Minimum non-viable product

Model monthly stock returns using

- Scraped from FRED using Selenium:
  - Growth (Leading index)
  - Inflation (CPI ex food & energy)
  - Monetary Tightness (10-year bond rate - 3-month T-bill rate)
  - All lagged 1 month (which I just realized I forgot to do)
- CRSP + Compustat
  - 55 years of monthly data, top 50% of stocks by market cap (varies from ~500 stocks at start of sample to ~1200 at end of sample, peaked at ~1400 stocks)
  - Value (Stock Price / Book Value)
  - Momentum (11-month change in stock through one month prior)
  - Industry (Fama 48 industries 1-hot with pd.get_dummies)

DATETIME	SYMBOL	PERMNO	INDUSTRY	VALUE	MOMENTUM	MONETARYPOLICY	GROWTH	INFLATION	RET
1963-01-31	T	10401	32	0.613024	-0.131919	0.92	99.939763	0.009615	0.036403
1963-01-31	GM	12079	23	0.408998	0.030786	0.92	99.939763	0.009615	0.079570
1963-01-31	XOM	11850	30	0.686189	0.163202	0.92	99.939763	0.009615	0.006303
1963-01-31	DD	11703	14	0.336631	0.115541	0.92	99.939763	0.009615	0.026096
1963-01-31	IBM	12490	34	0.126757	-0.306857	0.92	99.939763	0.009615	0.086538

See https://en.wikipedia.org/wiki/Carhart_four-factor_model (https://en.wikipedia.org/wiki/Carhart_four-factor_model)

1. The market model:

$$EXR_t = \alpha^J + \beta_{mkt} EXMKT_t + \epsilon_t$$

The intercept in this model is referred to as the “Jensen’s alpha”

2. The Fama-French three-factor model:

$$EXR_t = \alpha^{FF} + \beta_{mkt} EXMKT_t + \beta_{HML} HML_t + \beta_{SMB} SMB_t + \epsilon_t$$

The intercept in this model is referred to as the “three-factor alpha”

3. The Carhart four-factor model:

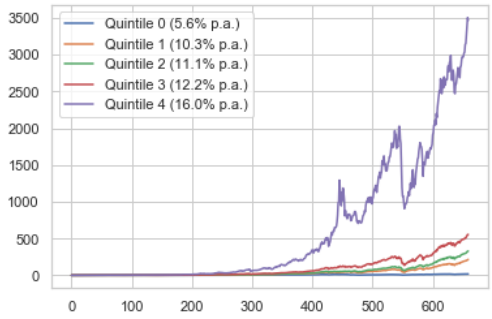
$$EXR_t = \alpha^c + \beta_{mkt} EXMKT_t + \beta_{HML} HML_t + \beta_{SMB} SMB_t + \beta_{UMD} UMD_t + \epsilon_t$$

In other words

- Original Markowitz CAPM model models stock returns as risk-free rate + beta * market factor + epsilon -
  - Factors explain ~70% of returns (epsilon ~ 30%)
  - http://book.ivo-welch.info/read/source.mba/chap10.pdf (http://book.ivo-welch.info/read/source.mba/chap10.pdf)
- Fama - French added value and size (market cap)
  - Factors explain ~90%
  - https://faculty.chicagobooth.edu/john.cochrane/teaching/35904_Asset_Pricing/Fama_French_multifactor_explanations.pdf (https://faculty.chicagobooth.edu/john.cochrane/teaching/35904_Asset_Pricing/Fama_French_multifactor_explanations.pdf)
- Carhart adds momentum
  - recent price changes often persist if they reflect hidden factors that take time to be appreciated, if success breeds success (i.e. successful companies whose stock is going up have good press, better hires, better credit etc., which leads to more success)
- https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1997.tb03808.x (https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1997.tb03808.x)
- AQR has made a lot of money on value/momentum https://www.aqr.com/Insights/Research/Journal-Article/Value-and-Momentum-Everywhere (https://www.aqr.com/Insights/Research/Journal-Article/Value-and-Momentum-Everywhere)

Baseline: Momentum only

- Bucket momentum quintiles
- Each month, go long the stocks in e.g. top momentum quintile
- Seems to have worked (surprisingly well?)

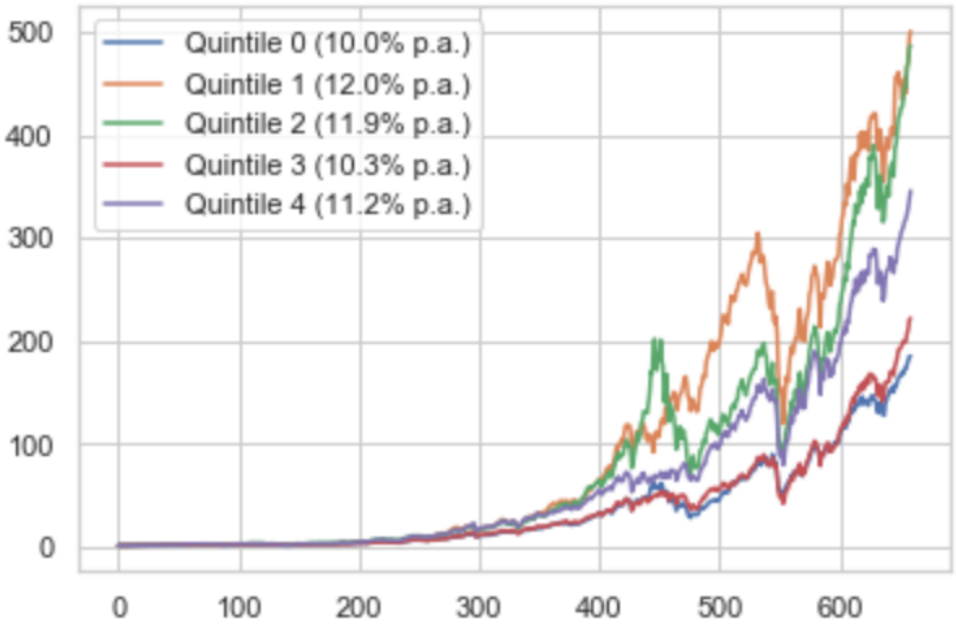


	Label	Annualized return	Annualized volatility
0	Quintile 0	5.580271	0.217176
1	Quintile 1	10.267456	0.164926
2	Quintile 2	11.115857	0.151383
3	Quintile 3	12.177722	0.156415
4	Quintile 4	15.974936	0.200819

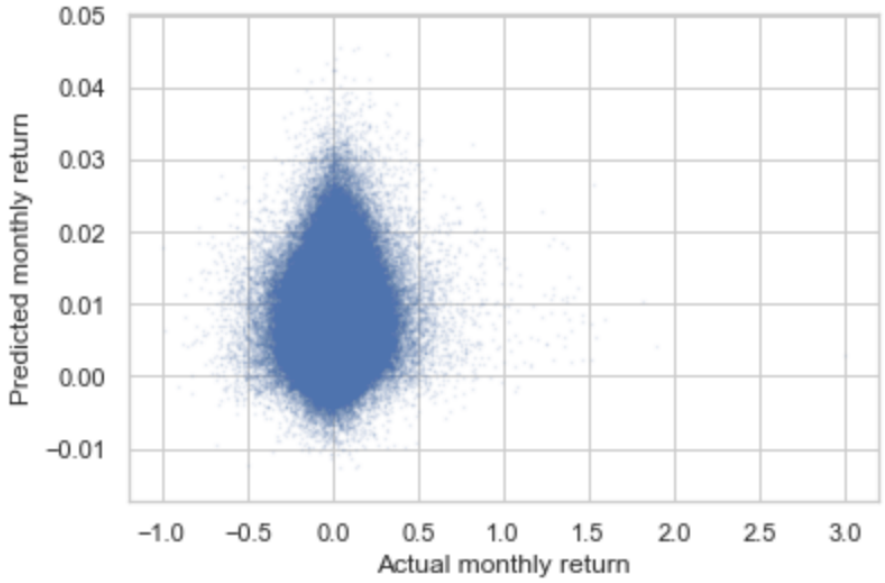
OLS model

- Fit monthly RET vs.
    - 2 Fundamental variables: Value, Momentum
    - 3 Econ variables
    - 48 industries (1-hot dummies)
- Get a tiny but nonzero R-squared - 0.1% out of sample
- Bucket predicted return into quintiles
- These quintiles have much less impact than momentum only
- Which is interesting because the prediction should have the momentum information + whatever information is in the other variables
- Thinking about how to fix, what I might be doing wrong

	Label	Annualized return	Annualized volatility
0	Quintile 0	9.958554	0.153951
1	Quintile 1	11.965378	0.173933
2	Quintile 2	11.907539	0.193946
3	Quintile 3	10.321383	0.176276
4	Quintile 4	11.213868	0.174644



```
R-squared: 0.002159
```




```
# out-of-sample test
y_test_pred = model.predict(X_test_scaled)
print ("R-squared: %0.6f" % sklearn.metrics.r2_score(y_test, y_test_pred))
```

```
R-squared: 0.001423
```

Things I tried

- basic profiling, pair plot
- Lasso, ElasticNet, Ridge, didn't improve OLS R-squared