Machine Learning Engineer Nanodegree

Capstone Project

Amrit Prasad
December 19th, 2017

Definition

Project Overview

On any given day, the stock market moves up or down in an ostensibly random manner. If these random fluctuations can be predicted via a model with more than an even probability, then a trader could utilise said model to profit from the market's moves. For this project, 5 years' worth of historical data relating to S&P 500 Index and VIX were downloaded from Yahoo Finance[1].

In the recent past, algorithmic trading has become the rage. It's essentially removing or automating away human involvement. It can be as simplistic as automating the data download via website crawling, to the creation of very complex models that use advanced math and finance domain knowledge that trade without human intervention. The advantage of using algorithms to trade instead of humans is that they lack human biases, execute faster, and have the ability to consume insane amounts of data. Machine Learning algorithms can readily be translated to solve these problems because they are essentially data fitters. Since the market interactions can be quite complex, and often beyond the purview of a single human to understand, ML techniques are also invaluable at figuring out links between disparate events which could then be analysed further to ascertain causality.

A few noteworthy ML applications include using sentiments' analysis (utilising positive/ negative news mentions) to trade on the relevant stock. There was an article on how Berkshire Hathaway stocks would increase every time Anne Hathaway was mentioned[2]. While definitely a false positive in this case, it's nevertheless an indication of how ML could be applied for trading. A very comprehensive ML application to stock market prediction can also be found here[3].

Problem Statement

The aim of this project is to utilise ML techniques, and try to come up with a model that would predict whether S&P 500 will close higher or lower than its open. Since there are two possible outcomes, this can be tackled as a classification problem, and I

aim to use a Decision Tree, SVM and Ensemble Leaners (Random Forests and Gradient Boosting), and choose the model with the highest cross validation F1 score. A label 1 should imply close above open the next day, while 0 should imply close below open. In case of predicted label 1, go long SPX, while go short in the other case. For unchanged index values, the labelling should be according to the average of the previous five trading days. This prediction can then be used by any amateur trader to profit by taking appropriate positions in SPY ETF/SPX Index Futures/derivatives.

Metrics

There are three evaluation metrics to consider here-

1) Accuracy score: This is defined as the fraction of correct labels out of all prediction.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ Population}$$

2) F1 score: This is defined as follows-

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Where,

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

This score is the harmonic mean of how accurately the actual positives have been labelled positive (recall), and how many of the positive labels are actual positives (precision).

3) XIRR of the strategy implementation: Once the models have been trained, and the best chosen on the basis of the above, we should try to see the XIRR of running the strategy assuming that the SPX Index can be bought and sold at the index values, which is not too bad an assumption (SPY ETF/Emini Futures do a pretty decent job of capturing the daily moves). The Internal Rate of Return (IRR) is the discount rate that makes the net present value of a series of cashflows zero. It tells you how much a given project/strategy would yield in a given period of time. Greater the IRR, better the strategy. A disadvantage of that is that it's not scaled by the time horizon, which is taken care of by XIRR. XIRR is the annualised IRR of a given set of cashflows. The annualization takes care of the previously discussed issue with IRR.

Accuracy alone isn't a very good metric to use for comparison because it weights the True labels way more than the False ones, and in some cases a False label might actually be quite damaging. Finance is one such domain. A false label on which you execute a trade could lead to disastrous losses. Hence, an F1 score is a better

alternative and should be chosen along with Accuracy while evaluating whether the final model is better or not.

Analysis

Data Exploration

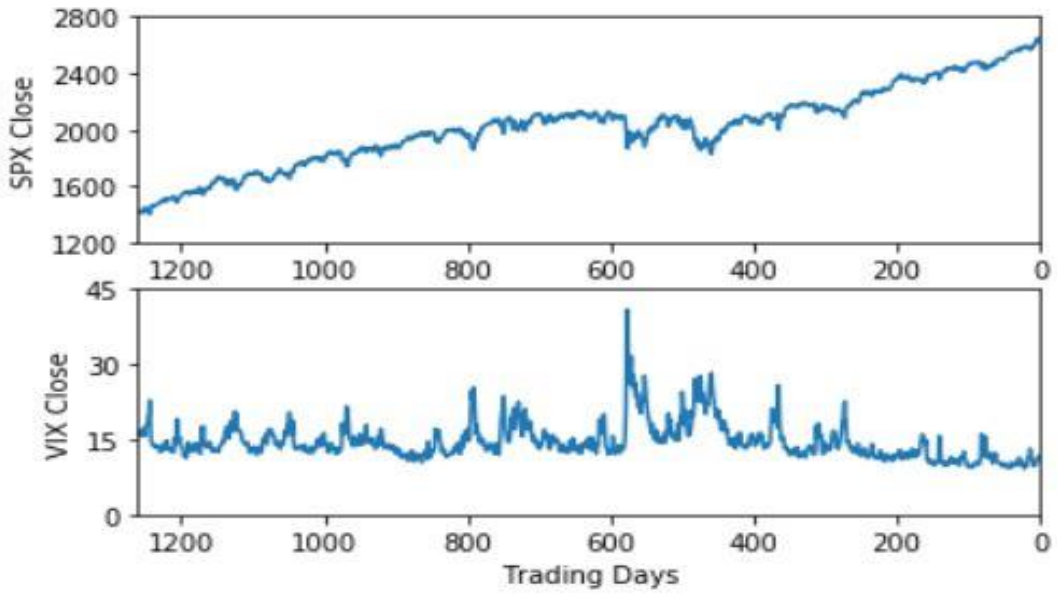
For this classification problem, I have tried to limit myself to numerical raw data like open/close price, volume and market expectation of 30-day volatility (VIX). The following raw data fields are going to be used in this problem-

1) Date: Date on which the data was recorded
2) Open_SPX: Open value of SPX Index
3) High_SPX: High value of SPX Index
4) Low_SPX: Low value of SPX Index
5) Close_SPX: Close value of SPX Index
6) Volume_SPX: Volume of SPX Index constituents traded
7) Open_VIX: Open value of SPX VIX
8) High_VIX: High value of SPX VIX
9) Low_VIX: Low value of SPX VIX
10) Close_VIX: Close value of SPX VIX

The above dataset was downloaded from Yahoo Finance[1] for the last 5 years of trading days from 6th Dec 2012 to 6th Dec 2017. Each data field has a value corresponding to the Date of trade. If any of the fields is missing/invalid, then it should be filled by the average of the 5 previous trading day values. There are 1260 trading day values corresponding to each of the above-mentioned features in the dataset used. A brief selection is showcased below-

Date	Open_SPX	High_SPX	Low_SPX	Close_SPX	Volume_SPX	Open_VIX	High_VIX	Low_VIX	Close_VIX
06-12-2012	1409.43	1413.95	1405.93	1413.94	3,229,700,000	16.59	16.85	16.31	16.58
07-12-2012	1413.95	1420.34	1410.90	1418.07	3,125,160,000	16.12	16.65	15.73	15.90
10-12-2012	1418.07	1421.64	1415.64	1418.55	2,999,430,000	16.47	16.47	15.96	16.05
11-12-2012	1418.55	1434.27	1418.55	1427.84	3,650,230,000	15.94	16.01	15.42	15.57
12-12-2012	1427.84	1438.59	1426.76	1428.48	3,709,050,000	15.60	16.09	15.41	15.95

The following graphs showcase the close of the S&P 500 and VIX Indices-



As can be seen from the graph above, SPX has increased from around 1500 to 2500, while VIX has fluctuated between 5 to 45. Clearly these numerical ranges aren't directly comparable, and new standardised features need to be engineered, that will be the input for training the model. These new features are mentioned in the Project Design section.

As mentioned in the Problem Statement above, close > open → label = 1, close < open → label = 0, and close = open → label = average of last 5 labels. If last 5 labels aren't available, then discard the data rows. After all the pre-processing was completed, the final tally was 532 out of 1007 data rows were labelled 1.

The raw dataset was balanced, and every chosen feature had an entry corresponding to every date. This is most likely because S&P 500 is probably *the* most tracked index on the planet, and hence its data quality is quite good. The dataset after the calculation of the new engineered features was unbalanced due to the presence of returns with varied time horizons (1d, 2d, 1yr etc.). Clearly 1yr returns won't be present for all dates that the 1d return would be present.

Algorithms and Techniques

Since I have constructed the problem in such a way that the prediction needs to be a binary quantity (close > or < open), and the label is readily calculated, supervised classification techniques would be ideal for attacking this problem. Four such techniques would be used in this project:

1) Decision Trees: These are intuitive rule based bifurcators, that split the data on the basis of feature thresholds. Their advantages are their interpretability, speed of training and prediction. Their disadvantages are a tendency to overfit, and high sensitivity to training data.

2) SVM: These are extremely versatile classifiers that through the use of kernels can create very accurate models. Their advantages are their accuracy, handling non-linear features, and low susceptibility to outliers since they concentrate only on the data points close to the margin. Their disadvantages are their non-interpretability, slow learning speed, and risk of overfitting in case of low domain knowledge or inability to translate that into the correct choice of kernel.
3) Random Forests: These are a kind of Ensemble Learning Method that train multiple weak learners (typically Decision Trees), and then predict on the basis of mode/mean of the individual models' predictions. They tend to reduce the overfitting issue endemic to Decision Trees.
4) Extreme Gradient Boosting: Gradient Boosting are another kind of Ensemble Learning Method that creates new learners after weighting the examples which were harder for the previous learner. They are the least prone to overfitting, but can be extremely slow while training, especially if the number of estimators are too high. Extreme Gradient Boosting are a faster implementation of the above, that were chosen because of the very slow training speeds of the Gradient Booster.

Benchmark Model

The benchmark model to be beaten would be the passive investing strategy in the most popular benchmark index of the stock/index's origin country, which advises buying and forgetting about the investment. This is the benchmark which 80% of fund managers are unable to beat year on year[4]. For our case, S&P 500 would be the benchmark index. We can invest in SPY ETFs/long dated forward contracts for simulating the benchmark in real life. For the purposes of this project, I'll be using the SPX Index values as the theoretical benchmark, which should correspond very closely with the real-life version.

Methodology

Data Preprocessing

As discussed in the Data Exploration section, we need to try and extract useful, standardised information out of the dataset (after it was cleaned as stated in the Data Exploration section), that could be used for predicting whether SPX closes above or below the open. A very important thing to keep in mind while working with this data is to ensure that look-ahead bias is avoided. Hence, if we are trying to predict something on trading day x, then all data should correspond to trading days x-1 and before. A few useful features that could help us in our prediction for trading day x are-

1) Trailing_1d_Return: Close(x-1)/Close(x-2) – 1 of SPX Index

2) Trailing_1d_Max_Move: High(x-1)/Low(x-1) – 1 of SPX Index
3) Trailing_2d_Return: Close(x-1)/Close(x-3) – 1 of SPX Index
4) Trailing_3d_Return: Close(x-1)/Close(x-4) – 1 of SPX Index
5) Trailing_4d_Return: Close(x-1)/Close(x-5) – 1 of SPX Index
6) Trailing_5d_Return: Close(x-1)/Close(x-6) – 1 of SPX Index
7) Trailing_10d_Return: Close(x-1)/Close(x-11) – 1 of SPX Index
8) Trailing_22d_Return: Close(x-1)/Close(x-23) – 1 of SPX Index
9) Trailing_63d_Return: Close(x-1)/Close(x-64) – 1 of SPX Index
10) Trailing_252d_Return: Close(x-1)/Close(x-253) – 1 of SPX Index
11) Trailing_1d_Return_VIX: Close(x-1)/Close(x-2) – 1 of SPX VIX
12) Trailing_1d_Max_Move_VIX: High(x-1)/Low(x-1) – 1 of SPX VIX
13) Trailing_5d_Return_VIX: Close(x-1)/Close(x-6) – 1 of SPX VIX
14) is_MA5_above_MA20: 1 if average of Close(x-1) to Close(x-6) > average of Close(x-1) to Close(x-21); else 0 of SPX Index
15) is_Trailing_1d_Vol_above_MA10_Vol: 1 if Vol(x-1) > average of Vol(x-1) to Vol(x-11); else 0 of SPX Index
16) Trailing_1d_Gap_Return: Open(x-1)/Close(x-2) – 1 of SPX Index

Close(y) above refers to the Close price on trading day y.

Once the above features have been calculated for however many days they can be, the next step involved cleaning up the data by dropping all NaN values, which will creep in if the date ranges are out of scope. Post the removal of NaN, close to 4 years' worth of data was left (the 1-year return was responsible for most of the data rows discarded). After the data cleaning comes the part where we split the data set into training, cross-validation and test sets. The remaining 4 years can then be split into 60%-20%-20%. A key thing to note here is that the data should be split *sequentially* rather than *randomly*, to avoid look-ahead bias.

Implementation

The first step is to import the evaluation metrics which is the F1 score. The accuracy score is also a good to have.

Post this, the benchmark metrics on the training/cross-validation/test sets were calculated. They were as follows-

Set	F1	Accuracy	XIRR
Training	0.6898	0.5265	5.70%
Cross-Validation	0.6968	0.5347	18.77%
Test	0.6906	0.5274	14.89%

Initially these models were trained on the training dataset to see how they performed while using the default values. Post that they were fine-tuned by changing the hyper parameters. The best tuned model out of them was chosen on the basis of the best F1 score on the cross-validation set. Out of these best tuned models, the one with the highest accuracy, F1 score and XIRR should be chosen for the comparison against the benchmark.

The Gradient Booster training times increase exponentially with the number of estimators (parameter n_estimators). This turned into a major problem while tuning because I had chosen quite large values of n_estimators. To counter this, I switched to Extreme Gradient Boosting, which is essentially a speeded-up version of Gradient Boosting. The training times for individual models were cut by a factor of upto 45, and allowed me to tune using relatively high values for n_estimators.

Also, as mentioned in the reflection part, the syntax for the creation of tuples that are inputs for the XIRR function was quite tricky due to multiple ways in which tuples can be joined. This resulted in multiple errors with not so clear error messages, and attempts to Google would invariably point to a different than the required way of joining tuples.

Refinement

This section details the fine tuning that was done to ensure that the models used ended up with the highest possible F1 score on the cross-validation set. The methodology used was fairly straightforward, in that a bunch of critical hyper parameters for each model were recognised, their lists of valid values created, and the models trained repeatedly after passing different values for the parameters.

The following results were observed-

1) Two Decision Tree models had better F1 score than the base model. The best one had a maximum depth of 4, and minimum samples per leaf of 9. Its F1 score was 0.6429
2) None of the SVMs trained were better than the base model, which calculated that the benchmark was the best fit.
3) Eight Random Forests were better than the base model. The best one had a max depth of 1, minimum samples per leaf of 7, split criterion of entropy, 60 number of estimators, and all features used in every constituent tree. Its F1 score was 0.6721
4) The Extreme Gradient Booster tuning resulted in the benchmark being calculated as the best fit again.

Post this a voting classifier was trained that combined the Decision Tree, Random Forest and the benchmark (Extreme Gradient Boosting was chosen, but it's the same

as the benchmark along with SVM). It resulted in a model that improved upon the F1 score of the best model (Random Forest) marginally, but saw a crash in its XIRR.

The final summary table with the results of the best tuned models is showcased below-

Best Learner Type	F1	Accuracy	XIRR
Decision Tree	0.6429	0.5545	33.69%
SVM	0.6968	0.5347	18.77%
Random Forest	0.6721	0.5990	62.86%
Gradient Booster	0.6968	0.5347	18.77%
Voting Classifier	0.6782	0.5396	26.90%

Results

Model Evaluation and Validation

None of the models did better on both F1 score and XIRR. In fact, SVM and Extreme Gradient Boosting returned the same values as the benchmark speaking of how hard it is to beat the passive strategy. This tends to verify that the weak form of the efficient market hypothesis holds for S&P 500. The weak form essentially states that all past price/volume movements have been incorporated into an asset's price, and the only way to 'beat the market' is by utilising fundamental analysis. The strong form invalidates the fundamental analysis aspect as well, but verifying that is out of the scope of this project.

Decision Tree and Random Forest were better on the accuracy and XIRR fronts though on the cross-validation set. The better of them, i.e. Random Forest was tested for robustness. For this, I calculated its F1 scores, and XIRR on the cross-validation set for 100 different values of random states. The results are intriguing-

1. F1 scores lie between 0.5674 and 0.6721 with a mean of 0.6196 and std. deviation of 0.0217. The max fails to beat the benchmark.
2. XIRRs lie between 19.90% and 745.09% with a mean of 75.19% and std. deviation of 88.72%. The min beats the benchmark.

A thing that's clearly obvious from the above results is that choosing random state doesn't impact whether the tuned Random Forest fails or succeeds to beat the benchmark. It consistently fails on the F1 metric, and succeeds on XIRR.

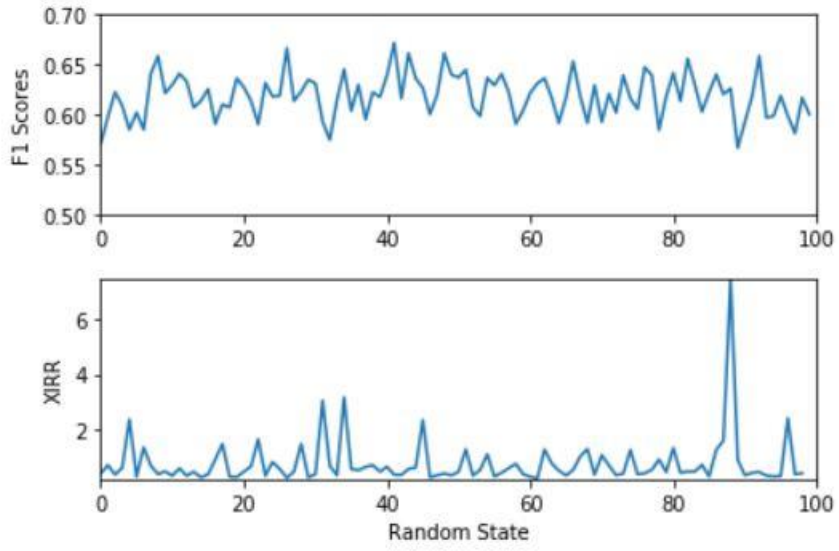
Justification

Random Forest was implemented on the test set as well, after being trained on the combined training + cross-validation sets. It again fails to beat the F1 score of the benchmark, but its XIRR remains exceptional. I would be sceptical of using this however for trading because it regularly fails to beat the F1 score even though the XIRR was regularly better, indicating that the returns could be due to fortuitous trades on the cross-validation set rather than the result of actual predictive power.

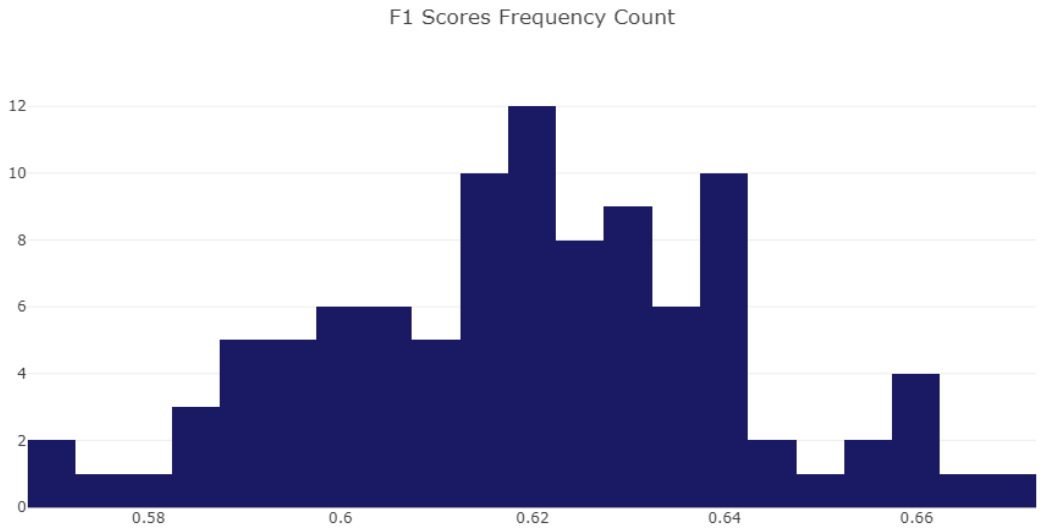
Conclusion

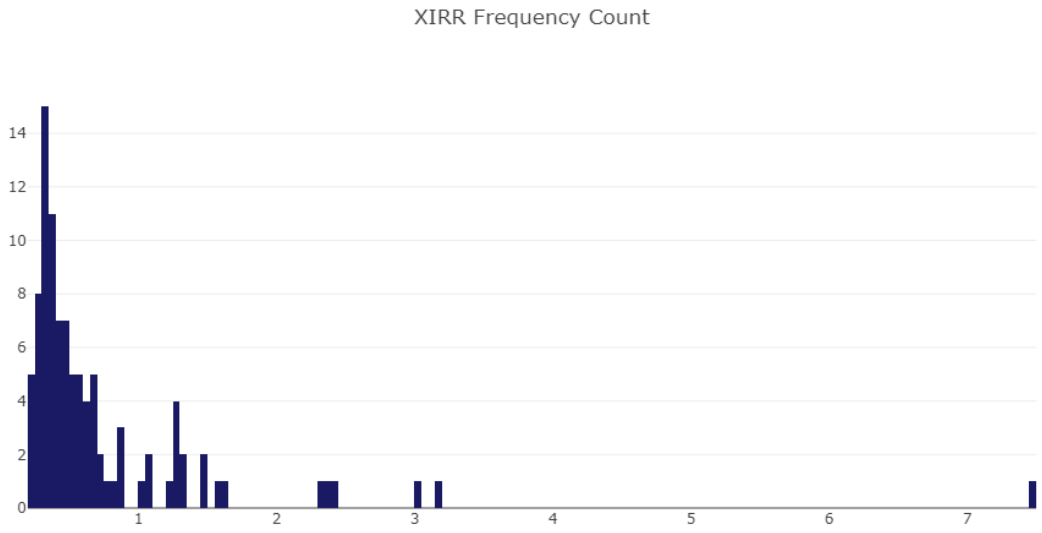
Free-Form Visualization

Something that really caught my attention while testing the tuned Random Forest for robustness, is how sensitive it is to the value of the random state. This is quite concerning since the random state is just the seed value that's passed to a pseudo-random number generator, and doesn't actually govern any aspect of the algorithm's behaviour. This is how the F1 scores and XIRR of the Random Forest varied according to the random state-



As you can see, the scores are not at all stable and highly dependent on the seed value. The distributions were as follows-





The plots clearly show that the extreme values of XIRR are a coincidental outlier, and should NOT be taken to mean that the Random Forest has performed exceptionally well.

Reflection

This project tried to classify whether SPX would close higher or lower than its open daily. This knowledge can easily be used to trade on the financial markets by going long in case predicted label 1 and short in case of predicted label 0.

SPX data was readily available on Yahoo[1], which was used to engineer features with an aim towards eliminating look-ahead bias by utilising only those data points which corresponded to older dates.

A benchmark of buy and hold (predicts all 1s) was chosen, and its metrics calculated on the training/cross-validation/test sets. Post that ML classification techniques like Decision tree, SVM, Random Forest and Extreme Gradient Boosting were trained on the training data set, tuned on the cross-validation data set, and the best chosen on the basis of XIRR and F1 score on the cross-validation data set. It was observed that none of the models succeeded to beat the benchmark on the F1 metric, but Random Forest and Decision Tree managed to get a better XIRR. Random Forest was the better out of the two.

A voting classifier was tried that tried to create an ensemble of the Random Forest, Decision Tree and the benchmark (tuned SVM and Extreme Gradient Boosting predicted all 1s). It was trained and made to predict exactly like the models before. It improved on the F1 score marginally than the Random Forest.

The Random Forest was checked for robustness by checking its metrics after changing its random state. 100 values were tried and the conclusion was that even though the random state made its F1 score and XIRR jump around significantly, it didn't impact the answer to the question, Does the Random Forest perform better than the benchmark on metric X? The Random Forest models continued to perform worse on F1, and better on XIRR.

Finally, the Random Forest and Voting Classifier were tested on the test sets. Random Forest continued the trend of mixed fortunes of lower F1 and higher XIRR, while the Voting Classifier was worse on both counts.

There were two parts of this project that particularly challenged me-

1) Getting the cashflows in the correct form of tuples for feeding into the XIRR function that I took from [5]. Joining tuples is a mess. They can be added as tup_1 + tup_2 or (tup_1,) + (tup_2,) or tup_1, tup_2. Only one of them gives the correctly accepted form, and googling for this error is a nightmare for a Python neophyte, because of the multiple ways of joining.
2) Figuring out what hyper parameters to tune, and get that F1 score to go beyond the benchmark's was the hardest part. Unfortunately, I didn't succeed there. A voting classifier didn't help either. This could be either due to S&P 500 being weak market efficient, or that the features chosen by me didn't have enough predictive power.

Improvement

The biggest improvement that I can think about making in this project would be to figure out a way to translate Finance domain knowledge into choosing parameters that would result in a model that would beat the benchmark. Currently the process utilised was quite crude in that datasets were created and models tuned manually on it by trying out random permutations of a list of hyper parameters. If they did end up beating the benchmark, they were to be promoted to run on the test set. But this would result in a model that *fits* well on three randomly chosen time series data of S&P 500, and doesn't necessarily have to capture something truly fundamental about the markets. Due to sheer coincidence, it is possible to find models that fit perfectly on the 3 specific datasets (training/cv/test), but are worthless otherwise[6]. This approach of throwing models at data to see what sticks can certainly be improved by figuring out a way to incorporate finance knowledge or seeing if it makes some financial sense.

I would expect such a model to perform *consistently* in the real markets, something which most active fund managers fail to achieve.

Bibliography

[1] https://finance.yahoo.com/quote/%5EVIX/history?period1=1354818600&period2=1512585000&interval=1d&filter=history&frequency=1d

[2] https://www.huffingtonpost.com/dan-mirvish/the-hathaway-effect-how-a_b_830041.html

[3] https://www.duo.uio.no/bitstream/handle/10852/51275/PredictingStocksWithMachineLearning.pdf?sequence=1

[4] http://us.spindices.com/documents/research/research-fleeting-alpha-evidence-from-the-spiva-and-persistence-scorecards.pdf

[5] https://github.com/peliot/XIRR-and-XNPV/blob/master/financial.py

[6] https://www.forbes.com/forbes/welcome/?toURL=https://www.forbes.com/sites/moneybuilder/2015/03/04/the-bangladesh-butter-indicator-says-buy/&refURL=https://www.google.com/&referrer=https://www.google.com/