Stock Trading Action Classification using Machine Learning

Thu Pham | Quang Lam

Luther College

Author Note

Our thanks to Data Science Department at Luther College, professor Shafqat Shad and professor Kent Lee for their support and instructions.

Please feel free to contact us at:

Thu Pham (phamtr01@luther.edu)

Quang Lam (lamqu01@luther.edu)

Abstract

Stock prices fluctuate within seconds and are affected by complicated financial and non-financial indicators. Hence, stock prices prediction is an ambitious project. However, thanks to Machine Learning techniques, we have the capacity of classifying trading action from massive amounts of data that capture the underlying stock price dynamics.

Keywords: trading action, deep learning, classification

Stock Trading Action Classification using Machine Learning

In this project, we utilized past data, technical indicators, and economic indexes and applied supervised learning methods to predict stock price action trading (long position/short position) on the next trading month.

As opposed to predicting the trend in short-term which is used in the high-frequency trading market, we intend to forecast the upward and downward movement in the weekly-basis not solely for algorithmic trading, but as a supplement to help investors alike on decision-making.

Our model is currently applied to a single stock symbol.

Method

Data Source

The project uses the free API from Alpha Vintage (alphavantage.co) to the monthly stock market price historical data in the past 20 years.

Data Collecting

```
# Config
symbl = 'MSFT'
apiKey = 'MSXSR0YHC991CZN6'

# Fetch Historical Data
# df = pd.read_csv('https://www.alphavantage.co/query?datatype=csv&function=TIME_SERIES_MONTHLY&symbol=' + symbl + '&ou

# Use local data
df = pd.read_csv('data/monthly_MSFT.csv')

# Setting index as date
df['timestamp'] = pd.to_datetime(df.timestamp, format='%Y-%m-%d')
df.index = df['timestamp']

print
```

Additionally, Alpha Vintage also provides the STOCH index data. By definition, the stochastic oscillator is a momentum indicator comparing a particular closing price of a security to a range of its price over a period of time.

Add STOCH indicator

```
symbl = 'MSFT'
apiKey = 'MSXSR0YHC991CZN6'

# Fetch SMA Data
stoch = pd.read_csv('https://www.alphavantage.co/query?function=STOCH&symbol=' + symbl + '&interval=monthly&time_perioc
```

Most importantly, the project uses the New York Times Articles API to retrieve all the news headlines New York Times published since January 2000. The code for this step can be found at https://github.com/thu2pham/StockForecasting/tree/master/data.

Data Preprocessing

To preprocess the stock market price historical data, we set an expected return value (for example, 2.5%), which is minimal change in the stock price compared to the previous month for a long position. With this threshold value, we add a new variable called the action with 2 values: 1 represents *long* position and 0 represents *short* position.

Then we use sentimental analysis to analyze the New York Times headlines and add 3 variables: **positive** represents the percentage of positive headlines, **appearance** and **positive2** represent the number of times the company appears on the news and the percentage of its positive appearance.

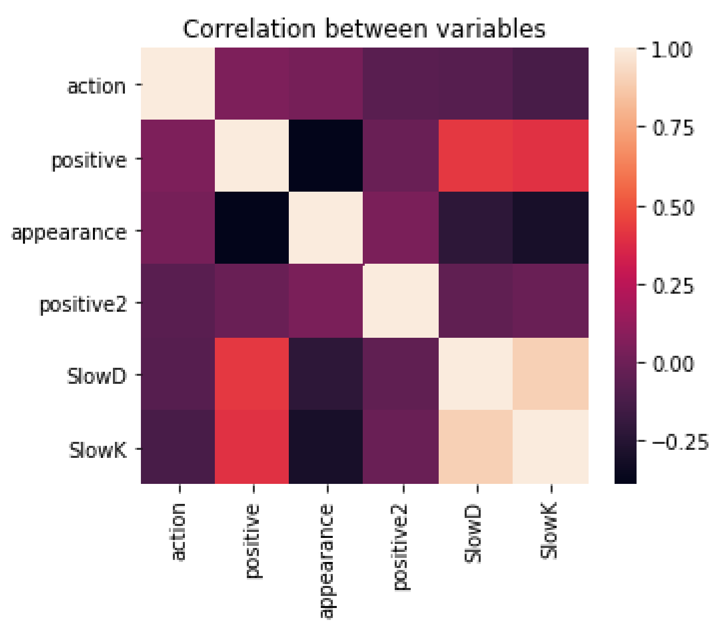
```
# Similarly for the second file
news_data2 = pd.read_csv("data/sentimental_data2.csv")
news_data2 = news_data2[::-1].reset_index().drop(columns=['index'])
news_data2 = news_data2[1:].reset_index()
sentimental_df = pd.concat([sentimental_df, news_data2], axis=1, join_axes=[sentimental_df.index])
sentimental_df.head(5)
```

	timestamp	open	high	low	close	volume	prev_close	action	index	year	month	positive	index	year	month	appearance	positive2
0	2019-05-14	130.530	130.65	123.04	124.73	285095390	130.60	0.0	1	2019	4	0.544137	1	2019	4	3	0.666667
1	2019-04-30	118.950	131.37	118.10	130.60	433157868	117.94	1.0	2	2019	3	0.562695	2	2019	3	4	0.500000
2	2019-03-29	112.890	120.82	108.80	117.94	589045341	112.03	1.0	3	2019	2	0.556610	3	2019	2	2	0.500000
3	2019-02-28	103.775	113.24	102.35	112.03	469095970	104.43	1.0	4	2019	1	0.548660	4	2019	1	9	0.500000
4	2019-01-31	99.550	107.90	97.20	104.43	714204787	101.57	1.0	5	2018	12	0.561755	5	2018	12	0	0.000000

Then **STOCH** variables and the three mentioned variables are matched with the upcoming month while action variable is matched with its real month. It means we would use other variables to predict what to do the upcoming months (action).

	timestamp	action	positive	appearance	positive2	SlowD	SlowK
0	2019-05-14	0.0	0.544137	3	0.666667	72.1060	89.5142
1	2019-04-30	1.0	0.562695	4	0.500000	59.1869	72.5735
2	2019-03-29	1.0	0.556610	2	0.500000	52.2875	54.2304
3	2019-02-28	1.0	0.548660	9	0.500000	58.3607	50.7567
4	2019-01-31	1.0	0.561755	0	0.000000	68.6283	51.8754

The correlation of our completed dataset is shown below, using heatmap:



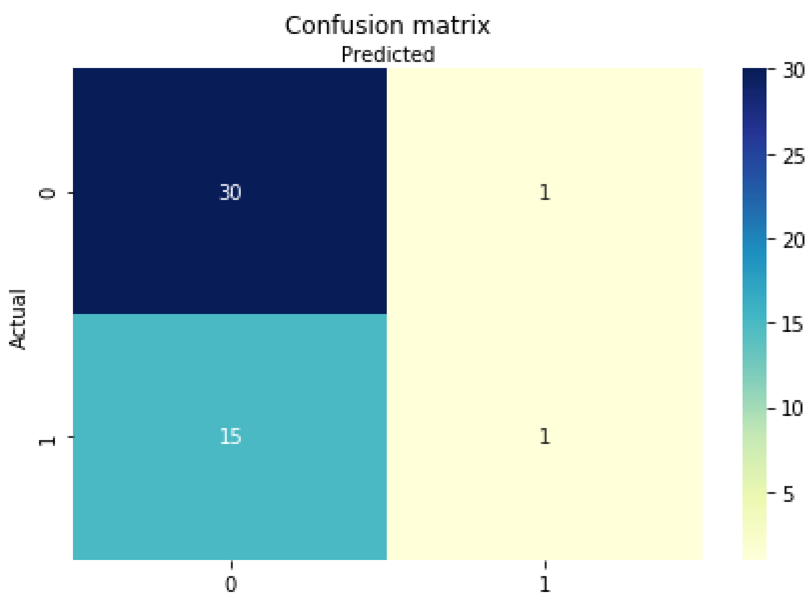
Model Implementation

Logistic Regression (LR). Logistic regression is a simple linear model for classification.

Logistic Regression

```
from sklearn.model_selection import train_test_split

all_data['timestamp'] = pd.to_datetime(all_data['timestamp'], format='%Y-%m-%d')
all_data.index = all_data['timestamp']

feature_cols = ['positive', 'appearance', 'positive2', 'SlowD', 'SlowK']
# feature_cols = ['positive2']
target = 'action'
X = all_data[feature_cols]
y = all_data[target]

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2)

# instantiate the model (using the default parameters)
logreg = LogisticRegression()

# fit the model with data
logreg.fit(X_train,y_train)

#
y_pred=logreg.predict(X_test)
y_conf=logreg.decision_function(X_test)
```

The accuracy of this method is approximately 66%. The confusion matrix is presented below:



Support Vector Machine (SVM). Similar to Logistic Regression, SVM is an algorithm used for classification problems.

Support Vector Machine (SVM)

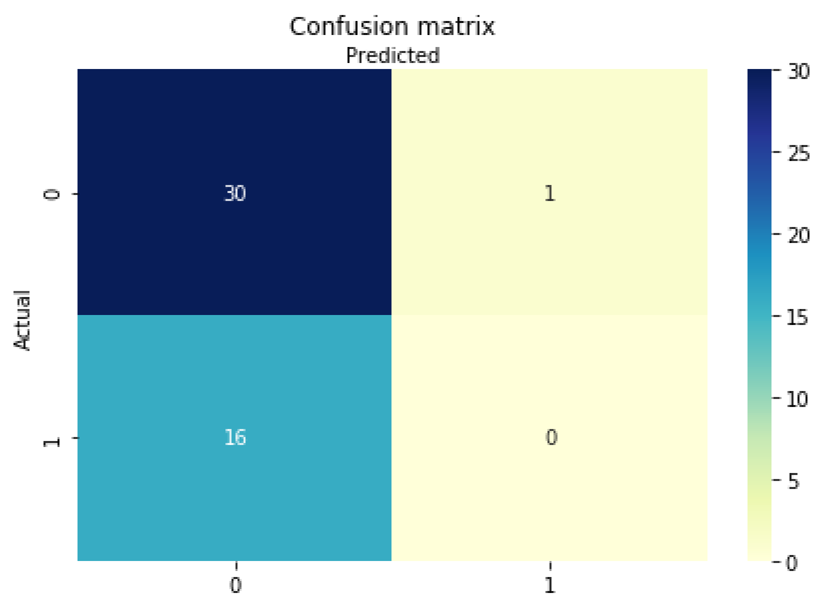
```
from sklearn import svm

# Initiate model
svc = svm.SVC(kernel='rbf')

# Fit model
svc.fit(X_train, y_train)
```

```
/Users/thupham/anaconda3/envs/py3k/lib/python3.7/site-packages/sklearn/svm/base.py:196: FutureWarning: The default va
lue of gamma will change from 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma ex
plicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

```
y_pred=svc.predict(X_test)
y_conf=svc.decision_function(X_test)
```

However, in large dataset, SVM performs marginally better and less sensitive to outliers than LR. In our case, this dataset is considered not big, we do not see an improvement in performance when applying SVM. The accuracy of SVM is approximately 64%.

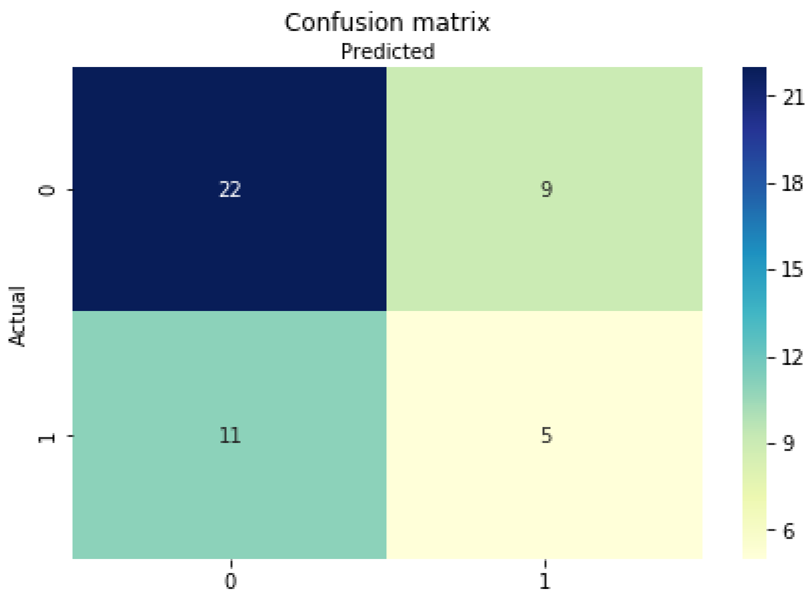


Neural Networks. Since we are performing binary classification, a multi-layer perceptron is an appropriate method for such model. We implement a Dense layer, which is a connecting layer. The first two layers take activation argument ‘relu’ while the last layer takes ‘ stigmoid’ .

```
# Import `Sequential` from `keras.models`
from keras.models import Sequential

# Import `Dense` from `keras.layers`
from keras.layers import Dense

# Initialize the constructor
model = Sequential()

# Add an input layer
model.add(Dense(12, activation='relu', input_shape=(5,)))

# Add one hidden layer
model.add(Dense(8, activation='relu'))

# Add an output layer
model.add(Dense(1, activation='sigmoid'))
```

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.fit(X_train, y_train,epochs=20, batch_size=1, verbose=1)
```

The accuracy of this model is roughly 57%



Results

The models successfully predict the actions at least 55% of the time with the expected return value close to 0. The expected return value significantly affects the accuracy of the models. The smaller the expected return value, the more likely the decision making fluctuates with the market, so the more least accurately the models predict.

Conclusion

As the models only predict between two values: long position or short position, we consider the accuracy of our trained models is not sufficient enough to use in the real world. With the results of this project, even though we intend to analyze the dataset on a monthly basis instead of daily or weekly to minimize to volatility, we conclude that the stock market is much more volatile than we expected and cannot be predicted accurately using the sentimental analysis of news headline but requiring additional data and methods. Also, our dataset in monthly basis is too small to train the value thoroughly.

Additionally, as we analyze the dataset using three different Machine Learning methods, including one using complex neural networks, the results are not much difference between them. So we conclude that the methodology of our project is not the primary factor of its inaccuracy.

To improve the results, we suggest adding new variables to the models, analyzing the data in the weekly-basis to have more breakpoints for the models to adjust to the volatile market.

References

Hutto, C.J. & Gilbert, E.E. (2014). *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14)*. Ann Arbor, MI, June 2014.

Gou, Xin. *How can Machine Learning Help Stock Prediction.* http://cs229.stanford.edu/proj2015/009_report.pdf. Stanford University, 2015.