Market Vector Auto Regression app for iPhone

Tutorial and example materials, updated June 2022

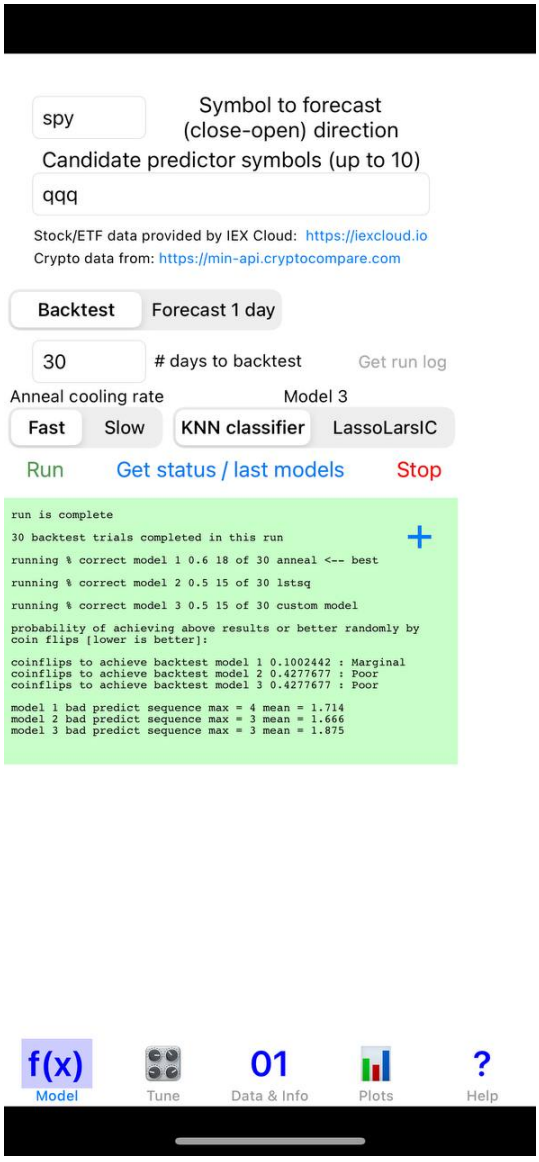


Figure A: Launch screen after a SPY, QQQ run, 30 day backtest, with arrow showing the best of 3 models in the backtest

Version 2.6

[Note: The `coinflips to achieve backtest` values in the screen shots and discussions in this document are slightly incorrect versus theory, mostly noticeable for

small backtest counts. A small fix to the code was added to Version 2.6 for correcting this.]

This app models the (close-open) price direction (today's price travel) of a given target asset... as a function of prior days' opening and closing prices of the asset in question and related assets which you specify. The app does not attempt to figure out what those related assets are, you have to specify them. However, the app does have a feature to discard unneeded variables from the more advanced models. Nonetheless, it is best not to include random asset symbols into the candidate predictor asset list, as the job of forecasting is difficult enough without including irrelevant data in a model on purpose. Included in "prior data" is a recent volatility estimate. Since historical volatility is essentially a moving average (though not of price), the models contain some concept of longer time history behavior and "stickiness" of information.

Terminology

This app also works for ETFs if they are available in the IEX Cloud historical data. As we see in the above Figure A example, we are studying SPY versus QQQ in the example run, which are both ETFs. It also works for a selected subset of crypto-currencies as provided by the cryptocompare.com data provider.

Since cryptocompare.com reports "closing" price for the always-open 24/7 crypto market at midnight GMT, one should not mix crypto and stock symbols in the same model, since stock prices report closing at 4PM US Eastern time. This mixing is possible theoretically if one takes care, but this app does not handle such care at the present writing. To avoid confusion, it is best not to mix crypto and stocks in the same model in this app, in the current release.

Quick start

1. Enter the symbol to forecast and up to ten candidate predictor symbols in the fields on the main Model tab of the app. In this example, we are trying to forecast SPY as a function of prior values of itself and QQQ. You don't need to re-enter SPY as a predictor variable, it is done automatically internally. It may be good to start with a small number of predictor symbols and see if the model gets better or worse in backtest as you add each new symbol. This is kind of a manual version of "forward stepwise regression" if you add one variable at a time and check backtest results at each before proceeding.

2. Enter the number of days to backtest. Or leave it at the default if you want. In this example, we set backtest days to 30 (these are trading days so 30 is more than a month).

3. Do a **Backtest** first (keep **Anneal cooling rate** set to **Fast**).
    a. Leave the backtest switch set to Backtest.

4. Press **Run.** Model backtest will run on the server. In a few seconds, you should see action in the green report-out screen. It will show intermediate results. You can look near the top of the output area to see "30 backtest trials completed" in the above example...this number will be reported out periodically as the backtest proceeds so you can see the progress. E.g. you might see 5, 7, 12, etc completed until it reaches the max you have specified.

5. Results are reported out in the green text field. Press the + button to expand the output to full screen. Press **…** (which will show up where the + was) to go back to the small view and see all the other buttons. You can also use the **export** button which appears on the expanded + view to save the contents of this field to a file or email etc.

   If you make a mistake and press Run before you are ready, press Stop to stop the run, and re-try.

6. After you have found a series of symbols and settings that yield good backtests, you can set the button to **Forecast 1 day ahead** and make a forecast. You should try this over several days and check to see how often your forecasts come true before acting on any such forecasts. Since this is a statistical model, one does not expect the forecast to always be true. The backtest may give an idea of how accurate the model might be in the future, but of course, the future may not be like the past and so future results may not be as accurate as the backtest results.

   An example forecast is shown in **Figure C** below. Since SPY and QQQ are always moving every trading day, if you run the same test after this document was written, you will see different numbers in these results.

   Once you have model settings yielding a decent long backtest, it may be viable for several days without re-tuning. Or you could recheck the backtest every day to be more sure.

7. As always, consult a professional investment advisor before trading. Forecasting is often “fraught with imminent peril,” as you should well know.

The top arrow in Figure B indicates which was the best model of the 3 models which are automatically run for every backtest and forecast. Here, it was our simulated annealing model, model 1. The bottom arrow indicates an estimate of model quality by estimating how easy it is to achieve this good of a result (60% directionally correct) by random coin flips. In this example, there is a 1 in 10 chance of achieving this 60% directionally

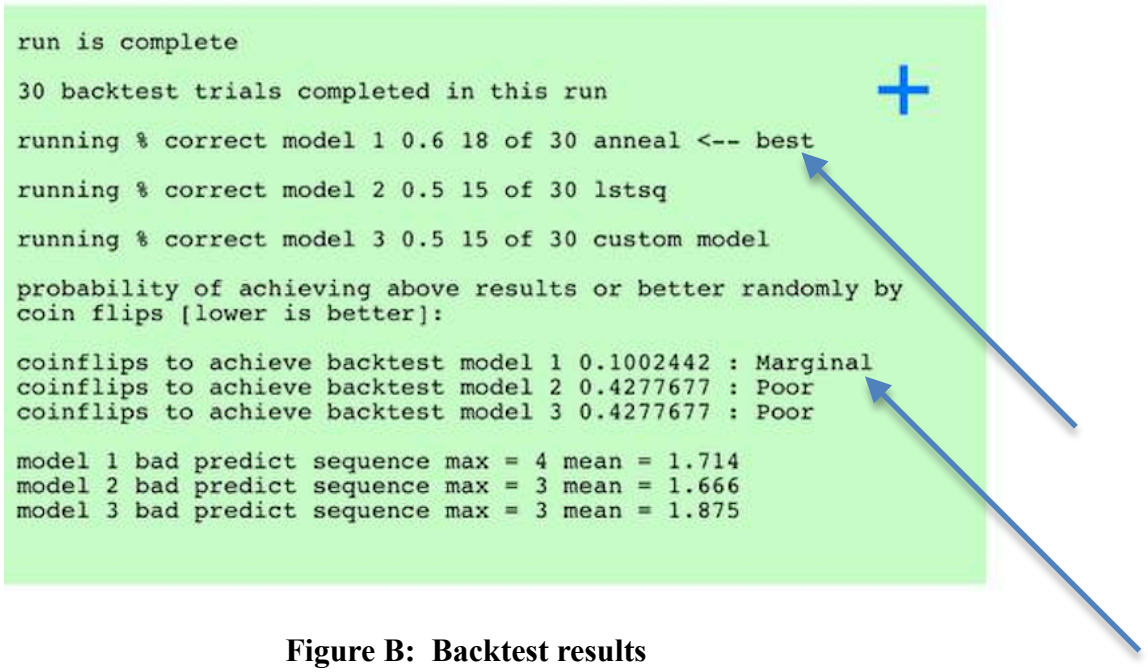


Figure B: Backtest results

correct result via coinflips over the long run (e.g. if we did many, many sequences of 30 flips). See the **Bonus features** section below for more info on this topic. Note that any individual sequence of 30 coin flips will widely vary in its "forecast" accuracy.

We note that this particular model's quality is therefore noted as "Marginal" corresponding to 0.1002 or about a 10% value.

this space intentionally left blank

The word choices of Marginal and Poor are not standard statistical terminology, but are kind of a red/green/yellow type of classification in the app to give "dashboard" / at-a-glance view of the models without having to mull over numbers and think too much ("is higher better? is lower better?").

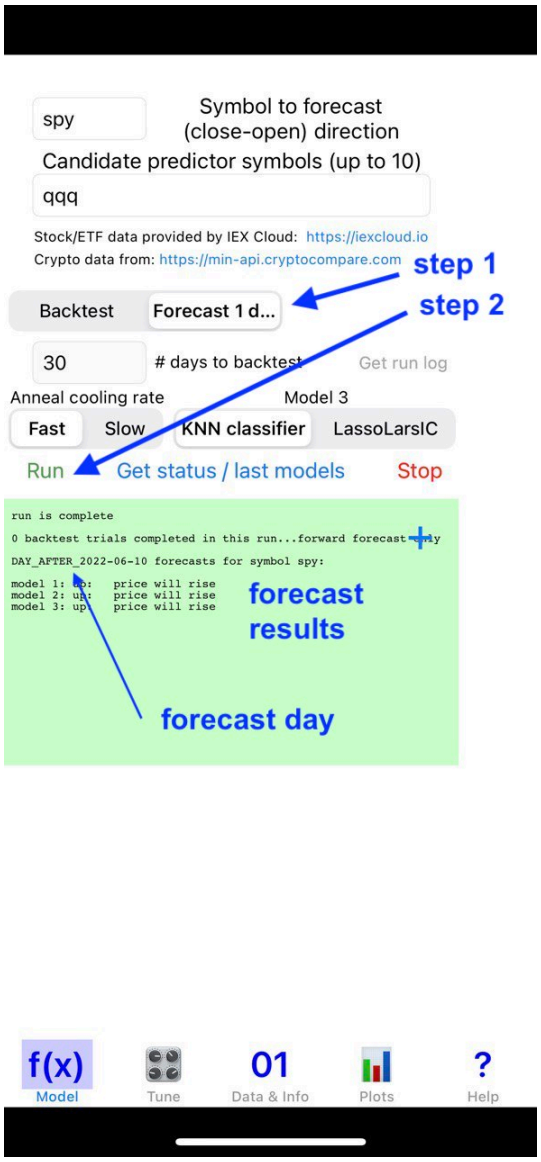


Figure C: First example run SPY = f(SPY_prior, QQQ_prior) for forecasting

Note that the other two models are only showing 50% directionally correct. These two models are therefore being described as Poor models, since they are approximately similar to coin flipping in terms of accuracy for this range of backtest.

If you like, you could use this Marginal-classified model 3 to make a forecast. Switch the [Backtest / Forecast 1 Day] button to "Forecast 1 Day," then press Run again.

Figure C shows the forecast results. All 3 models are forecasted from, even if you only want to pay attention to one of the models, and the 3 forecasts are presented independently. In this case, our best backtest was model 1, the annealing model. This forecasts that the price will rise in the forecasted day (close - open).

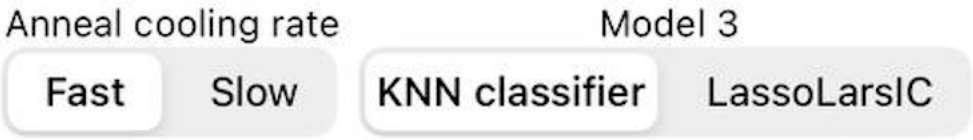


Figure D: Coarse tuning switches

```
run is complete

30 backtest trials completed in this run

running % correct model 1 0.6 18 of 30 anneal <-- best

running % correct model 2 0.5 15 of 30 lstsq

running % correct model 3 0.5 15 of 30 custom model

probability of achieving above results or better randomly by
coin flips [lower is better]:

coinflips to achieve backtest model 1 0.1002442 : Marginal
coinflips to achieve backtest model 2 0.4277677 : Poor
coinflips to achieve backtest model 3 0.4277677 : Poor

model 1 bad predict sequence max = 4 mean = 1.714
model 2 bad predict sequence max = 3 mean = 1.666
model 3 bad predict sequence max = 3 mean = 1.875
```

```
run is complete

30 backtest trials completed in this run

running % correct model 1 0.5666 17 of 30 anneal <-- best

running % correct model 2 0.5 15 of 30 lstsq

running % correct model 3 0.4666 14 of 30 custom model

probability of achieving above results or better randomly by
coin flips [lower is better]:

coinflips to achieve backtest model 1 0.1807973 : Marginal
coinflips to achieve backtest model 2 0.4277677 : Poor
coinflips to achieve backtest model 3 0.5722322 : Poor

model 1 bad predict sequence max = 3 mean = 1.444
model 2 bad predict sequence max = 3 mean = 1.666
model 3 bad predict sequence max = 5 mean = 2.285
```

Figure E: left = prior run output; right = first attempt at coarse tuning output

Note also the day/date indicator. We report the forecast day as the "trading day after" the last known day in our model data set to avoid having to account for holidays and weekends.

Note also that the other two models that did not backtest well also forecasted the same directional change in price (close - open) for the forecasted day. This is somewhat irrelevant, since our backtest showed that those two models did not backtest well and were similar to guessing the directional change. Hence, it may be a bad idea to pay attention to these two models.

However, we note this for future use, in case you get multiple models with reasonable backtest results. In this latter case of good backtests for more than one model, you could apply a kind of "voting" process with the 3 forecasts to see if they agree. In formal modeling, this is called an "ensemble" approach (an ensemble or collection of models where results from several models are blended together). We do not provide the ensemble computations in this simple forecaster app (yet?), but merely report the results out of the 3 models separately.

Model 1 and model 2 are mainly "reference" models which should be a baseline of accuracy. E.g. they are both very simple linear models.

Model tuning introduction

Next, we can attempt to do some manual tuning of the models to achieve better backtests.

The first level of tuning is found by adjusting two switches on the main Model tab of the app as shown above. These are the Anneal cooling rate and the Model 3 type.

These are coarse tuning parameters.

Flipping the Anneal cooling rate to Slow affects only model 1, and flipping the Model 3 switch to LassoLarsIC affects only model 3, so we can do both at once without causing confusion. Let's try this and make a run: Flip the two switches to their opposites, set the Backtest switch back to the Backtest position (instead of Forecast which we just did), and press Run again. With the anneal cooling rate set to "Slow," the models take a bit longer to run.

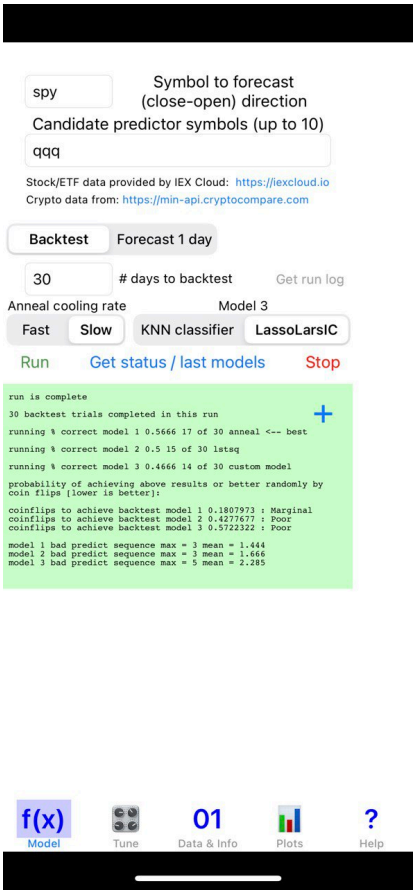


Figure F: Coarse tune example setup and output

In this first attempt at getting a better backtest, we see that things have not improved, but have in fact degraded. See Figure E and Figure F. The percent correct from each model is worse (except for model two, which is the same 0.5 = "fifty-fifty chance" coinflip low quality). Hence, there is no point to attempt a forecast with this model setup, as we don't think it will be better than our original trial based upon the backtests.

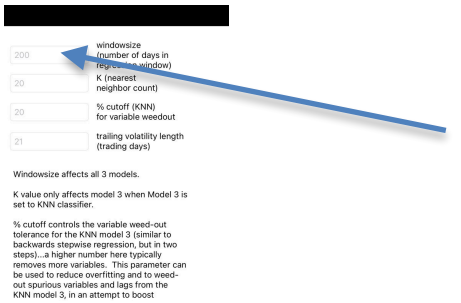
Original trial: model 1 percent correct was 0.6
New model: model 1 percent correct was 0.56

Similarly, model 3's percent correct dropped from 0.5 (already poor) to 0.47.

Note that when you run this study in the future from when this document was written, your numbers may vary, since the data for SPY and QQQ will be different. In your future scenario, doing this coarse model adjustment may result in *better* backtests than the default setup.

Hence, let's put the switches back to the way they were at the beginning (Fast anneal and KNN Classifier), and proceed with some more detailed refinements.

After you flip the main screen switches back to where they were, go to the Tune tab of the app.



Window size is the first editable parameter on this screen; default is 200 days. Try 100, see if your backtests get better. If not, try 150 or 250. You will not likely see much change in results if you change this number by a small amount.

Figure G: Tune tab screen.

Note the top data entry field, **windowsize.** This is a fairly coarse tuning parameter. It indicates how many prior days we want to feed into our model. If we include too many days, we are trying to fit too long of a time period with perhaps many market shocks or "regime change / everything is different" events into one simple model. If we include too few days, we may not have enough data to generalize from. The default we have set up in the app is 200 days. For a trial, let's cut this in half to 100 and then go back to the Model tab and and press Run again (in backtest mode of course).

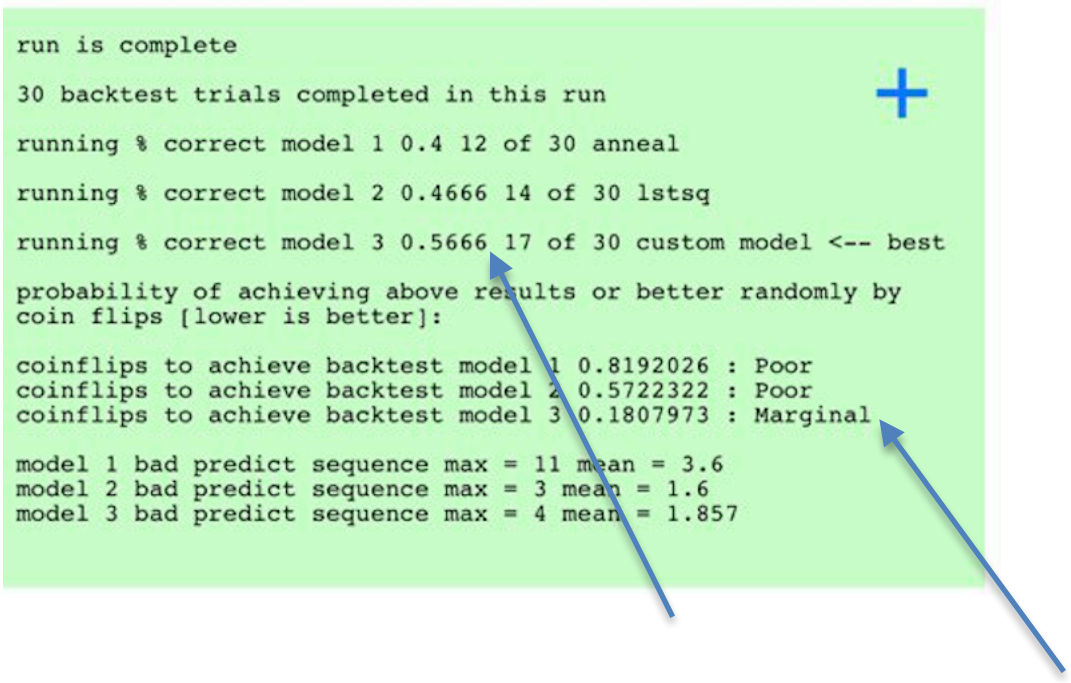


Figure H: 30 day backtest with window size set lower, to 100 (for the time period studied in this document). [Results may vary for your trials, which will be in a future time period.]

Here we see that by reducing the window size (number of points included in the model), the annealing model 1 got worse, and the model 3 (currently the KNN model) got better. The middle model 2 got worse also, but let's ignore this for the moment, as model 2 is a reference linear model that has all variables in it. Model 3 only got up to "Marginal" quality, and in fact is showing less than the 0.6 directional accuracy from our original model 1 annealing trial. Here we see model 3 accuracy is 0.566 directional accuracy (about 57%) during the 30 day backtest, versus our original annealing result of 0.6 (e.g. annealing was better...refer back to Figure B).

The positive results from this test is that model 3 is our most tunable model in this app, as we shall see in the next section. If model 3 got somewhat better by tuning one parameter, we may be able to boost its accuracy further by tuning additional parameters.

Those with some exposure to machine learning might start to recognize this as an example of "hyper-parameter tuning" of a model. Since hyper-parameters are merely ordinary parameters in this app's case (numbers or on/off type of switch settings), we

leave off the hyper- prefix to reduce confusion with new users. It starts to sound a little bit too "Star Wars" with the hyper- prefix, and it is not that at all.

Pro tip

> *Machine learning (hyper-)parameters must be tuned all together for best results, since they often interact with one another. If we tune them one at a time as we do in this example, we may get sub-optimal results. However, we may be able to get better results via this manual tuning than by taking all model parameters at their default settings. E.g., some less than optimal tuning may be better than no tuning at all.*

this space intentionally left blank

We will continue to adjust model 3 by changing the **% cutoff** parameter as shown in Figure I above. Default is 20; we set it to 30. What this does is: discards more internal variables that might be causing more problems than benefits in the model. It throws out variables that seem to have less effect on the results. After changing this number to 30, go back to the Model tab and press run again to backtest the new model settings.

Pro tip

> *This automatic elimination of variables that make the model worse is known as "feature selection" in machine learning parlance.*

Here we see that model 3 improved again. Removing more variables from the model improved the backtest. Which variables, you ask? This is a topic that shall wait for a

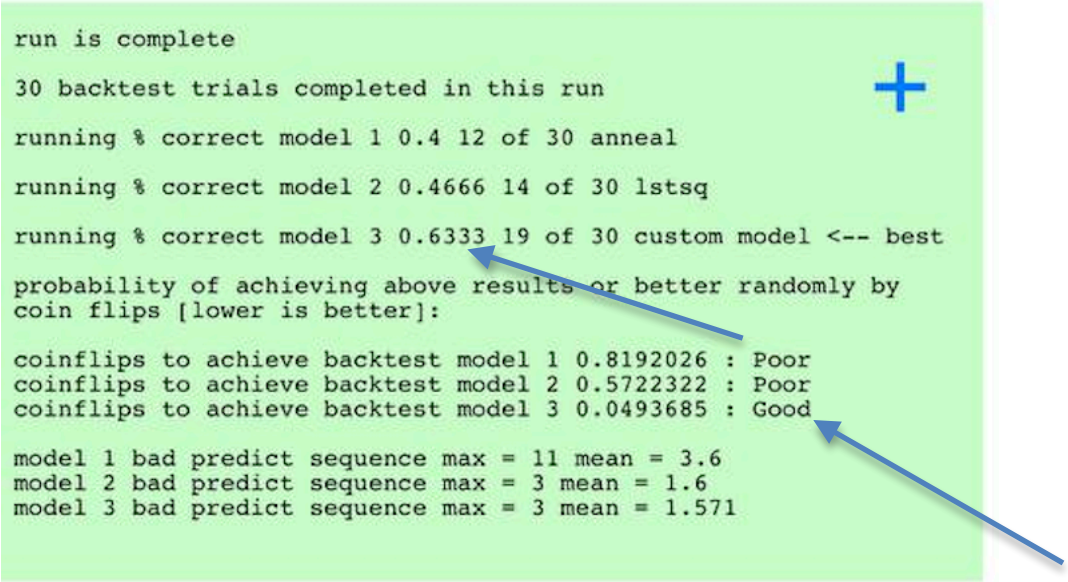


Figure J: 30 day backtest results (windowsize = 100, % cutoff for variable weedout = 30%). Model 3 is the model being tuned. Results improved to 63% directionally correct.

more detailed exposition. The short answer is: variables that seem to contribute less to the results overall are eliminated.

However, this improvement may not be the case for your model runs with different symbols than SPY and QQQ. You may increase **% cutoff** as we did here, and get *worse* backtest results. Conversely, you might see that if you instead *reduced* the **% cutoff** value, your backtest gets better. Such is a fact of life in modeling the ever-changing market with a simple model builder such as this, with manual model tuning.

But let us examine the results further: Now we see that our model 3 backtest is classified as Good, with a value of 0.049 (this is the estimated p-value for the statisticians out there). This suggests that to get 63% directionally accuracy by merely flipping coins 30 times, we would have to be fairly lucky: Such a 63% accurate result will only happen about 5% of the time if we did those batches of 30 coin flips many many times. A model that can be beat by randomness about 5% of the time or less is often considered a reasonable model in statistics. 5% is a 1 in 20 chance, a fairly good bet if you are on the 19 to 20 side of it [see Note 1].

Since we have improved our model backtest quality from our initial trial run with default parameters, let's run the model to FORECAST again. Flip the button over from Backtest to Forecast, and Run again.

Interestingly, the results for our time period are identical to our initial default setup. All 3 models forecasted "price will rise" on the next day. E.g. the results are identical to those shown in Figure C. This is partially due to the fact that we only attempt to forecast directional change (bullish/bearish) rather than specific percent changes of an asset.

Practical considerations and caveats

Note that in our best backtest above, it showed 63% correct directional accuracy, so even though the p-value (coinflip test) was the fairly low 5%, it still suggests that the model may be wrong 37% of the time (100-63). So if we follow this forecast and try to make a trade based upon it, the 30 day backtest shows that we may be wrong 37% of the time. Is this better or worse than following your "intuition" or recommendations from market commentators or the myriad of other sources of market information? These are questions we cannot answer at the moment with the data at hand, but questions you should think about if you use these models.

Also, there is the important consideration that even if a model might backtest well, what happens tomorrow may be an epoch change or bolt-from-the-blue change (one country invades another unexpectedly, terrible earthquake, general market crash for technical reasons, etc). E.g. Tomorrow may not be anything like the recent past at all.

Tuning discussion

As we see here, after a few trials and changing a couple of parameters, we were able to improve the model's backtest quite a bit. There is likely more that could be done to improve this backtest, either by changing the window size and % cutoff value more, in a more fine grained manner (e.g. iterating on those two variables' values to get better backtests), and/or tuning the other main parameter K... the nearest neighbor count. Those interested in tuning KNN models should look up the theory of them:

http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

In general, making K larger makes the model act more like a low-pass filter to filter out noise or un-modeled shocks (making the model flatter or more linear). Making K smaller allows the model to fit *known* data more accurately, with the downside that it may not be able to forecast *unknown* data as well. E.g. it may not be able to *generalize* to the future as well as it fits the past. Hence we often tend to larger values of K in a K-nearest model, but not too large, or else we might ignore some important data. This is why we dont have K set to a default of, say, 2 or 3. You can try this yourself by setting the K value to 2 or 3

and re-running a backtest. You will likely see model 3 showing up as "Poor" for any symbols you might have modeled.

The Method / Theory

You don't need to read this section to use the app, but you might be interested.

From Wikipedia:

"The only prior knowledge required is a list of variables which can be *hypothesized* to affect each other **intertemporally.**"

http://en.wikipedia.org/wiki/Vector_autoregression

For example, one might want to test whether the price change of General Motors today (GM) might depend on the recent prices of oil (USO is an oil ETF). This can be modeled using a tool such as this.

Such a *hypothesis* that the change in one asset price might affect another related asset in later days might not be true. It is only a hypothesis. There may not be a predictive relationship for your chosen symbols with the model types we are building. Nonetheless, a model can be built with the app for this case, though it will be a model that backtests and forecasts poorly. In this case, a model's backtest will be poor (low % correct) and the model is probably not useful for forecasting. Hence, *backtesting is critical* for the proper use of this app.

Only price travel Direction is modeled, not the actual price change in dollars. Internally, we actually do model the internal dollar change for close-open price for some of the model types. However, we only report out the overall gain or loss as a binary flag during the day to avoid interpretations of misleading accuracy.

Calculations are done on a server. This avoids undue battery drain on your iOS device.

We don’t report out traditional statistics such as how well the model actually fits the data in question (such as R-square, Adjusted R-square, or percent days fit properly), because those numbers can be *misleadingly accurate* for this type of model, where we want to find small signals amidst a lot of noise. Rather, we recommend looking at backtests to assess model quality as we do here.

Backtesting procedure internals

We already have the “day of forecast” prices for past data, so we just have to “hold our hand over” the prices that day and forget they are there when building the model, then run a forecast with the model and see if it matches our withheld data.

In the case where backtests give reasonable results over a series of prior days, the model may have some predictive power for 1 day ahead forecasts. How long of a successful backtest is needed for the model to be useful is an open question. Should you only rely on longer backtests, or should you rely on more recent (shorter) backtests before you apply a model to forecasting? There are benefits and drawbacks to both strategies: if your backtest is too long, you are pulling in data that may no longer be relevant to tomorrow's results. If you backtest is too short, it may not be robust enough and you may be tuning your model to recent one-off events that don't generalize into the future.

When backtesting, the model is rebuilt every day of the backtest and then used to forecast one day ahead.

Calculations for backtesting may take several minutes. You can start off a backtest trial then go back to the app later and press Get Status / last models to retrieve the results in progress or completed.

Example

Let us consider again the example of the price of GM being a function of some prior prices of USO (oil ETF).

For the GM = f(USO) example, the following data table is automatically generated internally. Then, models are fit to it using a variety of machine learning and regression methods.

Values included in the data table for this example are as follows, assembled from source data (IEX):

GM close - open (value to forecast; dependent variable)

GM close 1 day ago (yesterday)
GM close 2 days ago
GM open 1 day ago (yesterday)
GM open 2 days ago

similarly for USO, abbreviated:

USO close1
USO close2
USO open1
USO open2

This yields 1 variable as a function of 8 other variables.

The model form then becomes:

GMcloseMinusOpen = f(GMc1, GMc2, GMo1, GMo2, USOc1, USOc2, USOo1, USOo2)

Models are solved using "daily differenced" data rather than raw price levels to remove issues due to drift and other "non stationarity" issues. For more details, see:

http://en.wikipedia.org/wiki/Unit_root

Both the raw price levels and the daily differenced versions of the regression tables may be downloaded under the Data & Info tab for further exploration in other tools such as Apple's Numbers or Microsoft Excel or other machine learning / data analysis tools.

Note that all variables on the right side of the above equation are easily looked up before the market opens (all are prior days' data).

It is uncertain exactly when during the evening that IEX Cloud history data is posted for availability, but it is sometime after the close of the market of course.

By default, 200 recent historical data points are used to solve the model (e.g. these are the input data points to the models). We do not feed all historical data into the model but rather we use a rolling window of points, with the window size being user settable. This is tunable in the Tune tab as noted in the above tutorial. By limiting the window, we are allowing simpler models to be used, possibly to good results.

Three model types for f( ) are generated:

model 1: a simulated annealing classifier model
model 2: a simple linear least squares model
model 3: a K-Nearest classifier model or LarsLassoIC linear model

More information on these model types:

http://en.wikipedia.org/wiki/Simulated_annealing

http://en.wikipedia.org/wiki/Ordinary_least_squares

http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LassoLarsIC.html#sklearn.linear_model.LassoLarsIC

The final model type LassoLarsIC is a linear model type with automatic variable elimination based upon the somewhat mysterious (to the uninitiated at least; possibly also to the initiated) "information criteria" as noted in the above link. Since it is a linear model, it may be more useful for shorter time periods and shorter backtests rather than longer versions of each. E.g. it may be better at picking up "local" (in time) effects.

Classifier?

By a "classifier" model, it merely means that we are trying to classify the data into two groups: a **gain** group and a **loss** group, or, equivalently, a **bullish** group and a **bearish** group. The "zero change" value is not considered…the models are not accurate enough to predict the no-change condition to the penny.

Anneal Cooling Rate

Setting the cooling rate to Fast will sometimes yield less accurate answers for model 1 (the anneal model), but the overall backtest will run much faster (seconds instead of minutes). Conversely, sometimes you will find that the setting the anneal cooling rate to "fast" will give better backtests. It may be that when we set the cooling rate to slow, the model attempts to "overfit" the data [overfit = another machine learning vocabulary word, also relevant to standard statistical models such as linear regression]. The Fast cooling mode is useful if you are mainly interested in models 2 and 3 (least square and K-Nearest), in which case you don't need to use Slow Annealing since you will be ignoring that model. It may be a good idea to run a backtest with cooling set to Fast, then if you are not satisfied with models 2 and 3, re-run the test with cooling set to Slow and see if model 1 quality is any better than models 2 and 3. The annealing model is fairly crude so we do not provide fine-grained tuning of it, but we do provide the fast/slow switch for tuning this model in the current version of the app.

Bonus features

"Coinflips" info during backtesting:

When making forecasts of anything, correctness may depend on random circumstances rather than on your stellar predictive expertise. In other words, you may just get lucky. However, it's more difficult to be lucky over a longer period than over a shorter period.

The coinflip tests help you assess if the models you just built were "just lucky" over the backtest period or not.

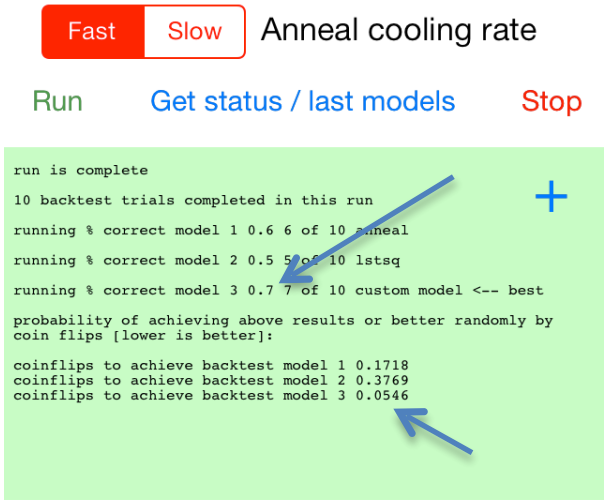
For example, a model may yield 75% correct results during backtesting. However, it's fairly easy to get 75% accurate results randomly if you only run 4 backtest trials (3 out of 4 correct = 75%).

On the other hand, if you run 20 trials and get 75% or more directionally correct, this is a good indicator that you may have a predictive model in the backtest period. It is much less likely that you will get 75% correct answers after 20 trials just by guessing. During backtesting, we report out this type of analysis.

More detailed information on this: http://en.wikipedia.org/wiki/Binomial_test

Backtest results interpretation

Let us look at the output of a particular example backtest run (this is a different model than the SPY/QQQ model above).



Here, our 10 backtest trial showed that 7 of 10 backtests were correct for the “custom model” (our K-Nearest classifier), or 70% accurate, reported as 0.7.

This is marked as the ← `best` model on the output (best for this backtest run sequence).

The “coinflips” value is 0.0546 (or about 5%), which means that there is only a 5% chance of getting 7 of 10 correct if we were to flip 10 coins.

So, this is a pretty good model. Not stellar, but pretty good in the backtest period.

The formal name for this "probability of getting the answer randomly" is the p-value*

http://en.wikipedia.org/wiki/P-value

*** Pro tip**

> *Here we are being conversational and colloquial. Statisticians have a very precise definition for the p-value of a regression, which definition is far out of scope of this document.*

This topic may be very confusing for the non-statistician to understand if he looks at those links, so don't worry too much about it. Just think: "how likely would it be for me to guess my asset price direction correctly for 20 days by flipping coins." Not very likely, right? Also consider that the likelihood (using the colloquial meaning of "likelihood") of getting 70% correct just by random chance goes down as you increase the number of backtest days to 30, 50, 100, etc.

Other buttons in the app

Stop	Stop a model run in progress (a backtest or a forecast).
Get Status / Last Models	Pull the status of a model run and the accuracy of the last models created. Useful when you launch a run, leave the app to send a text or etc, then return to the app to check it later.
Get log	Pull a detailed log file of the server run for debugging purposes and the curious. You can typically ignore this technical information if you are only interested in summary results. Here you will find unformatted outputs of the run on the server including some coefficient information if you are interested in what is going on behind the scenes.

Data & Info tab

You can use this tab to download the data tables generated for your own use in MS Excel, Apple Numbers, or other apps... including other machine learning software. This contains the historical data sets for all the symbols you had specified, properly aligned according to date and with 1 and 2 day lagged variables generated per each open and close price. When aligning, the target symbol is the master date column. All other symbols are aligned to it.

You can also audit the accuracy of data from these Downloadable files.
Missing data is not handled in any special way in this first release, it is just left out. If one of the Candidate predictor symbols has missing data in its history (i.e., does Not have data where it is supposed to be), the system may not solve properly. In general, we just skip an entire day's data if there is missing data in any of the symbols. If this happens infrequently, the one or few data points resulting may be outliers in the data set, and it may not have a noticeable effect on the model. "Your mileage may vary" when missing data is involved.

Historical data

All models are built from historical time series data from IEX Cloud or cryptocompapre.com for crypto data. As such, it only captures behavior that happened in the past and depends on the correctness of such data. Rare Black Swan events are not modeled specifically except as they get thrown into the regular modeling procedures. In fact, Black Swan type events are just treated as outliers in the data and may skew the results a bit, bullish or bearish. Such is the trouble with forecasting using these types of models. Nonetheless, the models are presented as-is for your consideration.

Technical details

As noted earlier, these types of models are a modified version of formal vector autoregression:

http://en.wikipedia.org/wiki/Vector_autoregression

Notable differences in this app:

1. Only 1 target variable is solved at a time, we don't solve for all inputs versus all outputs.

2. We are only trying to solve for price direction (up or down), not an actual value, though the "lstsq" (model 2) does solve internally for the actual (close - open) value. Only the correct/incorrect (direction) is reported in the current version of the app.

3. We introduce nonlinear solver methods, though the eventual model form is linear (e.g. a weighted average of prior values) for model 1 (annealing classifier) and model 2 (linear estimate).

Splits or repricings and New listings

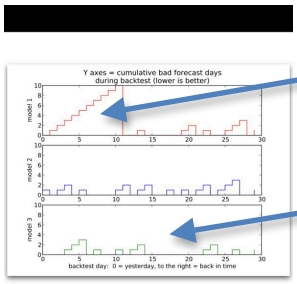
Models are not corrected for splits, reverse-splits or other re-pricings. A split or reprice will show up as an outlier in the data set and so may disrupt the model. *At this point, we don't recommend using the app on stocks or ETFs that have had repricings within the modeled and backtested time period.*

New listings with less than a year's worth of historical data may not work properly in the app, since we often use more than 200 trading days of historical data (by default) for building models. There is no particular check for short historical data, but the model will probably not run correctly by default unless you reduce this look-back window. This can be done in the Tune tab of the app. Basically we need enough data to cover the look-back window and the backtest window, perhaps with a few points extra to avoid "edge" failure cases in the code. When estimating how many points you need to run a model for a recent IPO or ICO, one should also include the length of the volatility computation (default 21 days), which is settable in the Tune tab. As always, let backtesting be your guide.

Miscellaneous / more advanced topics

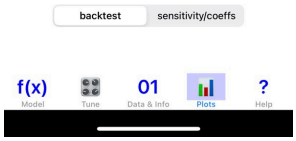
1. Tune tab: trailing volatility length (days)

    This allows you to tune how far back the volatility computation is performed. Changing this value may or may not help with model tuning and backtesting.



high stairsteps indicating
many failures in recent backtests

spread-out failures in backtest, not many in a row suggests a better model (model 3)



2. Plots tab: While the panel on the main Model tab showing model output displays a *summary* of backtest results and overall percent correct along with model quality estimates, the Plots tab shows the specific backtest results per day, and how many days in a row a bad forecast was made during the backtest. This allows you to see if a model is getting better or worse over time in the backtest period. The better case is if you *don't* see a large stack of stair-steps up (which indicate lot of failures in a row), but that the failures in the backtests are more spread-out (low stacks of stair-steps up). Note that on the horizontal axis of the stair-step plots, it shows the recent time periods are to the *left* of the graph rather than to the right as traditionally plotted-- more recent backtest failures show up on the left.

The sensitivity / coefficients plots allow you to see the estimated sensitivity of the forecast to the variables in the model. This may be useful if you decide to pull the regression table from the Data & Info tab and build your own model. It may provide a convenient starting point for which variables to include in such a model. The top graph is estimated (unsigned) sensitivities from model 3 (which has a variable type depending on the setting you use: currently the choices are K nearest neighbor and LassoLarsIC), and the bottom graph is signed coefficients of the full linear model.

3. Lag structure (advanced): We only use 2 days of lag in the model (lag in a time-series sense) as you can see if you download the data in the Data & Info tab or look at the sensitivity plots in the Plots tab, and we include opening prices and volatility (which by definition includes many days of data in it; 21 days by default; see the Tune tab in the app / volatility length value). We arrived at this by some coarse model testing and the general concept that "information more than 2 days old is likely priced into the market already." This is not always true and may preclude the model from picking up longer period cycles in the data (if any). Such is a topic for model enhancement.

4. Another difference from traditional VAR (Vector Auto Regression) models: This app models close-open price on the future day, not (close price minus prior day's close price) as is done in standard VAR presentations. We also include open prices and the historical volatility value in the model (also lagged), and are not limited to linear models of prior values as is traditional VAR.

5. We do not include the current day's opening price in the close-open forecast. If we were to do this, it would require the opening price to be known on the forecast day before we could make the forecast. E.g. we couldn't make the forecast the day or night before. While including the current day's opening price in this type of model might improve forecast quality, such is more suited for automated trading since it is more time-critical (e.g. the forecast should be run immediately after market open) than a partly-manual model builder such as this.

Notes

[Note 1] For more discussion on this topic, see the following document, especially the first line: "Since its invention 90 years ago, the p value has become the standard by which most quantitative research is judged; however, it was never intended for this purpose."

At least one issue noted in this document is addressed in our app in that we also provide the magnitude of correctness (the percent correct) during the backtest period. There are p-values with respect to variables and p-values with respect to whole model forecasts. This app provides the latter.

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5738950/