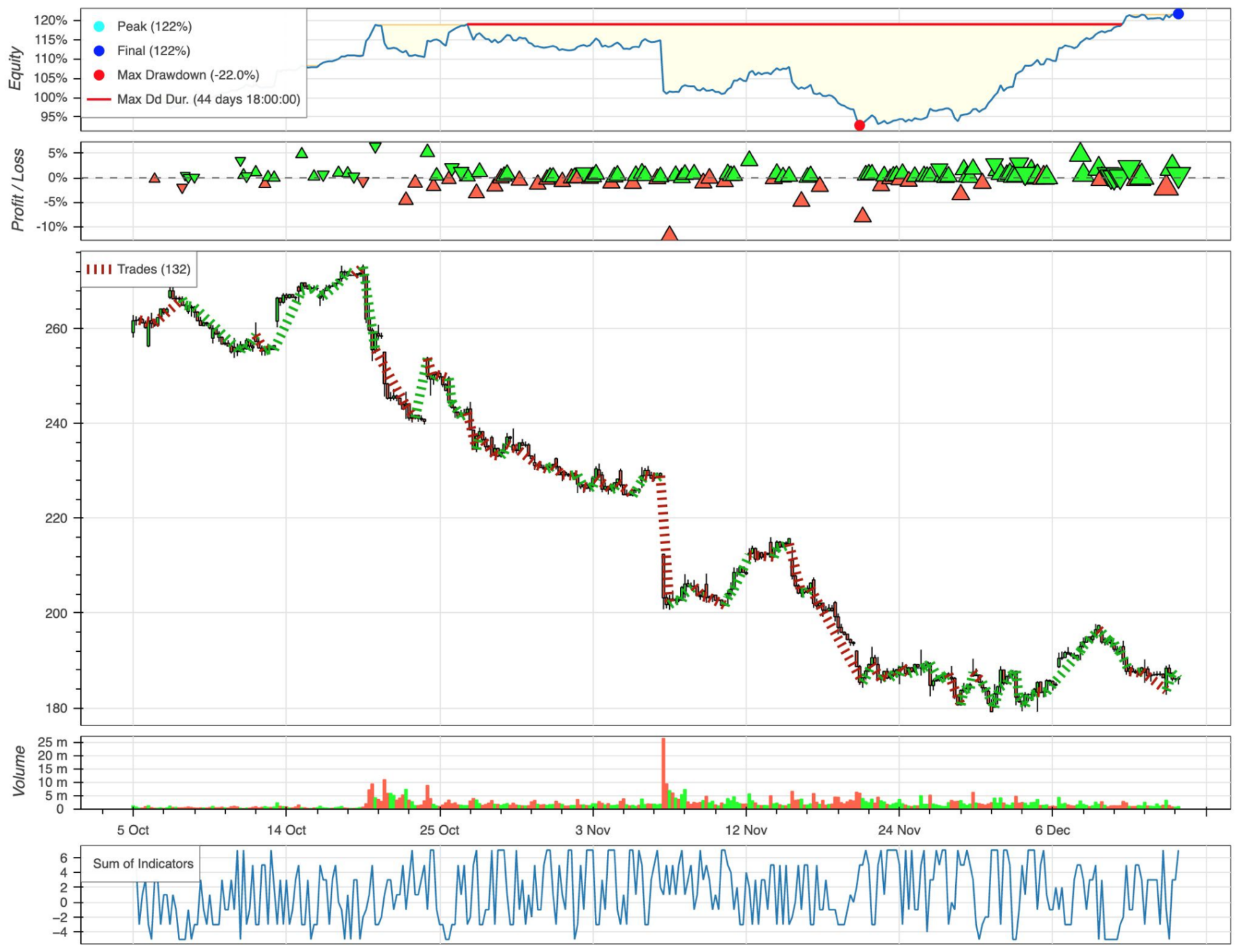


Figure 7: Backtesting results for PayPal

4 Discussion & Conclusion

During the experiments, prediction of the outcomes of stock data by checking the technical indicator signals has been tried and a combined indicator for future predictions is created. The purpose of these experiments was to see if there is a meaningful correlation between the technical analysis indicators and the outcomes. It is seen that the machine learning models are not generating accurate enough results for this use case, however, they still have a contribution on the trading process.

There is a common belief that investors are making their investments based on the technical analysis indicators, however, it is not all about that. Since the companies that we were analyzing are high-volume companies, their price changes are not only affected by the buy and sell. Therefore, there is no obvious correlation between the technical analysis indicator signals and the upcoming price change. Thus, we need much more data to create a better-performing indicator algorithm. On the other hand, Machine Learning and Deep Learning algorithms require relations between the input and output data. Stock market prices can be considered as a complex system with low predictability in the future. The upcoming price change is not directly dependent on the previous prices or the trend. Moreover, high volume processes make the predictability harder by adding lots of complexity.

5 Future Work

There are lots of data points that need to be taken into account, which are not easy to access and implement. For instance, social media and news data may carry a lot of information about companies and their value changes. On the other hand, fundamental analysis data can also be considered. Fundamental analysis is described as the financial performance of the company as well as the market situation depending on worldwide news. All of these data can be combined with technical analysis indicator signals, and a big matrix can be created altogether. Since there are different types of data in this case, such as text data, categorical data, and numeric data, it is not so easy to manage and combine all of them. In this case, there might be some problems with biased news and social media data, or there could be some data that do not carry relevant information about the stock change. These data should be cleaned carefully and they require detailed analyses. For processing financial text data, models such as FinBERT can be used. When a comprehensive model is generated by using various data sources and Natural Language Processing techniques, it can be used to catch manipulations and misleading signals in the stock markets by analyzing the combination of text data and financial data.

References

[1] J. Heaton and N. Polson, "Deep learning for finance: Deep portfolios," *SSRN Electronic Journal*, 2016. doi: 10.2139/ssrn.2838013. [Online]. Available: https://doi.org/10.2139/ssrn.2838013

[2] M. Nikou, G. Mansourfar, and J. Bagherzadeh, "Stock price prediction using DEEP learning algorithm and its comparison with machine learning algorithms," *Intelligent Systems in Accounting, Finance and Management*, vol. 26, no. 4, pp. 164–174, Oct. 2019. doi: 10.1002/isaf.1459. [Online]. Available: https://doi.org/10.1002/isaf.1459

[3] M. Vijh, D. Chandola, V. A. Tikkiwal, and A. Kumar, "Stock closing price prediction using machine learning techniques," *Procedia Computer Science*, vol. 167, pp. 599–606, 2020. doi: 10.1016/j.procs.2020.03.326. [Online]. Available: https://doi.org/10.1016/j.procs.2020.03.326

[4] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, Sep. 2017. doi: 10.1109/icacci.2017.8126078. [Online]. Available: https://doi.org/10.1109/icacci.2017.8126078

[5] A. Moghar and M. Hamiche, "Stock market prediction using LSTM recurrent neural network," *Procedia Computer Science*, vol. 170, pp. 1168–1173, 2020. doi: 10.1016/j.procs.2020.03.049. [Online]. Available: https://doi.org/10.1016/j.procs.2020.03.049

[6] "Sustainable stock exchanges initiative." [Online]. Available: https://sseinitiative.org/

[7] J. Chen, "Technical indicator definition," Dec 2021. [Online]. Available: https://www.investopedia.com/terms/t/technicalindicator.asp

[8] J. Fernando, "Moving average convergence divergence (macd)," Dec 2021. [Online]. Available: http://www.investopedia.com/terms/m/macd.asp

[9] M. Maitah, P. Prochazka, M. Cermak, and K. Šrédl, "Commodity channel index: Evaluation of trading rule of agricultural commodities," *International Journal of Economics and Financial Issues*, vol. 6, no. 1, p. 176–178, Jan. 2016. [Online]. Available: https://www.econjournals.com/index.php/ijefi/article/view/1648

[10] "Technical analysis library in python's documentation!" [Online]. Available: https://technical-analysis-library-in-python.readthedocs.io/en/latest/

[11] R. Agarwal, "The 5 feature selection algorithms every data scientist should know," Sep 2020. [Online]. Available: https://towardsdatascience.com/the-5-feature-selection-algorithms-every-data-scientist-need-to-know-3a6b566efd2

A Codes Used in Analyses

A.1 Data Gathering

```python
def get_multiple_tickers(ticker_names,start_date,end_date,
                         interval,add_ta=True):
    financedf = pd.DataFrame()
    features_df = pd.DataFrame()

    for ticker in ticker_names:
        df_data = get_single_ticker(ticker,start_date,end_date,interval)
        if add_ta:
            df_data = add_all_ta_features(
            df_data, open="Open", high="High", low="Low",
            close="Close", volume="Volume")
        df_data.columns = [ticker+'_'+c for c in df_data.columns]
        features_df = pd.concat([features_df,df_data],axis=1)

    return features_df


ts = ['AAPL','GOOGL','TSLA','MSFT','AMZN','FB','NVDA','ADI',
      'ADBE','NFLX','CMCSA','CSCO','AVGO','PYPL']

dfs_train = pd.DataFrame()
dfs_test = pd.DataFrame()
for ticker_name in ts:
    df = get_multiple_tickers([ticker_name],start_date="2020-01-03",
                              end_date=None,interval="60m")

    thr = 50*abs(df[ticker_name+'_Close'].pct_change()).mean()
    df.loc[((df[ticker_name+'_Close'].pct_change()*100)<=-thr),
           ticker_name+'_Change'] = -1
    df.loc[((df[ticker_name+'_Close'].pct_change()*100)>=thr),
           ticker_name+'_Change'] = 1
    df[ticker_name+'_Change'] = df[ticker_name+'_Change'].fillna(0)
    df[ticker_name+'_Change'] = df[ticker_name+'_Change'].shift(-1)
    df = df.drop(df.tail(1).index)
    df = df[df[ticker_name+'_Change']!=0]
    df = df.drop([ticker_name+'_trend_psar_down',
                  ticker_name+'_trend_psar_up'],axis=1)
    df = df.dropna()
    print(df[ticker_name+'_Change'].value_counts())
    df = add_signals(df)
    df['TickerName'] = ticker_name
    df = df.reset_index()
    split_point = int(len(df)*0.9)
    dfs_train = pd.concat([dfs_train,
                           pd.DataFrame(df.values[:split_point])])
    dfs_test = pd.concat([dfs_test,
                          pd.DataFrame(df.values[split_point:])])
```

A.2 Random Forest Grid Search for Depth

```
for depth in range(10,25):
    clf2 = RandomForestClassifier(max_depth=depth,random_state=31)
    clf2.fit(X_train,y_train)
    y_pred = clf2.predict(X_test)
    y_score = clf2.predict_proba(X_test)
    y_train_pred = clf2.predict(X_train)
    print('train:',clf2.score(X_train,y_train))
    print('test:',clf2.score(X_test,y_test))

    cm = mt.confusion_matrix(y_train,y_train_pred,normalize='true')
    sns.heatmap(cm,annot=True,fmt='.2f',
            xticklabels=['Decrease','Increase'],
            yticklabels=['Decrease','Increase'],
            vmin=0,vmax=1,cmap='RdYlGn')
    plt.title('Random Forest Train Data / Depth: {}'.format(depth))
    plt.show()

    cm = mt.confusion_matrix(y_test,y_pred,normalize='true')
    sns.heatmap(cm,annot=True,fmt='.2f',
            xticklabels=['Decrease','Increase'],
            yticklabels=['Decrease','Increase'],
            vmin=0,vmax=1,cmap='RdYlGn')
    plt.title('Random Forest Test Data / Depth: {}'.format(depth))
    plt.show()
    fig,ax = plt.subplots()
    ax.plot([0,1],[0,1],'--',label='Baseline')
    ax.set_title('Random Forest Train ROC Curve / Depth: {}'\
    .format(depth))
    mt.plot_roc_curve(clf2,X_train,y_train,ax=ax)
    plt.show()

    fig,ax = plt.subplots()
    ax.plot([0,1],[0,1],'--',label='Baseline')
    ax.set_title('Random Forest Test ROC Curve / Depth: {}'\
    .format(depth))
    mt.plot_roc_curve(clf2,X_test,y_test,ax=ax)
    plt.show()
```

A.3 LSTM Model

```
def multivariate_data(dataset, target, start_index,
                      end_index, history_size, target_size,
                      step, single_step=False):
    """
    dataset: whole dataset (np array)
    target: target column of whole dataset
            (put dataset if it has single variable) (np array)
    start_index: starting index of the dataset
    end_index: ending index of the dataset
               (put None if end index is the last element of dataset)
    history_size: how many data points should appear in past history
    target_size: how many data points you want to predict for future
    step: step size
    single_step: is single step
    """

    data = []
    labels = []
    start_index = start_index + history_size
    if end_index is None:
        end_index = len(dataset) - target_size

    for i in range(start_index, end_index):
        indices = range(i-history_size, i, step)
        data.append(dataset[indices])

        if single_step:
            labels.append(target[i+target_size])
        else:
            labels.append(target[i:i+target_size])

    return np.array(data), np.array(labels)

model = keras.Sequential()
model.add(keras.layers.LSTM(8, input_shape=(X_train.shape[1],
                            X_train.shape[2])))
model.add(keras.layers.Dense(y_train.shape[1]))

model.compile(loss='mean_squared_error',
              optimizer=keras.optimizers.Adam(0.001))
model.summary()
```

A.4 Backtesting

```
ts = ['AAPL','GOOGL','TSLA','MSFT','AMZN','FB','NVDA','ADI',
      'ADBE','NFLX','CMCSA','CSCO','AVGO','PYPL']
for btcomp in ts:
    ind_cols = ['rsi_sig',
    'wr_sig',
    'stoch_sig',
    'cci_sig',
    'macd_sig',
    'sma_sig',
    'ad_sig','bbi']

    dfs_ohlcv = pd.concat([dfs_test[['Open','High','Low','Close',
                                 'Volume']+our_gens]\
        .reset_index(drop=True),
        pd.DataFrame(y_pred,columns=['predictions'])],axis=1)
    dfs_ohlcv['Datetime'] = pd.to_datetime(dfs_ohlcv['Datetime'])

    df_tobt = dfs_ohlcv[dfs_ohlcv['TickerName']==btcomp]\
        .set_index('Datetime').drop('TickerName',axis=1)
    df_tobt = pd.concat([df_tobt,pd.DataFrame(df_tobt[ind_cols]\
        .sum(axis=1),columns=['sumofval'])],axis=1)
    df_tobt.index = df_tobt.index.tz_localize(None)
    df_tobt.sumofval = df_tobt.sumofval.shift(-1)
    df_tobt = df_tobt.dropna()

    class MLTrade(Strategy):
        def init(self):
            price = self.data.Close
            self.forecasts = self.I(lambda: np.repeat(np.nan,
                                                  len(self.data)),
                                                  name='forecast')
        def next(self):
            forecast = self.data.df.tail(1)['sumofval'].values[0]
            self.forecasts[-1] = forecast

            if forecast>5:
                self.buy()
            elif forecast<-5:
                self.sell()


    bt = Backtest(df_tobt, MLTrade, commission=0,
                exclusive_orders=True)
    stats = bt.run()
    print('Return %:',round(stats['Return [%]'],2))
    print('Buy&Hold Return %:',round(stats['Buy & Hold Return [%]'],2))
    print('Our model is',
          round(stats['Return [%]']-stats['Buy & Hold Return [%]'],2),
          '% better than Buy&Hold')
    print('*'*50)
```