Optimizing Portfolios to (Hypothetically) Make Money

Instructor: Rafid Mahmood

March 21, 2017

How I got into Analytics: Quantathon 2016

- Hackathon by UofT DCS & Waterfront Int. Ltd.
- 48 hours, 3 person teams
- Problem: given dataset of daily stock prices, construct portfolio strategy:
  - Rules for which stocks/how much to invest
  - Daily rebalancing: update portfolio every day

- Analytics approach:
  - Had no prior domain knowledge => ask questions
  - Data engineering principles => test hypotheses

The dataset: 4 years trading history x 100 stocks

- Don't know what stocks they are
- Daily data for Jan 4, 2000 – Dec 31, 2003
- Movement information:
  - Opening price (OP), Closing price (CP)
    Intraday high (IH), Intraday low (IL)
    Trading volume (TVL) Fill Indicator (IND)

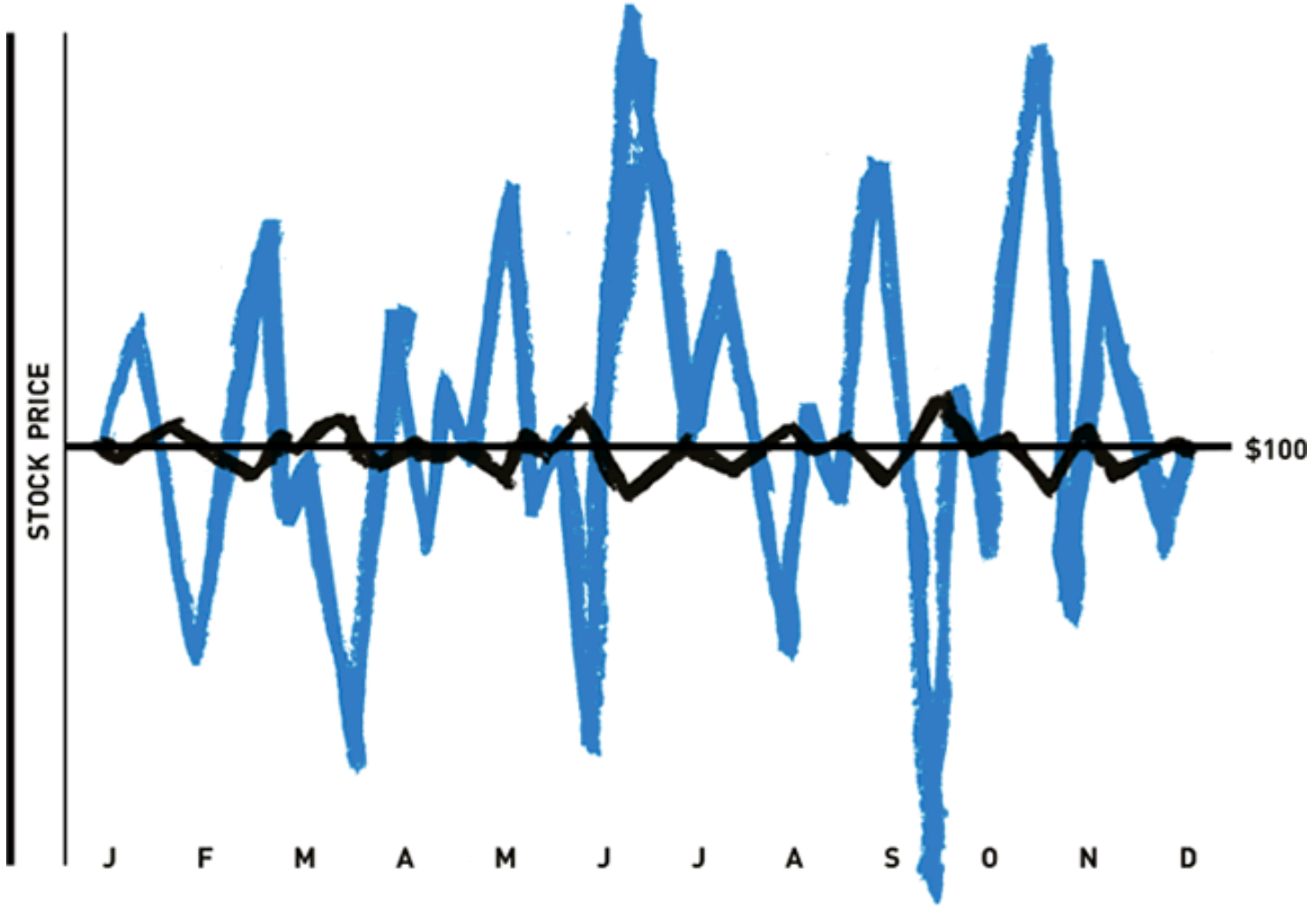
The general problem: decide what to invest every day

- Every day (t) and stock (j), calculate weights $W(t,j)$
  - $W(t,j) > 0$: long position (buying)
  - $W(t,j) < 0$: short position (selling)
- Construct our own function/strategy to get weights

$$W(t,j) = f(data)$$

- Used the 4 year dataset to construct strategy
- Evaluated on out-of-sample dataset from 2004-2015 (not available to us)

The goal: high expected returns BUT ALSO low risk



How do you evaluate the portfolio

- Return is change in price times amount invested

$$ROC(t,j) * W(t,j)$$

  - Where $ROC(t,j) = \frac{CP(t,j)}{OP(t,j)} - 1$ is daily open-to-close ratio

- Daily Return of Portfolio is normalized sum of returns

$$RP(t) = \frac{\sum_j W(t,j)ROC(t,j)}{\sum_j |W(t,j)|}$$

- Desire portfolios with high return and low variance

$$Sharpe\ Ratio = \frac{mean[RP(t)]}{variance[RP(t)]}$$

To summarize the problem

- Find strategy f(…) to get weights

$$W(t,j) = f(historical\ data)$$

$$\Rightarrow RP(t) = \frac{\sum_j W(t,j)ROC(t,j)}{\sum_j |W(t,j)|}$$

- Portfolio measured using Sharpe ratio

$$Sharpe = \frac{mean[RP(t)]}{variance[RP(t)]}$$

- Our overall objective

$$\max_{f(\dots)} \ Sharpe\ (\dots)$$

Determine weights using trade signals

- Signal: statistics from movement information

$$W(t,j) = f(signalA,\ signalB,\ \ldots)$$

- Causality: cannot use today's closing prices, volume, etc.
  - Only todays opening price and historical data

Signals can get very complicated

- Basic signals:
  - $RCO(t,j) = \frac{OP(t)}{CP(t-1)} - 1$
  - $ROO(t,j) = \frac{OP(t)}{OP(t-1)} - 1$

- Combine basic signals to more advanced signals:
  - $dRCO(t,j) = \frac{RCO(t,j) - mean(RCO|t)}{N}$

- Can go even further!
  - $Vol.Adj.dRCO(t,j) = \left(\frac{TVL(t-1,j)}{mean(TVL|t-1)}\right) dRCO(t,j)$

Part 1: Provided a basic model to start

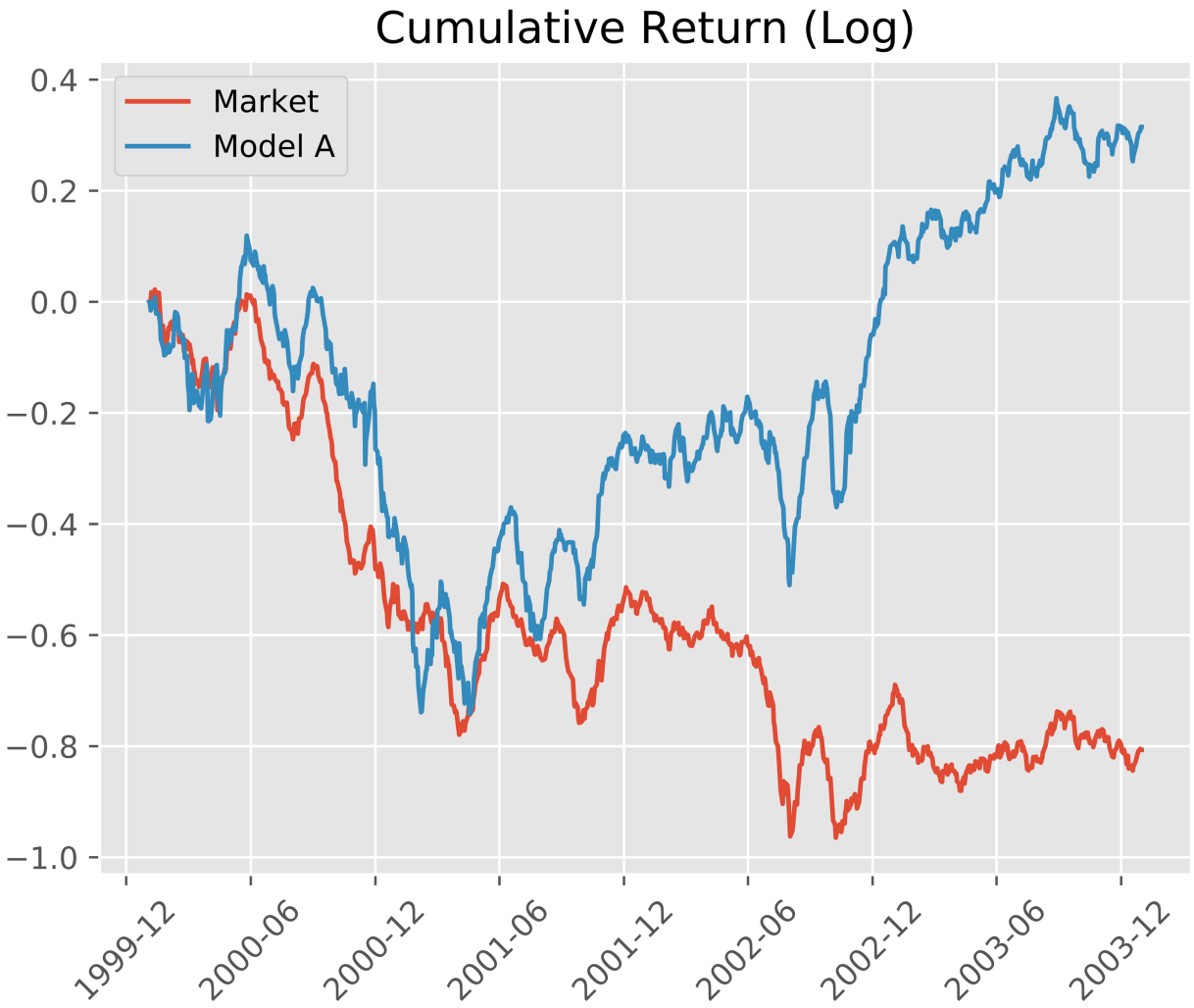
- Given 12 suggested signals
- $W(t,j) = a_1 * signalA + a_2 * signalB + \ \ldots$
- Find the optimal coefficients:

$$\max_{a_1 \ldots a_{12}} \ Sharpe\ Ratio\ (RP(W(a))$$

- Python => scipy.optimize => Different algorithms

This function is not easy to optimize

- Lots of local optima => different solution every time

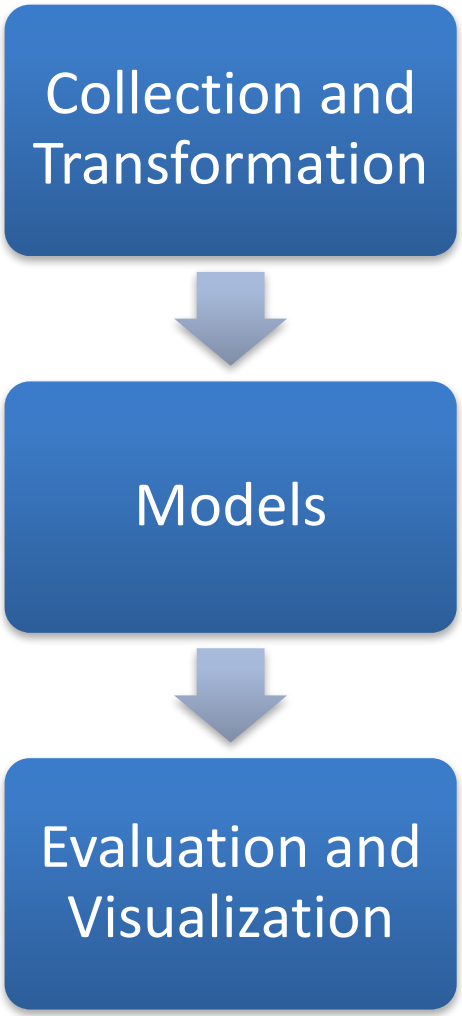


Part 2: constructing our own strategy

- **Data Science Step 0:** Build strong architecture/code that lets you easily test hypotheses and models

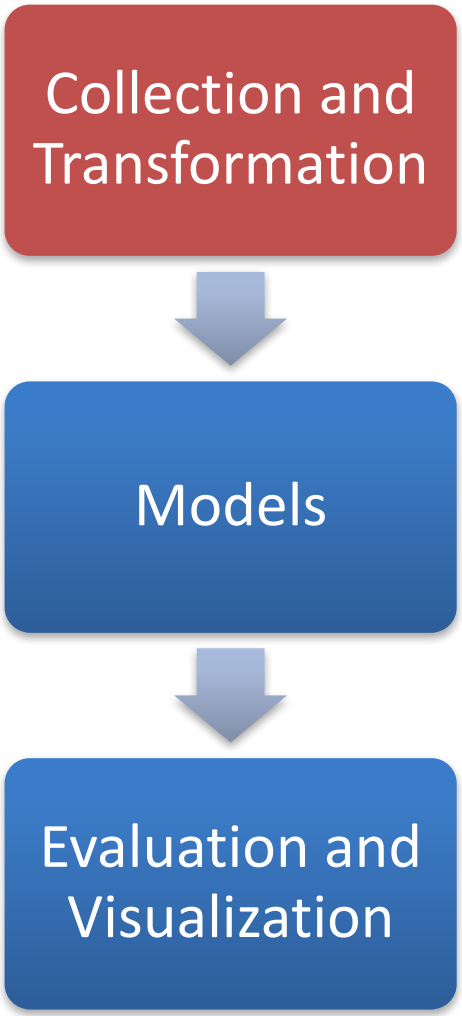
- **Data Science Step 1:** Test every hypothesis and model that you can come up with to see if it works

General rules for data science



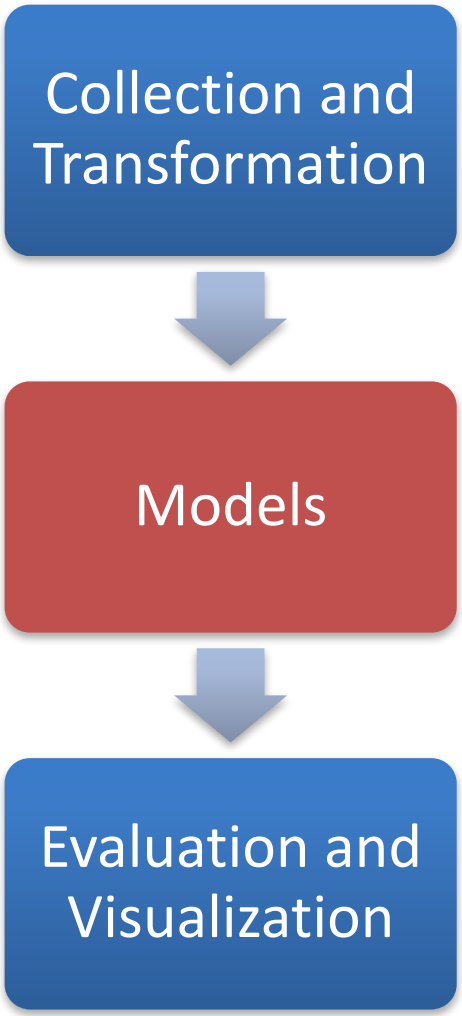
- Modularize each block
  - Keep code in separate functions/files for reusability
- Build generic functions
  - Don't hardcode parameters or you will forget to tune them
- Pair programming
  - Easier to catch bugs, search things, etc.
- Test for mistakes
  - If you can predict random numbers, something is wrong!

How we built our strategy



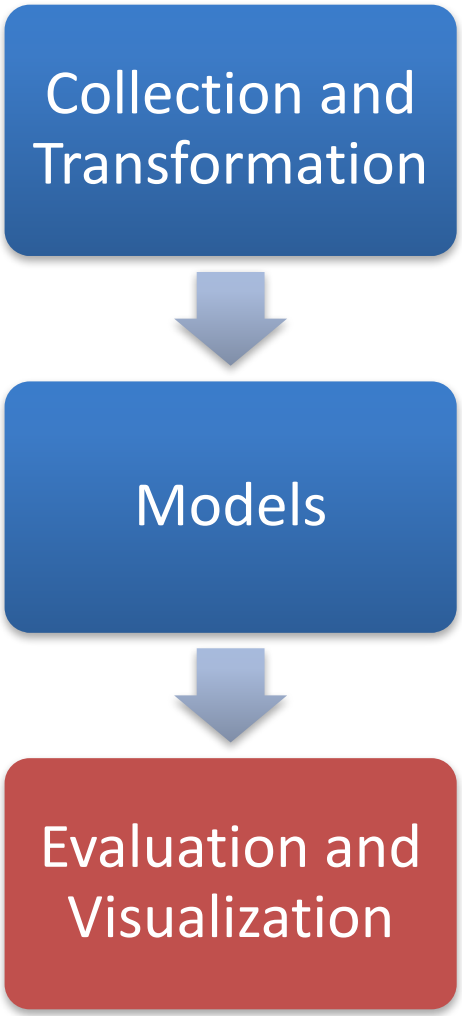
- Don't waste time loading big CSV files when you need data
  - Quick solution: save important data in *.mat, *.pickle files
  - Robust solution: convert data to databases, load with SQL, etc.
- We needed to compute large matrices with signal data:
  - Compute once, save to *.pickle, and load when needed

How we built our strategy



- Write generic functions that take arbitrary inputs
  - Don't hardcode model params!
  - You want to be able to easily test any hypothesis that you have
- Build different models and use grid searching to find the best
- We wanted to experiment with different signals, parameters
  - Not rewriting custom functions

How we built our strategy



- Evaluate all models with a consistent score/metric.
  - We tested strategies by calculating the Sharpe ratio
- Generate lots of different plots to visualize results
  - *Scores do not give the whole picture!*
- Be pessimistic about success
  - If you're doing unexpectedly well, you might be breaking causality

After thinking about the problem, we had an idea…

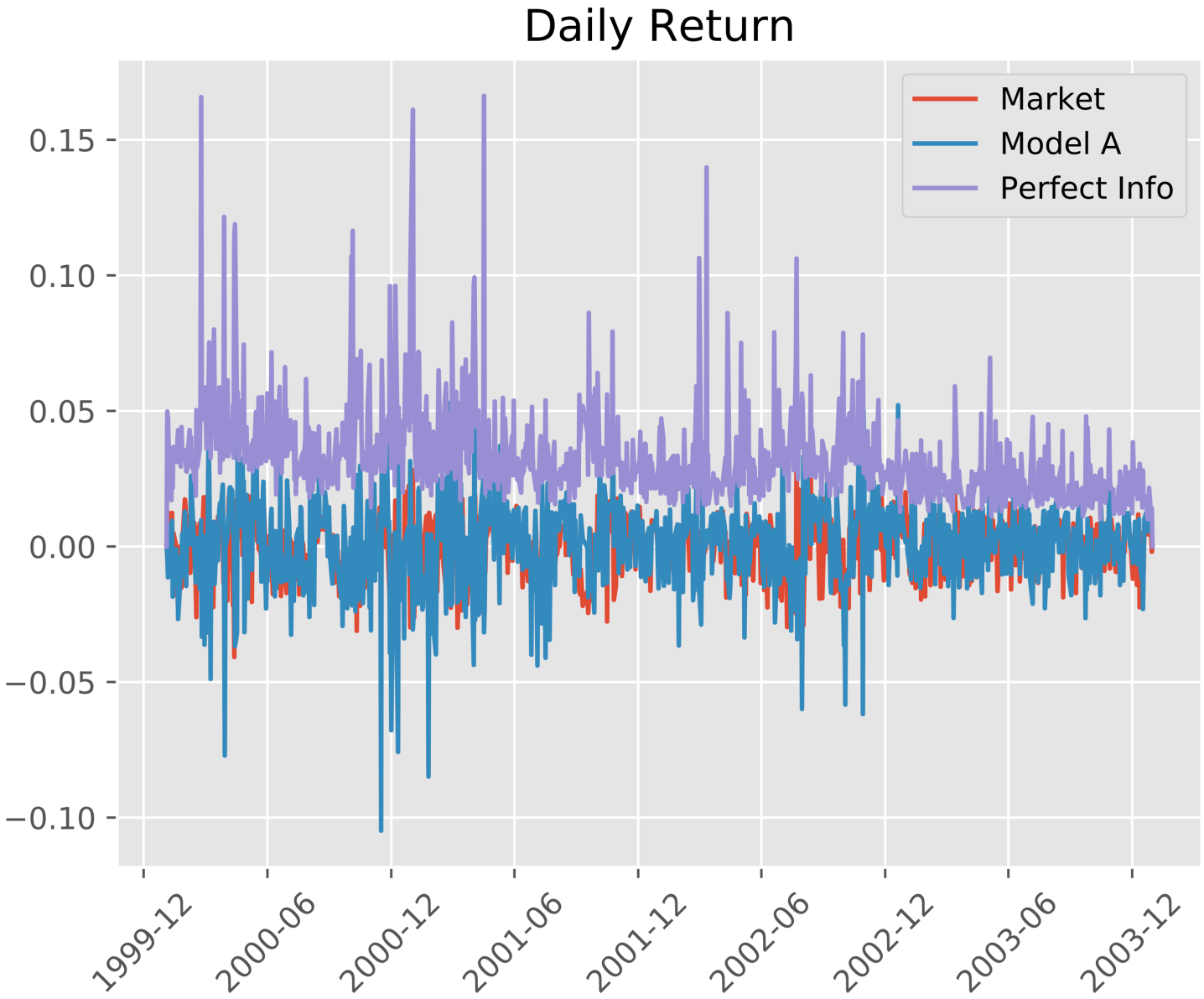
- Hypothesis: ideal scenario is if you know closing price
- Recall returns are calculated $W(t,j) * ROC(t,j)$, where

$$ROC(t,j) = \frac{CP(t)}{OP(t)} - 1$$

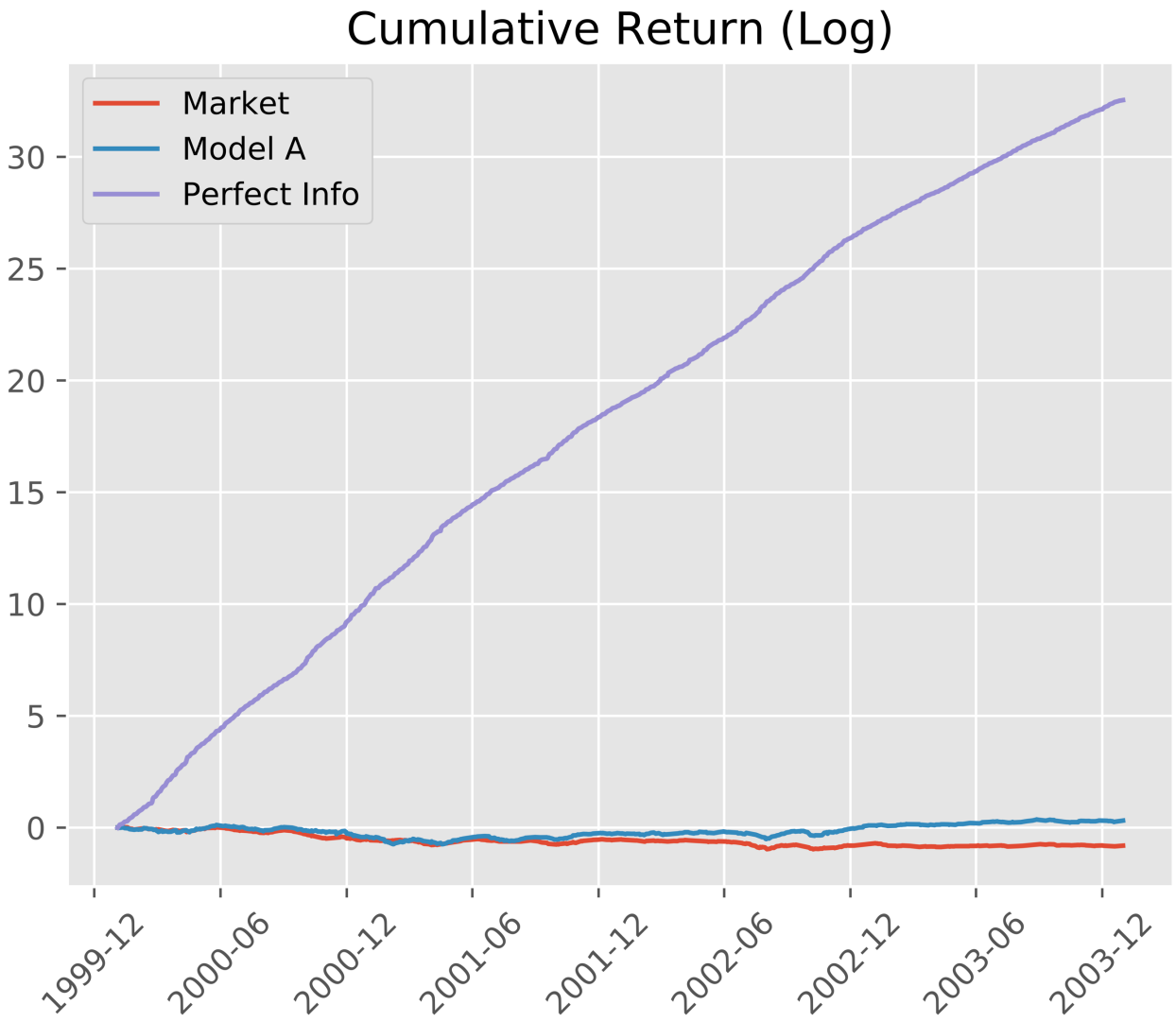
- What happens if you let $W(t,j) = ROC(t,j)$?

- Not a real strategy (due to causality), BUT
  - Instead of finding a f(…), we could just predict ROC(t,j)

Testing our hypothesis by fixing $W(t,j) = ROC(t,j)$



This (imaginary) strategy yields ridiculous returns



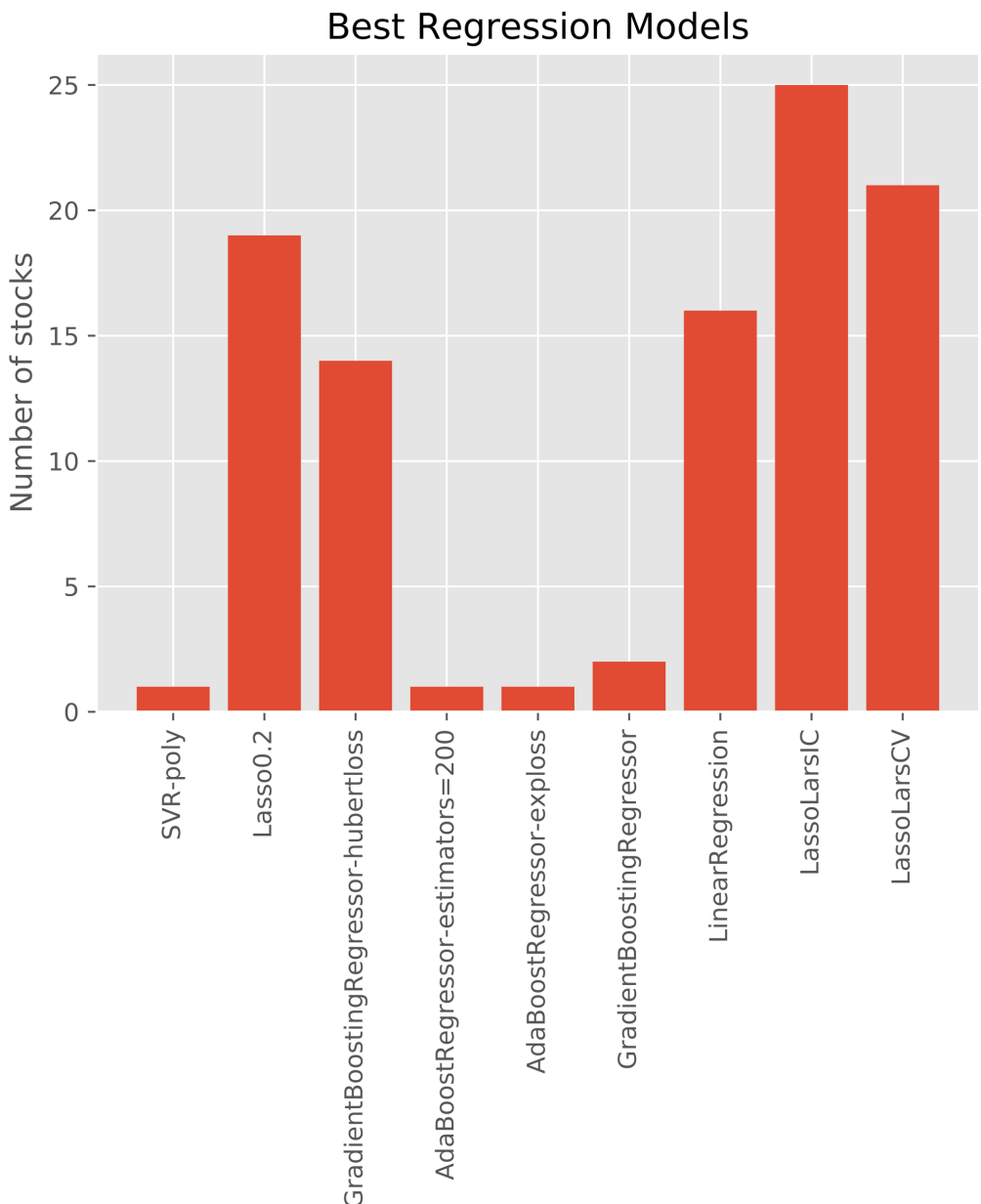
Our strategy: let W(t,j) = predicted ROC(t,j)

- W(t,j) = f_j (signal A, signal B, …) for the j-th stock
  - f_j(…) can be Regression, Forests, etc.
- Built separate prediction model for each stock
- 14 signals: suggested and some of our creations
- ~1000 data points/model, with 0.8 test/train split

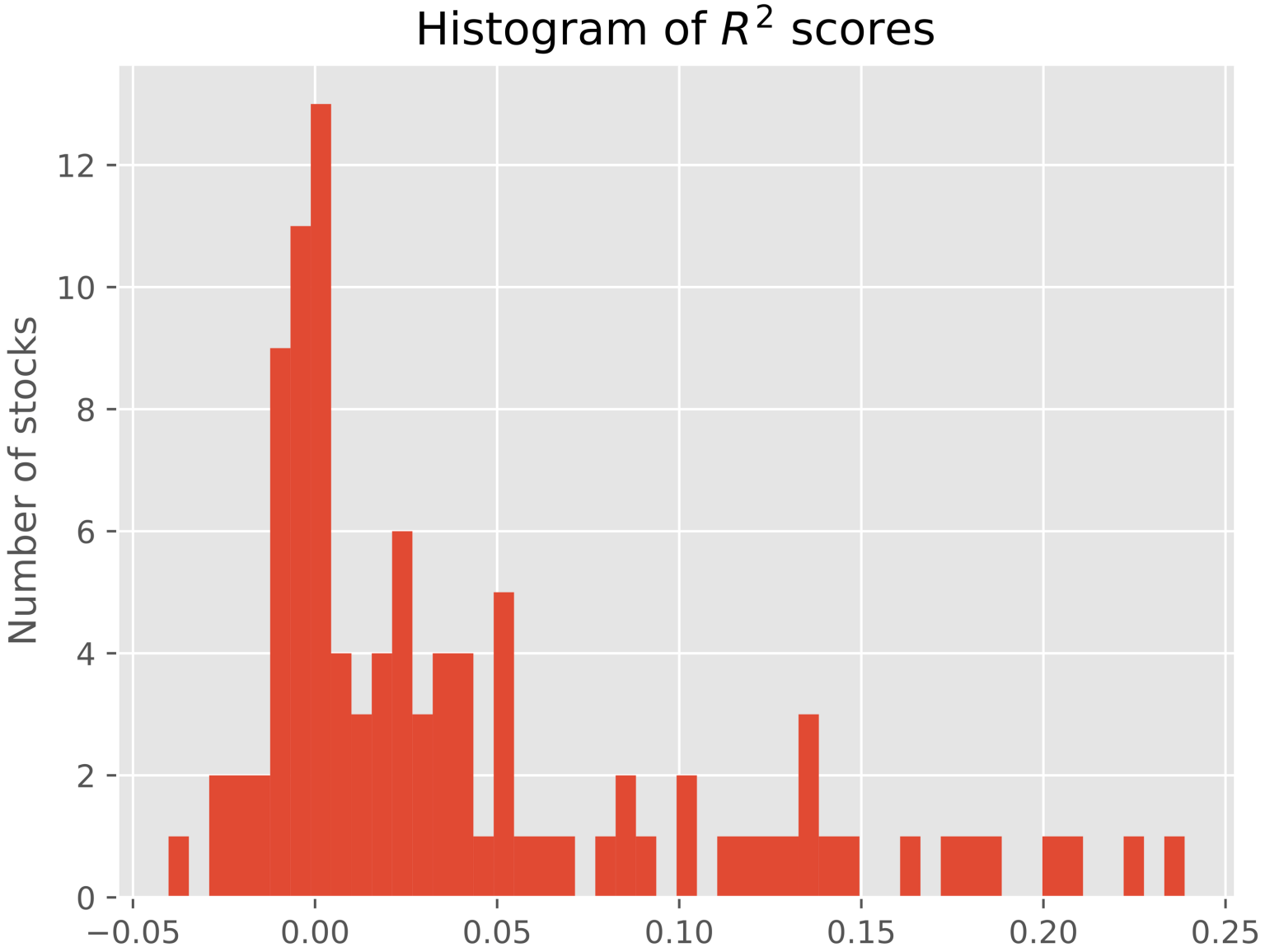
Grid searching: the quick + dirty way to predict

```
regressors = [
    ("AdaBoostRegressor-exploss",              AdaBoostRegressor(loss='exponential')),
    ("AdaBoostRegressor-estimators=200",       AdaBoostRegressor(n_estimators=200)),
    ("LinearRegression",                       LinearRegression()),
    ("Lasso0.2",                               Lasso(alpha=0.2)),
    ("Lasso0.1",                               Lasso(alpha=0.1)),
    ("Lasso0.8",                               Lasso(alpha=0.8)),
    ("BaggingRegressor",                       BaggingRegressor()),
    ("BaggingRegressor-bootstrapped",          BaggingRegressor(bootstrap_features=True)),
    ("GradientBoostingRegressor",              GradientBoostingRegressor()),
    ("GradientBoostingRegressor-hubertloss",   GradientBoostingRegressor(loss='lad')),
    ("LassoLarsIC",                            LassoLarsIC(criterion='aic')),
    ("LassoLarsCV",                            LassoLarsCV()),
    ("SVR-rbf",                                SVR(kernel='rbf')),
    ("SVR-poly",                               SVR(kernel='poly')),
    ("SVR-sigmoid",                            SVR(kernel='sigmoid')),
    ("DecisionTreeRegressor",                  DecisionTreeRegressor()),
    ("RandomForestRegressor",                  RandomForestRegressor()),
]

maxScore = -100
bestRegressor = None
for (name, r) in regressors:
    r.fit(XTrain, yTrain)
    score = r.score(XTest, yTest)

    if score > maxScore:
        maxScore = score
        bestRegressor = (name, r)

allBestRegressors.append(bestRegressor[0])
allScores.append(maxScore)
print('Best regressor for stock {} is {}, Score = {}'.format(j, bestRegressor[0], maxScore))
```

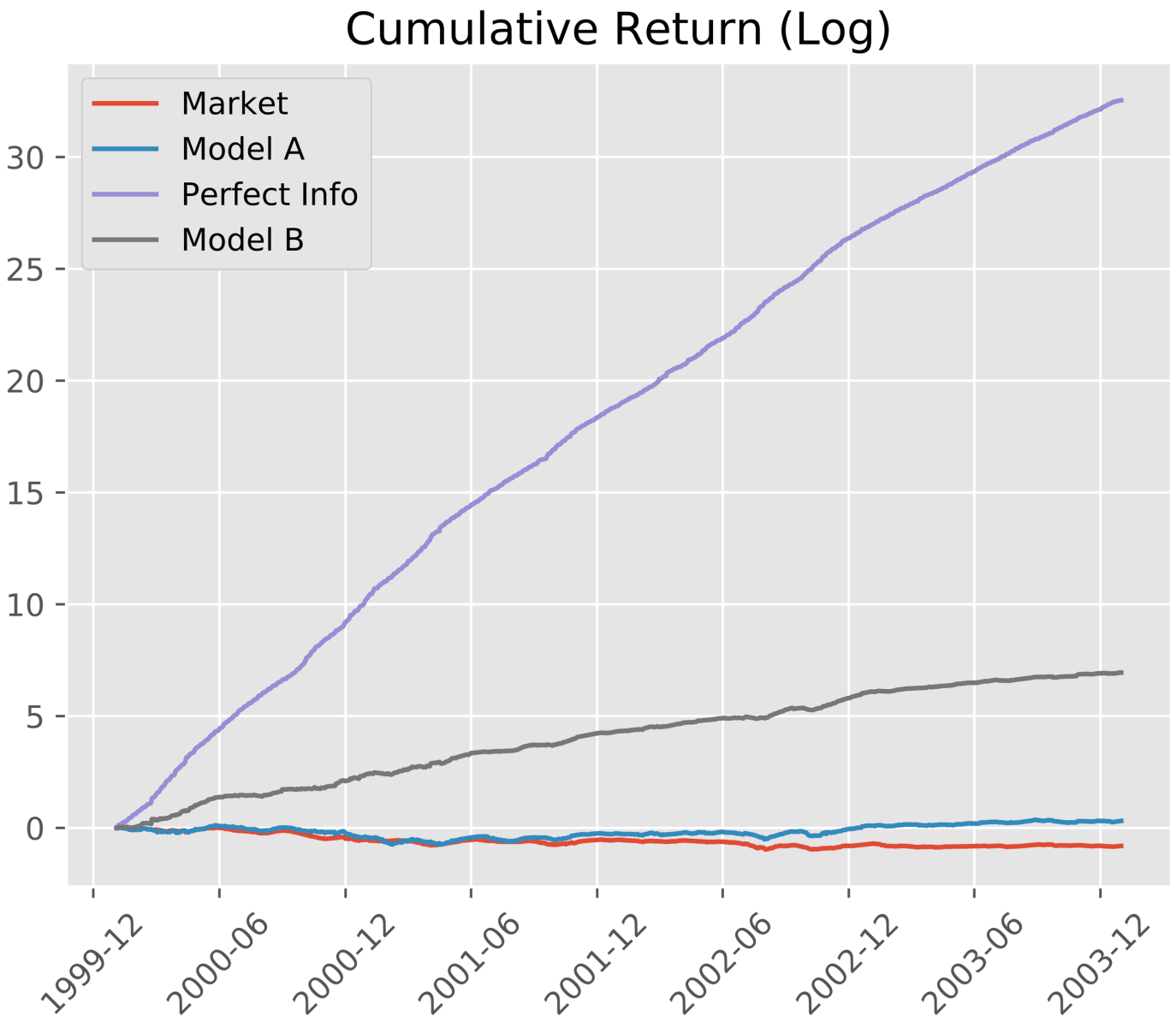
Different models work better with different stocks



Predicting ROC is still pretty hard



But our strategy improved significantly



How well does our model perform?

- **For Model A:**
- Log-Cum return: 0.3
- Sharpe Ratio: 0.43
- Largest consecutive loss: 0.22 over 12 days

- **For Model B:**
- Log-Cum return: 7
- Sharpe Ratio: 6.75
- Largest consecutive loss: 0.09 over 8 days

- **Comparing to industry standards:**
- Sharpe Ratio > 2 => decent strategy
  > 5 => Algorithmic Hedge Funds

Can I really make money with this method?

- Yes and no
  - P. D. Corte, R. Kosowski and T. Wang, **Market Closure and Short-Term Reversal**, AsianFA 2016 Conf.

- However several assumptions were made:
  - Transaction costs
  - Order filling/successful trades
  - Backtested on old data (2000-2003)

In the end, we won, and used the winnings to travel