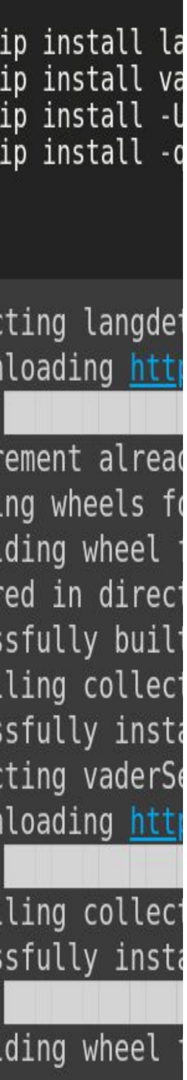
Algorithmic Trading in Python

Team 6



Introduction

Goal of the Project

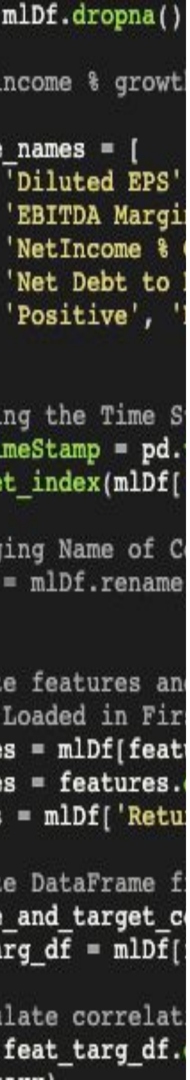
- Create "Financial Machine Learning Model" that predicts returns
- Implement "Algorithmic Trading" that utilizes fundamental and sentiment data to derive absolute return strategy both in bull and bear market.

What Strategy:

- "Micro Alpha model": Mainly Data-driven
- "General Sentiment" model: Measure the effect of sentiment in Social Media *vs* Technical
- Algorithm Upgrade: include 8 Fundamental data factors
- Buy Long at positive ML signal, Sell Short at negative ML signal
- Short-term trading: 1 day
- 1 stock: Red Hat (RHT)

Data collection

- Fundamental Data
- Alternative Data
- Sentiment Analysis



Fundamental Data Collection

Profitability

- Diluted EPS from Continuing Operations: Absolute number
- EBIT Margin, EBITDA Margin: %

Growth

- Revenue % Growth, EBITDA % Growth, EBIT % Growth, Sequential % Growth in Net Income from Continuing Operations

Financial Structure

- Net Debt to Equity

Feature of Quarterly Released Data: Look ahead bias

- Time delay of data: previous quarter’s data
- Difference in time horizon: “Daily” trading & sentimental analysis *vs* “Quarterly” data

Alternative Data Collection

From Github Repo collect the tweets

install Libraries

Load Libraries

Load past Data

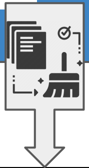
Load Twitter Past WebScraped

Load Fundamental past Data

Load Cleaned and Aggregated Tweets

Load Final Df for ML

Data Cleaning



What does it mean for B2B businesses when a tech giant like IBM acquires RedHat The Cloud is about to be a whole lot more important in 2019 and beyond.

Metrics

1.Positive Sentiment:
compound score >= 0.05

2.Neutral Sentiment:
(*compound* score >- 0.05)
& (*compound* score < 0.05)

3.Negative Sentiment:
compound score <=- 0.05

Results

```
aggDfDay.tail()
```

timeStamp	Favourites	retweets	Compound	Positive	Neutral	Negative
2019-01-10	330	128	25.3206	12.747	70.123	2.131
2019-01-11	541	112	33.4915	18.343	53.682	0.973
2019-01-12	116	63	10.1557	7.588	17.961	0.452
2019-01-13	182	75	9.1907	4.500	21.400	1.101
2019-01-14	154	89	10.8001	5.615	34.459	0.926

Data Processing

Data Preparation

Features

Fundamental

Diluted EPS from Continuing Operations

Revenue % Growth

EBIT Margin

EBIT % Growth

EBITDA Margin

EBITDA % Growth

Sequential % Growth in Net Income

Net Debt to Equity

Alternative

Favourites

Positive

Retweets

Neutral

Compound

Negative

Training- & Testing Data Split

Training Data 85%

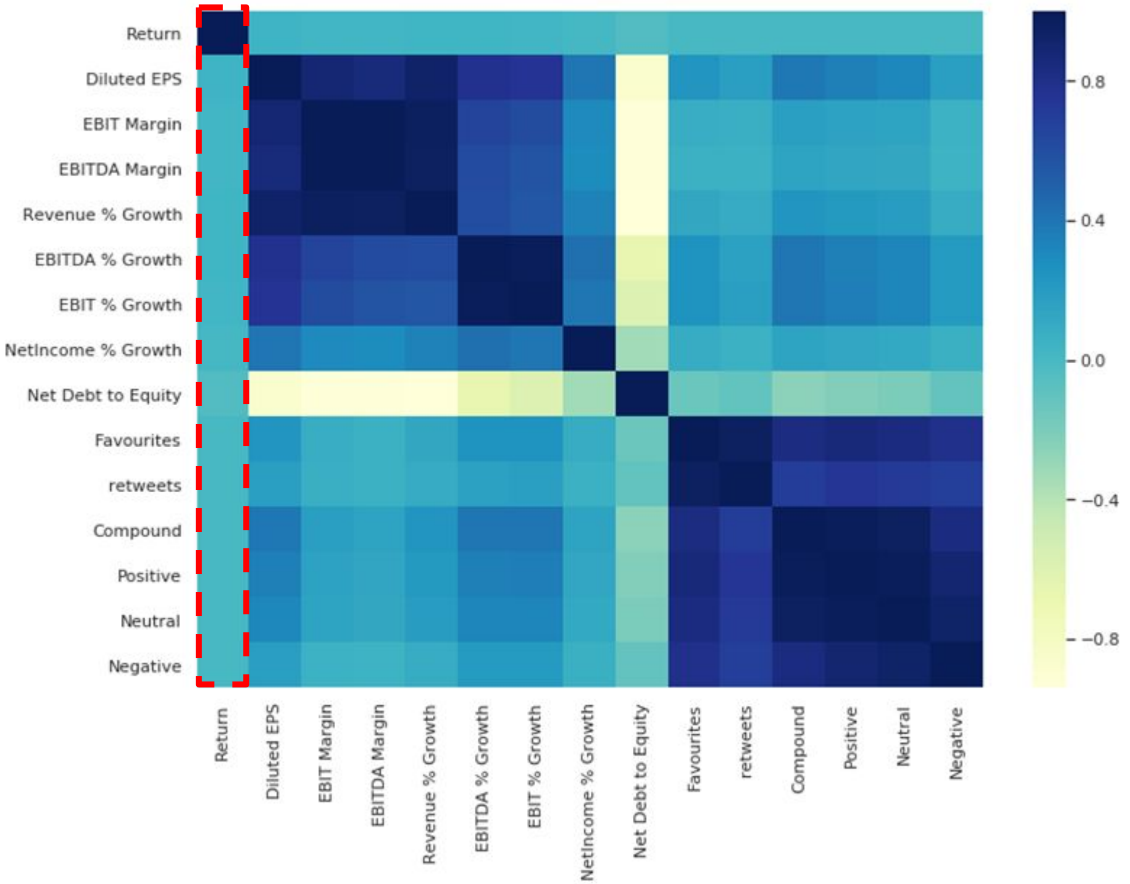
Testing Data 15%

Target

Return

```
from sklearn.
# Create the
rfr = RandomF
rfr.fit(train
# Look at the
print(rfr.sco
print(rfr.sco
from sklearn
depths = np.
num_leafs =
param_grid =
gs = GridSea
gs = gs.fit(
print('Best
depths = np.
num_leafs =
param_grid =
gs = GridSea
gs = gs.fit(
print('Best
```

Correlation between the Target Variable and the Features



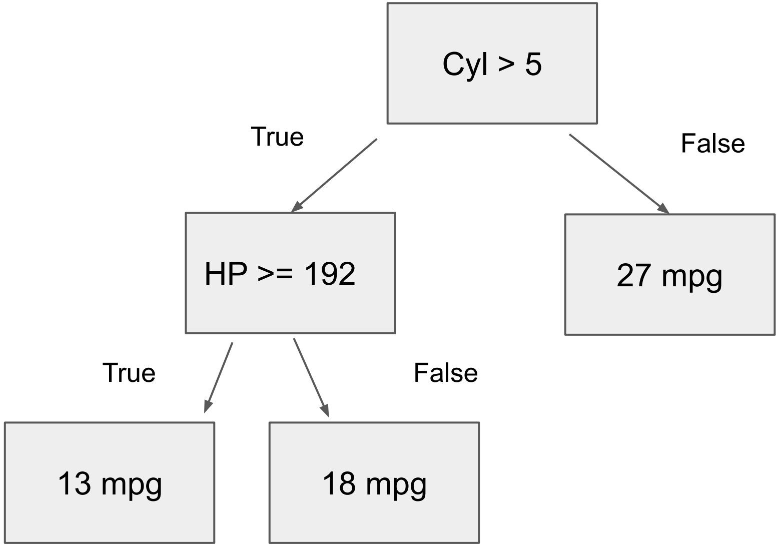
Correlation between the target variable and the features: near zero

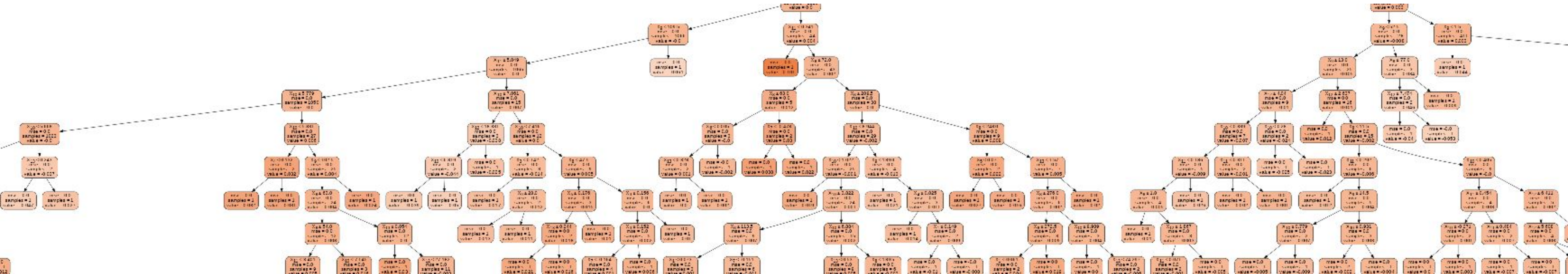
- Historical data
- Investors' expectations
- Unexpected events
- Nonlinear relationship

Machine Learning

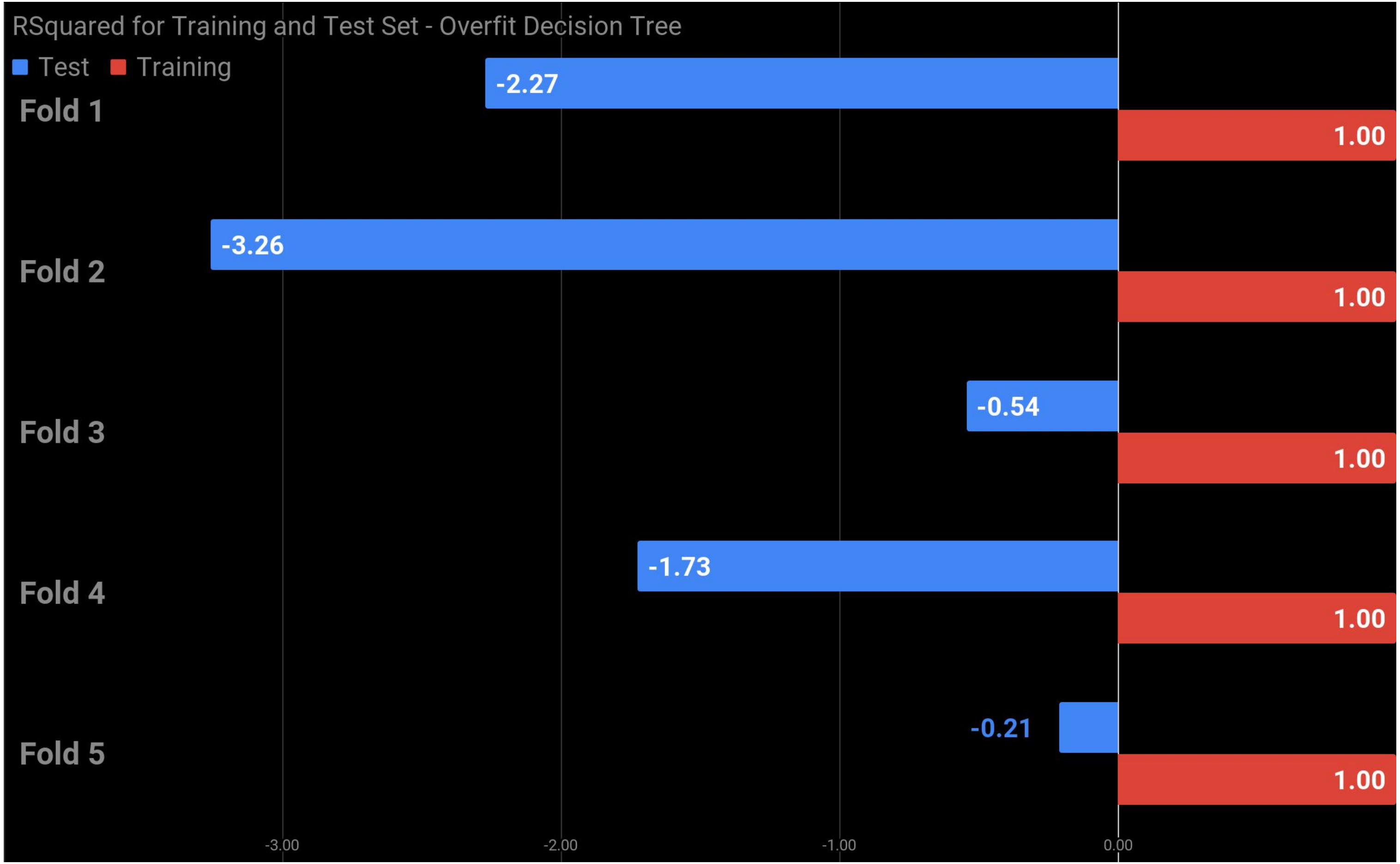
Decision Tree

- Prone to Overfitting
- R Squared of Training Set: 0.99
- R Squared of Test Set: -0.75





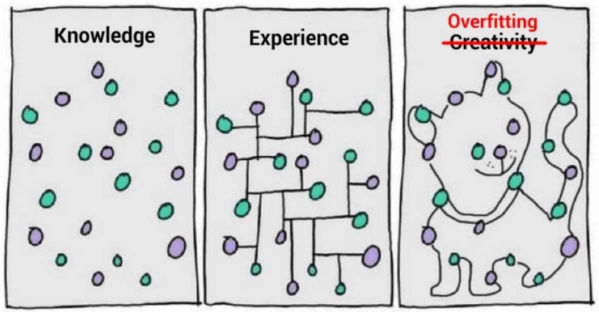
Decision Tree - Cross Validation



How easy is to Overfit a Model?

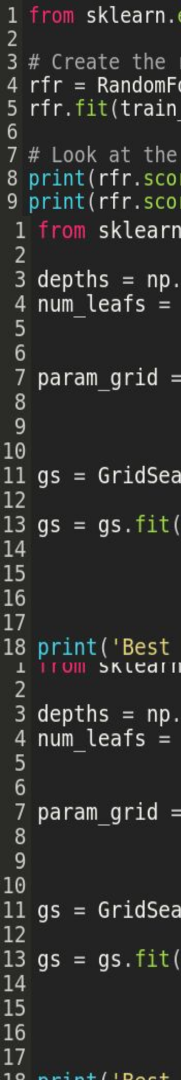
3 Lines of Code!

```
from sklearn.tree import DecisionTreeRegressor

# Create a decision tree regression model with default arguments
decision_tree = DecisionTreeRegressor(random_state=0)

decFit = decision_tree.fit(train_features,train_targets)
```

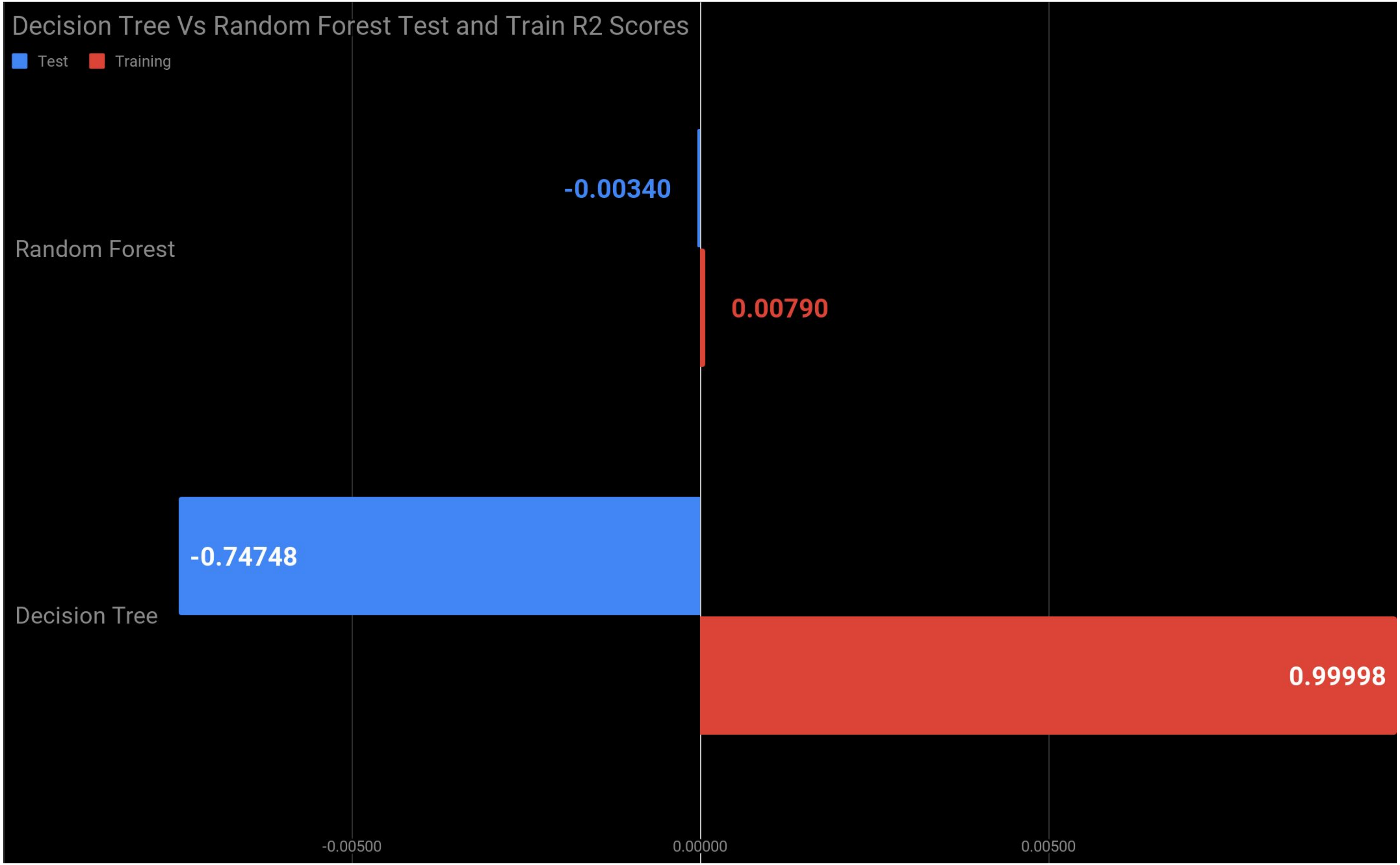


How easy is to lose all your money in the stock market!

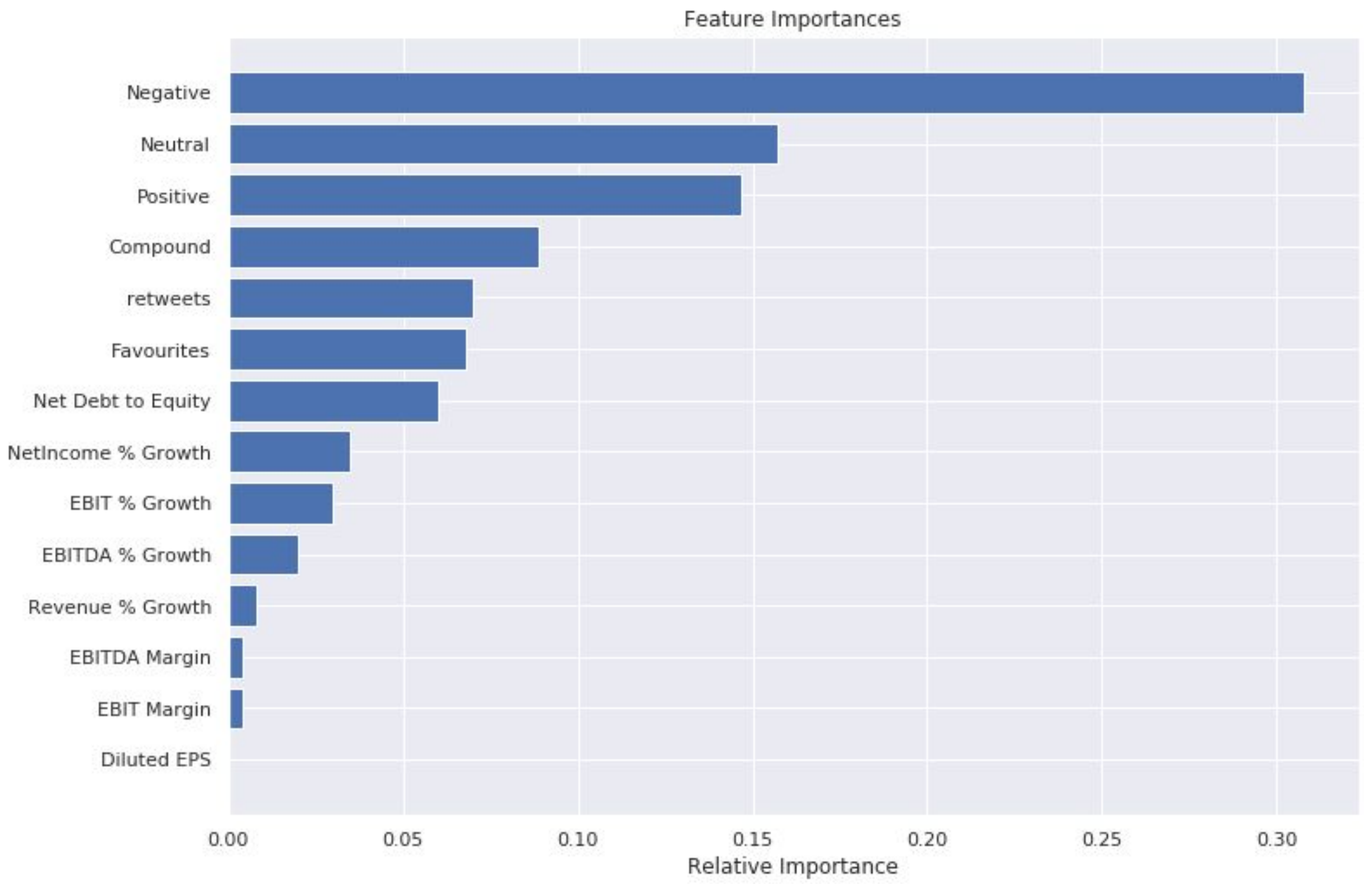
Random Forest - Why?



- Reduces noise
- Less Prone to overfit
- Feature Selection


Decision Tree Vs Random Forest Test and Train R2 Scores
Test
Training
Random Forest
-0.00340
0.00790
Decision Tree
-0.74748
0.99998
-0.00500
0.00000
0.00500

Random Forest - Feature Selection



Model For Backtesting

The hugely overfit Decision Tree will be used as predictor of returns.

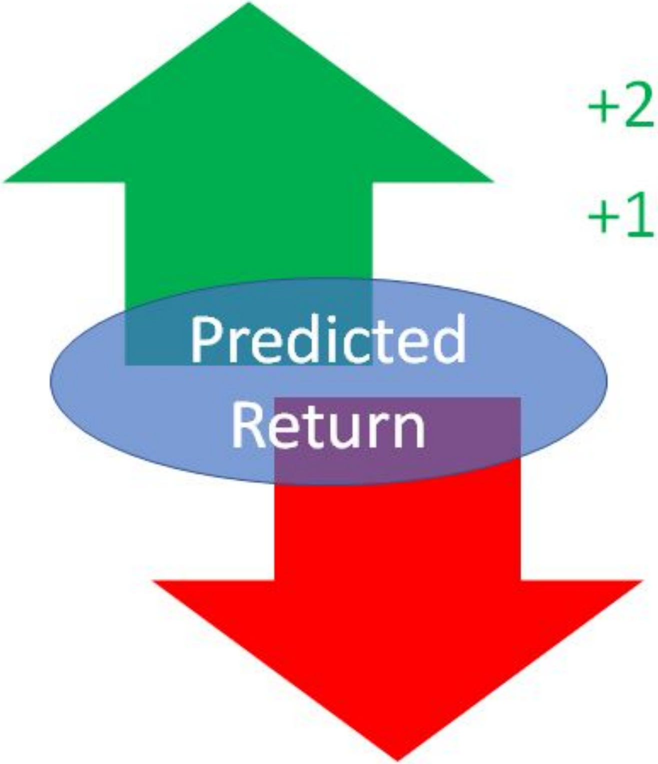
In order to answer the following questions:

- How will the overfitting affect the backtest?
- We will be able to identify it?

Backtesting

Trading Algorithm

- Rebalance according to Target Weight (TW)



+2% Strong Buy → TW +1.0

+1% Medium Buy → TW +0.5

-1% Medium Sell → TW -0.5

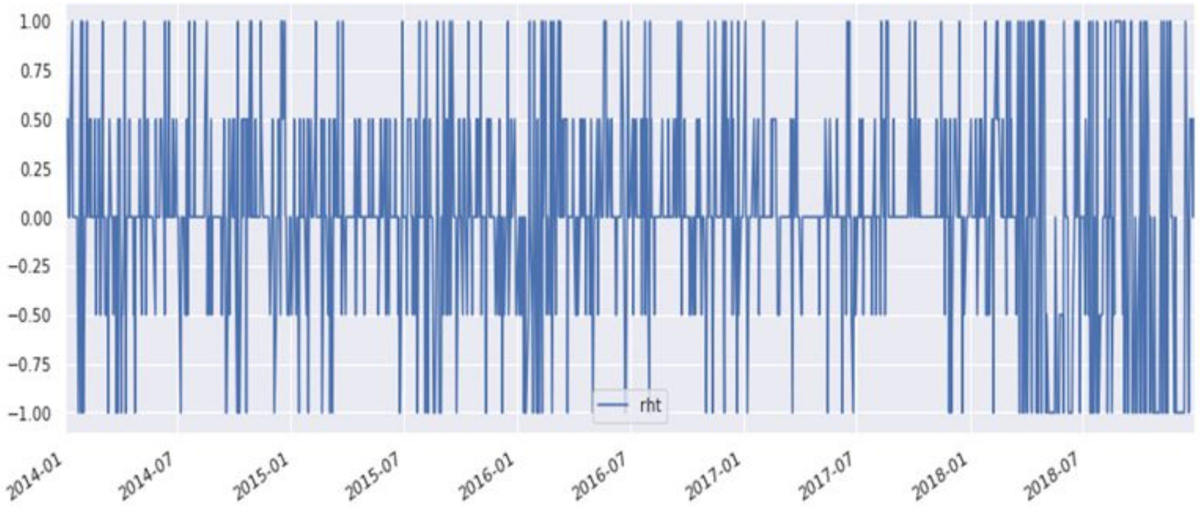
-2% Strong Sell → TW -1.0

Trading Algorithm

- Transaction Data

Date	Predicted Return	Target Weight	Quantity	Price
2014-01-03	1.1%	0.5	8,962	55.78
2014-01-06	1.9%	0.5	-44	56.34
2014-01-07	-0.7%	0	-8,918	57.42
2014-01-10	1.3%	0.5	8,927	56.81
2014-01-13	3.2%	1	8,810	57.56
2014-01-14	-0.3%	0	-17,737	59.38
2014-01-22	-1.4%	-0.5	-8,939	58.92
2014-01-23	-2.4%	-1	-9,304	58.12

- Security Weights



Comparison between Training vs Test

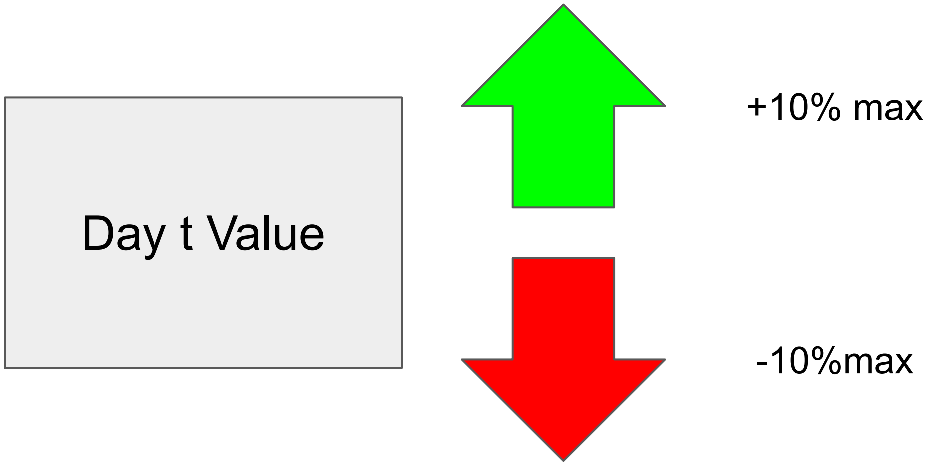
- Total Return & Risk Statistics

Total Return	Sharpe Ratio	Sortino Ratio	CAGR	Max Drawdown
40,323.22%	4.23	5.75	233.08%	-49.75%

Strategy 2: Adding Limit Deltas

Why?

Purpose is to limited the change in portfolio not more than 10% per day by using bt.algos.LimitDeltas()



Strategy 3: Adding Limit Deltas & 1 Day Lag

Why?

Because there is no way that we can know the closing price before the market close.

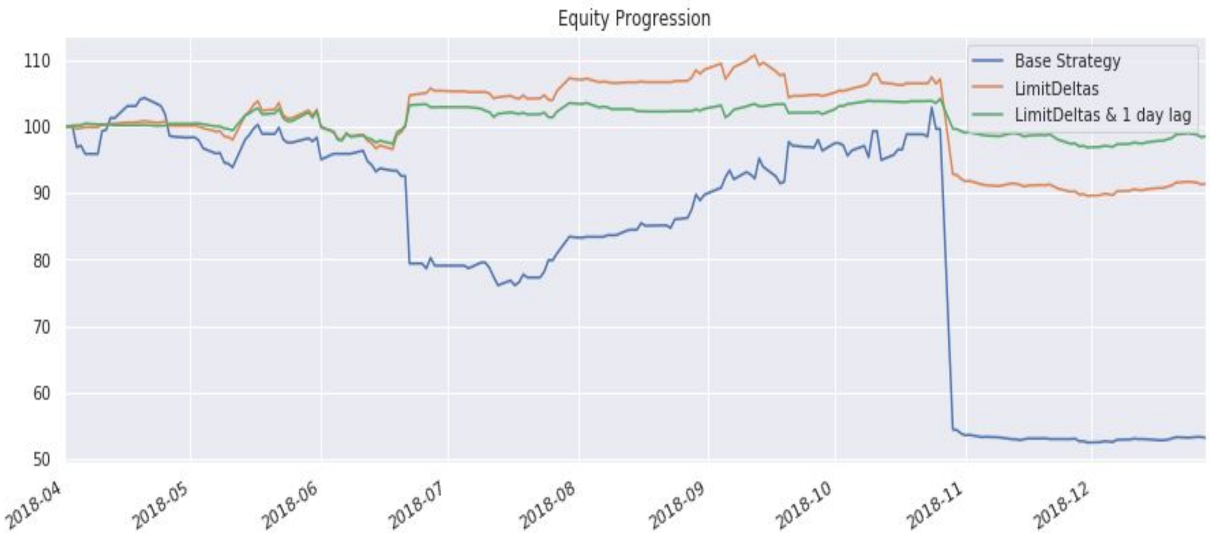
How?

By shifting the return for 1 day

	Predicted Return
2014-01-01	
2014-01-02	1.5%
2014-01-03	0.3%
2014-01-04	-0.8%
2014-01-05	-1.3%

Comparison of Strategies

	Base Strategy	Limit Deltas	Limit Deltas & 1 day lag
Total Return	-46.58%	-8.62%	-1.46%
Sharpe Ratio	-1.04	-0.54	-0.22
Sortino Ratio	-1.10	-0.64	-0.31
CAGR	-57.34%	-11.44%	-1.96%
Max Drawdown	-49.75%	-19.11%	-7.05%



Conclusion

Machine Learning

- ➔ Overfitting: Discrepancy between ML model and Backtesting
- Difference based on the period of training data: ex) ‘2005 – 2007’ vs ‘2008 – 2010’

Algorithmic Trading

- Accuracy of the measurement of sentiment effect
- Appropriateness of the choice and number of factors
- Trading effect on the market: capital size of the model
- Unexpected external events: market regime, competitors, market
- ➔ Vulnerable to lose big & fast

Backtesting

- ➔ Biases - Look ahead, Data Snooping, Shorting

"Lose big & fast"

But, could beat human traders!

Authors

- Diego Giménez
- Piya Thavornwong
- Jorge Betancourt
- Yoonhee Bae
- Myungsung Kim
- Xuan Lu

Thanks to machine-learning algorithms,
the robot apocalypse was short-lived.