ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

Introduction to Momentum Trading

Google Cloud

ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

auquan

Learning Objectives

- Identify the factors that contribute to the momentum feature
- Understand how trade entries and exits are selected using moving averages
- Evaluate and rank the effectiveness moving averages and moving average ribbons

Agenda

Establish a Momentum Trend

Specify Entry and Exit Signals with Moving Averages

Choosing Moving Average Length

Scoring Moving Averages

What is Momentum

What is a Momentum phase?

A momentum phase is a period in which an asset appears to be moving simply due to a reaction to prices rather than due to any stock specific fundamentals or news.

When prices move higher, it is known as a Bull phase and when prices move lower, it is known as a Bear phase.

Bull Momentum

Google Cloud

ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

auquan

What is Momentum

What are Trend Lines?

Trend lines are straight lines that connect the temporary highs and lows in a price series.

Technical analysts use trend lines above and below a series to identify potential buy or sell signals

Volume is a critical component to confirming a momentum trend.

Google Cloud

ESTABLISHED 1922 NEW YORK INSTITUTE OF FINANCE

auquan

What is Momentum

How do You Detect a Trend?

It is sometimes hard to spot a trend in a forest of price pars.

The slope of **rolling averages** or **moving averages** can tell you whether a stock is in an uptrend or downtrend

Bull Momentum

Google Cloud

ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

auquan

Agenda

Establish a Momentum Trend

Specify Entry and Exit Signals with Moving Averages

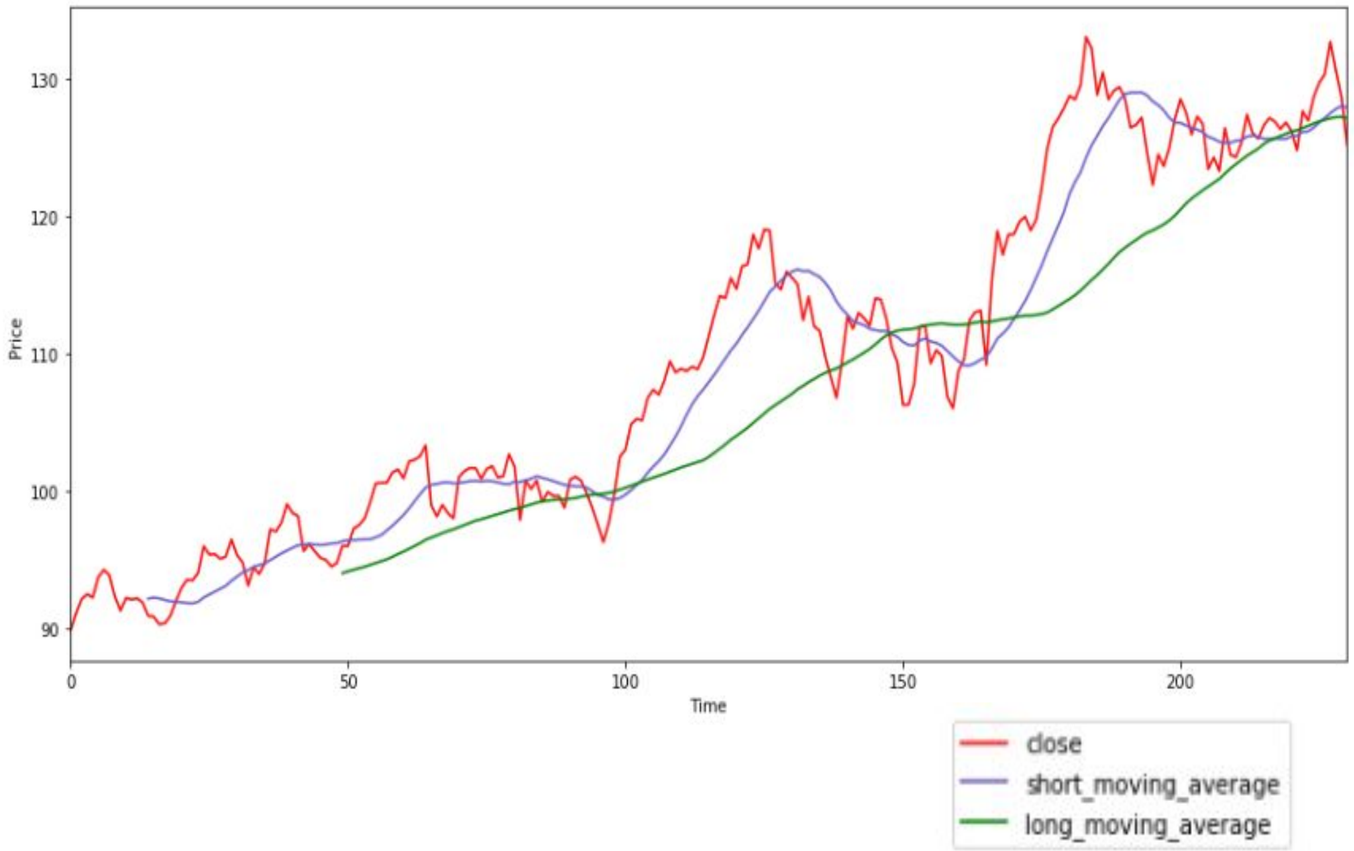
Choosing Moving Average Length

Scoring Moving Averages

Measuring Momentum

Moving Averages

Moving averages can be used to see stock price trends that may not be apparent on a OHLC bar graph.



Google Cloud

ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

auquan

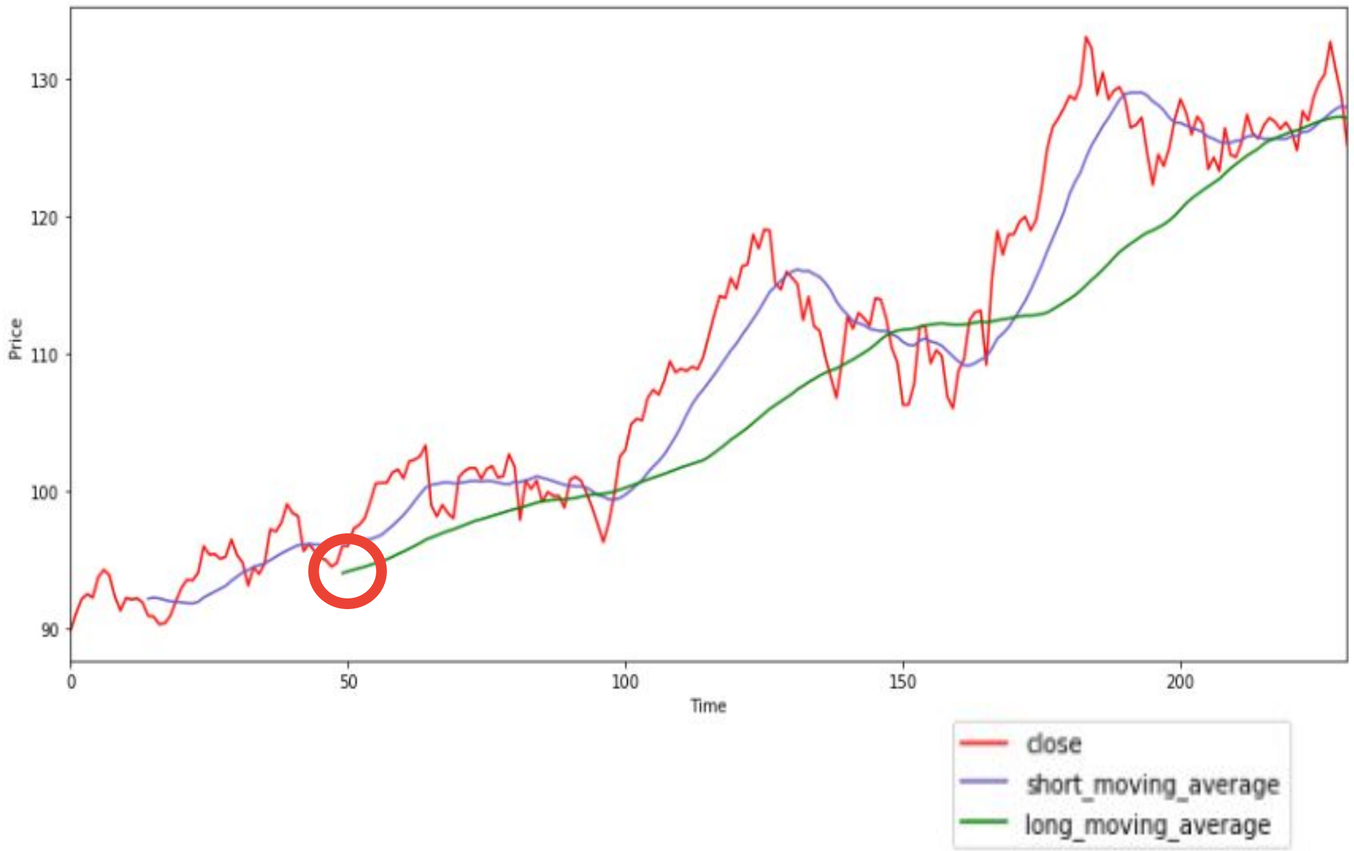
Measuring Momentum

Simple Moving Averages

50-d MA = $[\Sigma^{50}P_{CLOSE}] / 50$

Here you see the closing price plotted along with a 10-day and 50-day simple moving average.

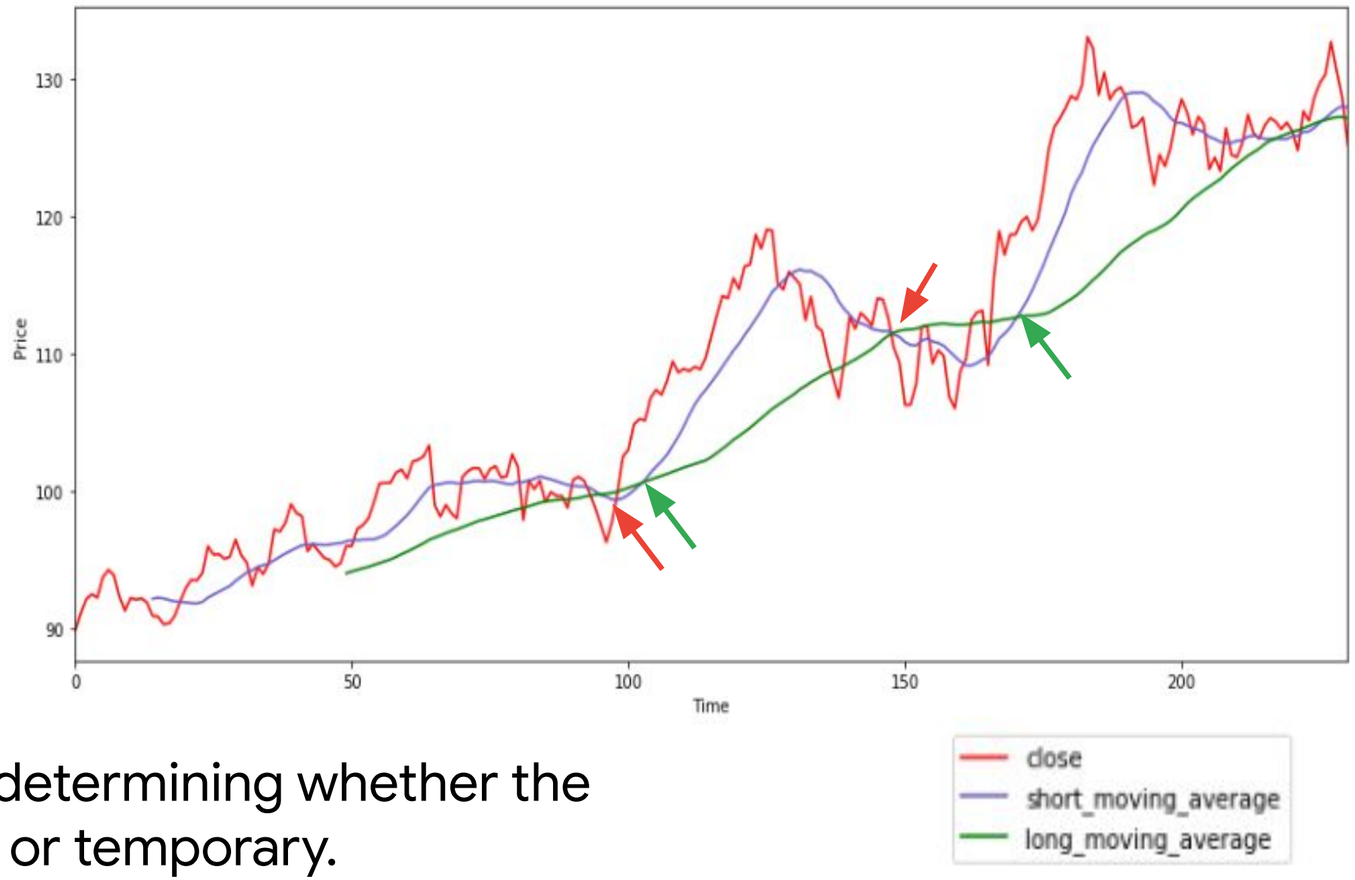
Each closing price in the window is equally weighted as opposed to an exponential average where more recent prices are given a heavier weight



Entry and Exit Signals

Moving Average Crossovers

A very common way to obtain a momentum signal is to look for moving average crossovers. This means computing two moving averages of different lengths, and waiting for one to cross the other. The direction of the cross will indicate the direction of the momentum.



Volume is a critical component of determining whether the crossover is a real change in trend or temporary.

Entry and Exit Signals

Moving Average Crossovers

A very common way to obtain a momentum signal is to look for moving average crossovers. This means computing two moving averages of different lengths, and waiting for one to cross the other. The direction of the cross will indicate the direction of the momentum.

Volume is a critical component of determining whether the crossover is a real change in trend or temporary.

Agenda

Establish a Momentum Trend

Specify Entry and Exit Signals with Moving Averages

Choosing Moving Average Length

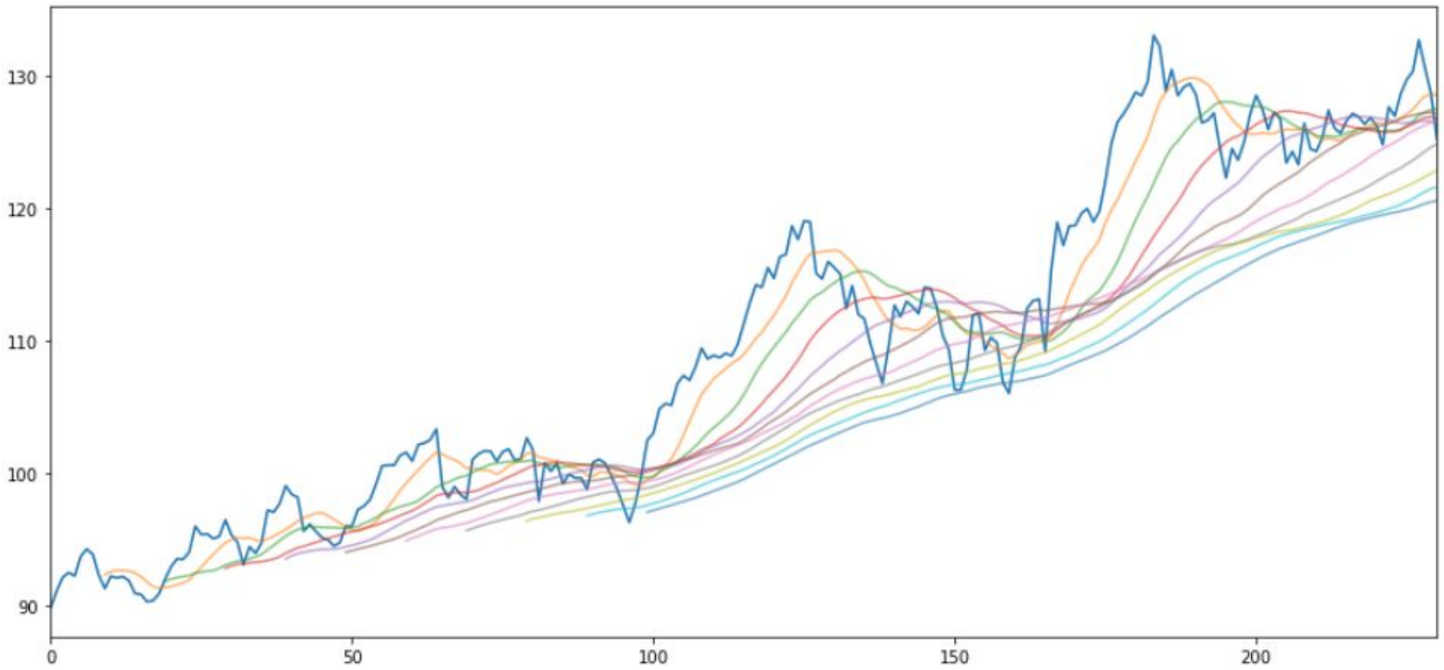
Scoring Moving Averages

Choosing Moving Averages

Dangers of Overfitting

Choosing Moving Average Lengths is very complicated. There are many choices of lengths such as 20, 30, 50, 65, 200, etc. In fact, any integer is a possible choice.

So you need to be aware of the dangers of Overfitting.



Google Cloud

ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

auquan

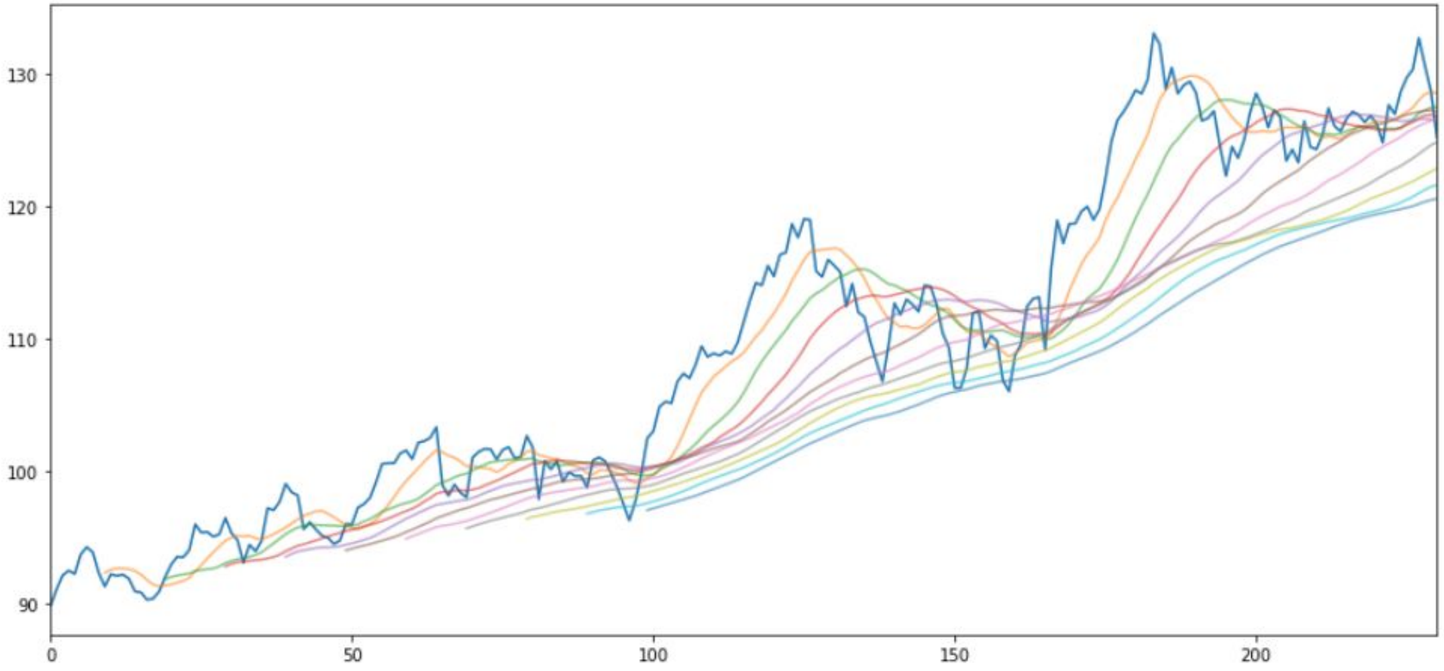
http://theory.stanford.edu/~sergei/slides/www10-metrics.pdf

Choosing Moving Averages

Dangers of overfitting

The choice of lengths will strongly affect the signal that you receive from your moving average crossover strategy. There may be better windows, and attempts to find them can be made with robust optimization techniques.

However, it is incredibly easy to overfit your moving window lengths.



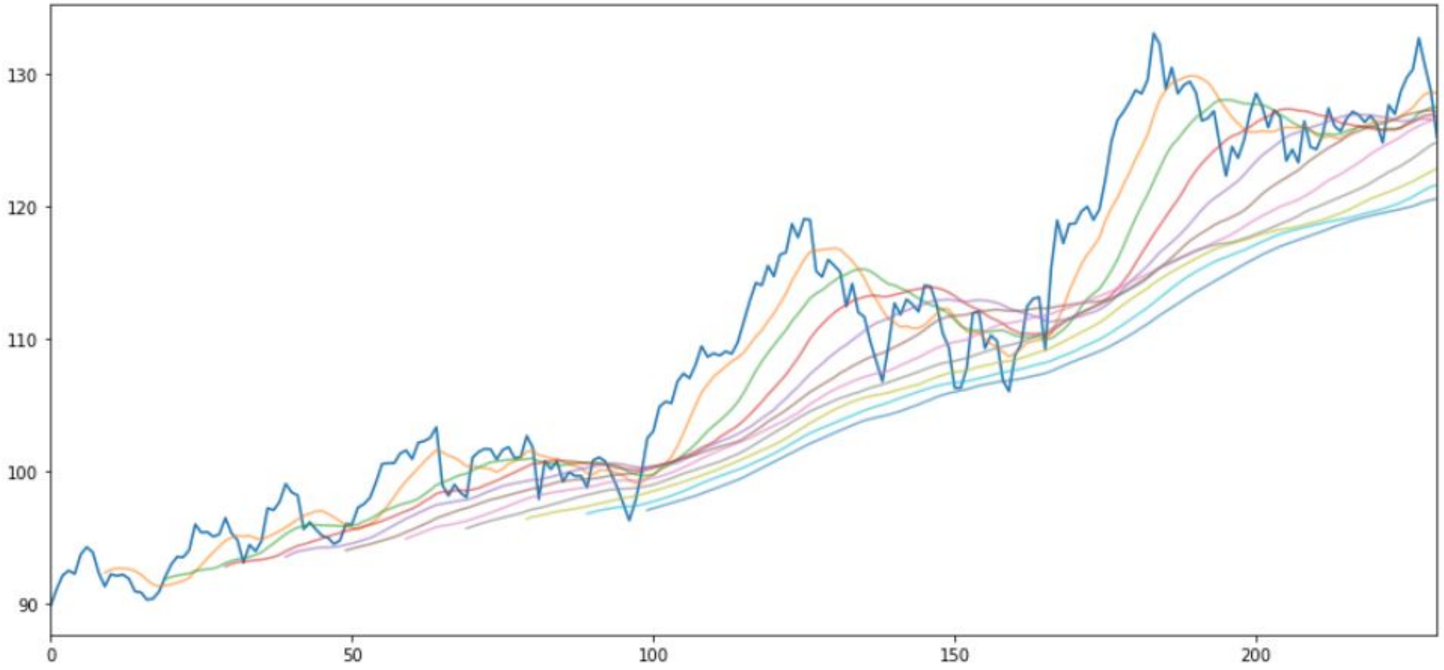
http://theory.stanford.edu/~sergei/slides/www10-metrics.pdf

Choosing Moving Averages

Dangers of overfitting

One approach is to use Moving Average Crossover Ribbons: draw many moving averages at a time, and attempt to extract statistics from the shape of the 'ribbon' rather than any two moving averages.

You can see how ribbons look like here. You can combine quantitative measures of ribbon shape into giving us a trading signal



Google Cloud

ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

auquan

http://theory.stanford.edu/~sergei/slides/www10-metrics.pdf

Agenda

Establish a Momentum Trend

Specify Entry and Exit Signals with Moving Averages

Choosing Moving Average Length

Scoring Moving Averages

Scoring Moving Averages; Distance Metric

Scoring

We take the rolling means and rank them from 1-10 and normalize to a score of 0 to 1.

10dMA < 20dMA <... 100dMA
⇒ **Score = 0**

10dMA > 20dMA>... 100dMA
⇒ **Score = 1**

A Score close to **0** means a **BUY** signal and a score close to **1** is a **SELL** signal.

Google Cloud
ESTABLISHED 1922 NEW YORK INSTITUTE OF FINANCE
auquan

Scoring Moving Averages; Distance Metric

Scoring

After ranking the rolling means and normalizing the scores and we plot them on the price chart.

We see a number of strong sell signals and two buy signals towards the end of the data.

Let's take a look at what happens after that Buy signal in the end!

Scoring Moving Averages

Scoring

When you look at what happens after the Buy signal from that previous chart, you see that the stock price rises for a bit and then drops by about 20-30% after the next few months.

So you have to be constantly on the lookout for better signals.

Not a great trading signal!

Google Cloud ESTABLISHED 1922 NEW YORK INSTITUTE OF FINANCE auquan

Scoring Moving Averages

Correlation

Another slightly better way may be to use correlation to rank the ribbons.

First we rank the ribbons using a Spearman's correlation and normalize the score once again.

A Score close to 1 means a buy signal and a score close to -1 is a sell signal

This is somewhat better since it triggered 3 sell signals pretty close to the short term top of the stock. This may be worth exploring some more!

Google Cloud ESTABLISHED 1922 NEW YORK INSTITUTE OF FINANCE auquan

Google Cloud

Building a Momentum Trading Model

Lesson made in partnership with **auquan**

Google Cloud

auquan

Agenda

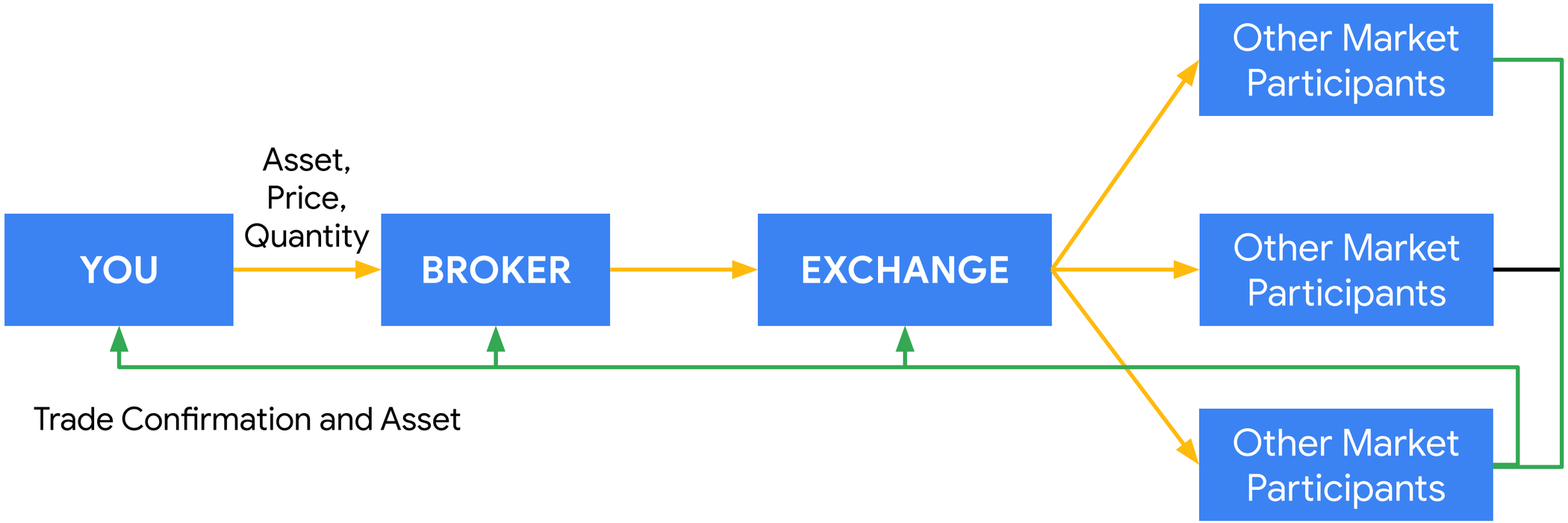
Elements of a Trading System

Developing a Trading Strategy using ML

Building a Momentum Trading Model using ML

What are the elements of a trading system?

In order to simulate Algorithmic trading on our computers or on the Cloud, there are several critical components. First, we need to simulate an exchange or connect to an exchange to trade.



Google Cloud auquan

What do we need to build and test a trading system?

Components of a trading system

- *Trading Strategy:* Decide which markets you want to trade, develop the logic with which you want to trade them and define parameters by which the trading logic is triggered. — You
- *Backtesting system:* Analyze your strategy's performance on historical data and remove biases — Toolbox
- *Execution system:* Link to a brokerage and minimize the transaction costs (cost of trading, commission, and slippage) — Toolbox
- *Risk management system:* Create pre and post trade checks to avoid losses — You

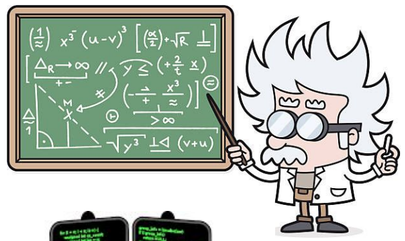
Google Cloud auquan

What do we need to build and test a trading system?

Components of a trading system

Auquan Toolbox *(pronounced "Awe-Qwan")*:

- Open source library
- Simulates an Exchange
- Backtests your algorithm
- Executes your trades
- Calculates your P&L
- Allows you to back test it any time frame you want



auquan

- Data sources
- Backtesting engine
- Execution engine (simulated)
- Calculate PnL
- Display Charts

Google Cloud auquan

Agenda

Elements of a Trading System

Developing a Trading Strategy using ML

Building a Momentum Trading Model using ML

Google Cloud auquan

Developing a profitable trading strategy is more than just finding a good idea

Google Cloud auquan

How to create an ML trading strategy

1. Create features from data:
    - Fundamental
        - P/E ratios, EPS, Cash Flows, etc.
    - Technical based on how your chosen securities behave:
        - Momentum features, Mean Reversion features
        - Correlated or Cointegrated features
    - Combined fundamental and technical indicators
2. Develop a trading strategy using these features:
    - Which instruments are a buy, a sell, or neutral,
    - When to buy/sell and
    - How much to buy/sell

Remember, it is better to lose funny money than real money.

Test!

Test!

Test!

Agenda

Elements of a Trading System

Developing a Trading Strategy using ML

Building a Momentum Trading Model using ML

Google Cloud auquan

Momentum Training Model

Step 1: Define the problem

Step 2: Collect the Data

Step 3: Creating Features

Step 4: Split the Data

Step 5: Select a Machine Learning Algorithm

Step 6: Backtest on Unseen Data

How to build a Trading System?

Developing a trading algorithm system consists of several elements:

- *Identify a strategy:* Decide which markets you want to trade, develop the logic with which you want to trade them and define parameters by which the trading logic is triggered — Auquan
- *Backtest your strategy:* Analyze your strategy's performance on historical data and remove biases — Auquan
- *Execute your strategy:* Link to a brokerage and minimize the transaction costs (cost of trading, commission, and slippage) — You
- *Manage Risk:* Create pre and post trade checks to avoid losses — You

What problem are you solving?

- Price Prediction
- Return/Pnl
- Buy/Sell Signal
- Portfolio Optimization
- Efficient Execution

What problem are you solving?

- Price Prediction
- Return/Pnl
- Buy/Sell Signal
- Portfolio Optimization
- Efficient Execution

Google Cloud auquan

What problem are you solving?

- Price Prediction
- Return/Pnl
- Buy/Sell Signal
- Portfolio Optimization
- Efficient Execution

What problem are you solving?

- Price Prediction
- Return/Pnl
- Buy/Sell Signal
- Portfolio Optimization
- Efficient Execution

What problem are you solving?

- Price Prediction
- Return/Pnl
- Buy/Sell Signal
- Portfolio Optimization
- Efficient Execution

What problem are you solving?

- Price Prediction
- Return/Pnl
- Buy/Sell Signal
- Portfolio Optimization
- Efficient Execution

What problem are you solving?

- Price Prediction
- Return/Pnl
- Buy/Sell Signal
- Portfolio Optimization
- Efficient Execution

Introducing Auquan

Auquan

- Pronounced “Awe Qwan”)
- Produce and maintain the Auquan Toolbox for Backtesting for trading strategies

Auquan Toolbox

- Open source library that will allow us to implement our momentum trading strategy
- Allows us to write minimal code to test our trading strategies and best of all, it is open source!

“Auquan is building the world’s first Data Science and Machine Learning platform for financial services.”

As well as their Toolbox, Auquan also host competitions and practice resources to help people learn to implement their data science skills on real problems sourced from top investment firms*

* From Auquan.com

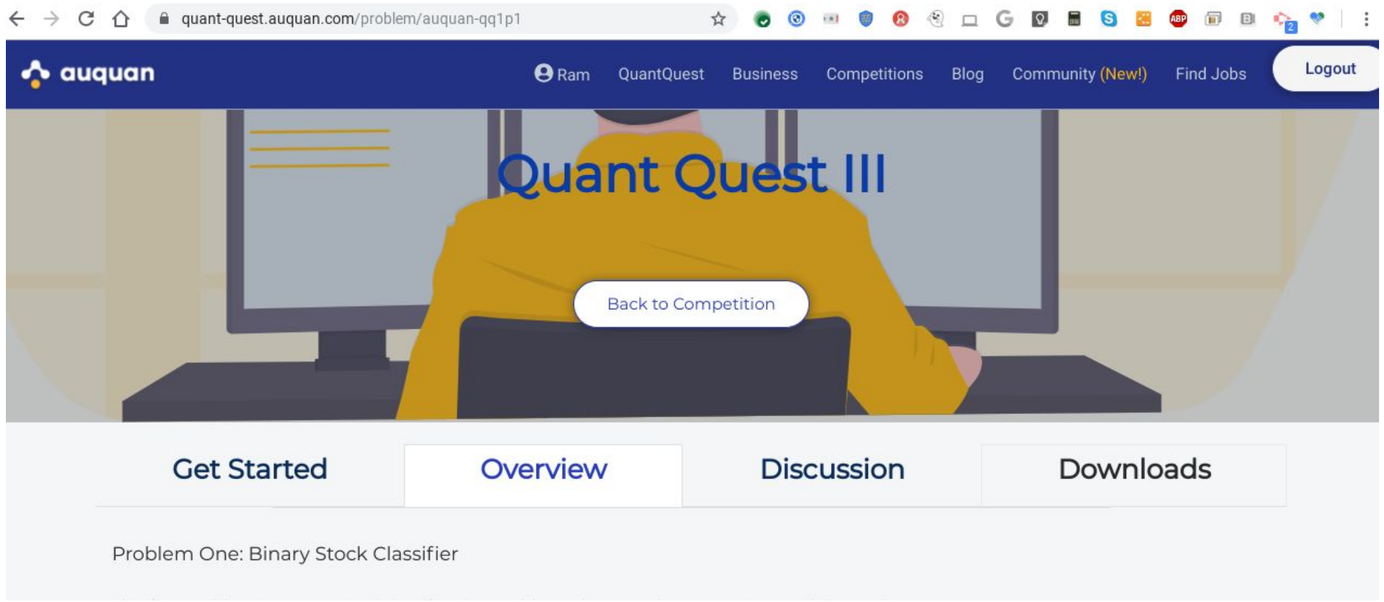
What problem are we solving?

We're going to use a problem from Auquan's web site called QuantQuest.

We are going to create a prediction model that predicts future expected value of basis using Machine Learning.

Problem can be found in following link (after registration):

https://quant-quest.auquan.com/problem/auquan-qq1p1



Google Cloud auquan

Current Problem

Problem: Create a prediction model that can help trade price difference between two assets

$$basis = Price\ of\ Stock - Price\ of\ Future$$

$$basis_t = S_t - F_t$$

Target Variable

Target: Expected Value in the next 5 minutes

$$Y = future\ expected\ value\ of$$

$$basis = Average(\mathbf{basis}_{(t1)},$$

$$\ldots \mathbf{basis}_{(t5)})$$

Objective: Create a model so that predicted value is as close as possible to Y

Evaluation Criteria: RMSE. We'll also use Total PnL as an evaluation criterion

Features (X)

Stock data at one-minute intervals for trading days over one month (~8000 data points):

- Stock Bid Price, Ask Price, Bid Volume, Ask Volume
- Future Bid Price, Ask Price, Bid Volume, Ask Volume
- Stock VWAP
- Future VWAP

*StockVWAP = BidPrice * AskVolume + AskPrice * BidVolume/(AskVolume + BidVolume)*

Data is already cleaned for Dividends, Splits, Rolls

Features (X)

Stock data at one-minute intervals for trading days over one month (~8000 data points):

- Stock Bid Price, Ask Price, Bid Volume, Ask Volume
- Future Bid Price, Ask Price, Bid Volume, Ask Volume
- Stock VWAP
- Future VWAP

*StockVWAP = BidPrice * AskVolume + AskPrice * BidVolume/(AskVolume + BidVolume)*

Data is already cleaned for Dividends, Splits, Rolls

Features (X)

Stock data at one-minute intervals for trading days over one month (~8000 data points):

- Stock Bid Price, Ask Price, Bid Volume, Ask Volume
- Future Bid Price, Ask Price, Bid Volume, Ask Volume
- Stock VWAP
- Future VWAP

*StockVWAP = BidPrice * AskVolume + AskPrice * BidVolume/(AskVolume + BidVolume)*

Data is already cleaned for Dividends, Splits, Rolls

Features (X)

Stock data at one-minute intervals for trading days over one month (~8000 data points):

- Stock Bid Price, Ask Price, Bid Volume, Ask Volume
- Future Bid Price, Ask Price, Bid Volume, Ask Volume
- Stock VWAP
- Future VWAP

*StockVWAP = BidPrice * AskVolume + AskPrice * BidVolume/(AskVolume + BidVolume)*

Data is already cleaned for Dividends, Splits, Rolls

Features (X)

Stock data at one-minute intervals for trading days over one month (~8000 data points):

- Stock Bid Price, Ask Price, Bid Volume, Ask Volume
- Future Bid Price, Ask Price, Bid Volume, Ask Volume
- Stock VWAP
- Future VWAP

*StockVWAP = BidPrice * AskVolume + AskPrice * BidVolume/(AskVolume + BidVolume)*

Data is already cleaned for Dividends, Splits, Rolls

Features (X)

Stock data at one-minute intervals for trading days over one month (~8000 data points):

- Stock Bid Price, Ask Price, Bid Volume, Ask Volume
- Future Bid Price, Ask Price, Bid Volume, Ask Volume
- Stock VWAP
- Future VWAP

  *StockVWAP = BidPrice * AskVolume + AskPrice * BidVolume/(AskVolume + BidVolume)*

Data is already cleaned for Dividends, Splits, Rolls

Features (X)

Stock data at one-minute intervals for trading days over one month (~8000 data points):

- Stock Bid Price, Ask Price, Bid Volume, Ask Volume
- Future Bid Price, Ask Price, Bid Volume, Ask Volume
- Stock VWAP
- Future VWAP

*StockVWAP = BidPrice * AskVolume + AskPrice * BidVolume/(AskVolume + BidVolume)*

Data is already cleaned for Dividends, Splits, Rolls

Sample Data

	Stock Price	Future Price	Basis	Y(Target
2017-01-06 09:17:00	777.150	769.025	5.425	5.410
2017-01-06 09:18:00	772.700	766.075	4.425	5.605
2017-01-06 09:19:00	774.425	767.000	4.950	5.705
2017-01-06 09:20:00	779.625	771.175	5.650	5.610
2017-01-06 09:21:00	780.750	771.050	6.400	5.340
2017-01-06 09:22:00	787.100	778.675	5.625	5.260
2017-01-06 09:23:00	787.900	779.750	5.400	5.315
2017-01-06 09:24:00	787.950	779.725	5.450	5.425
2017-01-06 09:25:00	787.825	779.975	5.175	5.545
2017-01-06 09:26:00	791.075	783.450	5.050	5.595

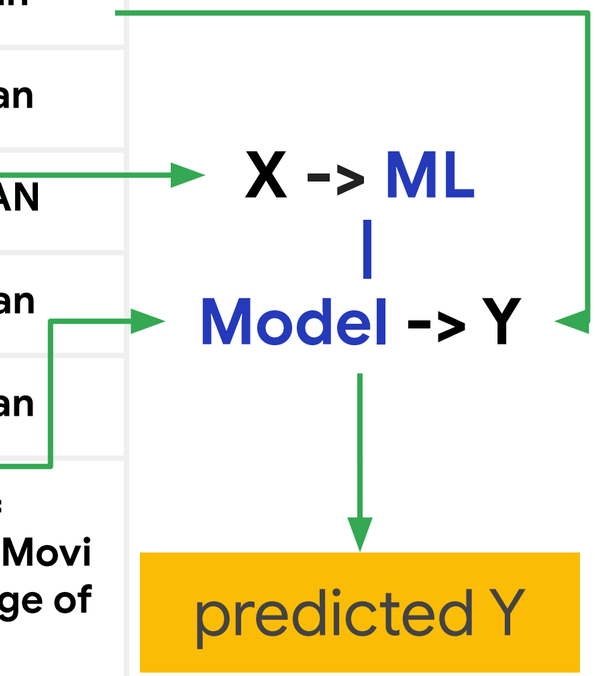
Target (Y)

HISTORICAL DATA

time

	Feature 1 (X1)	Feature 2 (X2)	Feature 3 (X3)	...	Target (Y)
t1	.	.	.		y1=nan
t2	.	.	.		y2=Nan
t3	.	.	.		y3=NAN
T4	.	.	.		y4=Nan
T5	.	.	.		y5=Nan
T6	.	.	.		Y6 = 5-periodMoving average of X1

what is Y?
To create Y, we are going to calculate 5-period moving average of Feature x1.



Google Cloud auquan

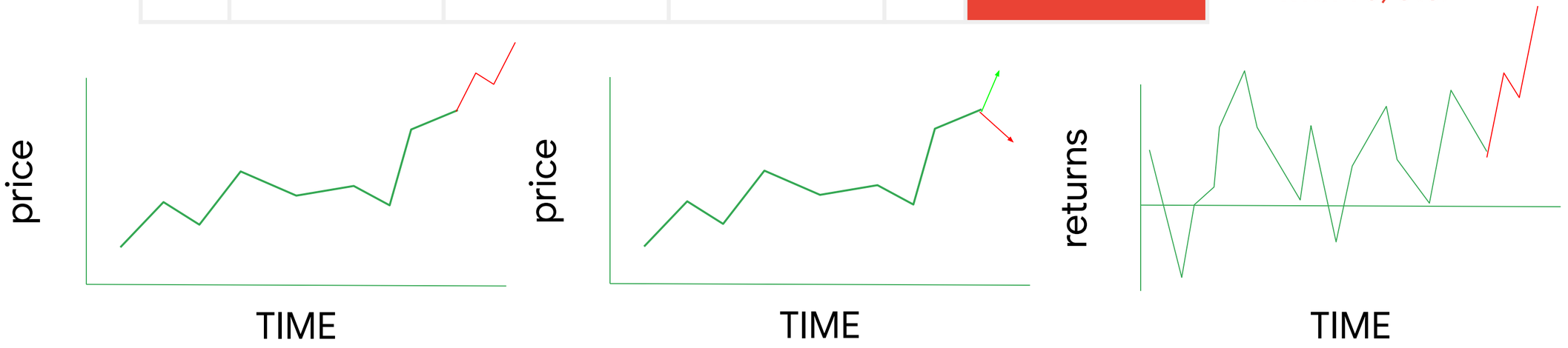
How to Align calculated Y with X

HISTORICAL DATA

time

	Feature 1 (X1)	Feature 2 (X2)	Feature 3 (X3)	...	Target (Y)
t1	.	.	.		Y6
t2	.	.	.		Y7
t3	.	.	.		Y8

What is Y?
To create Y, we are going to shift the future values upward ie. align T1 with Y6, etc.



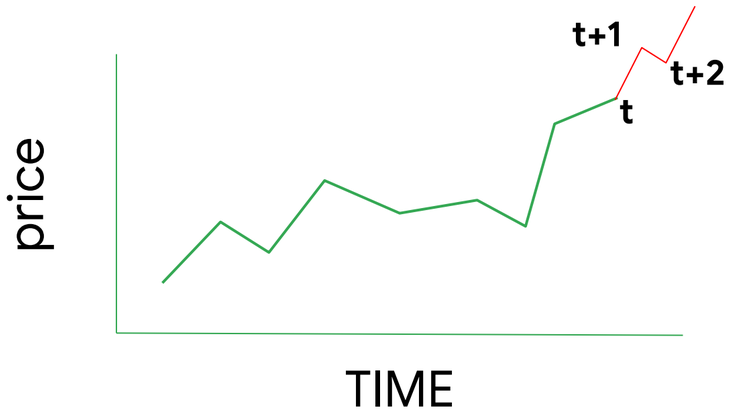
Google Cloud auquan

How do we evaluate the model?

HISTORICAL DATA

time

	Feature 1 (X1)	Feature 2 (X2)	Feature 3 (X3)	...	Target (Y)
t1	.	.	.		$Y1 = P_{t+1}$
t2	.	.	.		$Y2 = P_{t+2}$
t3	.	.	.		$Y3 = P_{t+3}$



how will you evaluate the model?

$$rmse = \sqrt{\frac{\Sigma(Y_{actual} - Y_{predicted})^2}{\text{number of samples}}}$$

Google Cloud auquan

Momentum Training Model

Step 1: Define the problem

Step 2: Collect the Data

Step 3: Creating Features

Step 4: Split the Data

Step 5: Select a Machine Learning Algorithm

Step 6: Backtest on Unseen Data

Data Collection

What data will have predictive power?

- Stock Price Data, Trade Volume Data
- Fundamental Data, Overall Market indicator (ex: Indexes)
- Correlated stocks data
- Similar assets price information

Check for accuracy, alignment

Clean for dividends, stock splits, rolls, etc.

Alternative Data

Alternative Data is Untapped Data

- Credit card transactions
- Email receipts
- Point-of-sale transactions
- Web site usage
- Obscure city hall records
- Satellite images
- Social media posts
- Online browsing activity
- Shipping container receipts
- Product reviews

Momentum Training Model

Step 1: Define the problem

Step 2: Collect the Data

Step 3: Creating Features

Step 4: Split the Data

Step 5: Select a Machine Learning Algorithm

Step 6: Backtest on Unseen Data

Feature Selection

Golden rule of feature selection
“predictive power should come primarily from the features and not from the model itself”

Feature Selection

Choice of features has a far greater impact on performance than choice of model

- Don’t randomly choose a very large set of features without exploring relationship with target variable
- Little to no relationship with target variable will lead to overfitting
- Rank candidate features according to Maximal Information Coefficient (MIC) or performing Principal Component Analysis (PCA) and other methods

Feature Selection

Choice of features has a far greater impact on performance than choice of model

- Don’t randomly choose a very large set of features without exploring relationship with target variable
- Little to no relationship with target variable will lead to overfitting
- Rank candidate features according to Maximal Information Coefficient (MIC) or performing Principal Component Analysis (PCA) and other methods

Feature Selection

Choice of features has a far greater impact on performance than choice of model

- Don’t randomly choose a very large set of features without exploring relationship with target variable
- Little to no relationship with target variable will lead to overfitting
- Rank candidate features according to Maximal Information Coefficient (MIC) or performing Principal Component Analysis (PCA) and other methods

Feature Selection

Choice of features has a far greater impact on performance than choice of model

- Don’t randomly choose a very large set of features without exploring relationship with target variable
- Little to no relationship with target variable will lead to overfitting
- Rank candidate features according to Maximal Information Coefficient (MIC) or performing Principal Component Analysis (PCA) and other methods

Feature Transform/Normalize

Normalization is tricky because future range of data is unknown.

- *Scaling:* Divide features by rolling standard deviation over a sample range
- *Centering:* Subtract rolling mean from current value
- *Normalization:* Both of the above (x-mean)/stdev over lookback period
- *Regular normalization:* Standardize data to the range -1 to +1 over lookback period (x-min)/(max-min) and re-center

Caveat: Since we are using historical mean, standard deviation, max or min over lookback period, the same normalized value of feature will mean different actual value at different times

Feature Transform/Normalize

Normalization is tricky because future range of data is unknown.

- *Scaling:* Divide features by rolling standard deviation over a sample range
- *Centering:* Subtract rolling mean from current value
- *Normalization:* Both of the above (x-mean)/stdev over lookback period
- *Regular normalization:* Standardize data to the range -1 to +1 over lookback period (x-min)/(max-min) and re-center

Caveat: Since we are using historical mean, standard deviation, max or min over lookback period, the same normalized value of feature will mean different actual value at different times

Feature Transform/Normalize

Normalization is tricky because future range of data is unknown.

- *Scaling:* Divide features by rolling standard deviation over a sample range
- *Centering:* Subtract rolling mean from current value
- *Normalization:* Both of the above (x-mean)/stdev over lookback period
- *Regular normalization:* Standardize data to the range -1 to +1 over lookback period (x-min)/(max-min) and re-center

Caveat: Since we are using historical mean, standard deviation, max or min over lookback period, the same normalized value of feature will mean different actual value at different times

Feature Transform/Normalize

Normalization is tricky because future range of data is unknown.

- *Scaling:* Divide features by rolling standard deviation over a sample range
- *Centering:* Subtract rolling mean from current value
- *Normalization:* Both of the above (x-mean)/stdev over lookback period
- *Regular normalization:* Standardize data to the range -1 to +1 over lookback period (x-min)/(max-min) and re-center

Caveat: Since we are using historical mean, standard deviation, max or min over lookback period, the same normalized value of feature will mean different actual value at different times

Feature Transform/Normalize

Normalization is tricky because future range of data is unknown.

- *Scaling:* Divide features by rolling standard deviation over a sample range
- *Centering:* Subtract rolling mean from current value
- *Normalization:* Both of the above (x-mean)/stdev over lookback period
- *Regular normalization:* Standardize data to the range -1 to +1 over lookback period (x-min)/(max-min) and re-center

Caveat: Since we are using historical mean, standard deviation, max or min over lookback period, the same normalized value of feature will mean different actual value at different times

Feature Transform/Normalize

Normalization is tricky because future range of data is unknown.

- *Scaling:* Divide features by rolling standard deviation over a sample range
- *Centering:* Subtract rolling mean from current value
- *Normalization:* Both of the above (x-mean)/stdev over lookback period
- *Regular normalization:* Standardize data to the range -1 to +1 over lookback period (x-min)/(max-min) and re-center

Caveat: Since we are using historical mean, standard deviation, max or min over lookback period, the same normalized value of feature will mean different actual value at different times

Feature Transform/Normalize

Normalization is tricky because future range of data is unknown.

- *Scaling:* Divide features by rolling standard deviation over a sample range
- *Centering:* Subtract rolling mean from current value
- *Normalization:* Both of the above (x-mean)/stdev over lookback period
- *Regular normalization:* Standardize data to the range -1 to +1 over lookback period (x-min)/(max-min) and re-center

Caveat: Since we are using historical mean, standard deviation, max or min over lookback period, the same normalized value of feature will mean different actual value at different times

Momentum Training Model

Step 1: Define the problem

Step 2: Collect the Data

Step 3: Creating Features

Step 4: Split the Data

Step 5: Select a Machine Learning Algorithm

Step 6: Backtest on Unseen Data

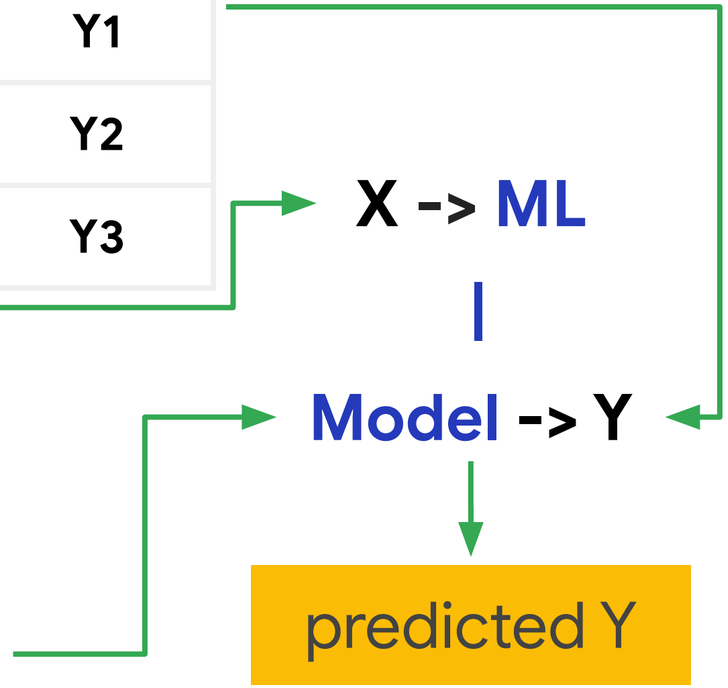
Sample Data

HISTORICAL DATA

time

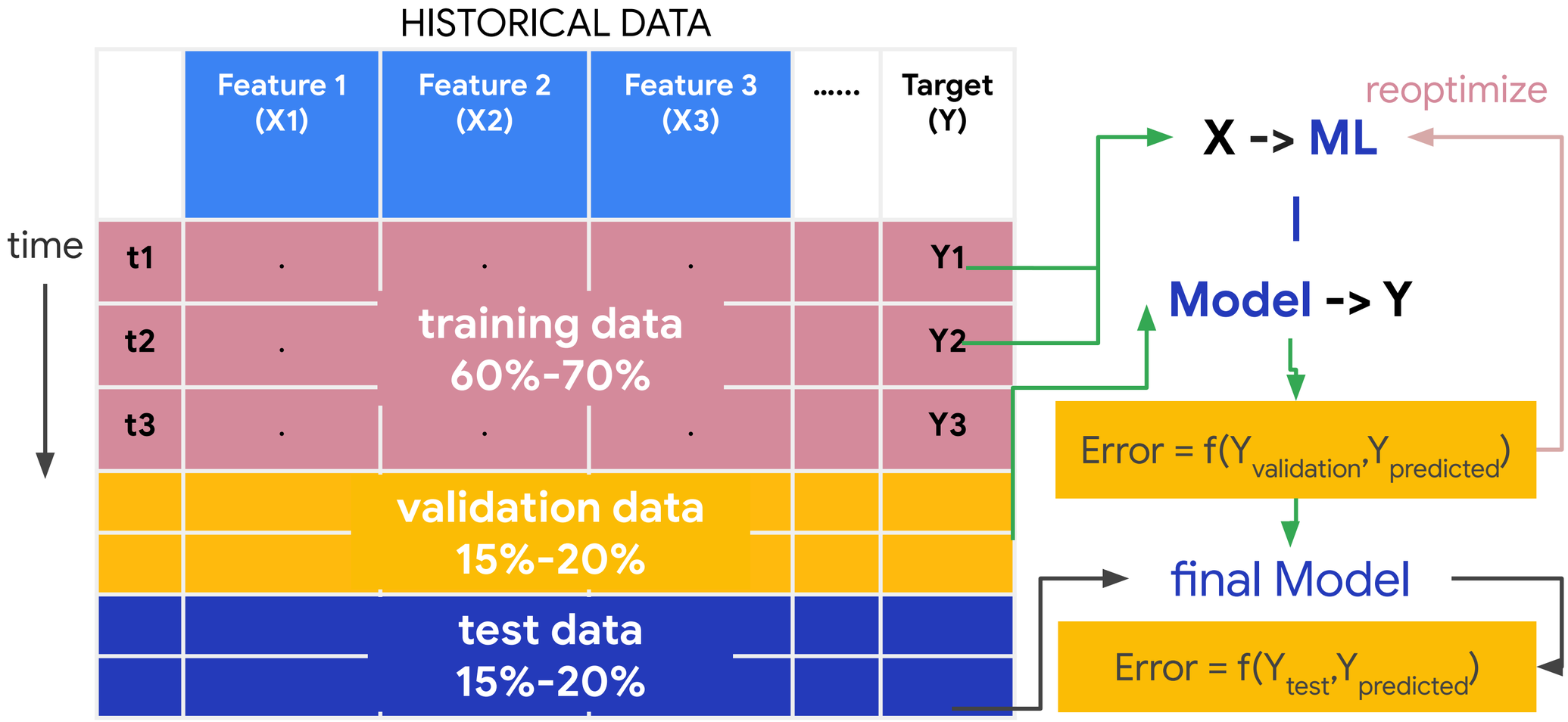
	Feature 1 (X1)	Feature 2 (X2)	Feature 3 (X3)	...	Target (Y)
t1	.	.	.		Y1
t2	.	.	.		Y2
t3	.	.	.		Y3

	(X1)	(X2)	(X3)	...
t10	.	.	.	
t11	.	.	.	



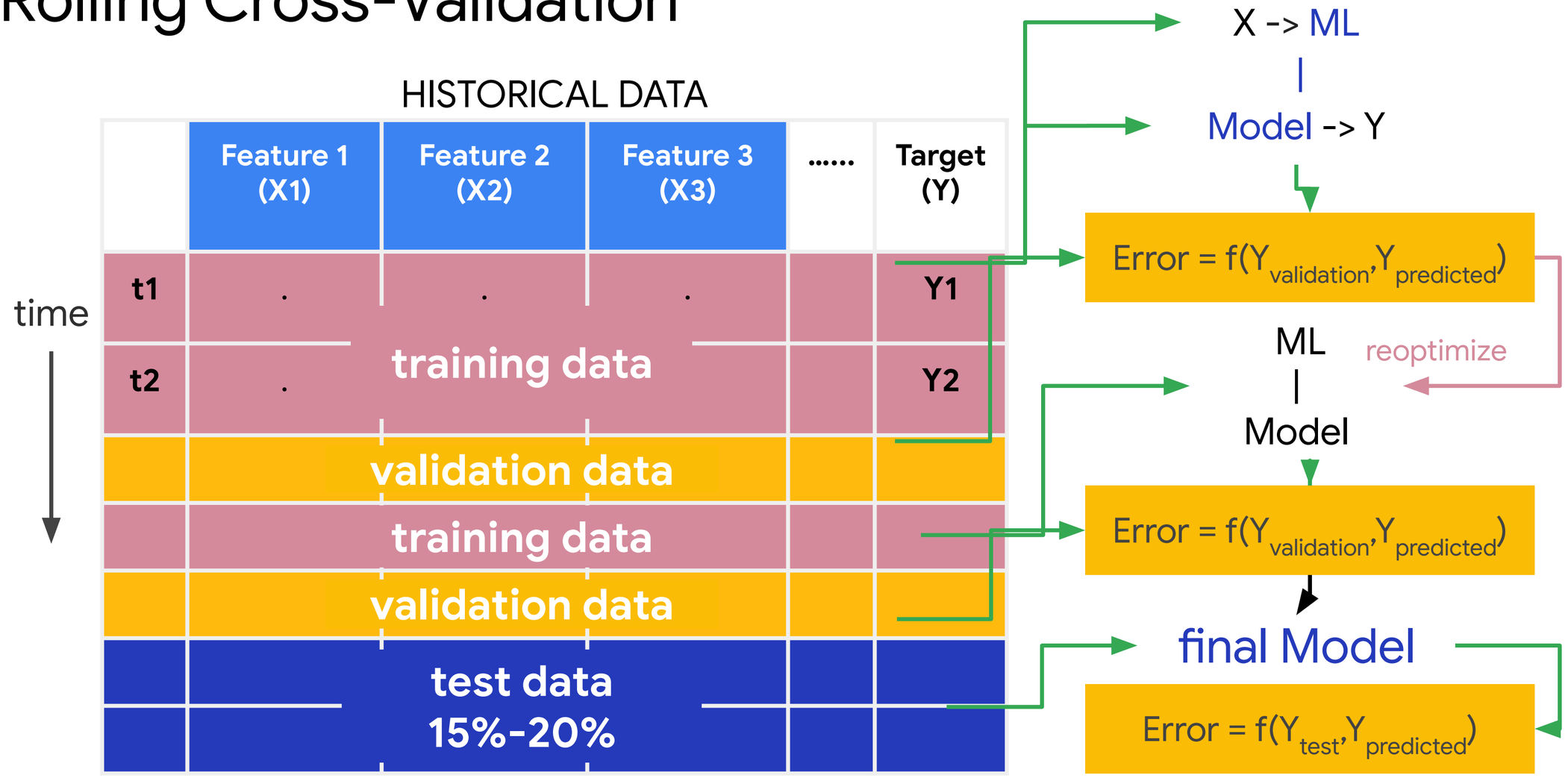
Google Cloud auquan

Splitting the Data into three sets



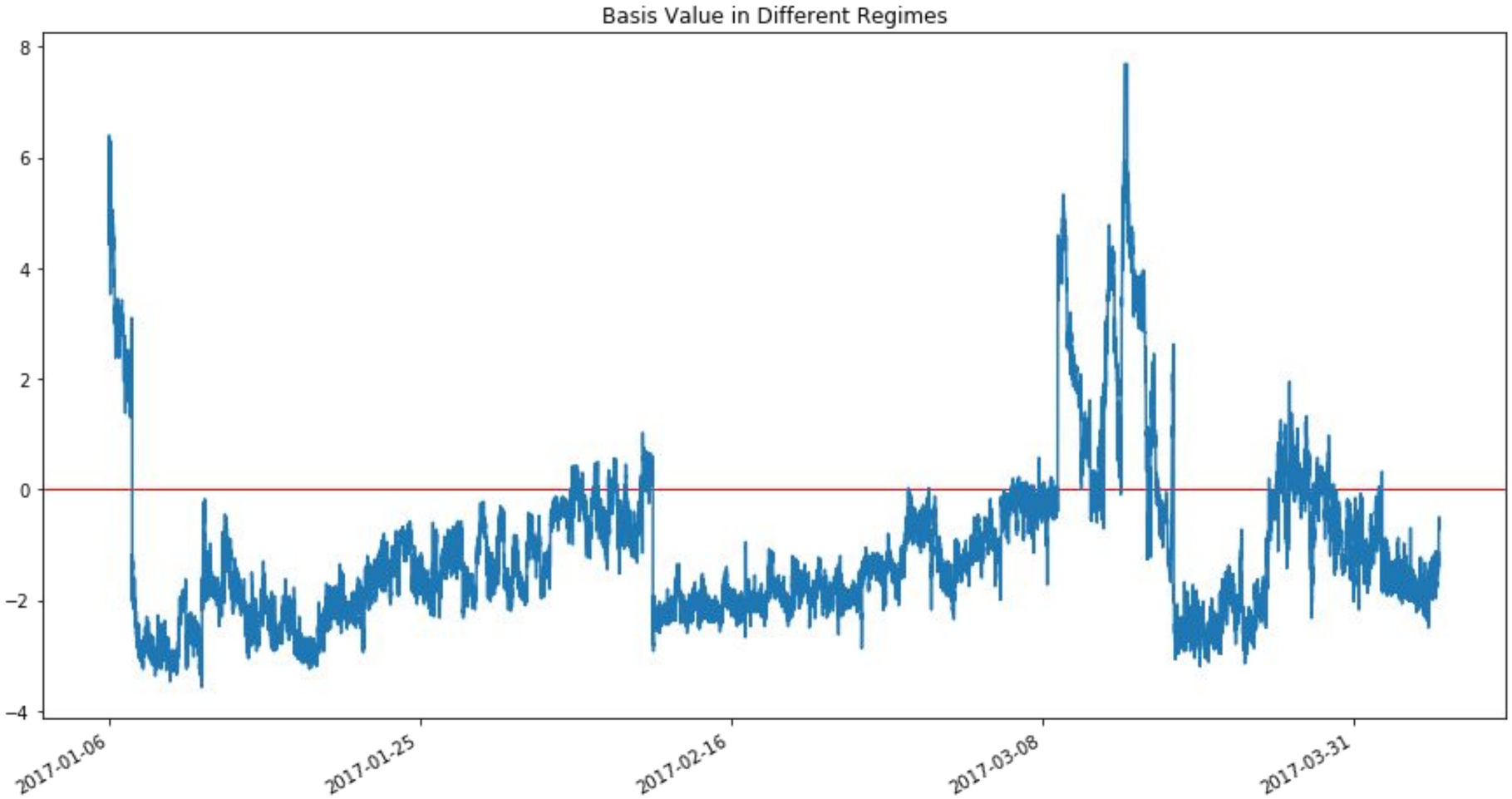
Google Cloud auquan

Rolling Cross-Validation



Google Cloud auquan

Market Regime Shifts



Google Cloud auquan

Momentum Training Model

Step 1: Define the problem

Step 2: Collect the Data

Step 3: Creating Features

Step 4: Split the Data

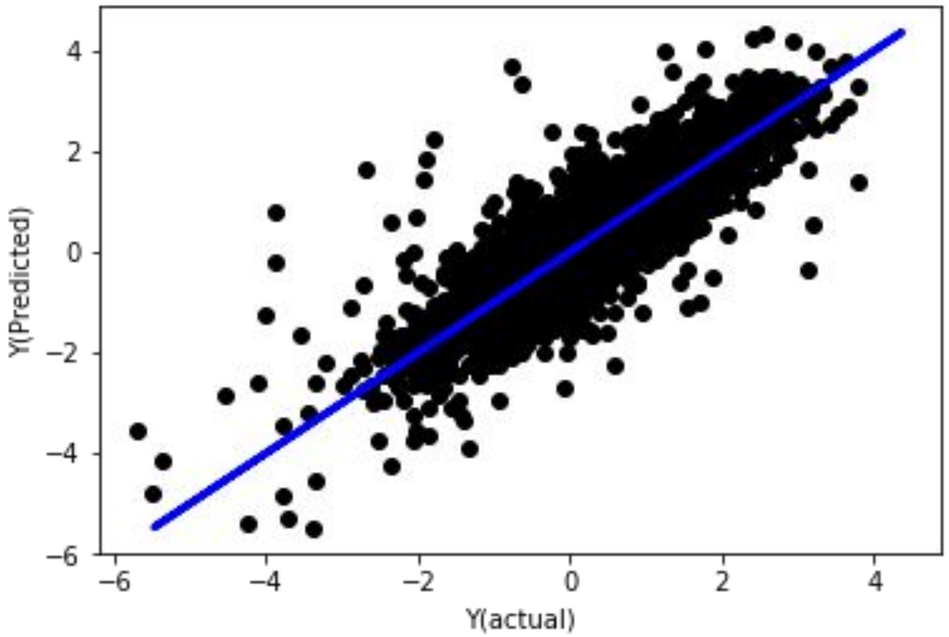
Step 5: Select a Machine Learning Algorithm

Step 6: Backtest on Unseen Data

Results with Linear Regression

29 Features

RMSE = 0.25 (Y Std = 0.59)
PnL = 8.06%

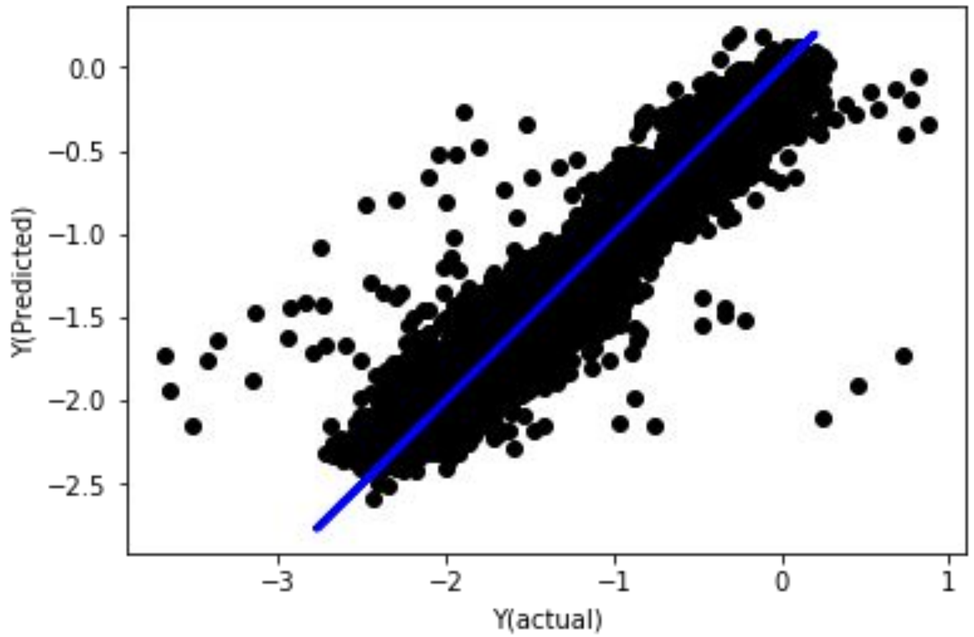


Google Cloud auquan

Results of Linear Regression after Normalization

```
for i in range(len(basis_X_train.columns)):
    basis_X_train[basis_X_train.columns[i]] = (basis_X_train[basis_X_train.columns[i]] \
                                               - basis_X_train[basis_X_train.columns[i]].rolling(30).mean())\
                                                / basis_X_train[basis_X_train.columns[i]].rolling(30).std()
```

RMSE = 0.27 (Y Std = 0.59)
PnI = 7.52%



Google Cloud auquan

Feature Importances / Coefficients

Pre Normalization

```
Coefficients:

19.11679
18.94110
4.70297
5.87735
3.54036
-56.48562
-0.00104
-0.00062
-0.00008
0.04345
0.01166
0.05763
-0.03063
6.02524
0.08399
0.00435
-0.00676
```

Post Normalization

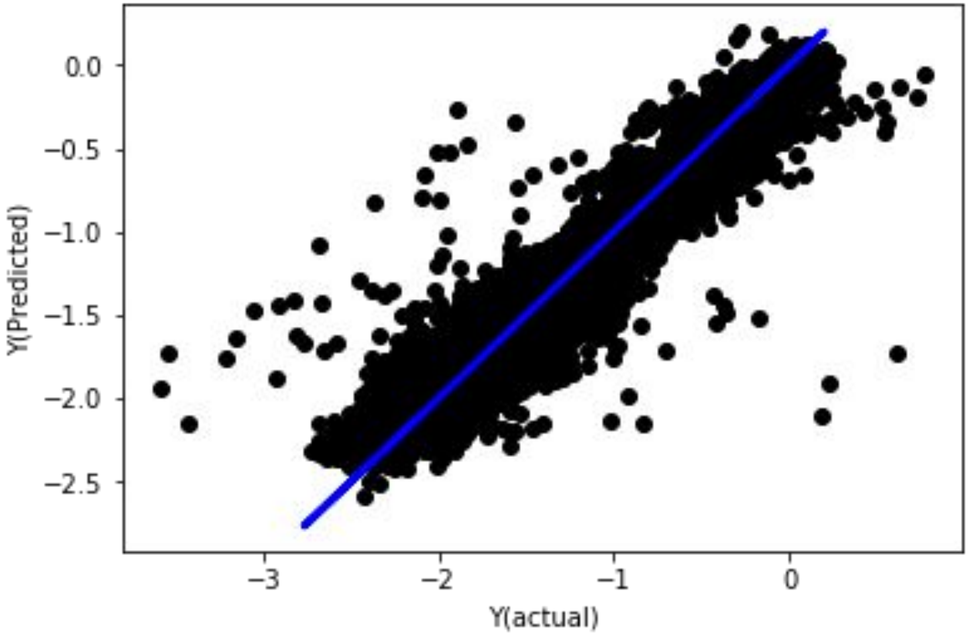
```
Coefficients:

-3.6256
-9.1527
1.4447
8.8779
-5.5569
19.8323
-0.0035
-0.0146
-0.0229
0.0089
0.0197
-0.0011
-0.0033
-7.9166
0.0499
0.0065
0.0113
2.0869
2.0322
```

Google Cloud auquan

After Normalization: Linear Regression

7 Features

RMSE = 0.22 (Y Std = 0.59)
Pnl = 8.18%

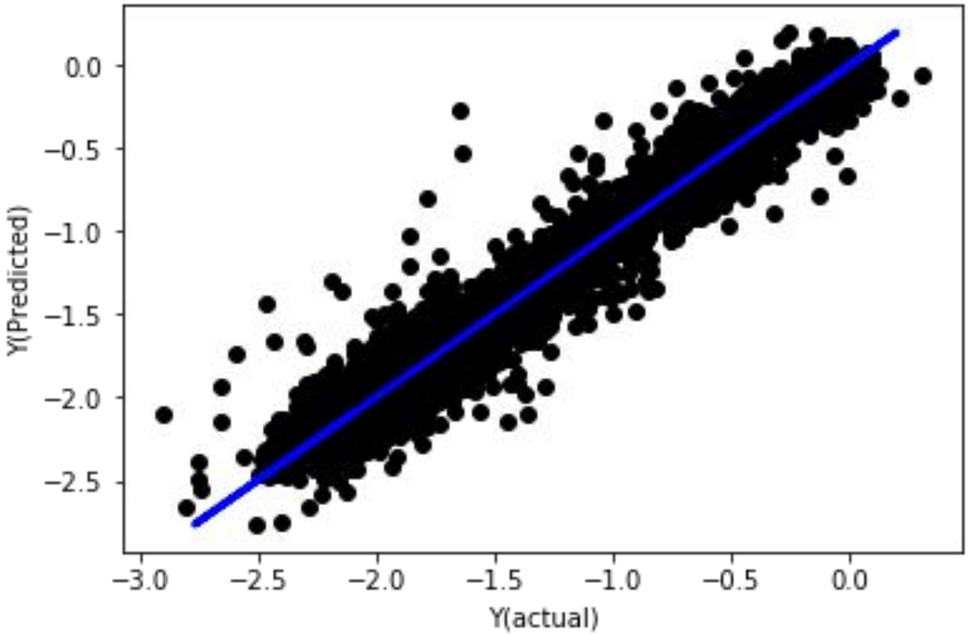


Google Cloud auquan

SVR

```
from sklearn.svm import SVR

model = SVR(kernel='rbf', C=1e3, gamma=0.1)

model.fit(basis_X_train, basis_y_train)
basis_y_pred = model.predict(basis_X_test)
```

RMSE = 0.182 (Y Std = 0.59)
Pnl = 10.14%



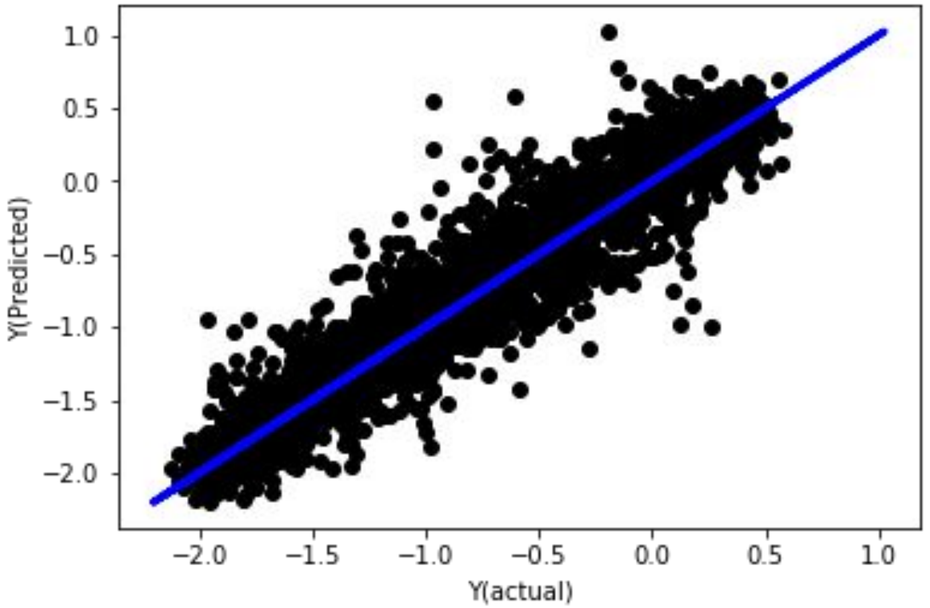
Google Cloud auquan

Gradient Boosting Regression

```
from sklearn import ensemble
model=ensemble.GradientBoostingRegressor()
model.fit(basis_X_train, basis_y_train)
basis_y_pred = model.predict(basis_X_test)
```

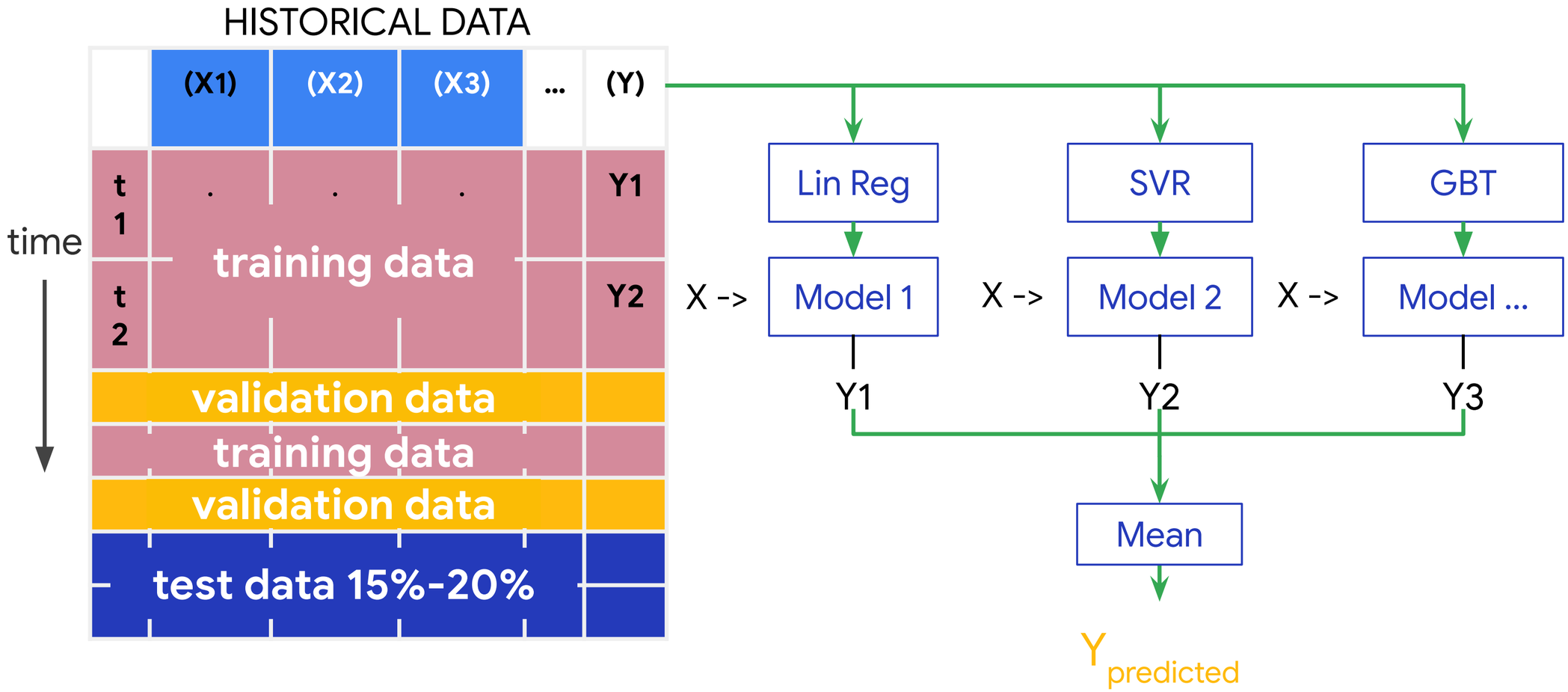
RMSE = 0.11

PnL = 22.3%



Google Cloud auquan

Ensemble Learning



Google Cloud auquan

Momentum Training Model

Step 1: Define the problem

Step 2: Collect the Data

Step 3: Creating Features

Step 4: Split the Data

Step 5: Select a Machine Learning Algorithm

Step 6: Backtest on Unseen Data

Google Cloud auquan

Current Problem - Regression

RMSE: 0.12
Total Pnl: 3.10%

Google Cloud auquan

Current Problem - Gradient Boosting

RMSE: 0.145
Total Pnl: 3.34%

Google Cloud auquan

Do’s and Don’ts

- AVOID OVERFITTING
- Don’t retain after every datapoint
- Avoid biases, especially lookahead bias
- Be wary of data mining bias

Avoid Overfitting!

- Overfitting is the most dangerous pitfall of a trading strategy
- A complex algorithm may perform wonderfully on a backtest but fails miserably on new unseen data - not really recovered any trend in data and no real predictive power
- Keep your systems as simple as possible
- Divide available data into training and test data

Google Cloud

Understanding the Code - Simple ML Strategies to generate Trading Signal

Trading Stock using Momentum factors

Preparation Steps

To run Auquan ToolBox with the first example problem, type this on your GCP shell or Colab shell or AI Platforms notebook:

```
pip3 install -U auquan_toolbox
```

- Installs all the required code for testing your code
- Automatically installs the auquan-toolbox
    - Contains useful features that are designed to make it easier for you to solve the problem

Trading Stock using Momentum factors

Preparation Steps

To run Auquan ToolBox with the first example problem, type this on your GCP shell or Colab shell or AI Platforms notebook:

```
pip3 install -U auquan_toolbox
```

- Installs all the required code for testing your code
- Automatically installs the auquan-toolbox
  - Contains useful features that are designed to make it easier for you to solve the problem

Google Cloud auquan

Trading Stock using Momentum factors

Preparation Steps

To run Auquan ToolBox with the first example problem, type this on your GCP shell or Colab shell or AI Platforms notebook:

```
pip3 install -U auquan_toolbox
```

- Installs all the required code for testing your code
- Automatically installs the auquan-toolbox
    - Contains useful features that are designed to make it easier for you to solve the problem

Trading Stock using Momentum factors

Preparation Steps

To run Auquan ToolBox with the first example problem, type this on your GCP shell or Colab shell or AI Platforms notebook:

```
pip3 install -U auquan_toolbox
```

- Installs all the required code for testing your code
- Automatically installs the auquan-toolbox
  - Contains useful features that are designed to make it easier for you to solve the problem

Trading Stock using Momentum factors

Preparation Steps

To run Auquan ToolBox with the first example problem, type this on your GCP shell or Colab shell or AI Platforms notebook:

```
pip3 install -U auquan_toolbox
```

- Installs all the required code for testing your code
- Automatically installs the auquan-toolbox
  - Contains useful features that are designed to make it easier for you to solve the problem

Trading Stock using Momentum factors

Preparation Steps

To run Auquan ToolBox with the first example problem, type this on your GCP shell or Colab shell or AI Platforms notebook:

```
pip3 install -U auquan_toolbox
```

- Installs all the required code for testing your code
- Automatically installs the auquan-toolbox
    - Contains useful features that are designed to make it easier for you to solve the problem

Trading Stock using Momentum factors

Understanding the ML problem

Problem: Use the trading data for a fictitious company (ticker: MQK) to predict the “basis” for the stock. Using its current and predicted basis, we will decide to buy or sell its stock at each point in time using purely momentum factors such as:

- 2-min exponential moving average of the basis
- 5 minute exponential moving average of the basis
- 10 minute exponential moving average of the basis

Trading Stock using Momentum factors

Understanding the ML problem

Problem: Use the trading data for a fictitious company (ticker: MQK) to predict the “basis” for the stock. Using its current and predicted basis, we will decide to buy or sell its stock at each point in time using purely momentum factors such as:

- 2-min exponential moving average of the basis
- 5 minute exponential moving average of the basis
- 10 minute exponential moving average of the basis

Trading Stock using Momentum factors

Understanding the ML problem

Problem: Use the trading data for a fictitious company (ticker: MQK) to predict the “basis” for the stock. Using its current and predicted basis, we will decide to buy or sell its stock at each point in time using purely momentum factors such as:

- 2-min exponential moving average of the basis
- 5 minute exponential moving average of the basis
- 10 minute exponential moving average of the basis

Trading Stock using Momentum factors

Understanding the ML problem

Problem: Use the trading data for a fictitious company (ticker: MQK) to predict the “basis” for the stock. Using its current and predicted basis, we will decide to buy or sell its stock at each point in time using purely momentum factors such as:

- 2-min exponential moving average of the basis
- 5 minute exponential moving average of the basis
- 10 minute exponential moving average of the basis

Trading Stock using Momentum factors

Understanding the ML problem

Problem: Use the trading data for a fictitious company (ticker: MQK) to predict the “basis” for the stock. Using its current and predicted basis, we will decide to buy or sell its stock at each point in time using purely momentum factors such as:

- 2-min exponential moving average of the basis
- 5 minute exponential moving average of the basis
- 10 minute exponential moving average of the basis

Trading Stock using Momentum factors

Understanding the ML problem

Problem: Use the trading data for a fictitious company (ticker: MQK) to predict the “basis” for the stock. Using its current and predicted basis, we will decide to buy or sell its stock at each point in time using purely momentum factors such as:

- 2-min exponential moving average of the basis
- 5 minute exponential moving average of the basis
- 10 minute exponential moving average of the basis

Trading Stock using Momentum Factors

How to download this code: click this link and download to your machine

Introduction to ML for Trading.ipynb

Let us understand it step by step.

```
from backtester.trading_system_parameters import TradingSystemParameters
from backtester.features.feature import Feature
from datetime import datetime, timedelta
from backtester.trading_system import TradingSystem
from backtester.version import updateCheck
from auquan_qq1_toolbox.problem1_trading_params import MyTradingParams

from sklearn import linear_model
from sklearn import metrics as sm
import pandas as pd
import numpy as np
import sys
```

By installing Auquan Toolbox, we get a backtester library containing:

- Ticker data
- Ability to buy and sell stock
- Develop a trading strategy

Google Cloud auquan

Trading Stock using Momentum Factors

How to download this code: click this link and download to your machine

Introduction to ML for Trading.ipynb

Let us understand it step by step.

```
from backtester.trading_system_parameters import TradingSystemParameters
from backtester.features.feature import Feature
from datetime import datetime, timedelta
from backtester.trading_system import TradingSystem
from backtester.version import updateCheck
from auquan_qq1_toolbox.problem1_trading_params import MyTradingParams

from sklearn import linear_model
from sklearn import metrics as sm
import pandas as pd
import numpy as np
import sys
```

By installing Auquan Toolbox, we get a backtester library containing:

- Ticker data
- Ability to buy and sell stock
- Develop a trading strategy

Google Cloud auquan

Trading Stock using Momentum Factors

How to download this code: click this link and download to your machine

Introduction to ML for Trading.ipynb

Let us understand it step by step.

```
from backtester.trading_system_parameters import TradingSystemParameters
from backtester.features.feature import Feature
from datetime import datetime, timedelta
from backtester.trading_system import TradingSystem
from backtester.version import updateCheck
from auquan_qq1_toolbox.problem1_trading_params import MyTradingParams

from sklearn import linear_model
from sklearn import metrics as sm
import pandas as pd
import numpy as np
import sys
```

By installing Auquan Toolbox, we get a backtester library containing:

- Ticker data
- Ability to buy and sell stock
- Develop a trading strategy

Google Cloud auquan

Step 1 and 2: Download data into data frames

1. **Download ticker data**

In [3]:

```
# Load the data
cachedFolderName = '/data/'
dataSetId = 'trainingData1'
startDate = '2017/01/06'
endDate = '2017/02/09'
instrumentIds = ['MQK']
downloadUrl = 'https://github.com/Auquan/auquan-historical-data/raw/master/qq2Data'

ds = CsvDataSource(cachedFolderName='historicalData/',
                               dataSetId=dataSetId,
                               instrumentIds=instrumentIds,
                               downloadUrl = downloadUrl,
                               timeKey = 'datetime',
                               timeStringFormat = '%Y-%m-%d %H:%M:%S',
                               startDateStr=startDate,
                               endDateStr=endDate,
                               liveUpdates=False,
                               pad=True)
```

```
Processing data for stock: MQK
```

2. **Save it into dataframes**

```
#Loading our data
def loadData(ds, id):
    data = ds._bookDataByInstrument[id].getBookData()

    data['Stock Price'] =  data['stockTopBidPrice'] +\
                           data['stockTopAskPrice'] / 2.0
    data['Future Price'] = data['futureTopBidPrice'] +\
                           data['futureTopAskPrice'] / 2.0
    data['Y(Target)'] = data['basis'].shift(-5)
    del data['benchmark_score']
    del data['FairValue']
    return data

data = loadData(ds,instrumentIds[0])
```

Google Cloud auquan

Step 3: Split Data into Train, Validation and Test

3. Split Data into Train, Validation and Test data sets

In [6]:

```
# Training Data
dataSetId =  'trainingData1'
ds_training = CsvDataSource(cachedFolderName='historicalData/',
                            dataSetId=dataSetId,
                            instrumentIds=instrumentIds,
                            downloadUrl = downloadUrl,
                            timeKey = 'datetime',
                            timeStringFormat = '%Y-%m-%d %H:%M:%S',
                            liveUpdates=False,
                            pad=True)

training_data = loadData(ds_training, instrumentIds[0])

# Validation Data
dataSetId =  'trainingData2'
ds_validation = CsvDataSource(cachedFolderName='historicalData/',
                            dataSetId=dataSetId,
                            instrumentIds=instrumentIds,
                            downloadUrl = downloadUrl,
                            timeKey = 'datetime',
                            timeStringFormat = '%Y-%m-%d %H:%M:%S',
                            liveUpdates=False,
                            pad=True)
validation_data = loadData(ds_validation, instrumentIds[0])
```

Download data sets URL link:

https://github.com/Auquan/auquan-historical-data/raw/master/qq2Data

```
# Test Data
dataSetId =  'trainingData3'
ds_test = CsvDataSource(cachedFolderName='historicalData/',
                            dataSetId=dataSetId,
                            instrumentIds=instrumentIds,
                            downloadUrl = downloadUrl,
                            timeKey = 'datetime',
                            timeStringFormat = '%Y-%m-%d %H:%M:%S',
                            liveUpdates=False,
                            pad=True)
out_of_sample_test_data = loadData(ds_test, instrumentIds[0])
```

```
Processing data for stock: MQK
Processing data for stock: MQK
Processing data for stock: MQK
```

Google Cloud auquan

Step 3 (continued): Preparing Data

3. Set up the Target variable

We add the target variable Y, defined as the average of next five values of basis :

```
def prepareData(data, period):
    data['Y(Target)'] = data['basis'].rolling(period).mean().shift(-period)
    if 'FairValue' in data.columns:
        del data['FairValue']
    data.dropna(inplace=True)

period = 5
prepareData(training_data, period)
prepareData(validation_data, period)
prepareData(out_of_sample_test_data, period)
```

Step 4: Feature Engineering

4. Define momentum functions

In order to add momentum factors we need to define functions that calculate momentum such as relative strength index and exponential moving averages:

```
def difference(dataDf, period):
    return dataDf.sub(dataDf.shift(period), fill_value=0)

def ewm(dataDf, halflife):
    return dataDf.ewm(halflife=halflife,ignore_na=False,min_periods=0,adjust=True).mean()

def rsi(data, period):
    data_upside = data.sub(data.shift(1), fill_value=0)
    data_downside = data_upside.copy()
    data_downside[data_upside > 0] = 0
    data_upside[data_upside < 0] = 0
    avg_upside = data_upside.rolling(period).mean()
    avg_downside = - data_downside.rolling(period).mean()
    rsi = 100 - (100 * avg_downside / (avg_downside + avg_upside))
    rsi[avg_downside == 0] = 100
    rsi[(avg_downside == 0) & (avg_upside == 0)] = 0

    return rsi
```

Step 4 (continued): Feature Engineering

4. Define momentum factors

Using momentum functions we create factors such as the following exponential moving averages with intervals:

- 2 minute
- 5 minute
- 10 minute

```
def create_features_again(data):
    basis_X = pd.DataFrame(index = data.index, columns =  [])

    basis_X['mom10'] = difference(data['basis'],11)

    basis_X['emabasis2'] = ewm(data['basis'],2)
    basis_X['emabasis5'] = ewm(data['basis'],5)
    basis_X['emabasis10'] = ewm(data['basis'],10)

    basis_X['basis'] = data['basis']

    basis_X['totalaskvolratio'] = (data['stockTotalAskVol']-data['futureTotalAskVol'])/100000
    basis_X['totalbidvolratio'] = (data['stockTotalBidVol']-data['futureTotalBidVol'])/100000

    basis_X = basis_X.fillna(0)

    basis_y = data['Y(Target)']
    basis_y.dropna(inplace=True)

    print("Any null data in y: %s, X: %s"%(basis_y.isnull().values.any(), basis_X.isnull().values.any()))
    print("Length y: %s, X: %s"%(len(basis_y.index), len(basis_X.index)))

    return basis_X, basis_y
```

```
basis_X_test, basis_y_test = create_features_again(validation_data)
basis_X_train, basis_y_train = create_features_again(training_data)
```

```
Any null data in y: False, X: False
Length y: 6499, X: 6499
Any null data in y: False, X: False
Length y: 8737, X: 8737
```

Google Cloud auquan

Step 5: Setting up a Model

5. **Define a model and running it on train, validation data**

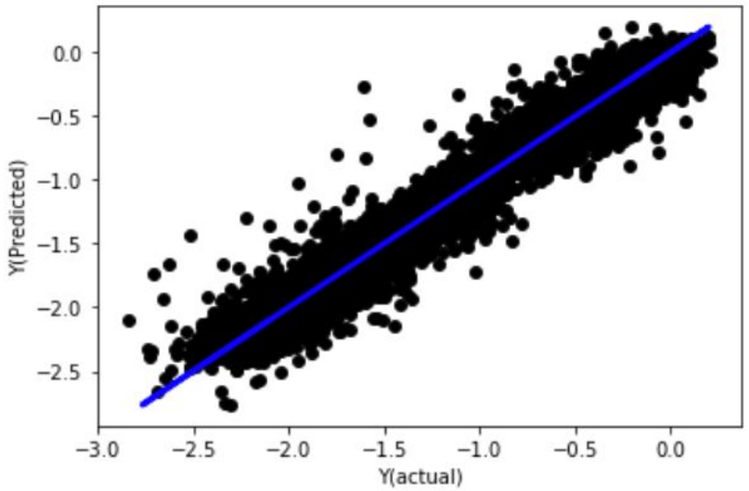
Defining an Ensemble model (Extra Trees Regressor) to:

- train
- test

```
from sklearn import ensemble
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score

model=ensemble.ExtraTreesRegressor()
model.fit(basis_X_train, basis_y_train)
basis_y_pred = model.predict(basis_X_test)

# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(basis_y_test, basis_y_pred))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(basis_y_test, basis_y_pred))
```

```
# Plot outputs
plt.scatter(basis_y_pred, basis_y_test,  color='black')
plt.plot(basis_y_test, basis_y_test, color='blue', linewidth=3)


plt.xlabel('Y(actual)')
plt.ylabel('Y(Predicted)')

plt.show()

basis_y_trees = basis_y_pred.copy()
```

```
Mean squared error: 0.02
Variance score: 0.95
```



Google Cloud auquan

Step 6: Backtest the Model

6. **Backtest the model and run it on train, test data**

Trained model (Extra Trees Regressor) on:

- train
- test

MyTradingParams() is the main function

Need to provide similar data as before

```
class MyTradingParams(TradingSystemParameters):
    def __init__(self):
        super(MyTradingParams, self).__init__()
        self.count = 0
        self.params = {}
        self.start = '2017/01/06'
        self.end = '2017/01/10'
        self.instrumentIds = ['MQK']
        filename = 'MQKmodel.pkl'
        with open(filename, 'rb') as handle:
            b = pickle.load(handle)
        self.model = {'MQK': b}
        self.updateFrequency = 1

    '''
    Returns an instance of class DataParser. Source of data for instruments
    '''

    def getDataParser(self):
        dataSetId = 'trainingData3'
        downloadUrl = 'https://github.com/Auquan/auquan-historical-data/raw/master/qq2Data'

        z= CsvDataSource(cachedFolderName='historicalData/',
                                dataSetId=dataSetId,
                                instrumentIds=self.instrumentIds,
                                downloadUrl = downloadUrl,
                                timeKey = '',
                                timeStringFormat = '%Y-%m-%d %H:%M:%S',
                                startDateStr=self.start,
                                endDateStr=self.end,
                                liveUpdates=True,
                                pad=True)
        return z
```

Google Cloud auquan

Step 6 (continued) : Backtest the Model

6. **What do we have to provide the backtester?**

   Provide the same features as the model was trained on:

   - EMA 2,5, 10

```
def getCustomFeatures(self):
    return {'prediction': TrainingPredictionFeature}

def getInstrumentFeatureConfigDicts(self):

    predictionDict = {'featureKey': 'prediction',
                          'featureId': 'prediction',
                          'params': {}}

    # ADD RELEVANT FEATURES HERE
    expma5dic = {'featureKey': 'emabasis5',
            'featureId': 'exponential_moving_average',
            'params': {'period': 5,
                          'featureName': 'basis'}}
    expma10dic = {'featureKey': 'emabasis10',
            'featureId': 'exponential_moving_average',
            'params': {'period': 10,
                          'featureName': 'basis'}}
    expma2dic = {'featureKey': 'emabasis2',
            'featureId': 'exponential_moving_average',
            'params': {'period': 2,
                          'featureName': 'basis'}}
    mom10dic = {'featureKey': 'mom10',
            'featureId': 'difference',
            'params': {'period': 10,
                          'featureName': 'basis'}}
    scoreDict = {'featureKey': 'score',
                 'featureId': 'score_fv',
                 'params': {'predictionKey': 'prediction',
                            'price': 'FairValue'}}
    return {INSTRUMENT_TYPE_STOCK: [expma5dic,expma2dic,expma10dic,mom10dic,
                                    predictionDict, scoreDict]}
```

Google Cloud auquan

Step 6 (continued) : Backtest the Model

6. **How do we get predictions?**

   Provide the same model that we trained earlier and feed it features:

   - EMA 2,5, 10

```
# Decision Tree Implementation:
from sklearn import ensemble

featureList = [mom10, emabasis2, emabasis5, emabasis10,
               basis, totalaskvol, totalbidvol]
for s in self.instrumentIds:
    Y = lbInstF.getFeatureDf('FairValue')[s]
    #Creating a dataframe to hold features for this stock
    X = pd.DataFrame(index=Y.index, columns=range(len(featureList)))        #DF with rows=timestamp and
    for i in range(len(featureList)):
        X[i] = featureList[i][s]

    # if this is the first time we are training a model,
    # start by creating a new model
    # we will update this model during further runs
    #if s not in self.model:
    if len(self.model) == 0:
        self.model[s] = ensemble.ExtraTreesRegressor()

    # if you are at the update frequency, update the model
    if (updateNum%self.updateFrequency==0) and (updateNum>59):

        # drop nans and infs from X
        X = X.replace([np.inf, -np.inf], np.nan).dropna()
        # create a target variable vector for this stock, with same index as X
        y_s = Y.loc[Y.index.isin(X.index)]

        print('Training...')
        # make numpy arrays with the right shape
        x_train = np.array(X)[:-1]                          # shape = timestamps x numFeatures
        y_train = np.array(y_s)[:-1].astype(int).reshape(-1) # shape = timestamps x 1
        self.model[s].fit(x_train, y_train)

    if (updateNum)<60:#self.updateFrequency:
        # we haven't trained a model yet
        predictions[s] = np.nan
    else:
        # make your prediction using your model
        if X.iloc[-1].replace([np.inf, -np.inf], np.nan).hasnans:
            # first verify none of the features are nan or inf
            print('Test Feature Data has nans')
            predictions[s] = np.nan
        else:
            predictions[s] = self.model[s].predict(X.iloc[-1].values.reshape(1,-1))


predictions.fillna(emabasis5,inplace=True)

print('Current basis: %.3f, predicted basis: %.3f'%(basis['MQK'], predictions['MQK']))
if updateNum>1:
    print('Current position: %.0f'%lbInstF.getFeatureDf('position').iloc[-1]['MQK'])

return predictions
```

Google Cloud auquan

Step 6 (continued) : Backtest the Model

6. **How do we convert predictions to trades?**

Provide model predictions to the backtester's execution system and provide it :

- Threshold Deviations
- Limits on long and short positions
- Limits on capital usage for a single trade
- Lot Sizes

```
'''
Here we convert prediction to intended positions for different instruments.
'''

def getExecutionSystem(self):
    return SimpleExecutionSystemWithFairValue(enter_threshold_deviation=0.5, exit_threshold_deviation=0.2,
                                              longLimit=250, shortLimit=250, capitalUsageLimit=0.05,
                                              enterlotSize=10, exitlotSize=10,
                                              limitType='L', price=self.getPriceFeatureKey())
```

Google Cloud auquan

Step 7: Check the Results

6. **How do we run the backtester?**

Backtester provides the following:

- P&L (return on capital as %)
- R-Squared (score)
- Max Drawdown
- Total loss
- Total profit
- Capital Usage

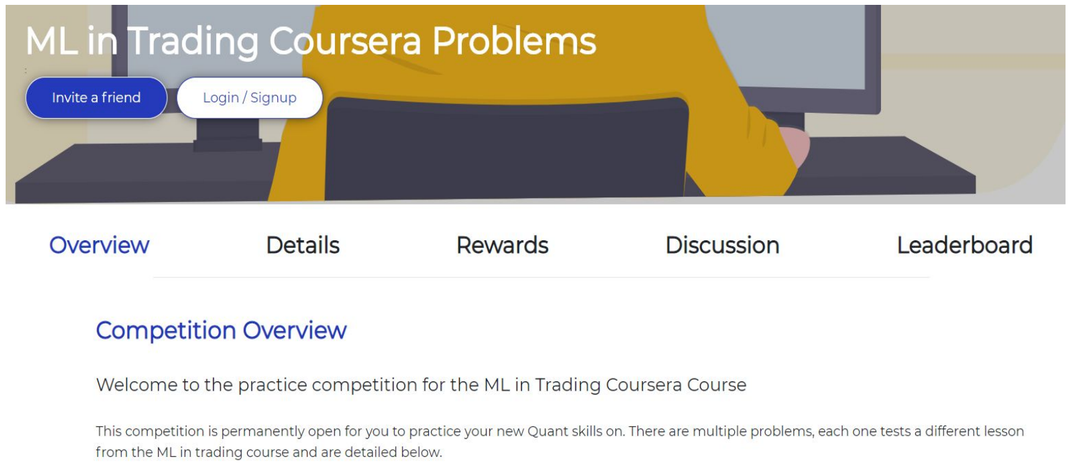
```
results
```

```
{'instrument_names': ['MQK'],
 'instrument_stats': [{'pnl': {'MQK': 0.11692330000000006},
   'score': {'MQK': 0.5270070414825269}}],
 'pnl': 0.11692330000000006,
 'trading_days': 3,
 'score': 0.5270070414825269,
 'portfolio_value': 11169.233,
 'maxDrawdown': 483.66499999999905,
 'maxPortfolioValue': 11230.483,
 'capitalUsage': 1324.7970000000041,
 'variance': 790.2561871673435,
 'capital': 9936.30299999999,
 'count_loss': 222,
 'total_loss': 4699.115000000002,
 'total_profit': 5868.347999999998,
 'count_profit': 228}
```

Apply your new skills to a real industry problem

If you're interested and want to explore the problem in more depth you can do so at:

quant-quest.auquan.com/competitions/coursera-real-world-problems



Google Cloud auquan

Auquan Sources Used to Create This Lesson

- Auquan's Medium Blog:
  medium.com/auquan/https-medium-com-auquan-machine-learning-techniques-trading-b7120cee4f05

  medium.com/auquan/momentum-simple-trading-strategies-part-2-188cf464ffcf

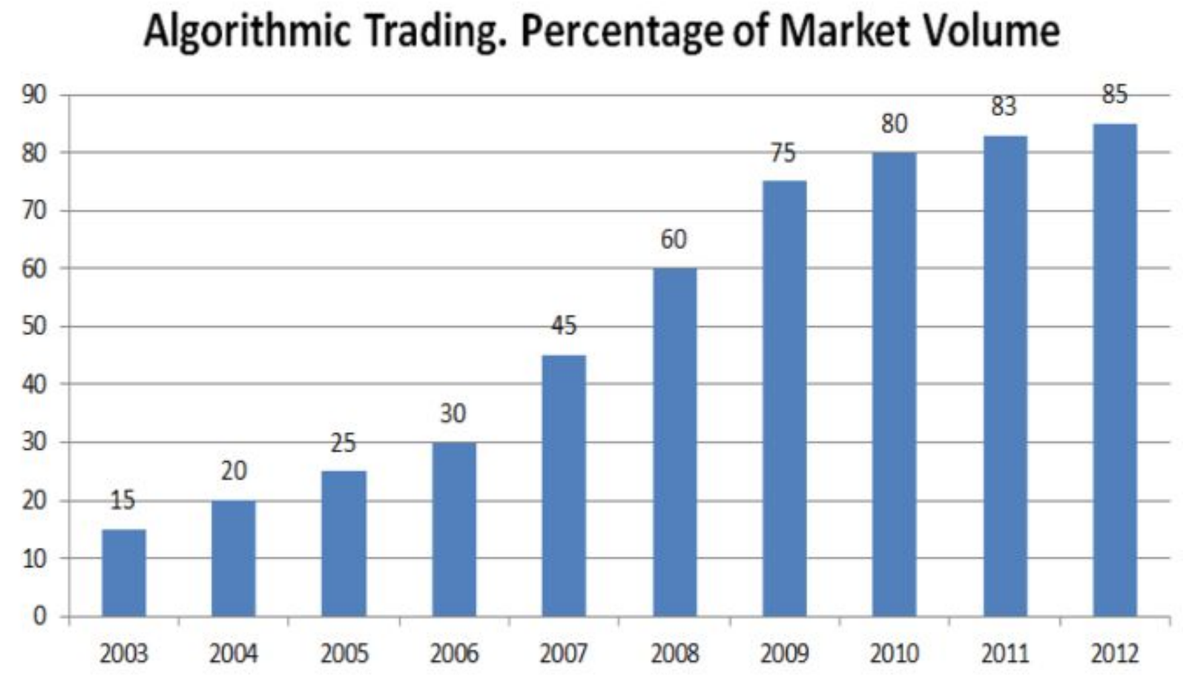
- Auquan's GitHub:
  github.com/Auquan/Tutorials/blob/master/Momentum%20Strategies.ipynb

- Auquan's Problems & Data:
  quant-quest.auquan.com/competitions

Google Cloud auquan

What does the future look like?

Algorithmic Trading is the future



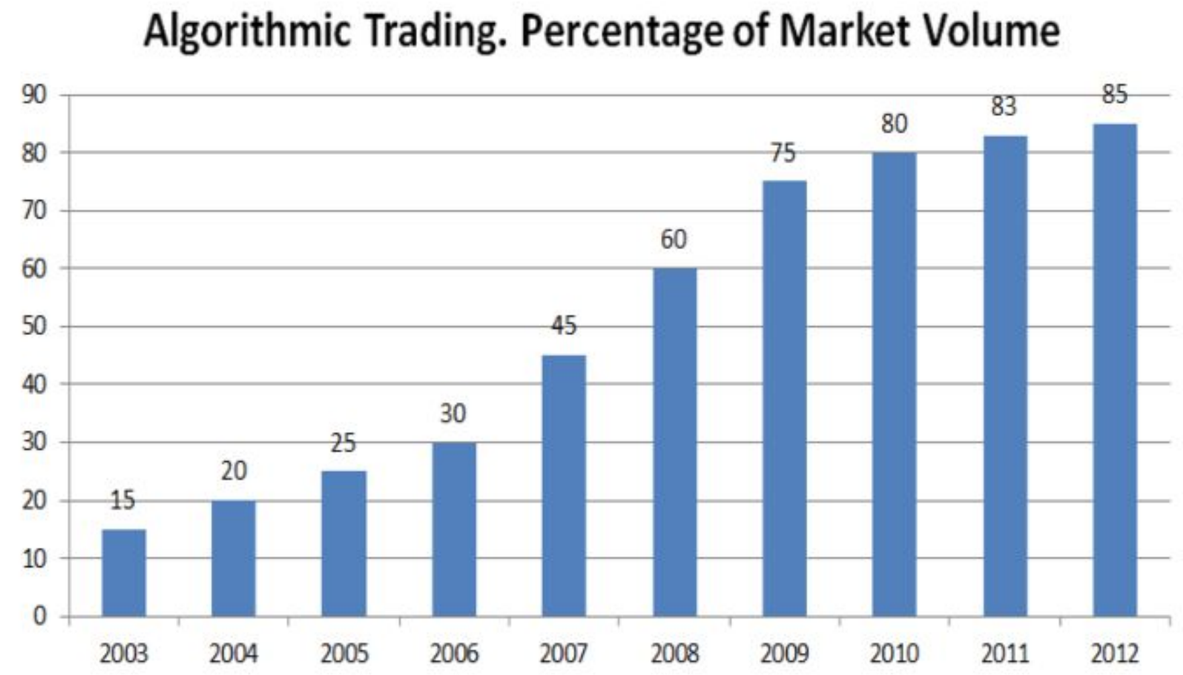
Algorithmic Trading. Percentage of Market Volume. data from Morton Glantz, Robert Kissell.

Google Cloud auquan

What does the future look like?

Algorithmic Trading is the future

Algorithmic trading volume is a critical component of market volume surpassing 85% as of 2012. It is probably more today.



Algorithmic Trading. Percentage of Market Volume. data from Morton Glantz, Robert Kissell.

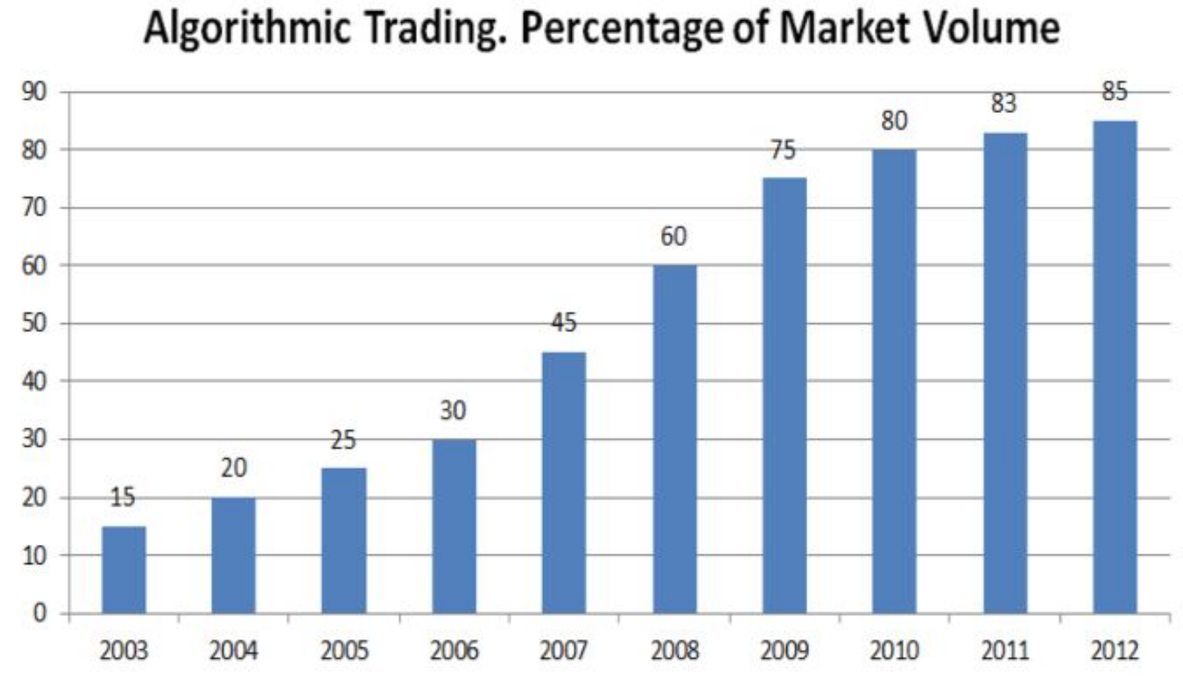
Google Cloud auquan

What does the future look like?

Algorithmic Trading is the future

Algorithmic trading volume is a critical component of market volume surpassing 85% as of 2012. It is probably more today.

Algorithmic Traders do not use their intuition or gut to trade, but use a computer program - though they may code their intuition in some cases into a program.



Algorithmic Trading. Percentage of Market Volume. data from Morton Glantz, Robert Kissell.

Google Cloud auquan

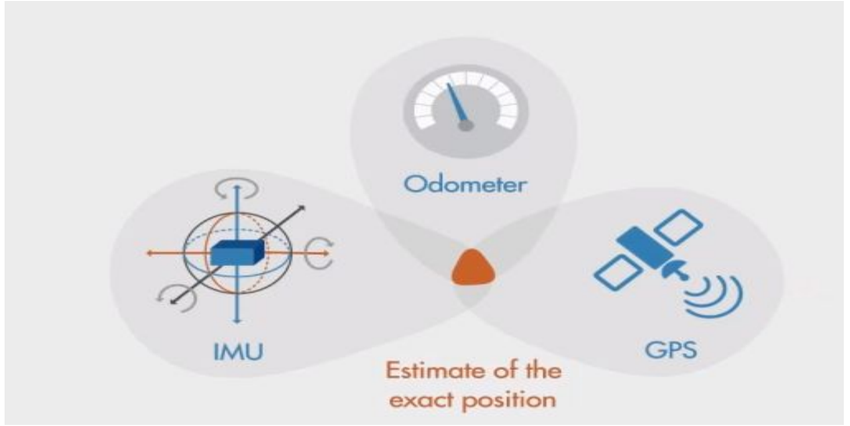
ESTABLISHED 1922

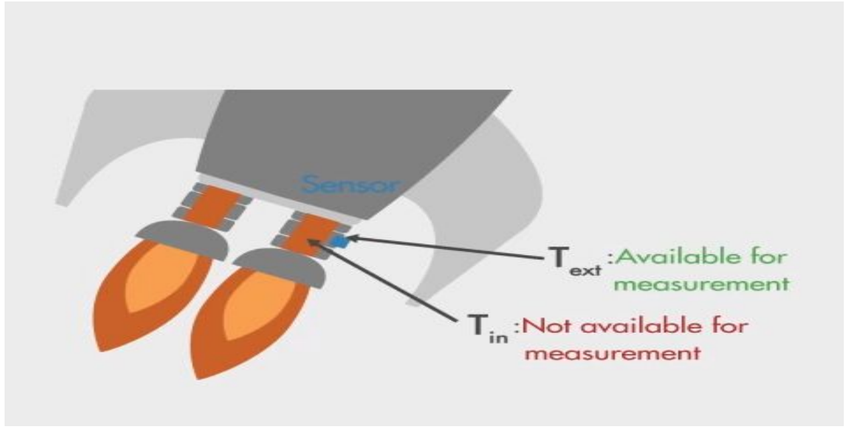
NEW YORK INSTITUTE OF FINANCE

Kalman Filter

What is a Kalman Filter?

- Named after Rudolf Kalman (1930-2016)
- One of the very first applications was in Project Apollo
- Common Applications include GPS systems, Computer Vision and Signal Processing

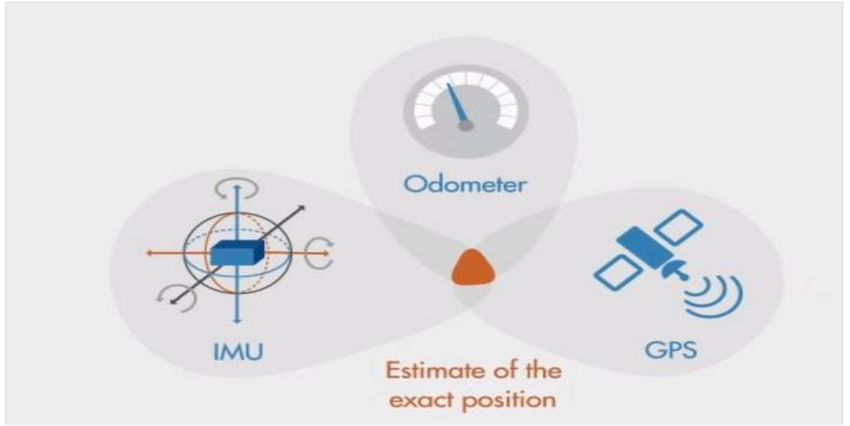


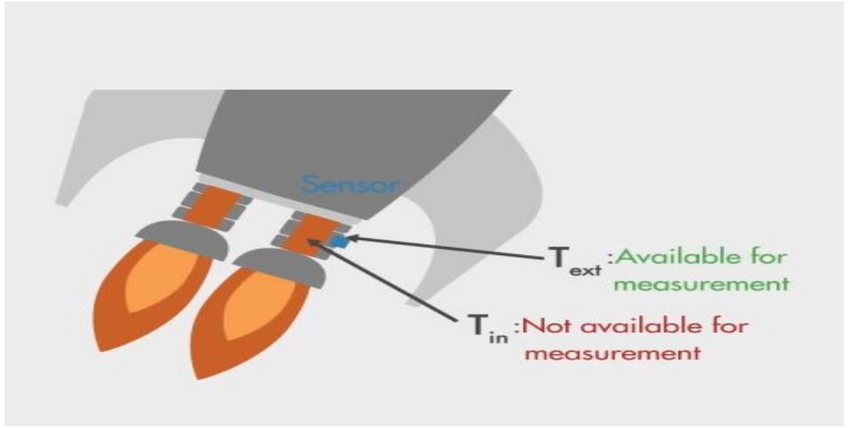


Source: https://www.youtube.com/watch?v=mwn8xhgNpFY

What is a Kalman Filter?

- Optimal Estimation Algorithm
- Not considered a Machine Learning technique by purists
- Supports estimation of past present and future states
- Even when the precise nature of the modeled system is unknown

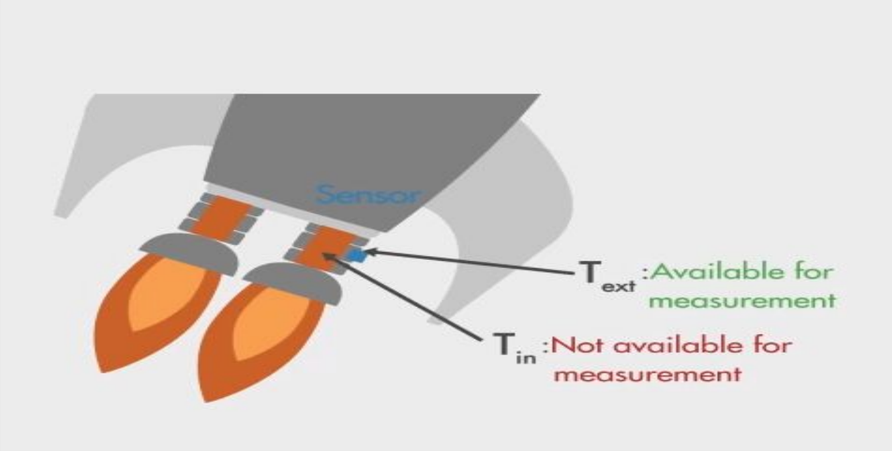




Source: https://www.youtube.com/watch?v=mwn8xhgNpFY

When is a Kalman Filter Needed?

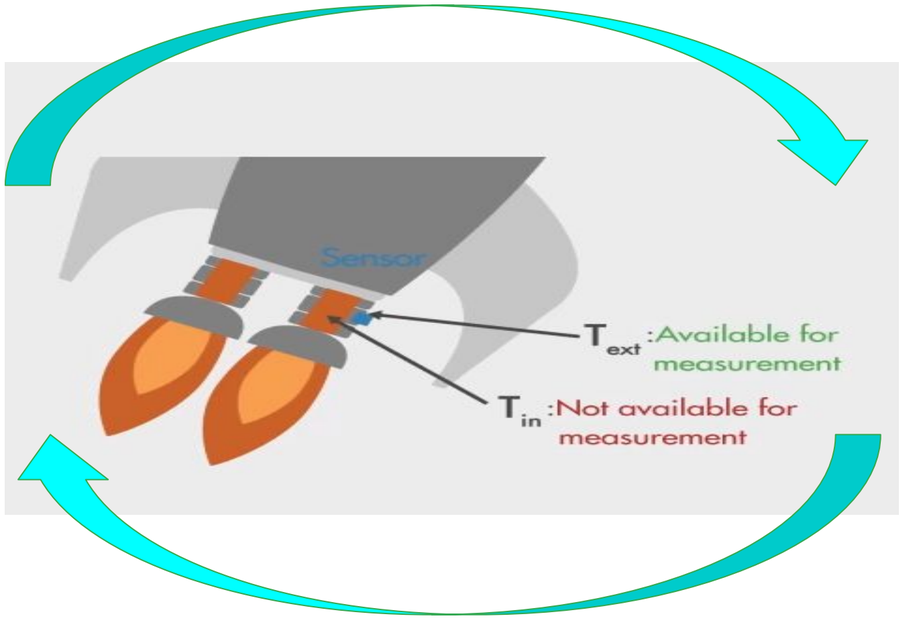
- The variables of interest can be measured only indirectly
- Or, measurements are available from multiple sources subject to Noise



When is a Kalman Filter Needed?

- Given a sequence of noisy measurements, the Kalman Filter is able to recover the “true state” of the underlying object being tracked

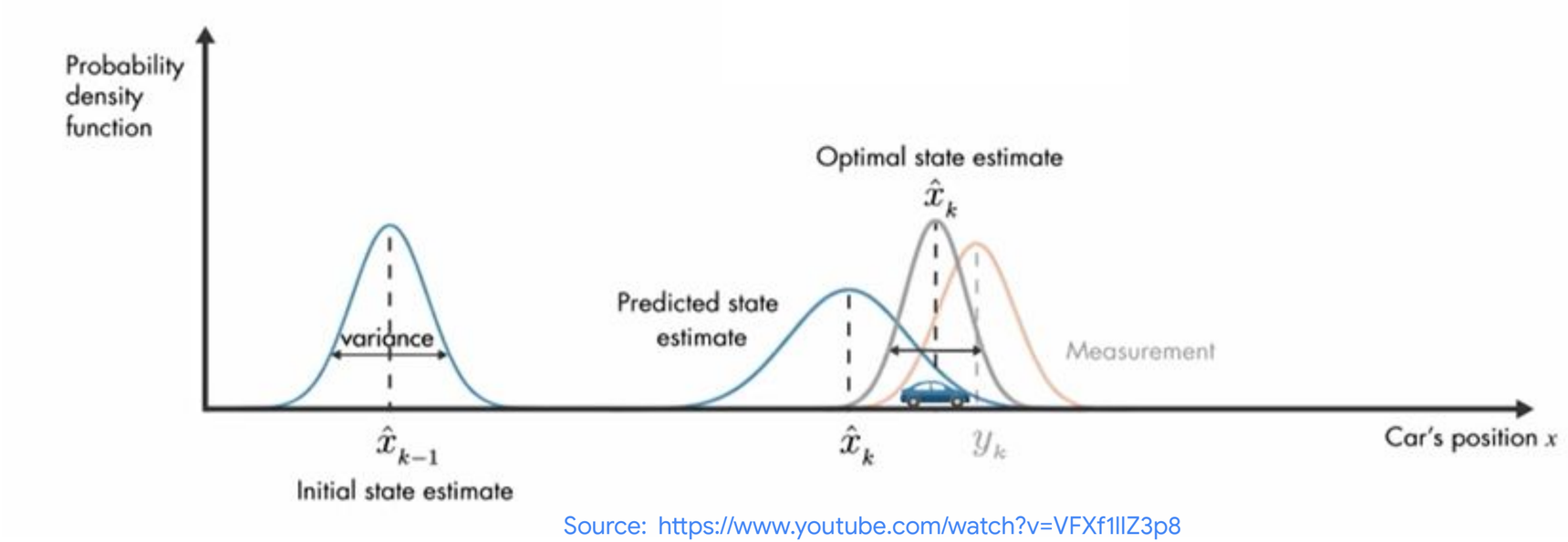
Time Update (“Predict”)



Measurement Update (“Correct”)

The Theory behind Kalman Filters

Let's try to estimate a Car's position using GPS Sensors. Our goal is to best estimate the Car's actual position using estimates of its observed state at various time intervals.



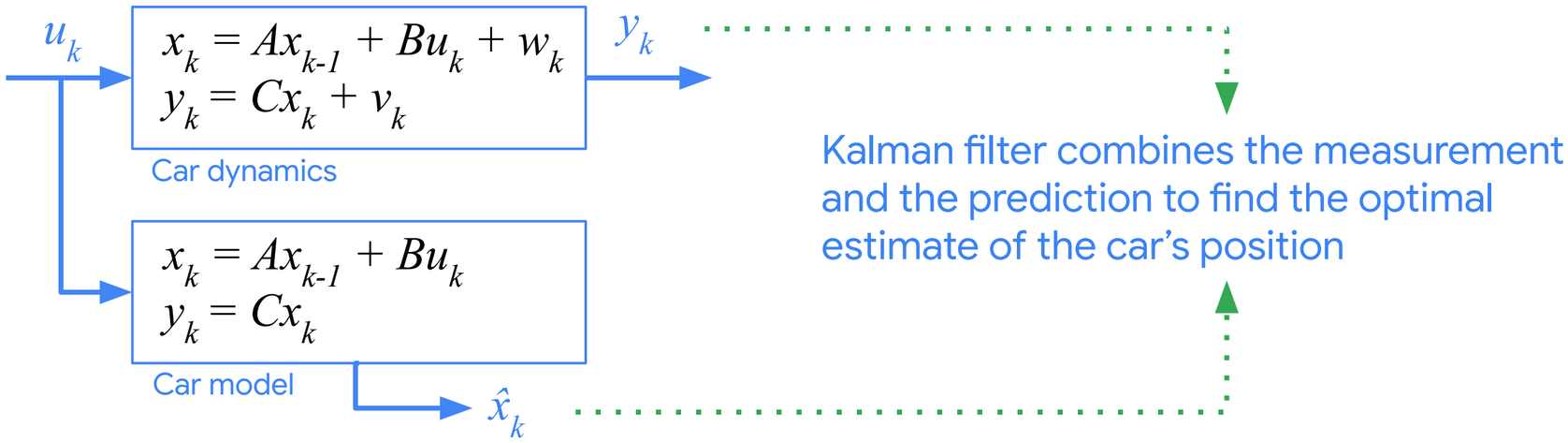
Source: https://www.youtube.com/watch?v=VFXf1llZ3p8

Google Cloud

ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

The Theory behind Kalman Filters

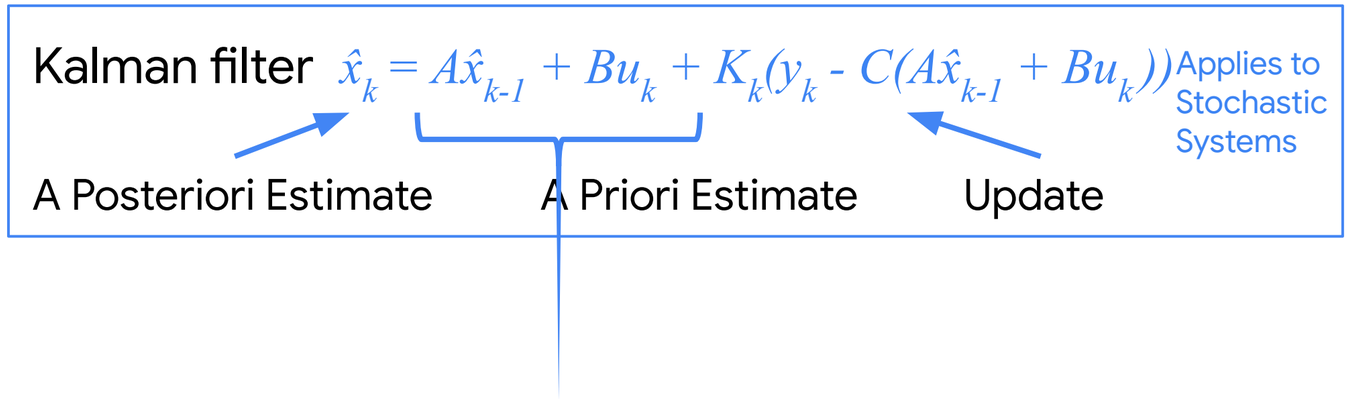
Kalman filter combines the measurement and the prediction to find the optimal estimate of the car's position



Source: https://www.youtube.com/watch?v=VFXf1llZ3p8

The Theory behind Kalman Filters

This iterative process of measurement and estimation continues until we find the optimal state. This is one way to use the Kalman filter in real world applications.



Source: https://www.youtube.com/watch?v=VFXf1llZ3p8

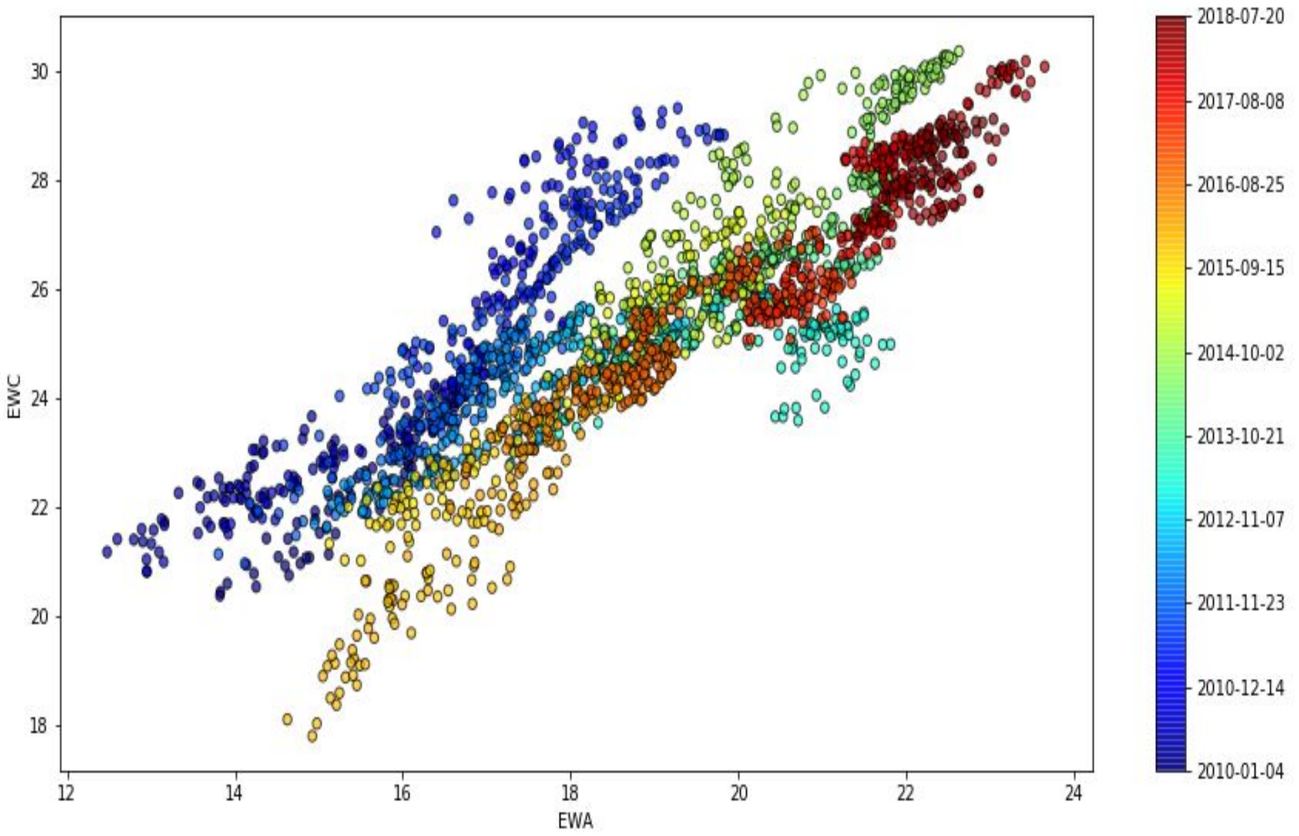
Different Kinds of Kalman Filters

State Estimator	Model	Assumed distribution	Computational cost
Kalman Filter (KF)	Linear	Gaussian	Low
Extended Kalman Filter (EKF)	Locally linear	Gaussian	Low (if the Jacobians need to be computed analytically) Medium (if the Jacobians can be computed numerically)
Unscented Kalman Filter (UKF)	Nonlinear	Gaussian	Medium
Particle Filter (PF)	Nonlinear	Non-Gaussian	High

Source: https://www.youtube.com/watch?v=Vefia3JMeHE

Kalman Filter: Stock Correlation Prediction Example*

- Let's assume two ETF's EWA and EWC are highly correlated to each other
- Knowing one, can we predict what the other's Price will be?



* Source: http://www.thealgoengineer.com/2014/online_linear_regression_kalman_filter/

Google Cloud

ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

Kalman Filter: Stock Correlation Prediction

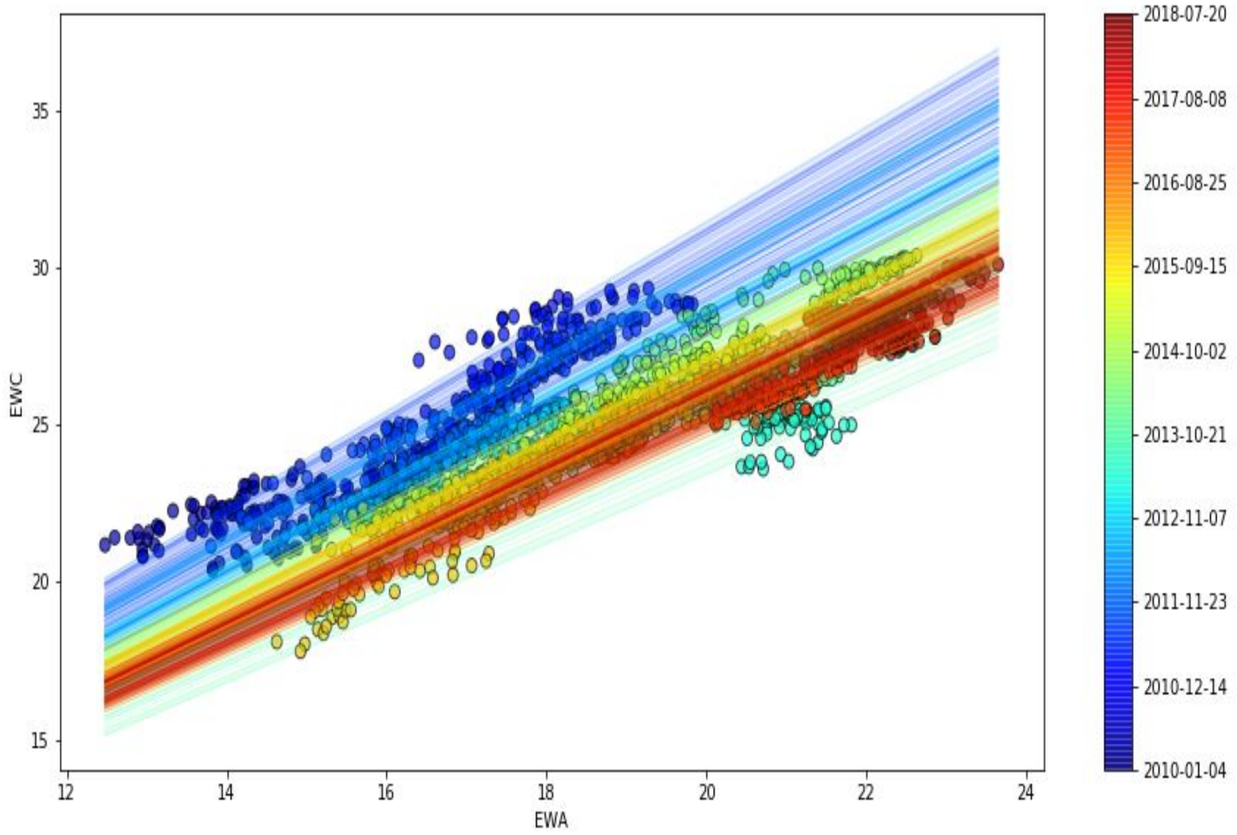
- Let's set up a Kalman Filter
- Let's use EWC to predict EWA
- We are going to set some initial values for the filter
- Then we feed EWC values to this filter and see what we get

Now, we can instantiate the `KalmanFilter` class from the pykalman module

```
kf = KalmanFilter(n_dim_obs=1, n_dim_state=2,
                  initial_state_mean=np.zeros(2),
                  initial_state_covariance=np.ones((2, 2)),
                  transition_matrices=np.eye(2),
                  observation_matrices=obs_mat,
                  observation_covariance=1.0,
                  transition_covariance=trans_cov)
```

Predicting EWA from EWC

How the Correlation Prediction lines look over the years



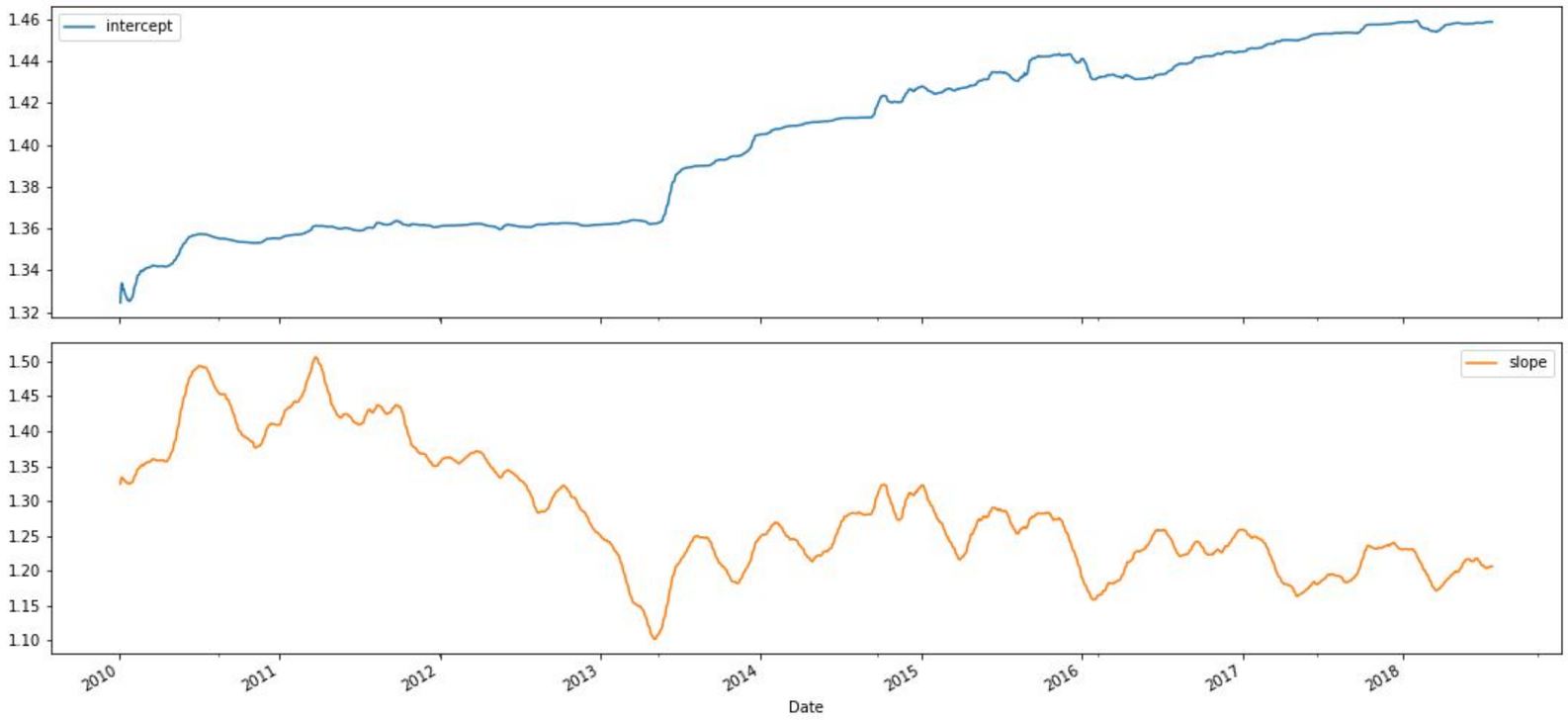
Google Cloud

ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

Kalman Filter: Stock Correlation Prediction

```
state_means, state_covs = kf.filter(data.EWC.values)
```

Look at the mean and covariance of the predicted states using the filter applied to EWC's values



Google Cloud

ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

Further Reading

If you are interested you can read more applications of **PREDICTING MARKET DATA WITH A KALMAN FILTER here:**
http://www.haikulabs.com/pmdwkf26.htm

ESTABLISHED 1922

NEW YORK INSTITUTE OF FINANCE

Kalman Filter: Trading Applications

Learning Objectives

- Design a Kalman estimator to replace moving average signals
- Use the Kalman framework to estimate regression coefficients

Learning Objectives

- Design a Kalman estimator to replace moving average signals
- Use the Kalman framework to estimate regression coefficients

Agenda

Kalman Estimator vs Moving Averages

Kalman Estimates of Regression Coefficients

Kalman Filter vs Moving Average

Kalman filters:

- Updates estimates at each time step
- Weights more recent observations more heavily
- Eliminates need to specify a window length

Kalman Filter vs Moving Average

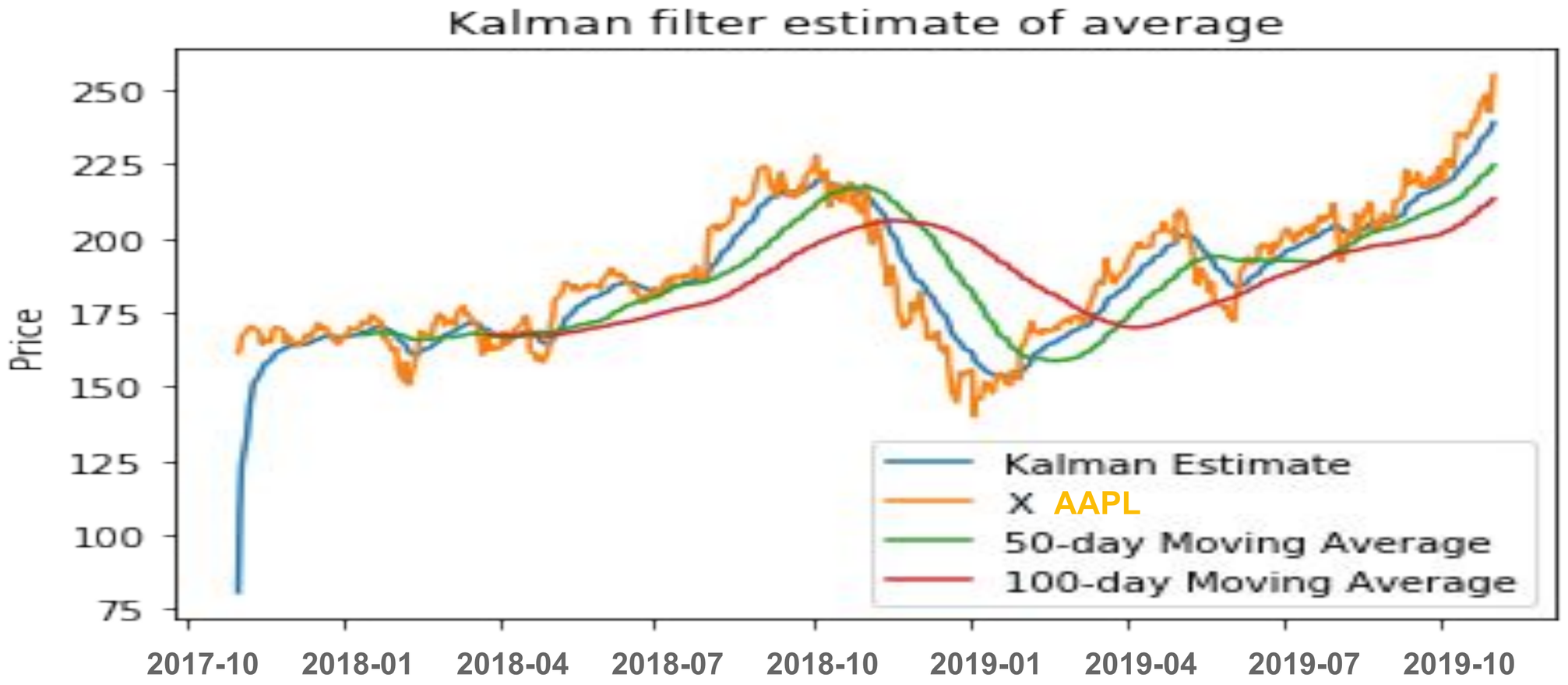
- Kalman filter vs 50 and 100 day moving average
- Kalman estimate evolves as a random walk with a small error term
- Model estimates the mean of the distribution which is our prediction of the next value
- Initial guess is mean = 0

Construct Kalman Filter and Moving Averages

```
# Load pricing data for a security
start = '2017-11-01'
end = '2019-11-01'
AAPL = pdr.DataReader('AAPL', data_source="yahoo", start=start, end=end)
AAPL = AAPL['Adj Close']
SPY = pdr.DataReader('LMT', data_source="yahoo", start=start, end=end)
SPY = SPY['Adj Close']
x = AAPL

# Construct a Kalman filter
kf = KalmanFilter(transition_matrices = [1],
                  observation_matrices = [1],
                  initial_state_mean = 0,
                  initial_state_covariance = 1,
                  observation_covariance=1,
                  transition_covariance=.01)
```

```
# Use the observed values of the price to get a rolling mean
state_means, _ = kf.filter(x.values)
state_means = pd.Series(state_means.flatten(), index=x.index)

# Compute the rolling mean with various lookback windows
mean50 = x.rolling(window = 50).mean()
mean100 = x.rolling(window =100).mean()

# Plot original data and estimated mean
plt.plot(state_means)
plt.plot(x)
plt.plot(mean50)
plt.plot(mean100)
plt.title('Kalman filter estimate of average')
plt.legend(['Kalman Estimate', 'X', '50-day Moving Average', '100-day Moving Average'])
plt.xlabel('Day')
plt.ylabel('Price');
```

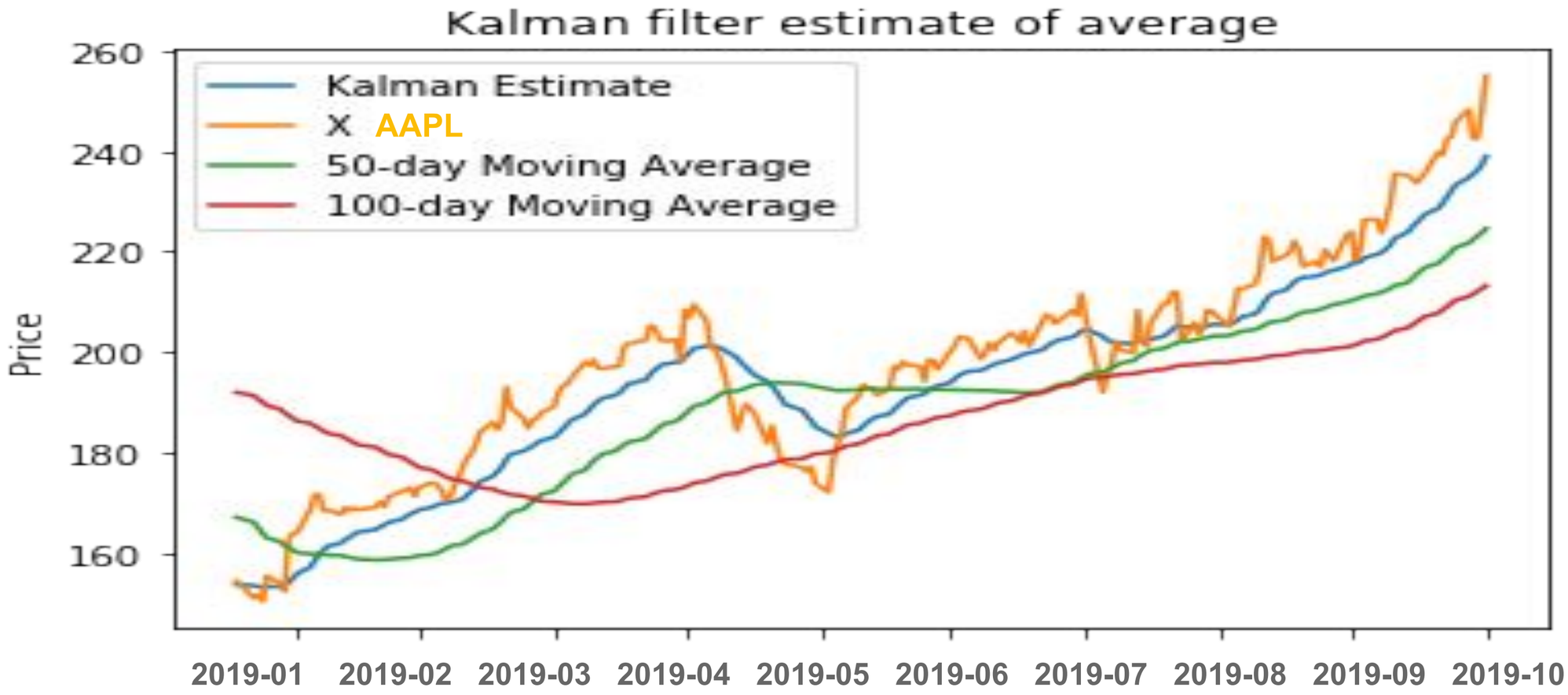
Kalman Estimate vs 50d and 100d Moving Averages



Google Cloud

ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

Construct Kalman Filter and Moving Averages



Google Cloud

ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

Agenda

Kalman Estimator vs Moving Averages

Kalman Estimates of Regression Coefficients

Inputs to Kalman Filter for Estimating α and β

- Initial system state is line that observations are following
- Initial guess for α and β is (0,0)
- New system state given by:

$$(\beta,\alpha) \cdot (x_i,1) \Rightarrow \beta x_i+\alpha \approx y_i$$

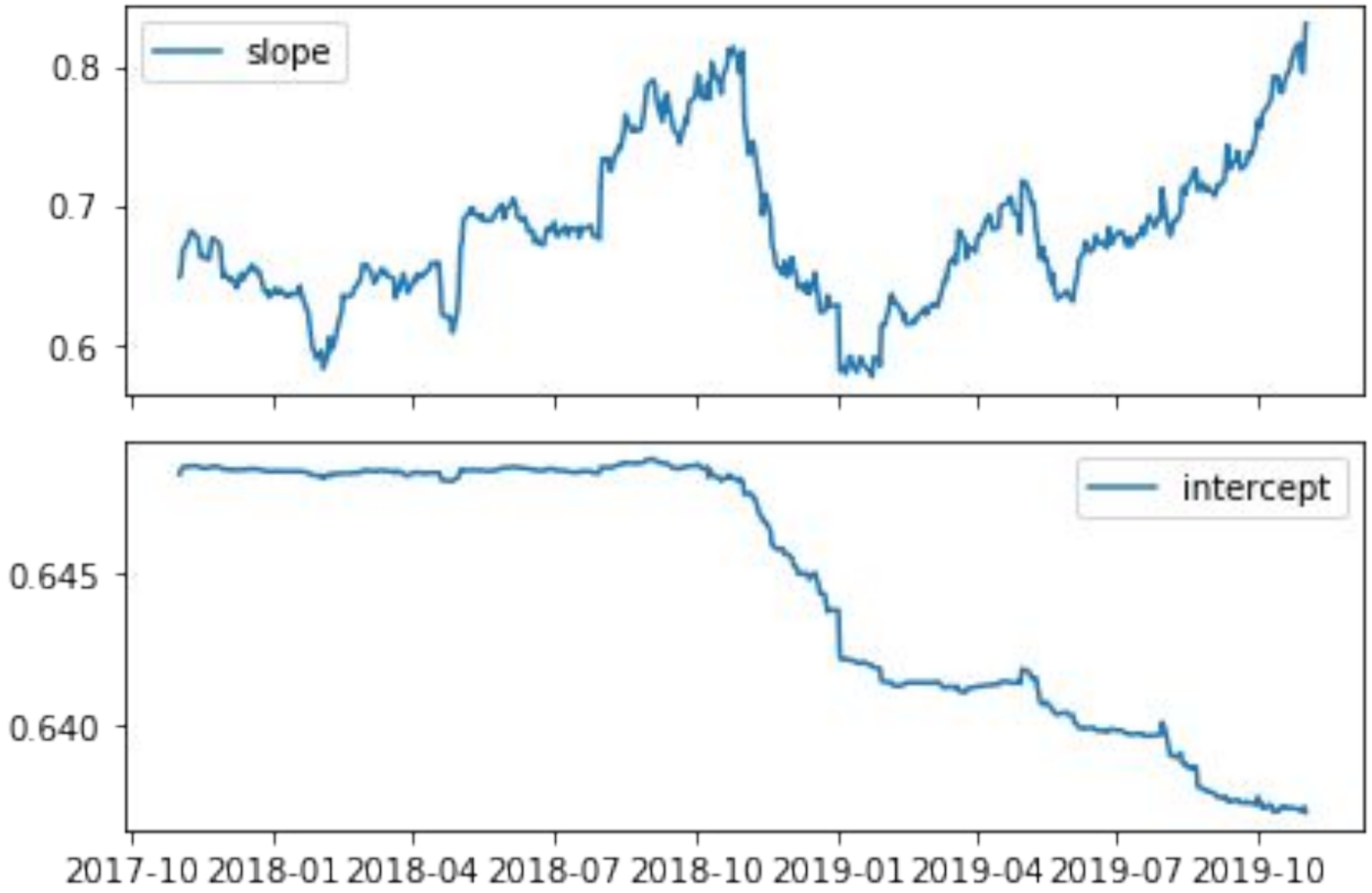
Code to estimate and plot α and β

```
delta = 1e-3
trans_cov = delta / (1 - delta) * np.eye(2) # How much random walk
wiggles
obs_mat = np.expand_dims(np.vstack([[x], [np.ones(len(x))]]).T, axis=1)

kf = KalmanFilter(n_dim_obs=1, n_dim_state=2, # y is 1-dimensional,
(alpha, beta) is 2-dimensional
                  initial_state_mean=[0,0],
                  initial_state_covariance=np.ones((2, 2)),
                  transition_matrices=np.eye(2),
                  observation_matrices=obs_mat,
                  observation_covariance=2,
                  transition_covariance=trans_cov)
# Use the observations y to get running estimates and errors for the
state parameters
state_means, state_covs = kf.filter(y.values)
To plot the means - our best estimates - of α and β over time
_, axarr = plt.subplots(2, sharex=True)
axarr[0].plot(x.index, state_means[:,0], label='slope')
axarr[0].legend()
axarr[1].plot(x.index, state_means[:,1], label='intercept')
axarr[1].legend()
plt.tight_layout();
```

Regression Coefficients: AAPL vs SPY

Alpha and Beta fluctuate a great deal over time.

If you are using beta for hedging within a trading strategy, it's critical to have an accurate and current estimate of beta.



Google Cloud

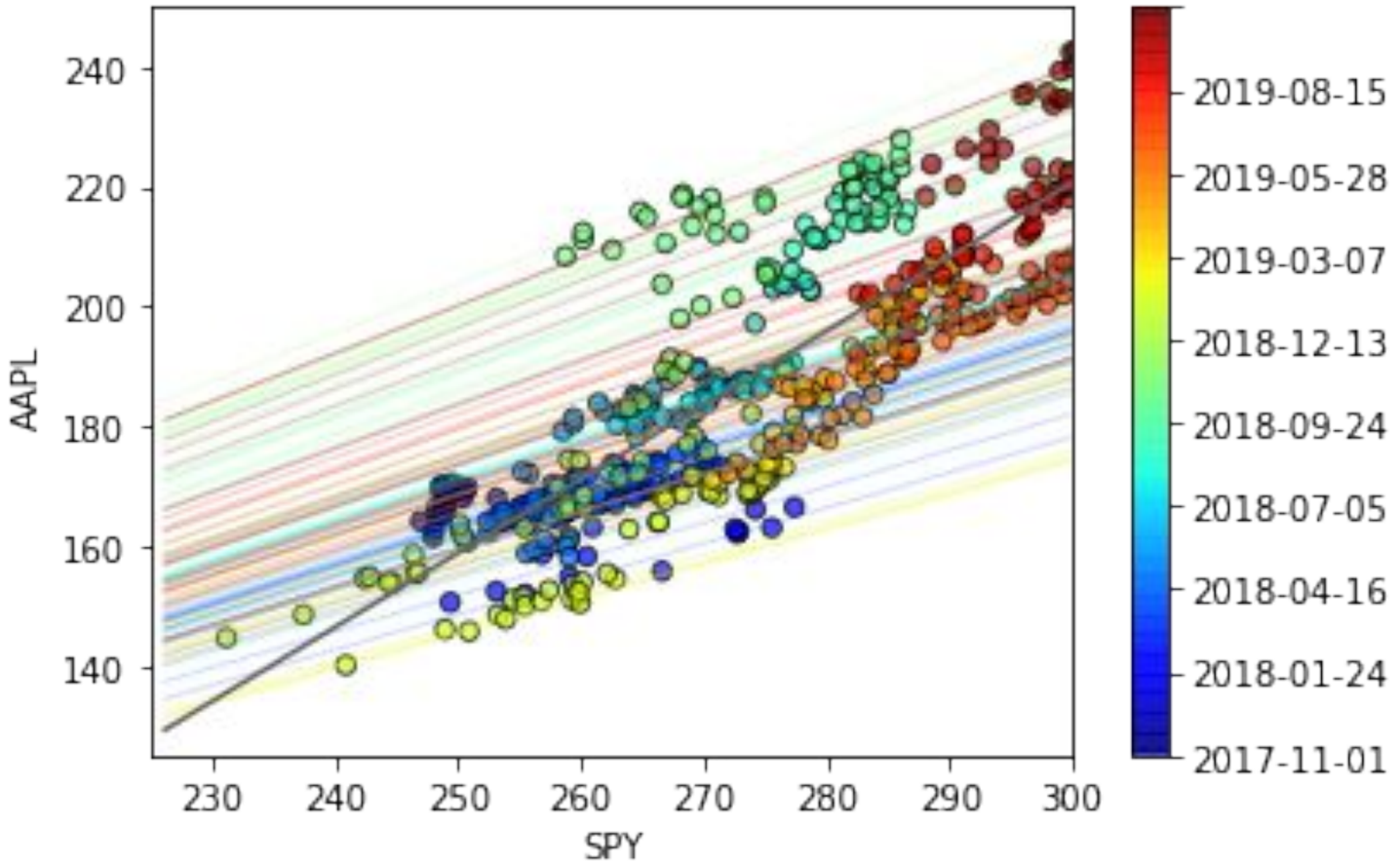
ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

Plot Kalman Estimate of Correlation: AAPL vs SPY

```
# Plot data points using colormap
sc = plt.scatter(x, y, s=30, c=colors, cmap=cm, edgecolor='k', alpha=0.7)
cb = plt.colorbar(sc)
cb.ax.set_yticklabels([str(p.date()) for p in x[::len(x)//9].index])

# Plot every fifth line
step = 5
xi = np.linspace(x.min()-5, x.max()+5, 2)
colors_l = np.linspace(0.1, 1, len(state_means[::step]))
for i, beta in enumerate(state_means[::step]):
    plt.plot(xi, beta[0] * xi + beta[1], alpha=.2, lw=1, c=cm(colors_l[i]))

# Plot the OLS regression line
plt.plot(xi, poly1d(np.polyfit(x, y, 1))(xi), '0.4')

# Adjust axes for visibility
plt.axis([225, 300, 125, 250])

# Label axes
plt.xlabel('SPY')
plt.ylabel('AAPL');
```

Regression Coefficients: AAPL vs SPY

Notice that although all of the state estimates take into account all previous observations, they fit the more recent data better than the older data. This allows the filter to adapt to structural changes in the data over time.



Use Returns of AAPL vs SPY to Plot Beta

```
# Get returns from pricing data
x_r = x.pct_change()[1:]
y_r = y.pct_change()[1:]

# Run Kalman filter on returns data
delta_r = 1e-2
trans_cov_r = delta_r / (1 - delta_r) * np.eye(2) # How much random walk wiggles
obs_mat_r = np.expand_dims(np.vstack([[x_r], [np.ones(len(x_r))]]).T, axis=1)
kf_r = KalmanFilter(n_dim_obs=1, n_dim_state=2, # y_r is 1-dimensional, (alpha, beta) is 2-dimensional
                    initial_state_mean=[0,0],
                    initial_state_covariance=np.ones((2, 2)),
                    transition_matrices=np.eye(2),
                    observation_matrices=obs_mat_r,
                    observation_covariance=.01,
                    transition_covariance=trans_cov_r)
```

```
To plot the means - our best estimates - of α and β over time
_, axarr = plt.subplots(2, sharex=True)
axarr[0].plot(x.index, state_means[:,0], label='slope')
axarr[0].legend()
axarr[1].plot(x.index, state_means[:,1], label='intercept')
axarr[1].legend()
plt.tight_layout();
```

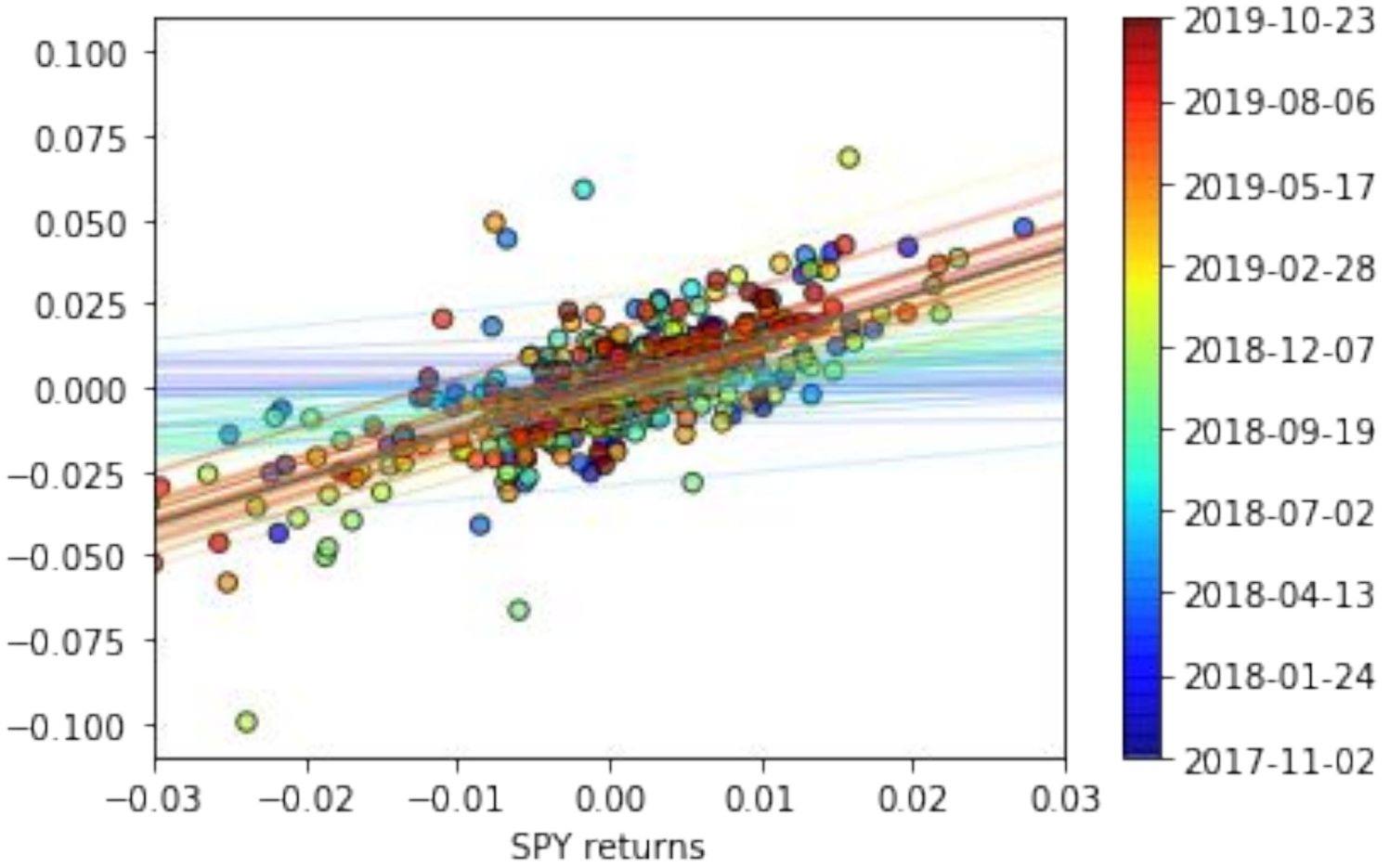
Google Cloud

ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

Kalman Estimates of Beta: AAPL vs SPY

With returns the data is less dispersed but you can clearly see that the Kalman slopes are close to zero initially.

Over time they increase until their slopes are about equal to the OLS regression (black line).



Google Cloud

ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

Further Reading

Kalman Filters

By Evgenia "Jenny" Nitishinskaya, Dr. Aidan O'Mahony, and Delaney Granizo-Mackenzie. Algorithms by David Edwards.

Lab

Momentum Trading Strategy

Google Cloud

ESTABLISHED 1922
NEW YORK INSTITUTE OF FINANCE

Lab Objectives

- Learn how a momentum trading strategy is implemented
- Modify a piece of code such that losing momentum trading strategy is turned into a strategy that makes money

Lab Objectives

- Learn how a momentum trading strategy is implemented
- Modify a piece of code such that losing momentum trading strategy is turned into a strategy that makes money

Lab Objectives

- Learn how a momentum trading strategy is implemented
- Modify a piece of code such that losing momentum trading strategy is turned into a strategy that makes money

Screencast