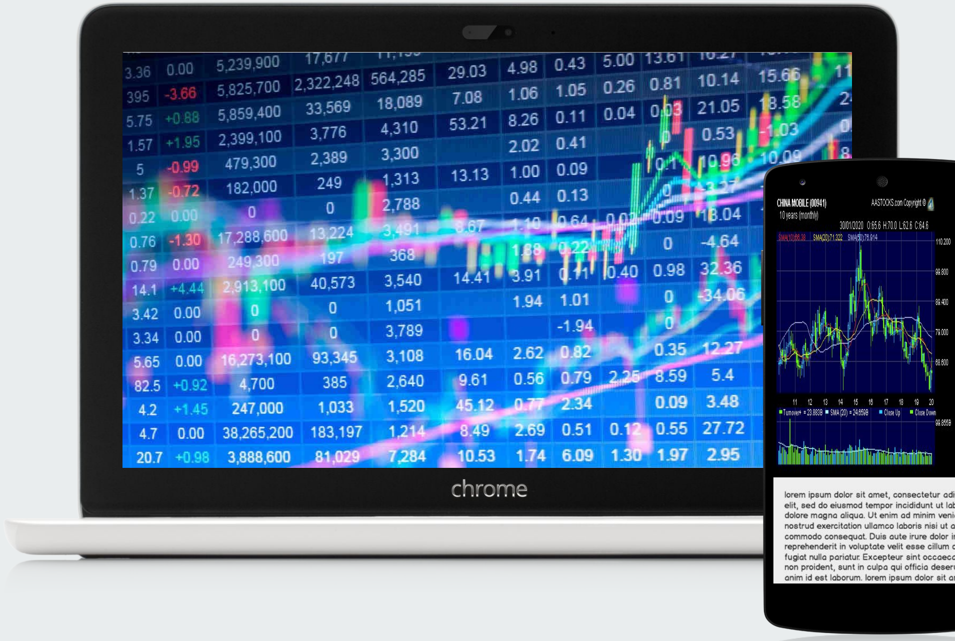
Stock Price Prediction

USING

MACHINE & DEEP LEARNING TECHNIQUES



Outline

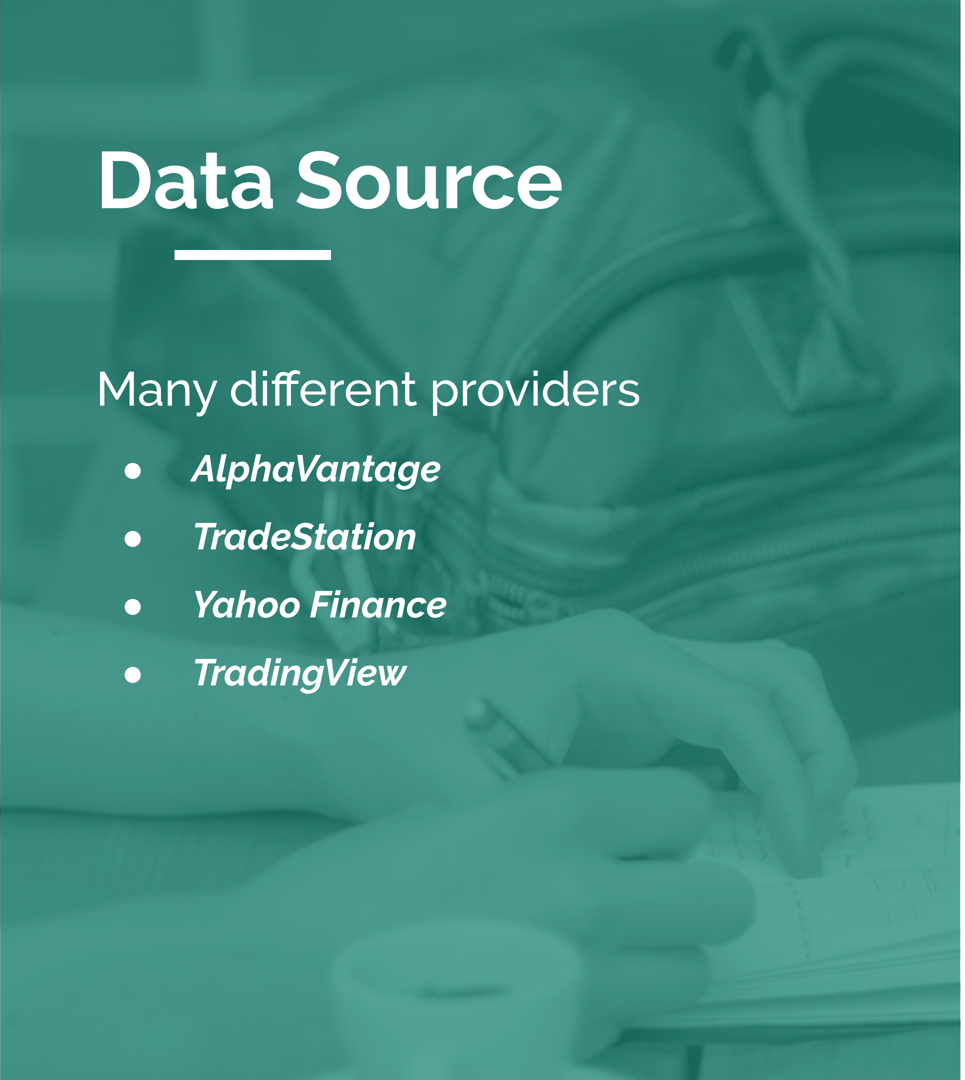
The Problem

Using traditional technical analysis to predict a stock performance is extremely challenging

Many different factors affect stock price

- ***Company news and performance***
- ***Industry performance***
- ***Investor sentiments***
- ***Economic factors***

How can we use ML and DL to complement TA to enhance decision making in trading decisions and strategies?



Data Source

Many different providers

- ***AlphaVantage***
- ***TradeStation***
- ***Yahoo Finance***
- ***TradingView***

Integrated suite - Data engineering, feature engineering, technical indicators, strategy testing, backtesting, etc

Key considerations

- Store in RDBMS for efficiency and effectiveness
- Apply what I've learned in course

Data Source

Which markets?

- US (NASDAQ, NYSE)
- HK (HSI)

Which intervals?

- Daily (EOD)
- Intraday - 1 minute (Live Streaming)

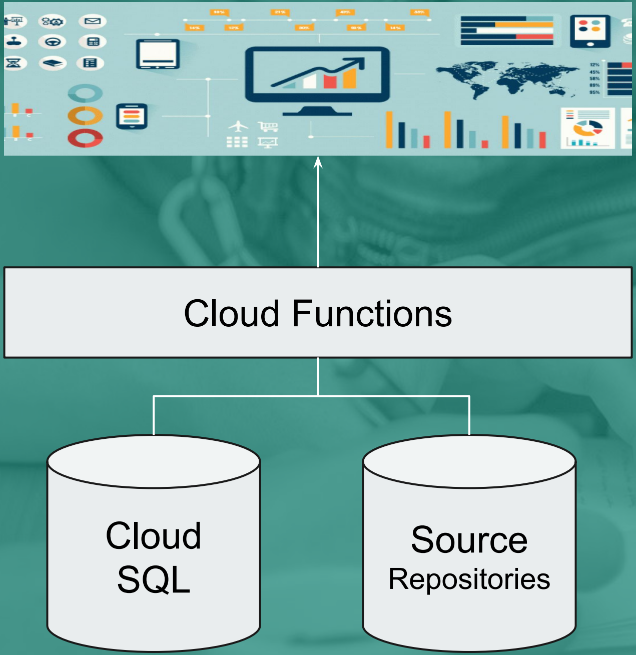
Which sectors?

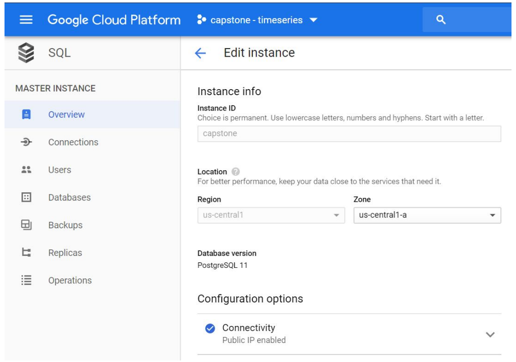
- Tech
- Utilities
- Consumers
- Multi

Which companies?

- Apple, Amazon, Microsoft
- NextEra
- Procter & Gamble
- General Electric

Data Source





Edit instance

Instance info

Instance ID
Choice is permanent. Use lowercase letters, numbers and hyphens. Start with a letter.
capstone

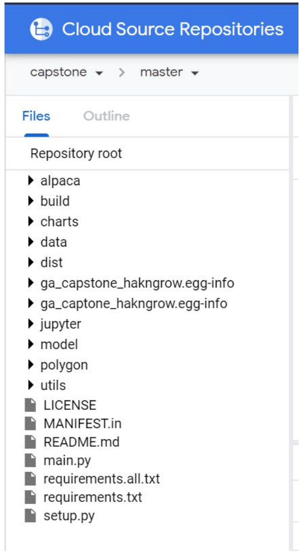
Location
For better performance, keep your data close to the services that need it.

Region: us-central1
Zone: us-central1-a

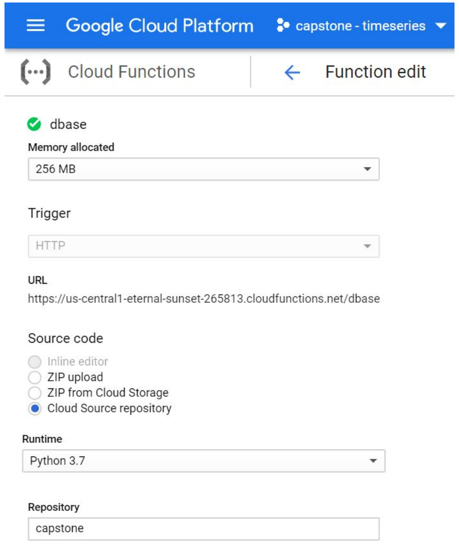
Database version
PostgreSQL 11

Configuration options

Connectivity
Public IP enabled



```
Repository root
▸ alpaca
▸ build
▸ charts
▸ data
▸ dist
▸ ga_capstone_hakngrow.egg-info
▸ ga_captone_hakngrow.egg-info
▸ jupyter
▸ model
▸ polygon
▸ utils
  LICENSE
  MANIFEST.in
  README.md
  main.py
  requirements.all.txt
  requirements.txt
  setup.py
```



Function edit

dbase

Memory allocated: 256 MB

Trigger: HTTP

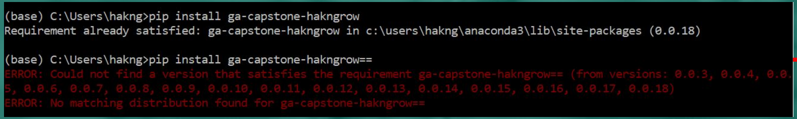
URL
https://us-central1-eternal-sunset-265813.cloudfunctions.net/dbase

Source code
- Inline editor
- ZIP upload
- ZIP from Cloud Storage
- [✓ Cloud Source repository]

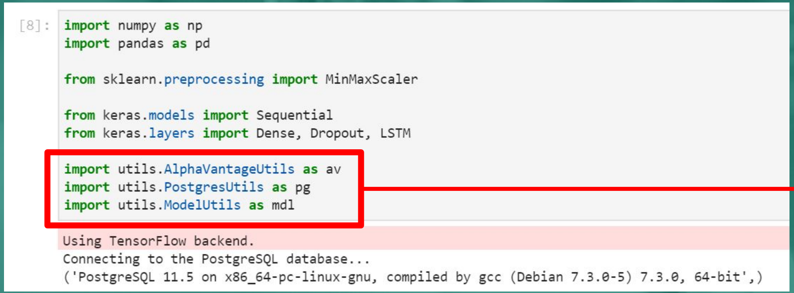
Runtime: Python 3.7

Repository: capstone

Data Source

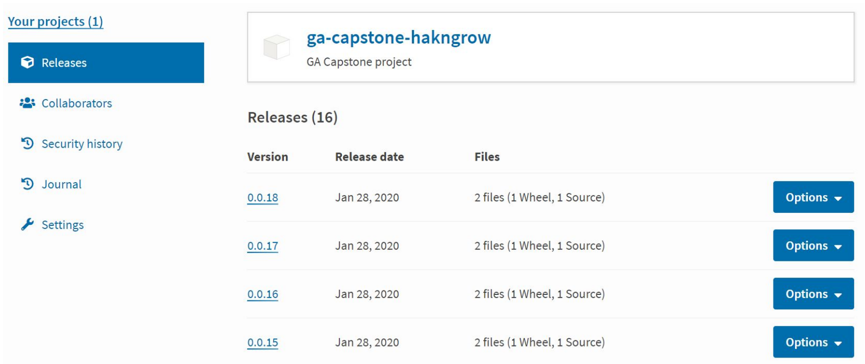


```
(base) C:\Users\hakng>pip install ga-capstone-hakngrow
Requirement already satisfied: ga-capstone-hakngrow in c:\users\hakng\anaconda3\lib\site-packages (0.0.18)

(base) C:\Users\hakng>pip install ga-capstone-hakngrow==
ERROR: Could not find a version that satisfies the requirement ga-capstone-hakngrow== (from versions: 0.0.3, 0.0.4, 0.0.
5, 0.0.6, 0.0.7, 0.0.8, 0.0.9, 0.0.10, 0.0.11, 0.0.12, 0.0.13, 0.0.14, 0.0.15, 0.0.16, 0.0.17, 0.0.18)
ERROR: No matching distribution found for ga-capstone-hakngrow==
```



```
[8]: import numpy as np
     import pandas as pd

     from sklearn.preprocessing import MinMaxScaler

     from keras.models import Sequential
     from keras.layers import Dense, Dropout, LSTM

     import utils.AlphaVantageUtils as av
     import utils.PostgresUtils as pg
     import utils.ModelUtils as mdl

     Using TensorFlow backend.
     Connecting to the PostgreSQL database...
     ('PostgreSQL 11.5 on x86_64-pc-linux-gnu, compiled by gcc (Debian 7.3.0-5) 7.3.0, 64-bit',)
```

Retrieve data directly from Jupyter Lab/Notebook



ga-capstone-hakngrow

GA Capstone project

Releases (16)

Version	Release date	Files	
0.0.18	Jan 28, 2020	2 files (1 Wheel, 1 Source)	Options
0.0.17	Jan 28, 2020	2 files (1 Wheel, 1 Source)	Options
0.0.16	Jan 28, 2020	2 files (1 Wheel, 1 Source)	Options
0.0.15	Jan 28, 2020	2 files (1 Wheel, 1 Source)	Options

Deployed on the Python Package Index

Feature Engineering

Date features

- Year, Month, Day
- Start of year, End of year
- Start of quarter, End of quarter
- Start of month, End of Month
- Start of week, End of week

Event features

- Company results/announcements
- Industry performance e.g. New home sales, crude oil inventories, etc
- Investor sentiments e.g. consumer confidence, business confidence,, Twitter, etc
- Economic factors e.g. FOMC, PMI, Goods trade balance, etc

Feature Engineering

- Continuous values e.g. RSI value of stock on a particular date, etc
- Binary or categorical e.g. Was the company results within expectations, was there a MACD crossover, etc
- Min/Max, standard scaling
- Traditional + ML features

Technical Indicator features

- Hundreds to choose from
- Trend - MACD (Lag)
- Momentum - RSI (Lead)
- Volatility
- Volume

Custom features

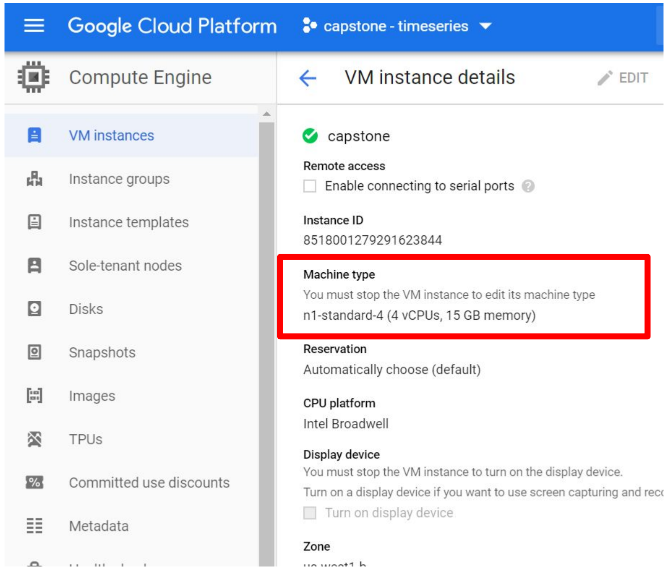
- Nvidia
- Game releases i.e. ShackNews

Modelling

Models used

- ***Linear Regression***
- ***KNN***
- ***Auto Arima***
- ***LSTM***

Companies - AAPL, AMZN, MSFT, PG, GE, NEE
Intervals - Daily, 1 min
Horizon - 20 years, 5 days
Price features - OHLC, C
Date features - YMD, DoY, WoY, DoY, SoY, EoY, etc
Indicator features - RSI, SMA
Metrics - RMSE



capstone

Remote access
Enable connecting to serial ports

Instance ID
8518001279291623844

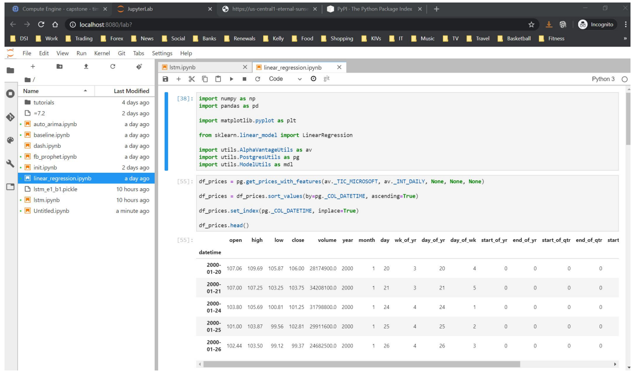
Machine type
You must stop the VM instance to edit its machine type
n1-standard-4 (4 vCPUs, 15 GB memory)

Reservation
Automatically choose (default)

CPU platform
Intel Broadwell

Display device
You must stop the VM instance to turn on the display device.
Turn on a display device if you want to use screen capturing and rec
Turn on display device

Zone



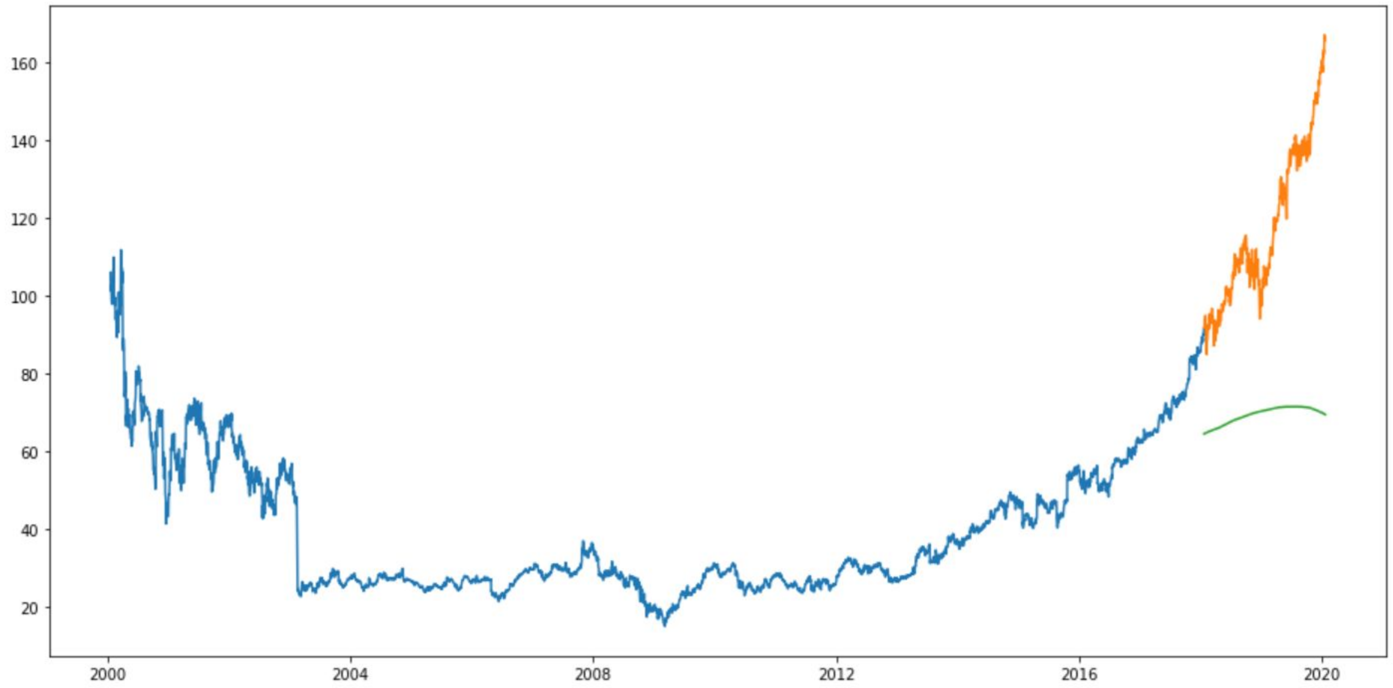
Name	Last Modified
tutorials	4 days ago
=7.2	2 days ago
auto_arima.ipynb	a day ago
baseline.ipynb	a day ago
dash.ipynb	a day ago
fb_prophet.ipynb	a day ago
init.ipynb	2 days ago
linear_regression.ipynb	a day ago
lstm_e1_b1.pickle	10 hours ago
lstm.ipynb	10 hours ago
Untitled.ipynb	a minute ago

```
[38]: import numpy as np
      import pandas as pd

      import matplotlib.pyplot as plt

      from sklearn.linear_model import LinearRegression

      import utils.AlphaVantageUtils as av
      import utils.PostgresUtils as pg
      import utils.ModelUtils as mdl
```

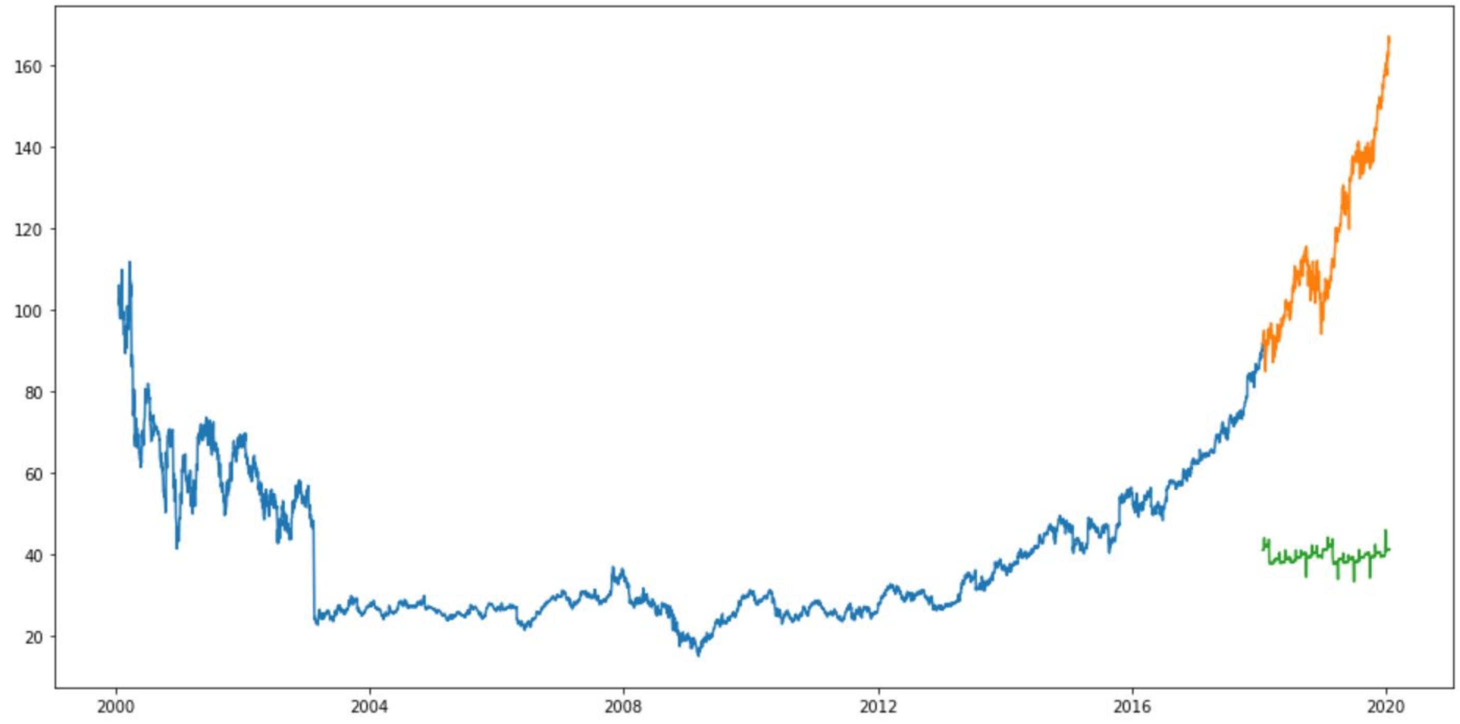
```
[55]: df_prices = pg.get_prices_with_features(av._TIC_MICROSOFT, av._INT_DAILY, None, None, None)

      df_prices = df_prices.sort_values(by=pg._COL_DATETIME, ascending=True)

      df_prices.set_index(pg._COL_DATETIME, inplace=True)

      df_prices.head()
```

```
[55]:          open    high    low    close      volume  year  month  day  wk_of_yr  day_of_yr  day_of_wk  start_of_yr  end_of_yr  start_of_qtr  end_of_qtr  start
   datetime
   2000-01-20  107.06  109.69  105.87  106.00  28174900.0  2000      1   20         3         20          4            0          0             0           0
   2000-01-21  107.00  107.25  103.25  103.75  34208100.0  2000      1   21         3         21          5            0          0             0           0
   2000-01-24  103.80  105.69  100.81  101.25  31798800.0  2000      1   24         4         24          1            0          0             0           0
   2000-01-25  101.00  103.87   99.56  102.81  29911600.0  2000      1   25         4         25          2            0          0             0           0
   2000-01-26  102.44  103.50   99.12   99.37  24682500.0  2000      1   26         4         26          3            0          0             0           0
```

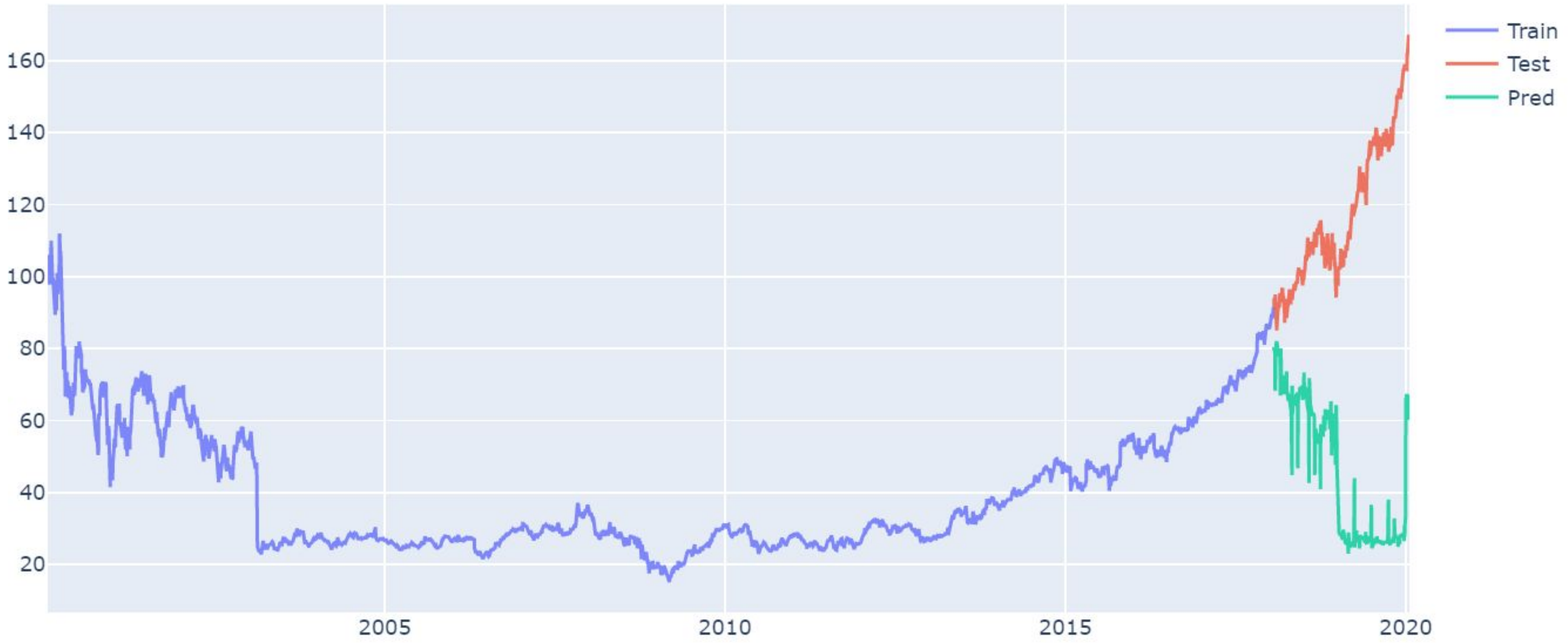
Model - Baseline



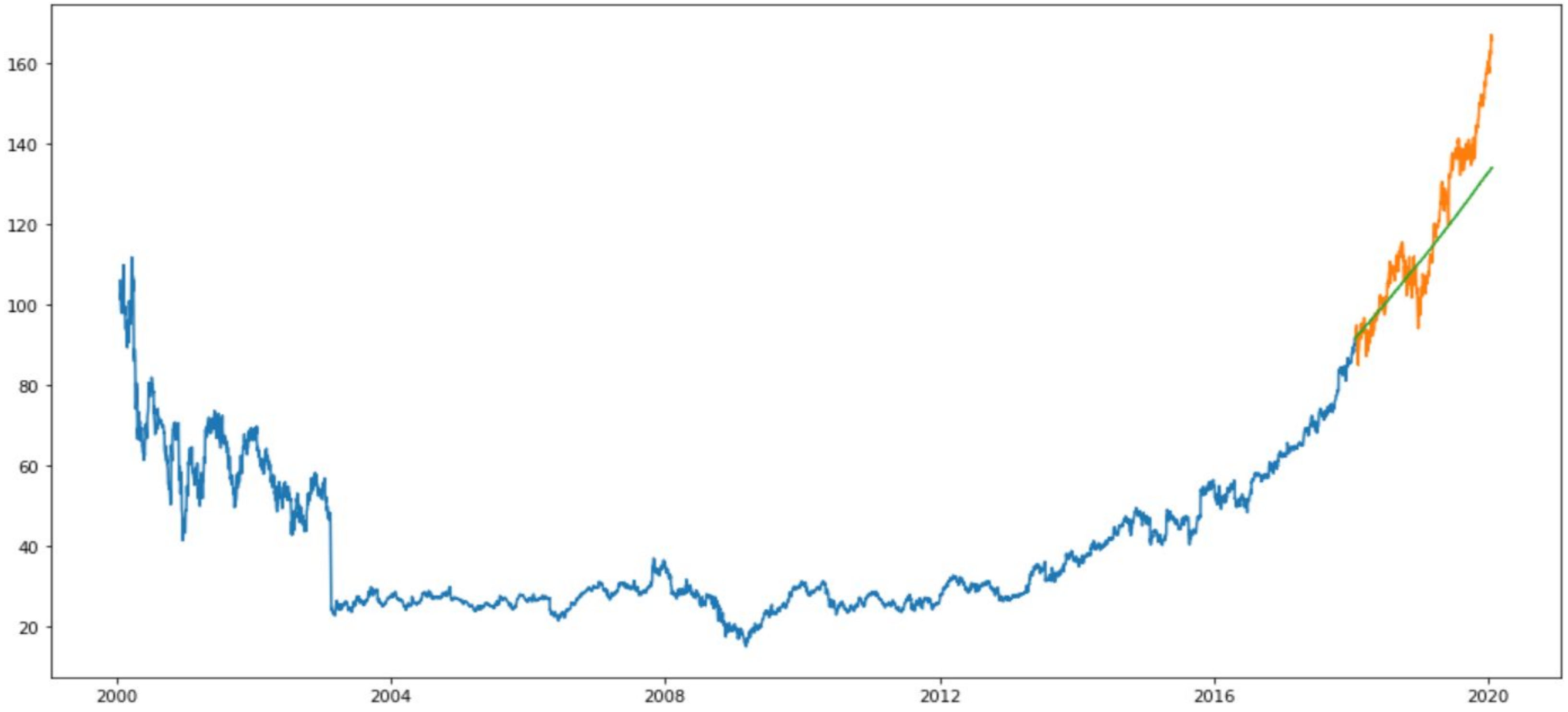
Model - Linear Regression



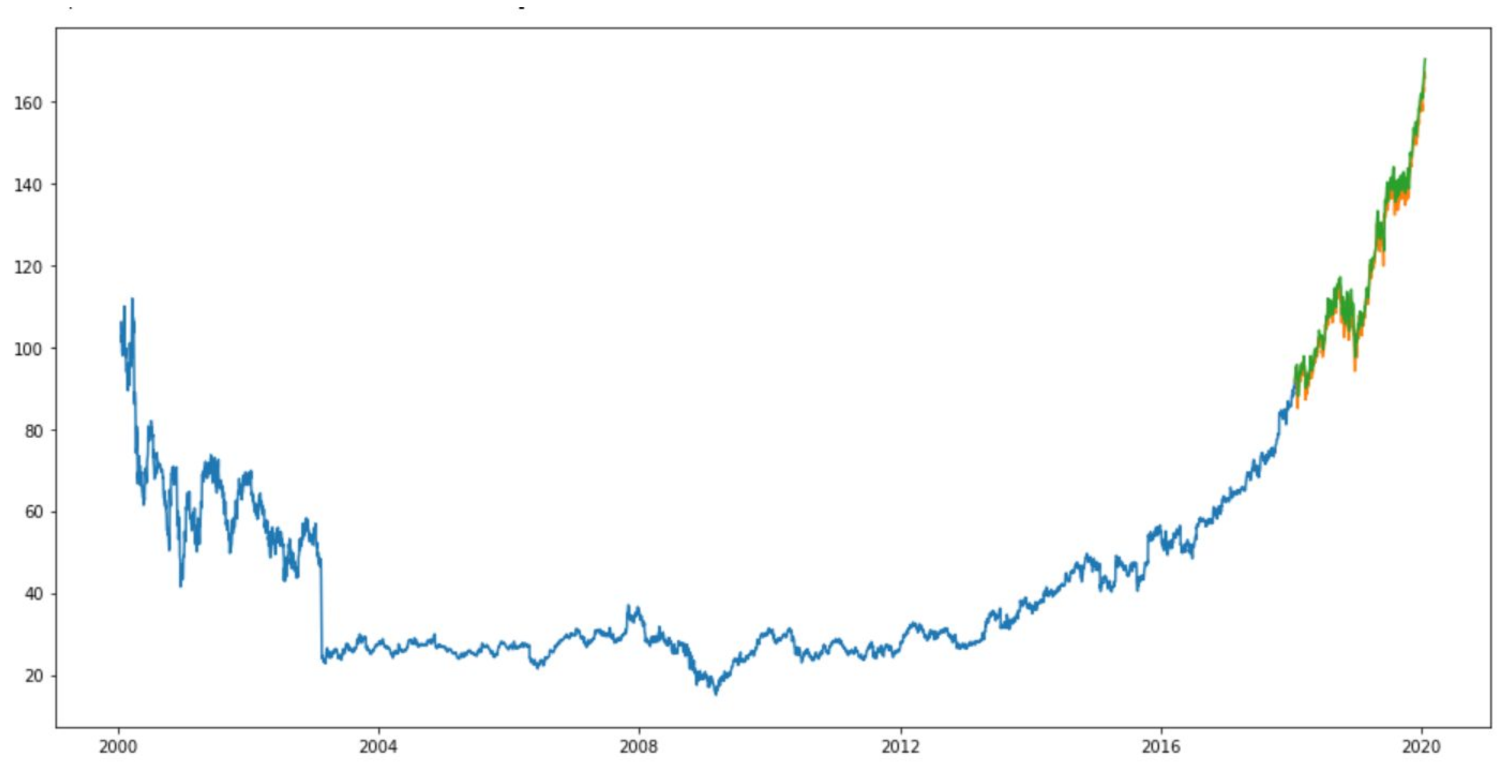
Model - KNN



Model - Auto Arima



Model - LSTM

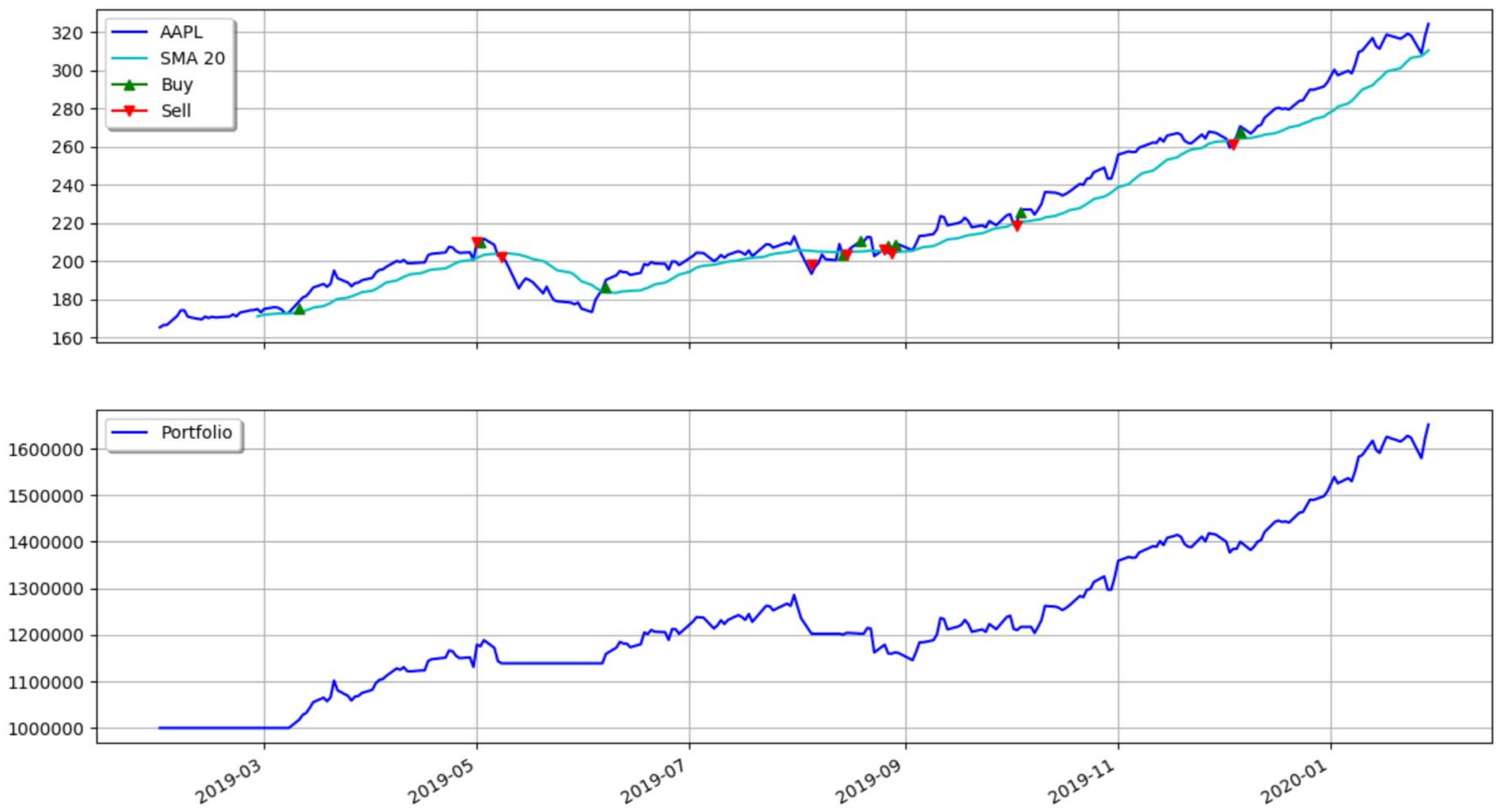


Strategy & Back-Testing

- Use LSTM model to complement traditional TA signals
- Use PyAlgoTrade frame for testing strategy and back-testing
- Simplify trading i.e. only 1 position open, no commission, no slippage.

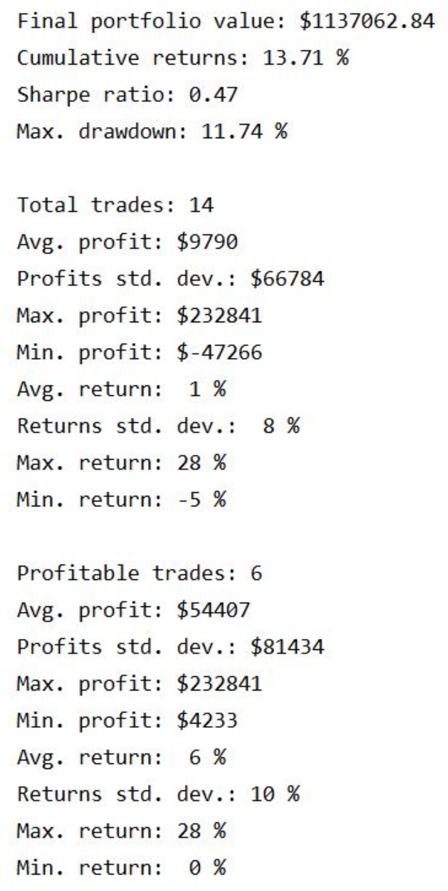
Rules

- An SMA period, entrySMA, for trend identification
- A smaller SMA, exitSMA, period for the exit point
- An RSI period, rsiPeriod, for entering both short/long positions
- An RSI oversold threshold, overSoldThreshold, for long position entry
- An RSI overbought threshold, overBoughtThreshold, for short position entry
- **Confirm position entry against LSTM model e.g. price is predicted to make a >= 10% move in the direction of the trade within 3 days**



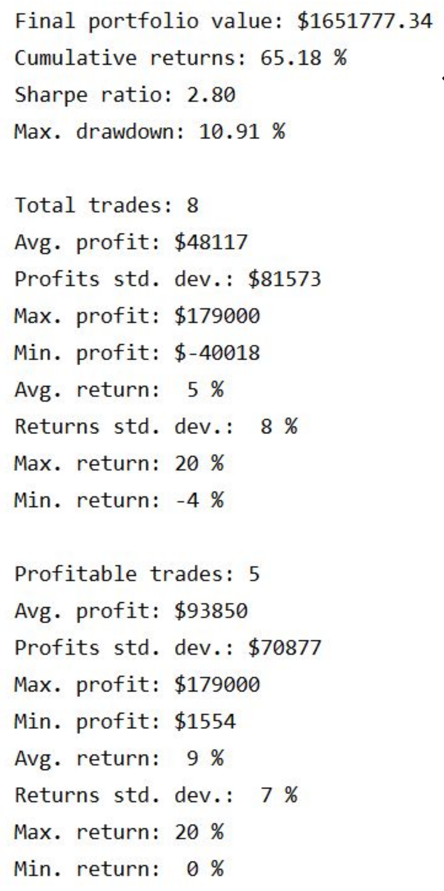
AAPL
SMA 20
Buy
Sell
Portfolio
320
300
280
260
240
220
200
180
160
1600000
1500000
1400000
1300000
1200000
1100000
1000000
2019-03
2019-05
2019-07
2019-09
2019-11
2020-01

Strategy



```
Final portfolio value: $1137062.84
Cumulative returns: 13.71 %
Sharpe ratio: 0.47
Max. drawdown: 11.74 %

Total trades: 14
Avg. profit: $9790
Profits std. dev.: $66784
Max. profit: $232841
Min. profit: $-47266
Avg. return:  1 %
Returns std. dev.:  8 %
Max. return: 28 %
Min. return: -5 %

Profitable trades: 6
Avg. profit: $54407
Profits std. dev.: $81434
Max. profit: $232841
Min. profit: $4233
Avg. return:  6 %
Returns std. dev.: 10 %
Max. return: 28 %
Min. return:  0 %
```

Without LSTM



```
Final portfolio value: $1651777.34
Cumulative returns: 65.18 %
Sharpe ratio: 2.80
Max. drawdown: 10.91 %

Total trades: 8
Avg. profit: $48117
Profits std. dev.: $81573
Max. profit: $179000
Min. profit: $-40018
Avg. return:  5 %
Returns std. dev.:  8 %
Max. return: 20 %
Min. return: -4 %

Profitable trades: 5
Avg. profit: $93850
Profits std. dev.: $70877
Max. profit: $179000
Min. profit: $1554
Avg. return:  9 %
Returns std. dev.:  7 %
Max. return: 20 %
Min. return:  0 %
```

With LSTM

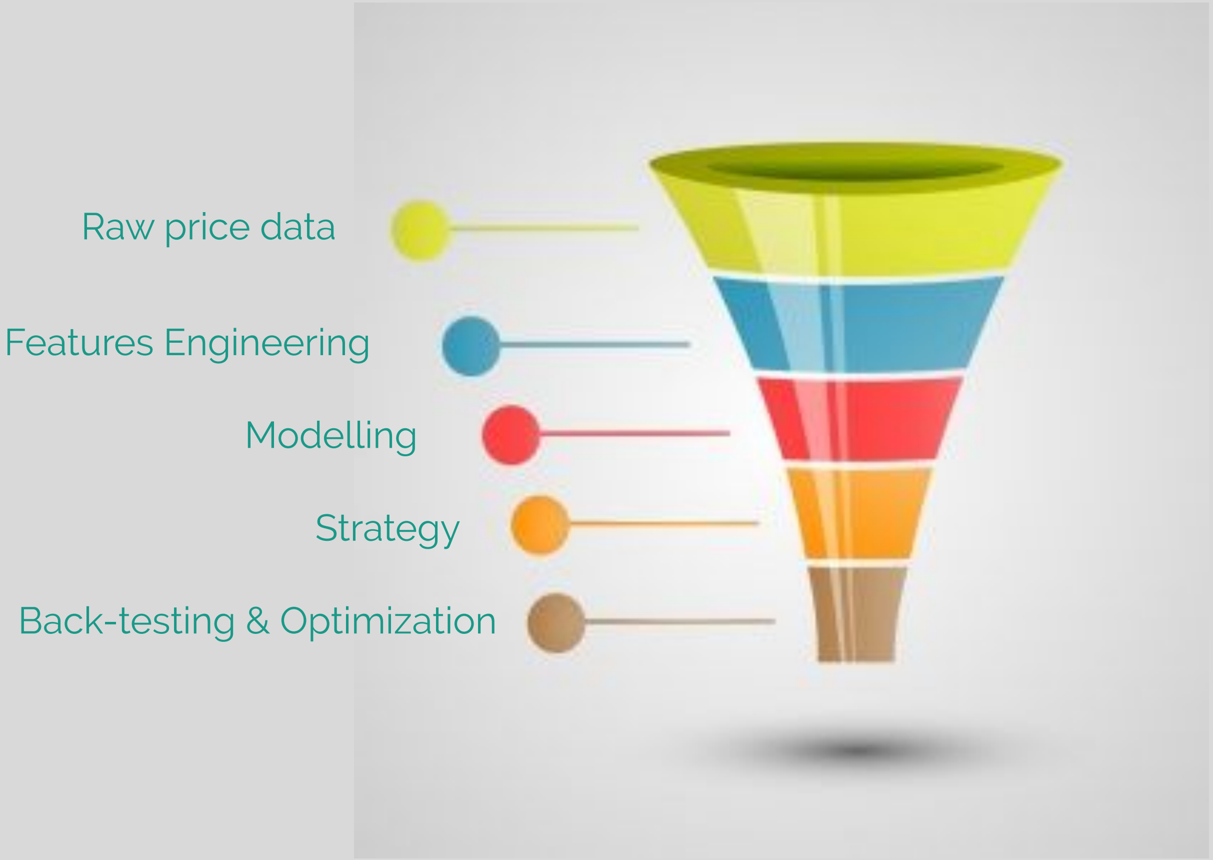
Strategy

Optimization

- In excess of 4,000,000 combinations
- Each combination takes 0.1 seconds
- It would take 4.6 days to find the set of best parameters using a single core computer
- Use PyAlgoTrade parallel processing framework

Parameters

- entrySMA [150 to 250]
- exitSMA [5 to 20]
- rsiPeriod [2 to 10]
- overSoldThreshold [5 to 25]
- overBoughtThreshold [75 to 95]
- **predictedPriceDelta [10 to 20]**
- **predictedPricePeriod [3 to 5]**


Raw price data
Features Engineering
Modelling
Strategy
Back-testing & Optimization

Conclusion