SPX Index Price Direction Prediction Model-Final Report

Team 53

Abstract

In this experiment, we attempted to predict whether the next day's closing price of the SPX index (S&P 500) would be higher or lower than the current day's closing price. We used features and technical indicators derived from open-high-low-close data, an implied volatility index, and macro-economic variables with the goal of building a prediction model that could provide a probabilistic trading edge.

Our goal was to build two or possibly three classification models using Logistic Regression (LR), Support Vector Machine (SVM) and/or an artificial neural network (ANN) and then select the best performing model. In the end, we were able to test a Logistic Regression and an SVM model with both models performing similarly in terms of accuracy (Logistic Regression slightly higher).

Initial tests of the LR model using variables and indicators derived from the open/high/low/close data of the SPX index showed high p-values and weak predictive power, however, results improved with the addition of features derived from macro-economic variables. Several of the variables showed statistically significant p-values below 0.05, and the model itself which included these variables also showed statistical significance. However, the best accuracy achieved by either model was still only 2%-3% above the baseline probability of naively guessing "up" every day which would have yielded accuracy of about 54% in our data set.

After examining our results in a confusion matrix, we were able to come up with a promising way to use either (or both) of the models as the basis for a trading edge. Please see details in our conclusion.

Introduction

The US stock market is the largest in the world and the SPX index consists of the largest 500 companies in this market. Therefore, we expect this market to be efficient and for there to be few, if any, inefficiencies to exploit. So why then would anyone wish to undertake the challenge of predicting whether the SPX index will close higher or lower on the next trading day versus the current day? The reason is that as of Q2, 2022, the CBOE began offering Tuesday and Thursday options expirations on the SPX index (in addition to Mon, Wed, Fri options), making it the only securities instrument with options expiring 5 days per week. Additionally, having options with strikes very close together (only 0.13% apart) makes expressing a binary outcome for the next trading day very easy to do.

So why would trading every day be worth doing? Consider, the risk and reward in any options position is generally commensurate with the odds of success in the position, and odds of success are determined by where the options strikes are positioned relative to the current SPX price. For example, 1-to-1 odds equates to a position that could either lose $50 or profit $50. Odds of 60/40 for a favourable outcome

would equate to a risk of $60 to make a profit of $40, and so forth. This means that any ability to predict better than the market is pricing can result in a profit over time.

Now imagine a trader uses a predictive model near the end of the current trading day to determine that the probability of an up move on the next day is 57% when the market is pricing a 52% chance of an up move. The trader now has a probabilistic edge over the market. The edge of 5% over the market will result in a positive expected value per trade when played out over a long period of time. The greater the probability edge over market, the more profitable the trades will be. With about 250 trading days per year, the strategy offers great potential for compound growth. Thus, even a small probabilistic edge could be the basis for a profitable trading strategy. Assembling macro-economic, fundamental, and technical analysis data, and combining these with domain knowledge, we are attempting to build a model with prediction accuracy at least 5% greater than the baseline rate of 54% (from guessing "up" each day).

The remainder of this paper will describe how (and from where) we have collected data, any feature-engineering steps we've taken, why the features chosen should have predictive power, how we built the LR and SVM models, and how we interpreted the results. Additionally, we'll discuss next steps we could take were we to extend our research beyond the scope of this project.

Sourcing, Cleaning, and Joining Datasets

Most data were obtained using the premium (paid) version of the TradingView charting application which allows for technical and fundamental data as well as open-high-low-close price data to be saved to csv files. While the same data could be obtained from free sources via API connections using R, the technical indicators would have to be calculated separately which could be very time-consuming and open the possibility of error in the calculations. Additional macro-economic data, including Real GDP and CPI, were gathered from the FRED (Federal Reserve Economic Data) website. Using the macro-economic data presented two challenges:

1) Joining the macro-economic data, which has frequency of quarterly and monthly with the SPX index data which has daily frequency and

2) How to estimate data points that will not be reported for several weeks or months. For example, it is currently Q4, but Q4 GDP will not be reported until Q1 of next year.

Joining the data was accomplished using Microsoft Power BI. Here, it was a matter of creating a contiguous date column spanning the entire range of the SPX data, then LEFT joining the respective datasets on the date column. Columns of data, now in a single dataset, were then cleaned by filling monthly or weekly data to daily frequency, removing null values, and changing data types as needed. Thus, the most recent data point will be used for current-period data that have not yet been reported (such Q4 GDP). From this process, we got a single csv file with all the datasets combined and ready to read into R. Once loaded into R, the data needed some additional cleaning steps as not all the data type formats were transferred correctly through the csv file.

Feature Selection

Our first iteration of the LR model was done with data taken directly from the SPX index. The SPX index data set consists of daily data from the beginning of 2013 until early October 2022 (over 2400 data points). TradingView has hundreds of technical analysis indicators to choose from and allows about 20 indicators to be placed on a chart. For this experiment, we selected the open/high/low/close (OHLC) plus several indicators that we felt could have predictive power in determining whether tomorrow's price would be higher or lower than today's price. The LR model built using only these variables exhibited high p-values for most variables and low predictive power. The next iteration of the LR model used all the data we gathered and the features we engineered. All the variables used, their descriptions and source are shown in the table below. The variables noted with * are the ones that made it into the final version of the models.

Variable Name	**Description**	Source
Date	Date applicable to the data point	NA
SPX_Index.Change %*	Day over day change of the closing price of SPX index, calculated as ln(current/previous)	TV
SPX_Index.UP_DOWN*	The dependent variable - a binary indicator of whether the next day's closing price was higher than the current day's closing price.	TV/engineered
SPX_Index.% from MA5*	The percentage difference between the current day closing price and the 5-day moving average of the index closing prices. Positive indicates short-term uptrend and negative indicates a short-term downtrend.	TV/engineered
SPX_Index.% from MA200*	The percentage difference between the current day closing price and the 200-day moving average of the index closing prices. Positive indicates long-term uptrend and negative indicates a long-term downtrend.	TV/engineered
SPX_Index.MA5 Slope	Binary (+1 or -1) indicator of short-term trend.	TV/engineered
SPX_Index.MA200 Slope	Binary (+1 or -1) indicator of long-term trend.	TV/engineered
SPX_Index.Closing Strength	A measure of how close to the high of the day was the closing price. Calculated as (close-low)/(high-low). The rationale is that a close near the high of the day might indicate strength going into the next day while a close near the low of the day would indicate weakness.	TV/engineered
SPX_Index.RSI Signal	Relative Strength Index oscillates between values of 0 and 100 and indicates whether a stock is overbought (above 70) or oversold (below 30). Signal generates a bullish +1 when the RSI crosses from below 30 to above 30, and a bearish -1 when it crosses from 70 to below 70.	TV/engineered
SPX_Index.ADX Signal	ADX – Average Directional Index indicates the strength of a trend but not its direction. Values above 25 indicates there is a strong trend, while values below 20 indicate absence of any trend. Signal generates 1 when a trend is present, otherwise 0. Rationale for including is that there may be better opportunities when the market is in a trending phase.	TV/engineered

SPX_Index.Volatility	A decimal representation of Volatility Percentile which indicates whether a stock is currently experiencing either high or low volatility as compared with the previous 30 trading days.	TV/engineered
VIX.VIX Change %	This is the current day percentage price increase or decrease compared to the prior day applied to the VIX index. VIX measures 30 day implied volatility on the SPX index.	TV/engineered
VIX.VIX Lag1 %	This is the previous day VIX change %. Rationale for including is that the VIX is forward-looking and an increase may indicate that markets will soon move lower.	TV/engineered
VIX.% from MA5	Indicates whether the VIX is currently above its short-term moving average.	TV/engineered
VIX.% from MA200	Indicates whether the VIX is currently above its long-term moving average.	TV/engineered
VIX.MA5 Slope	Indicates whether the VIX is in a short-term uptrend or downtrend.	TV/engineered
VIX.MA200 Slope	Indicates whether the VIX is in a long-term uptrend or downtrend.	TV/engineered
VVIX.VVIX Change %	VVIX measures implied volatility on the VIX index. Increasing levels indicate traders are hedging by using call options in the VIX index which could be in anticipation of sudden move lower in markets.	TV/engineered
VVIX.VVIX Lag1 %	This is the previous day VVIX change %. VVIX is also forward-looking.	TV/engineered
VVIX.% from MA5	Indicates whether the VVIX is currently above its short-term moving average.	TV/engineered
VVIX.% from MA200	Indicates whether the VVIX is currently above its long-term moving average.	TV/engineered
VVIX.MA5 Slope	Binary Indicator of whether the VVIX is in a short-term uptrend or downtrend. +1 for increasing, -1 for decreasing.	TV/engineered
VVIX.MA200 Slope	Binary Indicator of whether the VVIX is in a long-term uptrend or downtrend. +1 for increasing, -1 for decreasing.	TV/engineered
GDP.Change %	Percentage change in the real GDP since the previous quarterly reading. The number is filled forward in time until the next reading is published, typically 4 weeks after the quarter end.	FRED/engineered
GDP.Real GDP Slope*	Indicates directionally whether GDP (gross domestic product) is growing or shrinking. +1 increasing; -1 decreasing.	FRED/engineered
CPI.CPI Slope	Indicates directionally whether CPI (consumer price index) is growing or shrinking. +1 increasing; -1 decreasing.	FRED/engineered
FED_BS.Balance Sheet Change %	Indicates expansion or contraction in the Federal Reserve balance sheet. Expansion provides liquidity to markets and eases financial conditions - generally bullish for markets.	FRED/engineered

FED_BS.Balance Sheet Slope	Binary indicator of whether Fed balance is increasing or decreasing. +1 increasing; -1 decreasing.	FRED/engineered
FED_FUNDS.FedFunds Slope	A very short-term interest rate set by the Federal Reserve Open Market Committee. This rate ripples through all other interest rates in the economy. The slope indicates if the rate is rising or falling.	FRED/engineered
2YR_Bond.2YR Change %	Change in rate of the 2-year treasury bond. Closely related to the Fed Funds rate.	TV/engineered
2YR_Bond.2YR Slope	Binary indicator of whether the 2 year rate is increasing or decreasing.	TV/engineered
30YR_Bond.30YR Change %*	Change in rate of the 30-year treasury bond.	TV/engineered
30YR_Bond.30YR Slope	Binary indicator of whether the 30 year rate is increasing or decreasing.	TV/engineered
NFCI.NFCI*	National Financial Conditions Index - compiled by the Chicago Fed, this index tracks financial conditions in the money markets, debt, and equity markets. Rationale for using is that tightening financial conditions are negative for equity markets.	FRED/engineered
NFCI.NFCI Slope*	Binary indicator of whether the NFCI is increasing or decreasing.	FRED/engineered
Month Name	The month of the observation. Included to determine if there is a seasonal effect to the equity markets in certain months.	TV/engineered
Day Name	The day of the observation. Included to determine if there is a seasonal effect to the equity markets on certain days.	TV/engineered

The idea to encode some variables with their meaning from a technical analysis perspective instead of the raw values came from the paper by Patel and Shah. They tried two approaches, first using the raw (continuous) values for predicting, and then using what they termed “trend deterministic” data. They did this by discretizing the continuously-valued data. Our procedure is not exactly the same as theirs, nor are we using the same technical indicators. However, the goal in both cases is to clearly encode what the technical indicators and price action are saying about the current and developing trend the same way a trader looking at these indicators on a chart would do. Patel and Shah assert that the “trend deterministic” features did produce better results than the continuously-valued data in their models. We did not see any instances where a trend deterministic variable became statistically significant if the continuously-valued version of the variable was not, however, we did not directly test a model based solely on trend deterministic variables versus one based solely on continuous variables.

Experiment Design and Results (Logistic Regression)

Importing and Preprocessing (First Iteration)

After importing the required libraries, the csv file containing the raw data and the engineered features is imported into R. The engineered features and the date column are kept, but columns containing the

raw data (used for building the features) can now be dropped as can any rows containing null or na data. Columns are then renamed for clarity and the dependent variable column is encoded as a factor.

A scan of the dependent variable column indicates that the data set contains 54.4% “up” samples and 45.6% “down” samples. This makes 54.4% our baseline for measuring model accuracy as an investor could simply bet “up” every day and be correct 54.4% of the time.

Splitting Data into Train and Test

Various ways to split the data into train-test sets have been mentioned in the literature. We have explored three different methods:

1. Chronological Split - Data is split into two different timelines. Longer earlier period is taken as training dataset (2013-2019). Shorter recent period is taken as test dataset (2020-2022).
    a. Advantage: We will be testing and determining the accuracy on more recent data. Higher model accuracy in more recent data means better prediction for tomorrow.
    b. Disadvantage: Model might be biased by extended bull or bear runs, as well as other factors (micro or macro level). In the split above, training data period represents a healthier stable economic period, and most probably a monotonic bull run from 2013-2019. Whereas the testing data period encompasses the more turbulent Covid years. As such, this split is already biased.
    c. Stratified Split – Earlier, we saw that the overall data has 54% uptrend and 46% downtrend. So, it might be more scientific to preserve that proportion in both the training and test dataset. Advantage of such stratification is that it helps to reduce the selection bias in the train-test split.
2. Multi-Stratified Split – Rather than splitting only on one parameter, we can split based on multiple parameters. One more parameter we are thinking about is splitting by year, so that both train and test have same proportion of data from all years on top of preserving uptrend-downtrend proportion. It offers similar advantages as the singular stratified split, but with a higher level of granularity in the split proportions.

Modeling without Cross-Validation

Using the stratified split for training/testing sets, we run a logistic regression model, regressing the dependent “up_down” variable on all other variables except “time”. Testing the model’s predictions using the test data set results in a prediction accuracy of 54.7% which is essentially the same as naively guessing “up” every day. Examining the confusion matrix based on the model’s output suggests that that is basically what the model is doing. Of the 614 data points in the test set, the model is selecting “up” in 556 of those cases (or 90.6% of the time). 306 of the “up” predictions are correct but 250 of them are incorrect. Only 30 of the 280 “down” predictions are correct. This would appear to be the main weakness of the model, at this stage. Varying the cut-off from 0.5 to a higher number would seem like an easy way to counter the preponderance of “up” predictions. However, trying this, we discover that while a cut-off of 0.54 to 0.55 does equalize the number of up and down predictions, it does nothing to improve the accuracy of the overall model. Cut-offs above 0.6 produce nearly all “down” predictions, and cut-offs less than 0.45 produce almost entirely “up” predictions.

Modeling with Cross-Validation (CV)

One problem with singular model fit is that the model might be over-fitting. Another problem is selection bias (which we have addressed to a certain extent by carrying out stratified train-test split). To address these issues, we will model the data with cross-validation (CV). This methodology works by building the best model determined by using multiple train-test splits.

Our initial trial of the CV model produced results very similar to the model using a single split. Time-permitting, we'll re-examine to confirm we're getting the intended results as a CV model would typically show a lower accuracy score due to less over-fitting

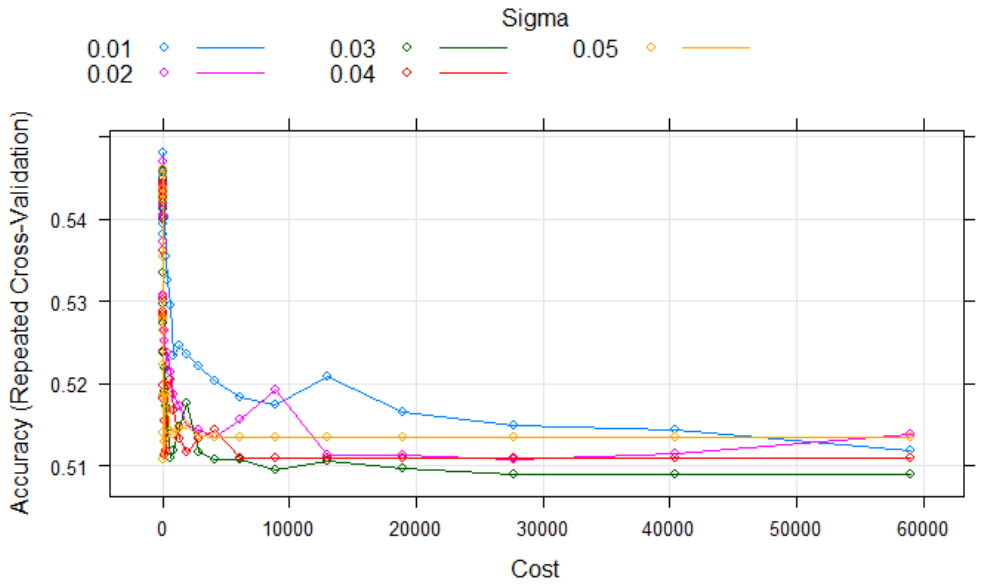
Next Iterations of Logistic Regression Model

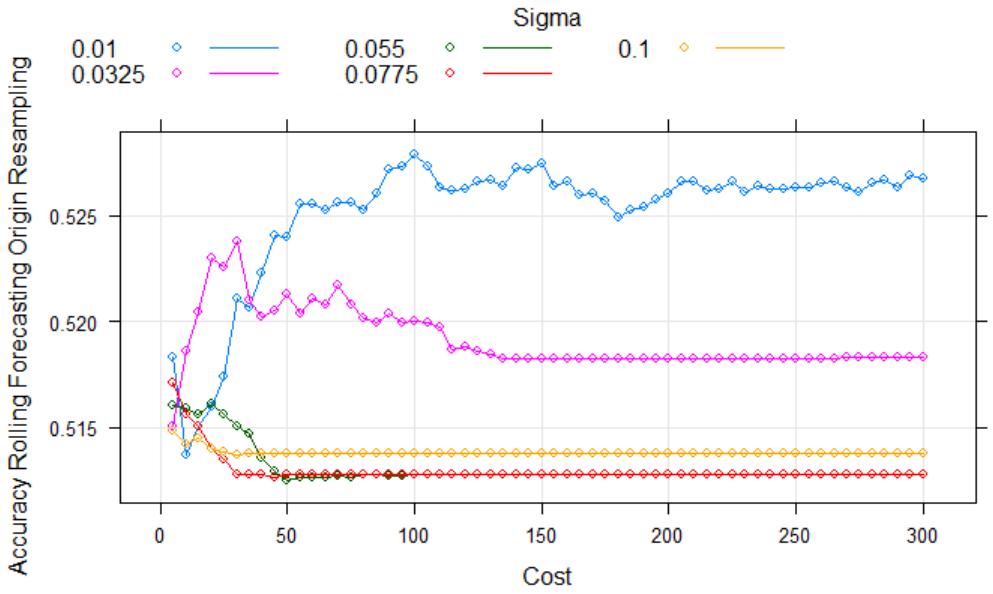
While the first trial of the LR model, using only variables taken directly from the SPX index, produced less than stellar accuracy results, the second trial, which incorporated macro-economic variables as features, performed noticeably better. Several of the features had low p-values, indicating statistical significance, and the overall model had statistical significance albeit relatively low predictive power. Running the same model (using all features) in Radiant produced basically the same results as coding in R, however, the Radiant model summary includes a pseudo-R-squared measure which was 0.032, suggesting our model was explaining about 3% of outcomes in the data. The final model removed the statistically insignificant variables to simplify the model and avoid overfitting. This caused the pseudo-R-squared to drop to 0.017 when using Radiant. This is the same (final) model for which we are reporting accuracy of 56%.

SVM Modeling Approach

We also built an SVM model to predict the direction of the dependent variable. Support Vector Machine (SVM) model is a supervised learning classification algorithm. The model was built with the same independent variables that have already been discussed above. Since Linear SVM did not provide good model accuracy, other variants of SVM were also tested, and SVM with Radial basis kernel function provided the best result. So, SVM with Radial Basis was adopted for further analysis. The SVM model was cross validated on 75% of the data samples and tested on the remaining 25% of the samples. Another method of cross validation was also used where the training data was sampled into slices of time period of 90 days and the model was validated on one time slice each time while training on remaining data/slices. The two hyperparameters – cost (C) and sigma – were tuned during cross validation by preparing a grid of these parameters. The results of the SVM modeling are explained below.

With cross-validation, the highest accuracy we were able to acquire was 55%. Hyperparameter tuning showed some interesting results, but the tuning did not help improve on model accuracy by much. The effects of various values of the hyperparameters on the accuracy have been shown in Figure 1 below.





Below we discuss the observations on effects of changing the two hyperparameters for SVM model:

1. There was not much difference between 10-fold cross validation and cross validation with time slices as both these methods returned similar accuracy of prediction, but with slightly different (insignificant) element values in the confusion matrix.
2. The main issue was that the model prediction was mostly skewed towards positive signal and there were very few predictions of the negative signal.
3. Highest accuracy was obtained at sigma value of 0.01, and all other trained value of sigma had lower accuracy.

4. When C close to 0: Accuracy of 55% is obtained, and this is the highest accuracy attained. But the drawback of this model was that every signal was predicted as positive. Setting the cost parameter close to 0 means no penalty for wrong prediction. And this encouraged the model to skew completely towards positive signals to maximize the accuracy.
5. When C is high: As the value of C increases from 0, at first the accuracy decreases drastically and then it slightly increases for the value of C between 300-12k. But after that the accuracy decreases again and becomes asymptotic.
6. Thus, $C=0$ is one extreme which despite gaining highest accuracy is the same as assuming that every day the signal is going to be positive. And that is a very naive approach.
7. On the other hand, the models for $C \gt 0$ suffer from lower accuracy. These models cannot provide better prediction than assuming every signal is positive. Various reasons for this could be:
    a. The covariates chosen are not able to explain the direction of the signal with higher accuracy.
    b. The SPX index is too efficient (I.e. movements appear random) for the model to discern signal from noise.
8. Despite that, our model has one notable strength:
    a. The approach is sound and could be applied to other indices, stocks, bonds or currencies with little change in the model.

Conclusion and Final Observations

Our goal was to find a probabilistic trading edge in daily price moves of the SPX (S&P 500) index. We knew this would be an extremely challenging endeavor as the S&P 500 is the largest, most widely traded, and most efficient stock market index in the world. It is also a market with no shortage of multi-billion-dollar hedge funds employing algorithmic trading methodologies. In building our predictive model, we included several technical indicators and macro-economic variables as features, not knowing which ones would prove statistically significant and which ones would not. The variables that did prove to be most statistically significant were not the first ones we tried nor the ones we intuitively felt would be most meaningful. We were surprised that some of the more powerful predictors included longer-term variables such as Real GDP, 30Yr Bond, and distance from 200-day moving average. Perhaps this indicates that market direction based on longer-term macroeconomic conditions is more predictable than “noisy” short-term indicators.

After removing variables with low predictive power from our model, we were left with a model that had statistical significance but relatively low accuracy above the baseline rate of 54.4%. This result is not completely unexpected given the challenges cited above. Performance of the model was better on positive (up) predictions than on negative (down) predictions, therefore the model’s value is greater if trades are placed only when the model generates an “up” signal and ignored whenever the model generates a “down” signal. This would suggest that Precision is at least as important a metric as Accuracy for evaluating this type of model.

So, the big question is, “can this model be used to generate a trading profit?” Theoretically, even a small edge in probabilities can accrue into a large profit as the law of large numbers plays out; this is the

business model of casinos. However, trading real-world stock markets is not a theoretical exercise and there are additional costs in the form of commissions, bid-ask spreads, and slippage that can take a strategy from being profitable on paper to loss-making in practice. The greater the edge in probabilities, the greater the ability of the strategy to overcome "friction" generated by the transactions. The only way to know for sure would be to backtest the strategy with the various trading costs, something that was beyond the scope of this experiment.

There are other paths that could be taken on the road to finding a more robust strategy. One approach might include applying the same modelling approach to individual stocks or ETFs. Because these other products are not traded as widely as the SPX index, there could be a greater opportunity to exploit inefficiencies using machine learning models. One of the research papers we read prior to conducting our experiment was based on two stocks and an index in the Indian stock market and their results showed better accuracy than our model based on the SPX index. To work on other products, the strategy would need to be adapted because the availability of options will not be as flexible as the SPX index, however many products have weekly options, expiring every Friday. Therefore, trades could be placed once a week on Thursday.

Another approach would be to base a model on a time frame longer than daily (perhaps weekly or monthly). Intuitively, a longer time frame should have fewer, seemingly random price fluctuations, and be more aligned to macro-economic fundamentals. This might allow a model to better use the "signal" from data rather than the "noise" which is impossible for the best model to predict. The downside would be fewer trading opportunities and slower growth as a result of less frequent compounding.

There are also many technical and macroeconomic indicators that we did not try as well as potential interactions between them that could potentially increase model accuracy. Lastly, there are models such as artificial neural networks that might provide better accuracy than the models we tested.

Appendix

SPX index price action and technical indicators displayed on a chart

	time	open	high	low	close	MA5	MA200	RSI	ADX	volatility percentile	Change %	UP/DOWN	% from MA5	% from MA200	MA5 Slope	MA200 Slope	Closing Strength	RSI Signal	ADX Signal	Volatility
2	2013-01-07T14:30:00Z	1466.47	1466.47	1456.62	1461.89	1455.268	1391.516	62.32617148	16.08610607	33.64383562										
3	2013-01-08T14:30:00Z	1461.89	1461.89	1451.64	1457.15	1461.46	1391.7873	60.11853771	16.38738674	34.02739726	0.00	1	0.00	0.05	1	1	0.54	0	0	0.340273973
4	2013-01-09T14:30:00Z	1457.15	1464.73	1457.15	1461.02	1461.18	1392.1285	61.32308999	16.82874571	30.84931507	0.00	1	0.00	0.05	-1	1	0.51	0	0	0.308493151
5	2013-01-10T14:30:00Z	1461.02	1472.3	1461.02	1472.12	1463.73	1392.50355	64.62348782	17.65532432	30.46575342	0.01	0	0.01	0.06	1	1	0.98	0	0	0.304657534
6	2013-01-11T14:30:00Z	1472.12	1472.75	1467.58	1472.05	1464.846	1392.78125	64.5860584	18.44741212	29.80821918	0.00	0	0.00	0.06	1	1	0.86	0	0	0.298082192
7	2013-01-14T14:30:00Z	1472.05	1472.05	1465.69	1470.68	1466.604	1393.07205	63.80712728	18.9874942	28.82191781	0.00	1	0.00	0.06	1	1	0.78	0	0	0.288219178
8	2013-01-15T14:30:00Z	1470.67	1473.31	1463.76	1472.34	1469.642	1393.40605	64.3678835	19.28345472	28.49315068	0.00	1	0.00	0.06	1	1	0.90	0	0	0.284931507
9	2013-01-16T14:30:00Z	1472.33	1473.96	1467.6	1472.63	1471.964	1393.7528	64.47144615	19.60337924	28.21917808	0.00	1	0.00	0.06	1	1	0.79	0	0	0.282191781
10	2013-01-17T14:30:00Z	1472.63	1485.16	1472.63	1480.94	1473.728	1394.11515	67.39575603	20.62074022	28.05479452	0.01	1	0.00	0.06	1	1	0.66	0	0	0.280547945
11	2013-01-18T14:30:00Z	1480.95	1485.98	1475.81	1485.98	1476.514	1394.44985	69.05913973	21.61420394	28.10958904	0.00	1	0.01	0.07	1	1	1.00	0	0	0.28109589
12	2013-01-22T14:30:00Z	1485.98	1492.56	1481.16	1492.56	1480.89	1394.84575	71.1299748	22.91966349	28.2739726	0.00	1	0.01	0.07	1	1	1.00	0	0	0.282739726
13	2013-01-23T14:30:00Z	1492.56	1496.13	1489.9	1494.81	1485.384	1395.325	71.82440333	24.32792929	28.43835616	0.00	1	0.01	0.07	1	1	0.79	0	0	0.284383562
14	2013-01-24T14:30:00Z	1494.81	1502.27	1489.46	1494.82	1489.822	1395.8087	71.82764677	25.95512702	28.65753425	0.00	1	0.00	0.07	1	1	0.42	0	1	0.286575342
15	2013-01-25T14:30:00Z	1494.82	1503.26	1494.82	1502.96	1494.226	1396.4125	74.40997995	27.5165685	28.65753425	0.01	0	0.01	0.08	1	1	0.96	0	1	0.286575342
16	2013-01-28T14:30:00Z	1502.96	1503.23	1496.33	1500.18	1497.066	1397.12045	71.98322716	28.96647844	28.93150685	0.00	1	0.00	0.07	1	1	0.56	0	1	0.289315068
17	2013-01-29T14:30:00Z	1500.18	1509.35	1498.09	1507.84	1500.122	1397.8161	74.45531324	30.63863933	28.76712329	0.01	0	0.01	0.08	1	1	0.87	0	1	0.287671233
18	2013-01-30T14:30:00Z	1507.84	1509.94	1500.11	1501.96	1501.552	1398.38805	69.39359347	32.22195232	28.60273973	0.00	0	0.00	0.07	1	1	0.19	-1	1	0.286027397
19	2013-01-31T14:30:00Z	1501.96	1504.19	1496.76	1498.11	1502.21	1399.0273	66.2192442	33.12324373	28.10958904	0.00	1	0.00	0.07	1	1	0.18	0	1	0.28109589
20	2013-02-01T14:30:00Z	1498.11	1514.41	1498.11	1513.17	1504.252	1399.7453	71.67709915	34.5362382	27.7260274	0.01	0	0.01	0.08	1	1	0.92	0	1	0.277260274
21	2013-02-04T14:30:00Z	1513.17	1513.17	1495.02	1495.71	1503.358	1400.26995	59.64524855	35.35196917	27.34246575	-0.01	1	-0.01	0.07	-1	1	0.04	-1	1	0.273424658
22	2013-02-05T14:30:00Z	1495.71	1514.96	1495.71	1511.29	1504.048	1400.9007	65.25064857	36.21585647	27.61643836	0.01	1	0.00	0.08	1	1	0.81	0	1	0.276164384
23	2013-02-06T14:30:00Z	1511.29	1512.53	1504.71	1512.12	1506.08	1401.5767	65.52537899	37.01803754	27.45205479	0.00	0	0.00	0.08	1	1	0.95	0	1	0.274520548
24	2013-02-07T14:30:00Z	1512.12	1512.9	1498.49	1509.39	1508.336	1402.231	63.74035615	36.70980045	27.45205479	0.00	1	0.00	0.08	1	1	0.76	0	1	0.274520548
25	2013-02-08T14:30:00Z	1509.39	1518.31	1509.39	1517.93	1509.288	1402.98595	66.78828899	36.83525057	27.34246575	0.01	0	0.01	0.08	1	1	0.96	0	1	0.273424658
26	2013-02-11T14:30:00Z	1517.93	1518.31	1513.61	1517.01	1513.548	1403.71115	66.14325579	36.95173996	26.90410959	0.00	1	0.00	0.08	1	1	0.72	0	1	0.269041096
27	2013-02-12T14:30:00Z	1517.01	1522.29	1515.61	1519.43	1515.176	1404.35485	67.04486243	37.35919977	25.20547945	0.00	1	0.00	0.08	1	1	0.57	0	1	0.252054795
28	2013-02-13T14:30:00Z	1519.43	1524.69	1515.93	1520.33	1516.818	1404.9566	67.39263826	37.9110103	21.64383562	0.00	1	0.00	0.08	1	1	0.50	0	1	0.216438356
29	2013-02-14T14:30:00Z	1520.33	1523.14	1514.02	1521.38	1519.216	1405.5467	67.81931893	38.06045719	16.32876712	0.00	0	0.00	0.08	1	1	0.81	0	1	0.163287671
30	2013-02-15T14:30:00Z	1521.38	1524.24	1514.14	1519.79	1519.588	1406.1561	66.40234061	38.29033864	11.17808219	0.00	1	0.00	0.08	1	1	0.56	0	1	0.111780822
31	2013-02-19T14:30:00Z	1519.79	1530.94	1519.79	1530.94	1522.374	1406.7817	70.98114856	39.01713436	6.136986301	0.01	0	0.01	0.09	1	1	1.00	0	1	0.061369863

SPX index prices, indicators, and engineered variables in tabular format

Research papers / sources:

- Huang, W., Nakamori, Y., & Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. Computers & Operations Research, 32(10), 2513–2522. https://doi.org/10.1016/j.cor.2004.03.016

- Vijh, M., Chandola, D., Tikkiwal, V. A., & Kumar, A. (2020). Stock Closing Price Prediction using Machine Learning Techniques. Procedia Computer Science, 167, 599–606. https://doi.org/10.1016/j.procs.2020.03.326

- Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques. Expert Systems with Applications, 42(1), 259–268. https://doi.org/10.1016/j.eswa.2014.07.040

Logistic Regression Test Models v1 & v2

Team 53

2022-10-25

Importing necessary libraries

```
library(tidyverse)
library(lubridate)
library(xts)
library(caret)
```

Importing the stock price data of the security

```
imported_data = read_csv('Features_spx.csv', show_col_types = FALSE,
                na = c('', 'NA')) %>%
                glimpse()
## Rows: 2,457
## Columns: 20
## $ time                    <dttm> 2013-01-07 14:30:00, 2013-01-08 14:30:00, 201…
## $ open                    <dbl> 1466.47, 1461.89, 1457.15, 1461.02, 1472.12, 1…
## $ high                    <dbl> 1466.47, 1461.89, 1464.73, 1472.30, 1472.75, 1…
## $ low                     <dbl> 1456.62, 1451.64, 1457.15, 1461.02, 1467.58, 1…
## $ close                   <dbl> 1461.89, 1457.15, 1461.02, 1472.12, 1472.05, 1…
## $ MA5                     <dbl> 1455.268, 1461.460, 1461.180, 1463.730, 1464.8…
## $ MA200                   <dbl> 1391.516, 1391.787, 1392.129, 1392.504, 1392.7…
## $ RSI                     <dbl> 62.32617, 60.11854, 61.32309, 64.62349, 64.586…
## $ ADX                     <dbl> 16.08611, 16.38739, 16.82875, 17.65532, 18.447…
## $ `volatility percentile` <dbl> 33.64384, 34.02740, 30.84932, 30.46575, 29.808…
## $ `Change %`              <dbl> NA, -3.247646e-03, 2.652349e-03, 7.568717e-03,…
## $ `UP/DOWN`               <dbl> NA, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, …
## $ `% from MA5`            <dbl> NA, -0.002949106, -0.000109501, 0.005731931, 0…
## $ `% from MA200`          <dbl> NA, 0.04696314, 0.04948645, 0.05717504, 0.0569…
## $ `MA5 Slope`             <dbl> NA, 1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
## $ `MA200 Slope`           <dbl> NA, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
## $ `Closing Strength`      <dbl> NA, 0.53756098, 0.51055409, 0.98404255, 0.8646…
## $ `RSI Signal`            <dbl> NA, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ `ADX Signal`            <dbl> NA, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, …
## $ Volatility              <dbl> NA, 0.3402740, 0.3084932, 0.3046575, 0.2980822…
```

Preprocessing data

Preprocessing tasks performed:

1. Dropping various columns
2. Dropping out rows with NA
3. Renaming columns
4. Changing data types of columns

```
data = imported_data %>%
      select( -c(2:10)) %>%
      na.omit() %>%
      rename('up_down' = 'UP/DOWN') %>%
      mutate(up_down = as.factor(up_down)) %>%
      glimpse()
## Rows: 2,456
## Columns: 11
## $ time               <dttm> 2013-01-08 14:30:00, 2013-01-09 14:30:00, 2013-01-…
## $ `Change %`         <dbl> -3.247646e-03, 2.652349e-03, 7.568717e-03, -4.75516…
## $ up_down            <fct> 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, …
## $ `% from MA5`       <dbl> -0.002949106, -0.000109501, 0.005731931, 0.00491792…
## $ `% from MA200`     <dbl> 0.04696314, 0.04948645, 0.05717504, 0.05691400, 0.0…
## $ `MA5 Slope`        <dbl> 1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
## $ `MA200 Slope`      <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
## $ `Closing Strength` <dbl> 0.53756098, 0.51055409, 0.98404255, 0.86460348, 0.7…
## $ `RSI Signal`       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0,…
## $ `ADX Signal`       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, …
## $ Volatility         <dbl> 0.3402740, 0.3084932, 0.3046575, 0.2980822, 0.28821…
prop.table(table(data$up_down))*100
##
##        0        1
## 45.60261 54.39739
```

This tells us that the next day stock price has uptrended 54% of times and downtrended 46% of times.

Splitting the data into train-test

Various ways to split the data to train-test set have been mentioned in the literature. Below we have explored various ways to execute the split.

Chronological Split:

Data is split into two different timelines. Longer earlier period is taken as training dataset (2013-2019). Shorter recent period is taken as test dataset (2020-2022).

```
train = data %>%
        filter(time >= as.Date('2013-01-01') & time <= as.Date('2019-12-31'))
test = data %>%
  filter(time >= as.Date('2020-01-01') & time <= as.Date('2022-12-31'))
```

Advantage:

1. We will be testing and determining the accuracy on more recent data. Higher model accuracy in more recent data means better prediction for tomorrow.

Disadvantage:

1. Model might be biased by extended bull or bear runs, as well as other factors (micro or macro level). In the split above, training data period represents a very healthy, and most probably a monotonic bull run from 2013-2019. Whereas the testing data period encompasses the more turbulent Covid years. As such, this split is already biased.

Stratified split:

Earlier, we saw that the overall data has 54% uptrend and 46% downtrend. So, it might be more scientific to preserve that proportion in both the training and test dataset. Stratified split from `caret` package allows such split.

```
trainIndex = caret::createDataPartition(y = as.factor(data$up_down),
                                        p = 0.75,
                                        list = F)
train = data %>%
          slice(trainIndex)
test = data %>%
          slice(-trainIndex)


prop.table(table(train$up_down))
##
##         0         1
## 0.4560261 0.5439739
```

Multi-stratified split:

Rather than splitting only on one parameter, we can split based on multiple parameters and this process is called multi-stratified split. One more parameter we are thinking about is splitting by year as well so that

both train and test have same proportion of data from all years on top of preserving uptrend-downtrend proportion.

Logistic Regression

Single modeling without cross-validation

First, let's run a singular Logistic Regression model and determine its accuracy and other metrics.

```
model1 = glm(up_down ~ . - time , data = train, family=binomial)
summary(model1)
##
## Call:
## glm(formula = up_down ~ . - time, family = binomial, data = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.481  -1.251   1.045   1.099   1.659
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.002615   0.179131  -0.015   0.9884
## `Change %`         -14.287630   7.524927  -1.899   0.0576 .
## `% from MA5`         2.963590   7.724853   0.384   0.7012
## `% from MA200`       0.215733   1.342804   0.161   0.8724
## `MA5 Slope`         -0.024804   0.060505  -0.410   0.6818
## `MA200 Slope`        0.105801   0.094809   1.116   0.2644
## `Closing Strength`   0.108542   0.188879   0.575   0.5655
## `RSI Signal`         0.116421   0.285243   0.408   0.6832
## `ADX Signal`         0.045343   0.102091   0.444   0.6569
## Volatility           0.059403   0.187093   0.318   0.7509
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2539.3  on 1841  degrees of freedom
## Residual deviance: 2531.0  on 1832  degrees of freedom
## AIC: 2551
##
```

```
## Number of Fisher Scoring iterations: 4
```

Calculating accuracy and other metrics

```
threshold = 0.5
predicted_probability = model1 %>% predict(test, type = 'response')
predicted_val = ifelse(predicted_probability > threshold, 1, 0) %>%
                as.factor()
actual_val = test$up_down

# Model Accuracy
mean(predicted_val == actual_val)
## [1] 0.5472313
# Confusion matrix
conf_mat = caret::confusionMatrix(predicted_val, actual_val)
conf_mat
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  30  28
##          1 250 306
##
##                Accuracy : 0.5472
##                  95% CI : (0.5069, 0.5871)
##     No Information Rate : 0.544
##     P-Value [Acc > NIR] : 0.4521
##
##                   Kappa : 0.0249
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.10714
##             Specificity : 0.91617
##          Pos Pred Value : 0.51724
##          Neg Pred Value : 0.55036
##              Prevalence : 0.45603
##          Detection Rate : 0.04886
```

```
##    Detection Prevalence : 0.09446
##       Balanced Accuracy : 0.51166
##
##        'Positive' Class : 0
##
```

Modeling with Cross-Validation

One problem with singular model fit is that the model might be over-fitting. Another problem is selection bias (which we have addressed to a certain extent by carrying out stratified train-test split).

To address these issues, we will model the data with cross-validation.

```
train_control = caret::trainControl(method = 'repeatedcv',
                                    number = 10,
                                    repeats = 10)


model_cv = caret::train(as.factor(up_down) ~ . -time,
                     data = train,
                     trControl = train_control,
                     method = 'glm',
                     family = binomial())
summary(model_cv)
##
## Call:
## NULL
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.481  -1.251   1.045   1.099   1.659
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -0.002615   0.179131  -0.015   0.9884
## `\\`Change %\\``         -14.287630   7.524927  -1.899   0.0576 .
## `\\`% from MA5\\``         2.963590   7.724853   0.384   0.7012
## `\\`% from MA200\\``       0.215733   1.342804   0.161   0.8724
## `\\`MA5 Slope\\``         -0.024804   0.060505  -0.410   0.6818
## `\\`MA200 Slope\\``        0.105801   0.094809   1.116   0.2644
## `\\`Closing Strength\\``   0.108542   0.188879   0.575   0.5655
```

```
## `\\`RSI Signal\\``          0.116421   0.285243   0.408   0.6832
## `\\`ADX Signal\\``          0.045343   0.102091   0.444   0.6569
## Volatility                  0.059403   0.187093   0.318   0.7509
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2539.3  on 1841  degrees of freedom
## Residual deviance: 2531.0  on 1832  degrees of freedom
## AIC: 2551
##
## Number of Fisher Scoring iterations: 4
model_cv
## Generalized Linear Model
##
## 1842 samples
##   10 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 1658, 1657, 1658, 1658, 1658, 1658, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.5330605  -0.007266147
```

Calculating accuracy and other model metrics

```
threshold = 0.5
predicted_probability = model_cv %>% predict(test, type = 'prob')
predicted_val = ifelse(predicted_probability[, 2] > threshold, 1, 0) %>%
                as.factor()
actual_val = test$up_down

# Model Accuracy
mean(predicted_val == actual_val)
```

```
## [1] 0.5472313
# Confusion matrix
conf_mat = caret::confusionMatrix(predicted_val, actual_val)
conf_mat
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  30  28
##          1 250 306
##
##                Accuracy : 0.5472
##                  95% CI : (0.5069, 0.5871)
##     No Information Rate : 0.544
##     P-Value [Acc > NIR] : 0.4521
##
##                   Kappa : 0.0249
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.10714
##             Specificity : 0.91617
##          Pos Pred Value : 0.51724
##          Neg Pred Value : 0.55036
##              Prevalence : 0.45603
##          Detection Rate : 0.04886
##    Detection Prevalence : 0.09446
##       Balanced Accuracy : 0.51166
##
##        'Positive' Class : 0
##
```