Capstone Project

Retail Investor Behavior Predictor in US Stock Market

Machine Learning Engineer Nano-degree

Veronica Tang

Nov 21st, 2019

Definition

Project Overview

Retail investor behavior has always been an area of study in finance given their unique characteristics as well as their importance in the market place. According to regulatory disclosure from retail brokerage firms such as Charles Schwab, TD Ameritrade, etc., it is estimated that retail investor makes up about 15%-20% of volume in US equities market. Understanding how they behave is an interesting topic for any other market participants such as institutional investors and market makers.

Online stock trading app Robinhood has successfully garnered interest from millennials retail clients by offering no-commission products. According to news report from CNBC in May 2018, Robinhood has 4 million brokerage accounts on its platform compared to E-trades 3.7 million accounts1, which makes Robinhood's user data significant and representative of a big market share. The stock popularity data they publishes provides a rare opportunity for us to peek into retail investor activities.

Problem statement

The problem of interest in this project is to find out if we can predict retail investor will buy or sell next day, given today's information. It is common for retail investors to make investment decisions based on readily observable information they can get easy access to, such as price, volume, and if peers surrounding them are also buying or selling. There might be hidden code in these information that can help us find patterns in their behavior. In this project, I tackled the problem in the following steps:

1) I identified which type of stocks is suitable for the analysis. Only stocks that have attracted enough retail activities are in scope for my analysis. For stocks with a low retail investor base and low trading volume, the user holding data can be misleading and filled with noises.
2) I found out what features are useful among public information retail investors derive their decisions from. This problem is composed of two parts: a) what kind of data I should look at b) how many days of look-back I should include of these data.

[1]CNBC news on Robinhood: https://www.cnbc.com/2018/05/10/trading-app-robinhood-surpasses-e-trade-in-user-numbers.html

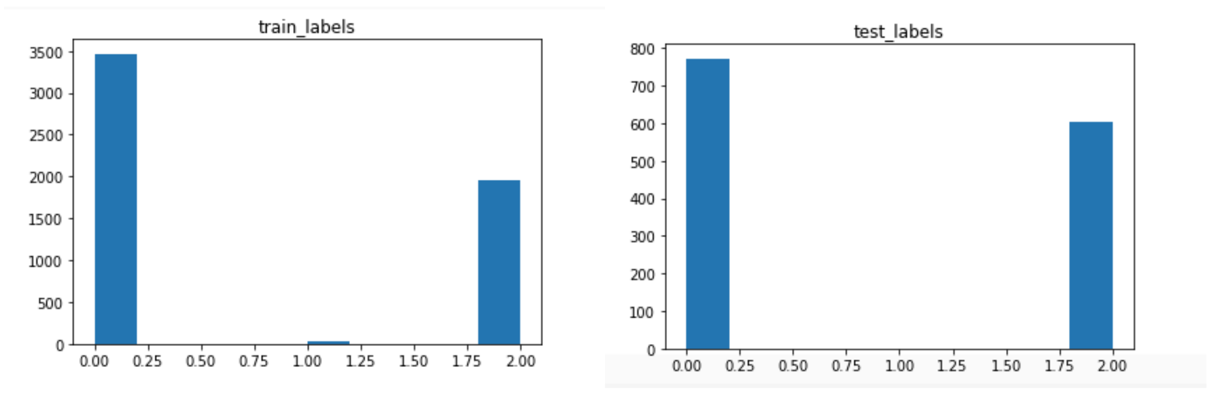
3) I employed machine learning models to try to predict if user holding will go up or down or remain flat on next day given those features defined in step 2) on category of stocks chosen from step 1).

Metrics

F1 score = 2* Precision*Recall / (Precision + Recall)
Precision = True Positive / (True Positive + False Positive)
Recall = True Positive / (True Positive + False Negative)

Given the target of prediction (percentage change of user holding from previous day to today) has 3 classes — increase, decrease or unchanged, and the cases where user holdings remained unchanged is a minority class that only contains a tiny number of the data, the target is therefore imbalanced. F1 Score is a suitable measure to use in this case that will help us seek a balance between Precision and Recall.

CHART 1: TRAIN AND TEST LABELS HISTOGRAM (0 = USER HOLDING INCREASED COMPARED TO PREVIOUS TRADING SESSION, 1 = UNCHANGED, 2 = DECREASED)



I would also like to highlight here that evaluation metrics will be calculated in micro-average instead of macro-average[2]. A macro-average of the metric is calculated independently for each class and then take a simple average among the classes, which essentially assumes an equal weight of the results from different classes. A micro-average will aggregate the contributions of all classes to compute the average metric. Since our target classes are imbalanced, micro-average is preferable.

Analysis

Data Exploration

The original data was downloaded from sources below in the following format:

[2] A systematic analysis of performance measures for classification tasks: http://rali.iro.umontreal.ca/rali/sites/default/files/publis/SokolovaLapalme-JIPM09.pdf

1. Stock price volume file: Historical end of day stock data for US stocks. It contains 8516 unique symbols with data from 2014-06-23 to 2019-06-11.

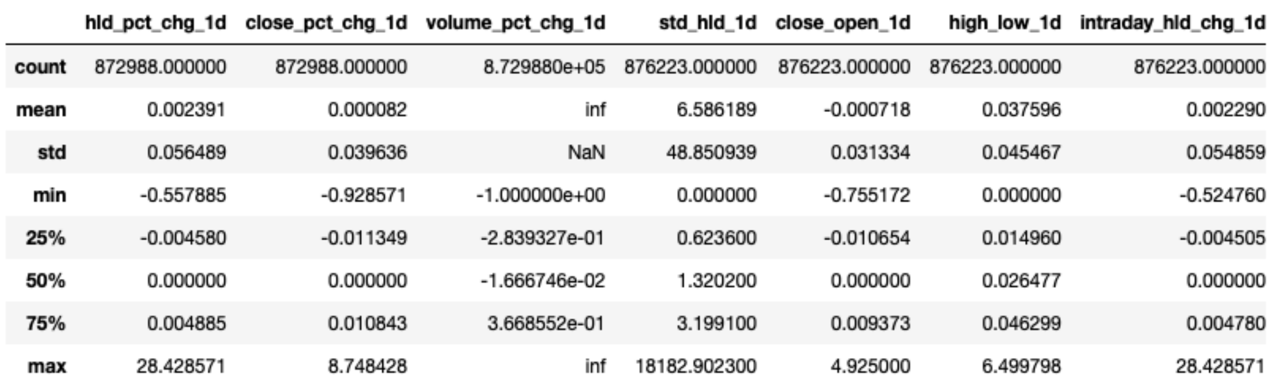
data source link: https://backtest.pl
original data source: The author downloaded data through IEX API.
columns: date, open, high, low, close, volume
format: saved in separate csv files with stock ticker as the file name

2. Robinhood file: Robinhood user holding files. After some simple pre-processing described in the next section (i.e. discarding stock files with lower than 100 user holdings at any point of time), there are 3235 unique symbols with data from 2018-05-02 to 2019-10-17.

data source link: https://robintrack.net/data-download
original data source: The author downloaded data through Robinhood API.
columns: timestamp, user_holdings
format: saved in separate csv files with stock ticker as the file name

After examining the data, it is obvious that there exists a lot of drastic volume change, extremely low trading volume as well as low user holding situations. In some cases volume traded on the day was zero and resulted in infinity volume percentage changes compared to pervious day (volume_pct_chg_1d in table below). Though 25 and 75 percentile data looks fine, the extremely big max values in all the features are concerning. Taking volume change for example, 62% of the data points experienced > 20% or < -20% volume change.

TABLE 1: STATISTICS OF FEATURES ON ENTIRE DATASET



	hld_pct_chg_1d	close_pct_chg_1d	volume_pct_chg_1d	std_hld_1d	close_open_1d	high_low_1d	intraday_hld_chg_1d
count	872988.000000	872988.000000	8.729880e+05	876223.000000	876223.000000	876223.000000	876223.000000
mean	0.002391	0.000082	inf	6.586189	-0.000718	0.037596	0.002290
std	0.056489	0.039636	NaN	48.850939	0.031334	0.045467	0.054859
min	-0.557885	-0.928571	-1.000000e+00	0.000000	-0.755172	0.000000	-0.524760
25%	-0.004580	-0.011349	-2.839327e-01	0.623600	-0.010654	0.014960	-0.004505
50%	0.000000	0.000000	-1.666746e-02	1.320200	0.000000	0.026477	0.000000
75%	0.004885	0.010843	3.668552e-01	3.199100	0.009373	0.046299	0.004780
max	28.428571	8.748428	inf	18182.902300	4.925000	6.499798	28.428571

In reality, many of these cases are due to stock split, merger and arbitrage or other one-off corporate actions events. Another possibility is due to inactivity of trading in some less-known names that rarely receives attention or only sporadically receives attention from investors, which makes the percentage change data extremely volatile.

A significant challenge in dealing with outliers in this project was, it was hard to simply remove them without damaging the integrity and continuity of the time-series. In addition, it was also difficult to make a judgement arbitrarily on what is the right threshold to use to throw out bad data. What's more, it was challenging to distinguish true outliers from valid exuberant market activities. For example, there might have been positive news on the stock that caused the frenzy reaction from investors to buy a stock, which led to a spike in price, volume, and user holdings. In this case, the large values in percentage change in price, volume and holdings could be very useful insights that contribute to how retail investors behave on the next day. If a threshold was applied to filter out extremely large numbers, it risked eliminating this type of legitimate market moves at the same time of removing truly problematic outliers.

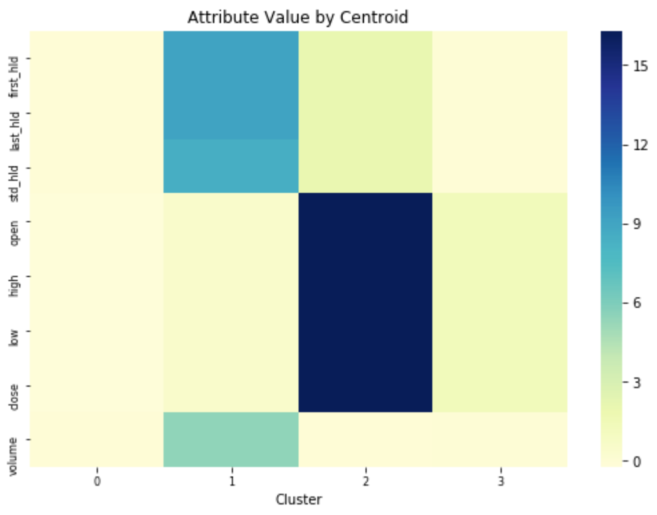
In order to deal with the problem of abnormal values in our features, and focus on stocks that have sufficient retail investor participation, I used a clustering model to categorize stocks first to identify the appropriate stocks in scope for our analysis. This process will be described in details in "Methodology - Data Preprocessing" section.

As I narrowed down the universe of stocks, data visualization presented in the following section will all be concentrated on the cluster of stocks that I ended up predicting retail behavior on, instead of the entire universe of stocks available in the raw data.

Exploratory Visualization

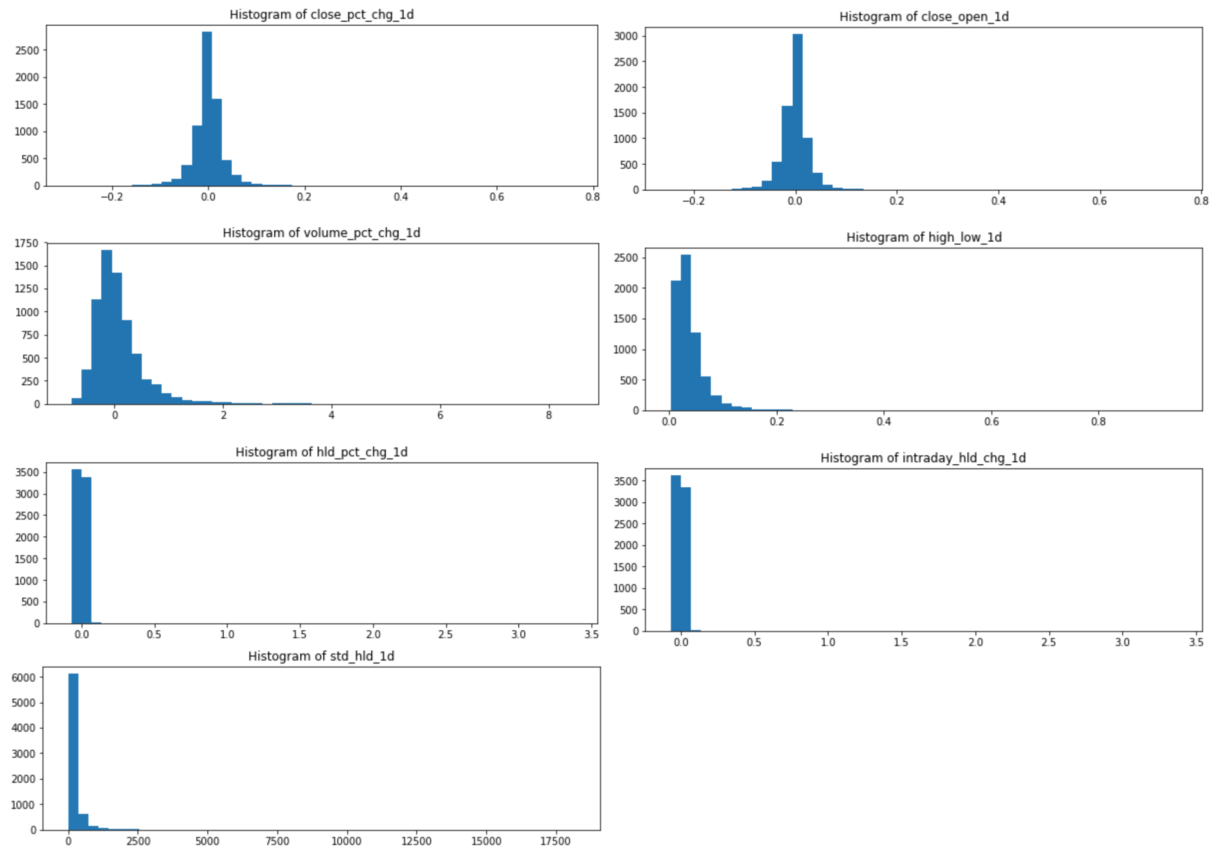
After performing clustering, I found the category of stocks that was characterized by high volume, high user holding and medium price (cluster 1). Intuitively, this made the category an ideal candidate for our analysis. In general, high volume reduces the extreme volume change, and high user holding from Robinhood data proves the stock is indeed frequently traded by retail investors. Moderate stock price also makes sense given high price stocks can hardly be afforded even if retail investor only buy 1 round lot, while penny stocks are usually small caps less known and hence less interest from retail.

CHART 2: STOCK CLUSTERING WITH K = 4



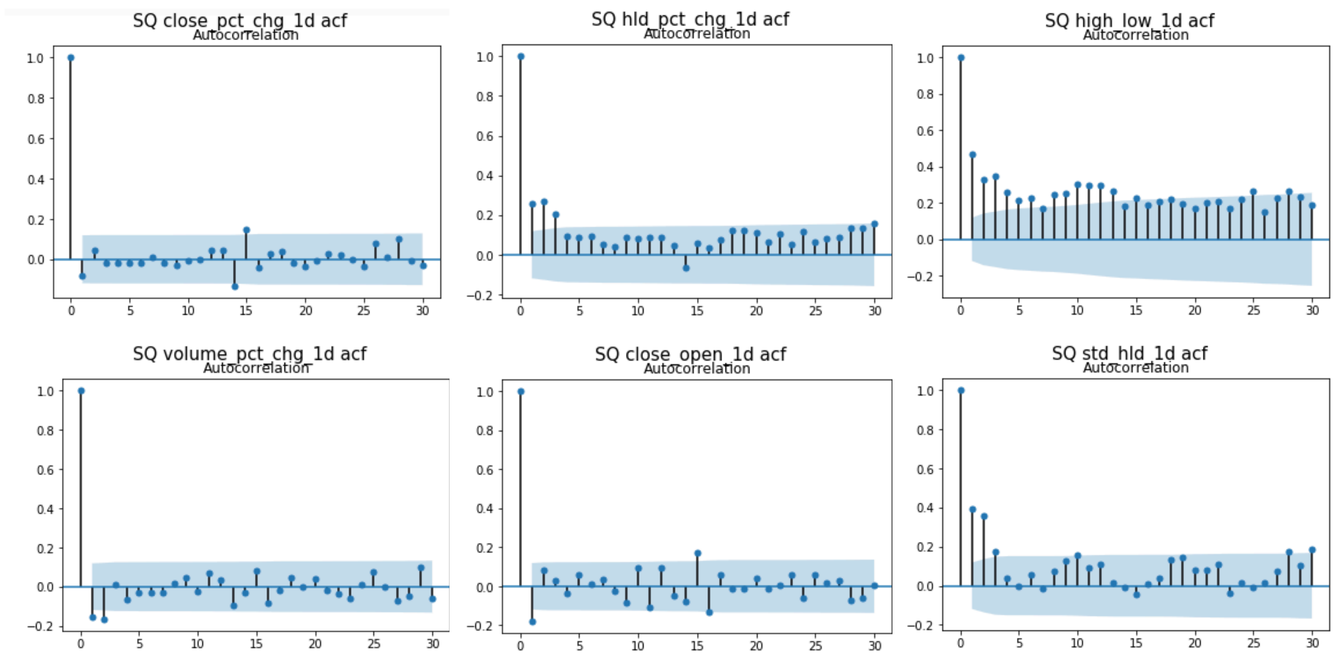
Within cluster 1, the distribution of all the features in percentage change terms are now within a normal range.

CHART 3: HISTOGRAM OF CLUSTER 1 SELECTED FEATURES



In order to understand if features in previous days can influence the target, I randomly chose stocks from cluster 1 and drew their autocorrelation chart.

CHART 4: AUTOCORRELATION



From above, I found using 5 days as the looking back period will be a good start as it covers the high autocorrelation number of lags for most features.

TABLE 2: CORRELATION MATRIX

	std_hld_1d	close_pct_chg_1d	volume_pct_chg_1d	hld_pct_chg_1d	close_open_1d	high_low_1d	intraday_hld_chg_1d
std_hld_1d	1.00	0.10	0.21	0.41	0.11	0.36	0.43
close_pct_chg_1d	0.10	1.00	0.14	0.23	0.82	0.09	0.24
volume_pct_chg_1d	0.21	0.14	1.00	0.06	0.06	0.31	0.06
hld_pct_chg_1d	0.41	0.23	0.06	1.00	0.26	0.38	0.97
close_open_1d	0.11	0.82	0.06	0.26	1.00	0.07	0.27
high_low_1d	0.36	0.09	0.31	0.38	0.07	1.00	0.39
intraday_hld_chg_1d	0.43	0.24	0.06	0.97	0.27	0.39	1.00

After checking correlation between features, I found close_pct_chg_1d and close_open_1d, hld_pct_chg_1d and intraday_hld_chg_1d have high correlations (>0.8), which makes sense given they are depicting the same thing except the first two values are day-to-day changes and are more inclusive of overnight moves. This high correlation would be taken care of after PCA was performed.

Algorithms and Techniques

1. K-means Clustering: In the first step of identifying suitable stocks to predict retail behavior, I used a K-means clustering algorithm, one of the simplest and popular unsupervised machine learning algorithms. K-means clustering aims to partition N observations into K clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster[3]. One important choice here is the number of clusters to form (K).
2. Principal Component Analysis: In order to make the training more efficient, I used Principal Component Analysis in making the final decision on choosing features. Principal component analysis (PCA) simplifies the complexity in high-dimensional data while retaining trends and patterns. It does this by transforming the data into fewer dimensions, which act as summaries of features[4].
3. Algorithms for predicting the target (increased / unchanged / decreased user holding): I chose to use Long Short Term Memory networks (LSTM), a type of recurrent neural networks. It is a

[3] Wikipedia on K-means Clustering: https://en.wikipedia.org/wiki/K-means_clustering

[4] Nature on definition of PCA: https://www.nature.com/articles/nmeth.4346

popular state-of-the-art for dealing with sequential data[5], capable of learning long-term dependencies. The outputs are an assigned probability for each class, which I will transform into 1 or 0 by selecting the highest probability class.

The following parameters can be tuned to optimize the classifier:

a. Training length (number of epochs)
b. Dropout value
c. Batch size (how many data to look at once during a single training step)
d. Number of hidden neurons

Benchmark

There is no existing study in the field specifically related to this topic. So I used a basic logistic regression model as my benchmark model.

There are a few parameters to be selected[6]:

1. C: Inverse of regularization strength; smaller values specify stronger regularization. After trying a few different values, I selected C=1e-2.
2. multi_class: If the option chosen is 'ovr', then a binary problem is fit for each label. For 'multinomial' the loss minimised is the multinomial loss fit across the entire probability distribution, *even when the data is binary*. After trying for both, I selected multi_class='multinomial'.
3. solver: For small datasets, 'liblinear' is a good choice, and it is therefore selected.

The benchmark model yielded a decent result with F1 score = 65%.

Methodology

Data Preprocessing

The preprocessing process was done in the "merge_price_files.py" and "merge_popularity_files.py" in the following steps:

1. Merge stock price volume files: Get stock ticker from csv file name, add the information as a column, concatenate all individual csv files.
2. Merge Robinhood files: Get stock ticker from csv file name, add the information as a column, concatenate individual csv files if the stock's data is available in the stock price volume file; group data by date, for each date, get the first user holding data and last user holding data reported, calculate standard deviation of the user holding data reported on the day. If any

[5] Understanding LSTM Networks: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[6] scikit-learn documentation on logistics regression algorithm: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

stock ticker's user holding is below 10 user holding at any point of time, I want to totally discard the data of that stock. This is because stocks with low user holdings can have user number fluctuations due to random irrelevant reasons. For example, Robinhood gives 1 lot of a stock for free to new users, this could cause noise when I interpret changes observed in user holdings.

The following steps were conducted after the raw data above was uploaded to S3, and were executed in Amazon SageMaker's Notebook (capstone_project_main.ipynb):

3. Merge stock price volume file and Robinhood file: Merge on stock ticker and date. At this stage, with absolute level data obtained above (first holding, last holding, standard deviation of holding, open price, highest price, lowest price, close price, volume) on each day on 3235 unique symbols from 2018-05-02 to 2019-06-11.
4. Create percentage change data from absolute values: Firstly, percentage change from previous day is calculated for close price, volume, Robinhood user holding. Since close price and volume data are snapped at end of the day, I used the last timestamp of Robinhood user holding to make it consistent. Secondly, I transformed some intraday values in different columns into intraday percentage data, i.e. calculating percentage change from the day's open to close, low to high, the day's first Robinhood holding reading to last reading, to further normalize those columns and make them more relevant under the time-series context.
5. Examine distribution of these percentage changes by looking at their percentiles and via visualizing them through histograms as well as checking the number of outliers.
6. Examine the target closely - Robinhood user holding percentage change. Check how balanced the data is and the data quality. Check when there was no change in user holding, if it was due to bad data in a specific stock or wide-spread phenomenon that infrequently happened across stocks.
7. Given the number of outliers found described in Analysis - Exploratory Visualization stage, I clustered stocks and selected cluster 1 stocks only, and performed step 5 and 6 again.
8. Further eliminate outliers with excessive volume jump: If a stock's mean volume change is larger than 100% and max volume change is larger than 1000%, then I will remove its entire time series. In Cluster 1, only one such stock was found and its entire time series was deleted.
9. Check autocorrelation and correlation of features.
10. Create feature data shift based on analysis of the autocorrelation pattern. For each date T, percentage change numbers from T-5 to T-1 of all the relevant features are created.
11. Convert the target value, user holding percentage change to 3 classes: increased = 0 , unchanged = 1, decreased = 2.

Implementation

The implementation process can be split into three main stages, as described briefly in "Definition - Problem Statement" and "Analysis - Algorithms and Techniques"

1. K-means Clustering to find the suitable stocks

1.1 Load raw data and preprocess as described previous section “Data Preprocessing”

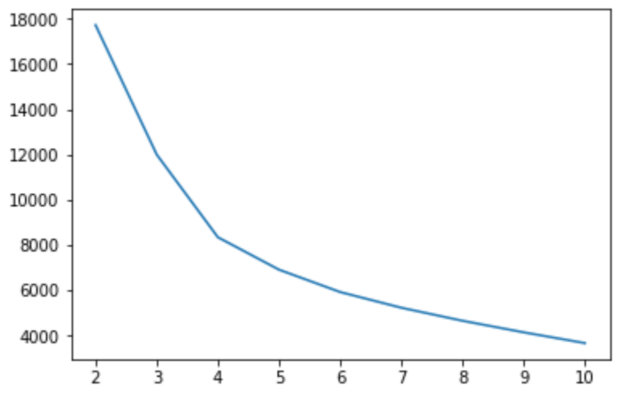
1.2 At each stock level, the average of raw data values (open, close, volume ,etc. in absolute terms instead of percentage changes) were calculated across all dates.

1.3 All the features are standardized using StandardScaler() from sklearn.preprocessing

1.4 K-means clustering was run from K = 1 to 10 based on those features of each stock. After plotting K vs. sum of distance vs centroids, the elbow appeared at K = 4 so K is selected as such.

Cluster 1 which has high volume and high user holdings while moderate stock price, has the best potential for predictable user holding change patterns.

CHART 5: K (X AXIS) PLOTTED AGAINST SUM OF DISTANCE TO CENTROIDS (Y AXIS)



The list of stock tickers in cluster 1 is shown as below:
['AAPL', 'AMD', 'APHA', 'BABA', 'BAC', 'CGC', 'CHK', 'CRON', 'DIS', 'F',
'FB', 'FIT', 'GE', 'GPRO', 'MSFT', 'MU', 'NFLX', 'NIO', 'NVDA', 'PLUG',
'SNAP', 'SPY', 'SQ', 'T', 'TSLA', 'TWTR', 'ZNGA']

2. PCA on features: After performing data preprocessing on cluster 1 stocks (described in previous section), I used PCA to trim down the number of features to make the training process later more efficient.

Before PCA, I had 7 unique features (listed below) * 5 days (from T-1 to T-5) = 35 features.

4 intraday columns:

std_hld_1d: standard deviation of user holdings data across all the timestamps on the day

close_open_1d: percentage change from the day's open price to close price

high_low_1d: percentage change from the day's highest price to lowest price (always >= 0)

intraday_hld_pct_chg_1d: percentage change from the day's first Robinhood holding reading to last reading

3 day-to-day columns:

close_pct_chg_1d: percentage change from previous trading day's close price to today's

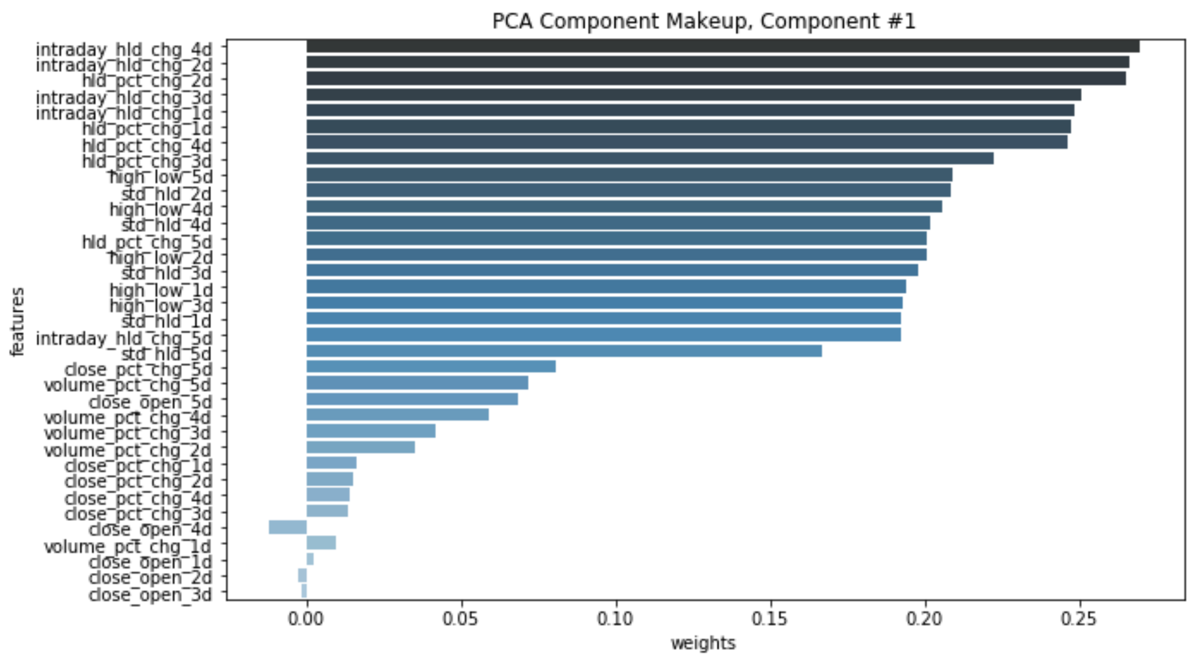
volume_pct_chg_1d: percentage change from previous trading day's volume to today's hld_pct_chg_1d: percentage change from previous trading day's Robinhood user holding to today's

2.1 Train/test split: For each ticker, sort by date and choose the earlier 80% dates as training data and the later dates as test data, instead of random shuffling to avoid look-ahead bias . One tricky part here is the data shouldn't be solely partitioned by date, but should be partitioned by date per stock. This is done by looping over stock names and splitting train/test for each of them.
In the same train_test_split() function, the train and test labels were also one-hot encoded during this step to be prepared for training.

2.2 All the features are standardized using StandardScaler() from sklearn.preprocessing

2.3 PCA fit and transform: By trying different number of components and calculating their explained variance ratio, I ended up using 20 components which sums to retain 93% of variance ratio. This is a significant improvement from having 35 components.

CHART 6: EXAMPLE - MAKEUP OF COMPONENT #1 AFTER PCA



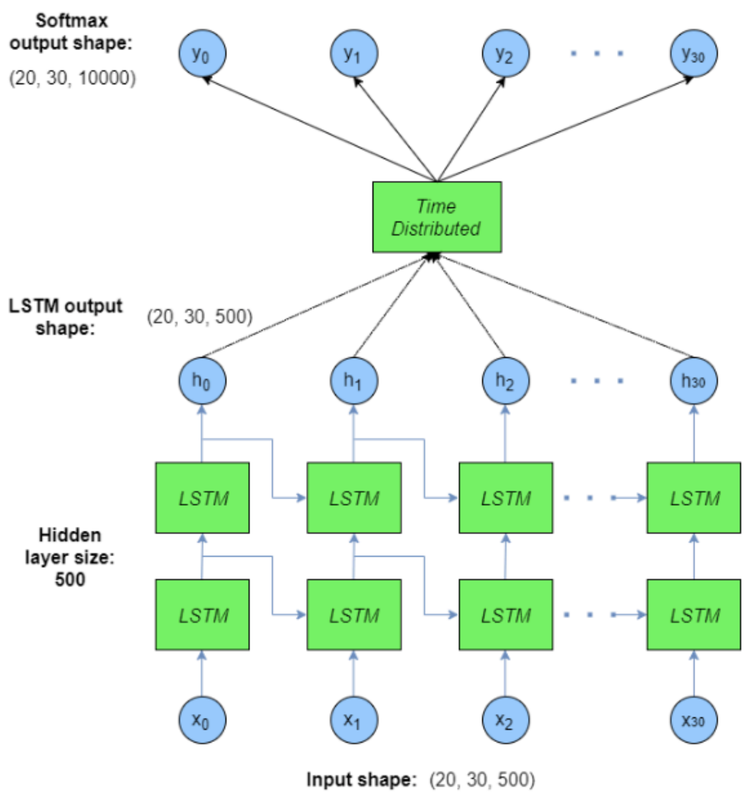
3. Running classification models

3.1 Logistics Regression: As described in details in "Benchmark" section, from training a logistic regression model, the F1 score produced on test set is 65%.

3.2 LSTM: The training process was done in the following steps:

- Define the network architecture and training parameters
- Define activation = softmax
- Define the loss function = categorical_crossentropy, optimizer = adam, metrics = accuracy, which are commonly used for multi-class classification problems
- Train the network, logging the validation/training loss and the validation accuracy
- Plot the logged values and review the pattern
- Try different values of parameters if result is not ideal

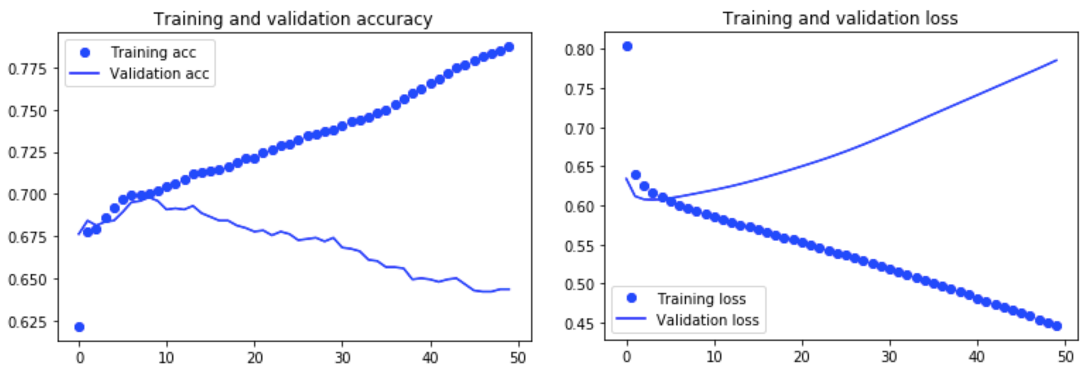
Below is a chart that describes how the model works[7]:



Refinement

By starting with no dropout value, I found loss function of training and validation data diverged, which indicated overfitting. Dropout works by probabilistically removing, or "dropping out," inputs to a layer, which may be input variables in the data sample or activations from a previous layer[8].

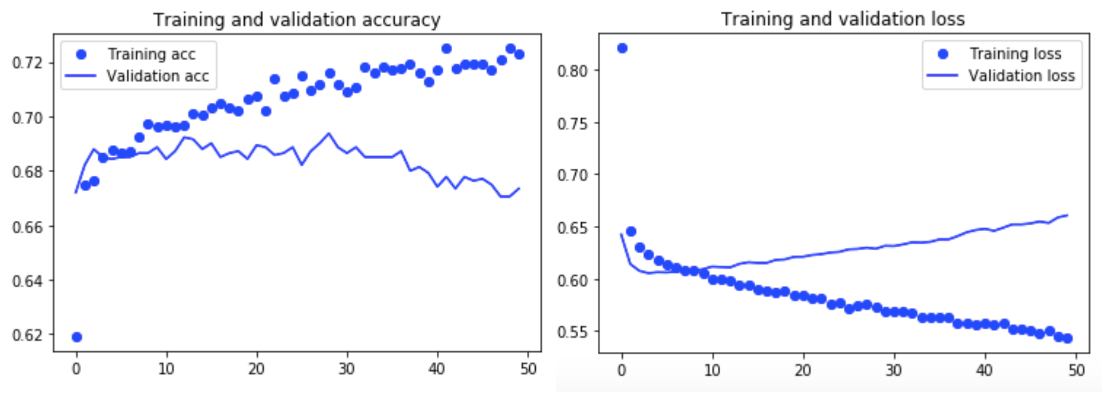
CHART 7: EXAMPLE OF OVERFITTING: LSTM LAYER = 300, EPOCH = 50, BATCH SIZE = 20, DROPOUT = NOT TURNED ON



[7] Chart is from: https://adventuresinmachinelearning.com/keras-lstm-tutorial/

[8] Machine Learning Mastery on dropout: https://machinelearningmastery.com/how-to-reduce-overfitting-with-dropout-regularization-in-keras/

CHART 8: EXAMPLE OF ADDING DROPOUT HELPED: LSTM LAYER = 300, EPOCH = 50, BATCH SIZE = 20, DROPOUT = 0.5



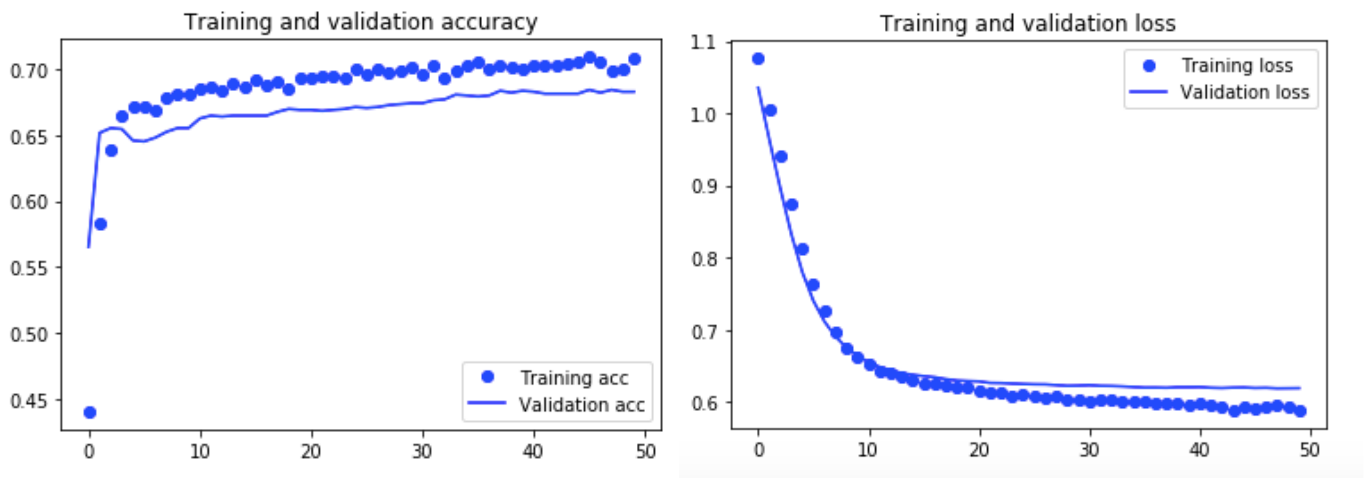
The batch size limits the number of samples to be shown to the network before a weight update can be performed. The introduction of hidden layer(s) makes it possible for the network to exhibit non-linear behavior. I performed similar trial and error on different combinations of parameter values, and found turning up batch size and number of hidden layers both helped too.

Results

Model Evaluation and Validation

After trying for different values of the combinations of the algorithm architecture and hyperparameters, here is what I decided to use:

- number of hidden neurons = 100
- dropout = 0.5
- epoch = 50
- batch size = 260
- activation=‘softmax'
- loss=‘categorical_crossentropy'
- optimizer='adam'



- metrics='accuracy'

F1 score achieved as 68.29%, better than the benchmark model.

Justification

From the perspective of F1 score, LSTM model did beat the logistic regression model, but by only marginally 3%. The result is not convincing enough for me to claim LSTM is better than the simple logistic regression in this case.

However, both models have presented promising predictive power when it comes to retail investor behavior. Through what I learned from industry practitioners, it is already pretty good to get better than 50% time correct. Because market is efficient and filled with salient and sophisticated professional investors. If there had been any opportunity from publicly known information, it could be easily arbitraged away. In addition, for a lot of strategies that runs on high leverage and high notional, a better than 50% accurate model is sufficient for making money.

Conclusion

Reflection and Potential Area for Improvement

As mentioned earlier, it is indeed a challenging task to make prediction in finance, since the market digests information very efficiently and is highly competitive. That is also why I chose a prudent goal when I designed the project, and only tried to predict direction of retail buying / selling instead of predicting asset prices directly.

In the futures, there are a few areas for improvement in this project:

1. Refine market data by adjusting for some basic known stock market effects such as seasonality.
2. Separate out effect from stand-alone one-off effects such as corporate actions and earnings, which will result in smaller number of outliers and make more data usable.
3. I should carefully evaluate if using the same clustering method / the threshold implied by the current cluster 1 will yield stable and consistent result when I replicate the analysis in the future.