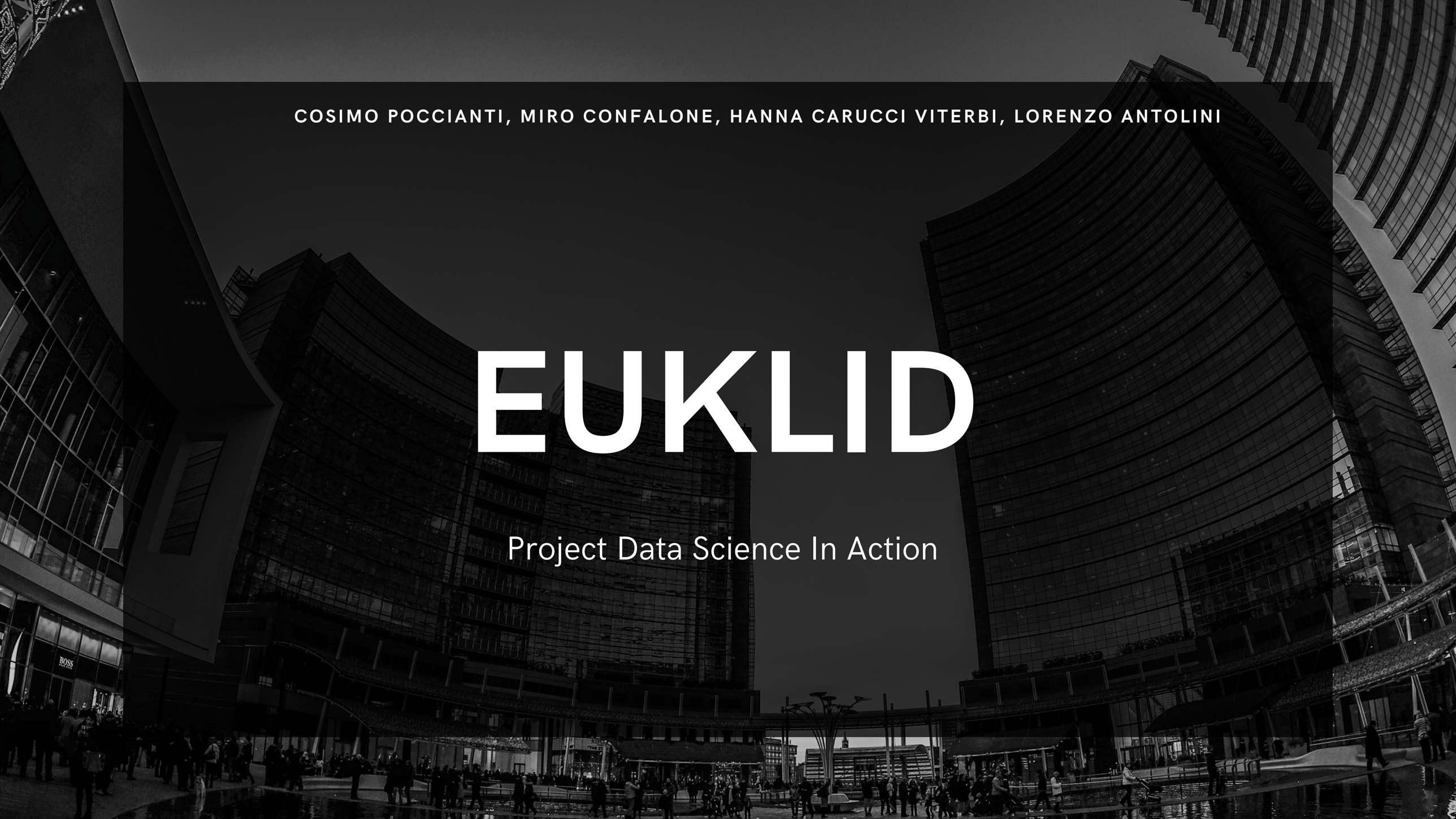
COSIMO POCCIANTI, MIRO CONFALONE, HANNA CARUCCI VITERBI, LORENZO ANTOLINI

EUKLID

Project Data Science In Action



LAYOUT

project's objectives

NX4 results

N algorithms have been run on each dataset to detect particular patterns and differences between the times series. Different approaches have been explored.

Ensemble model

The models have been used as weak learners for an Ensmble Model (Random Forest) with all the outputs of the algorithms

Results

Two main models with high accuracies and performances have been obtained: one as en ensemble model of all the algorithms and the second with the LSTM.

Asset

EUKLID

Historical series from December 1999 to February 2022

IBM

Stock

IBM is a multinational technology company operating in 171 countries. The primary market for trading the stock is the NYSE.

NASDAQ

Index

Nasdaq is an online global marketplace for buying and trading securities and the world's first electronic exchange.

GOLD

Commodity

In the free market system, gold is a currency. Gold has a price, and that price will fluctuate relative to other forms of exchange, such as the U.S. dollar, the euro, and the Japanese yen.

WTI

Commodity

WTI is the underlying commodity of the New York Mercantile Exchange's (NYMEX) oil futures contract and is considered a high-quality oil that is easily refined.

LUISS

DATA COLLECTION

Historical values of IBM, NASDAQ, GOLD, and WTI have been downloaded from Yahoo Finance API. The adjusted close values were provided by Euklid.

All datasets include:

- Date: the date to which the row refers
- Open: the daily opening price of the underlying asset
- Close: the daily closing price of the underlying asset
- AdjClose: the daily closing price after adjustments for all applicable splits and dividend distributions
- High: the highest daily price that the underlying asset has reached
- Low: the lowest daily price that the underlying asset has reached
- Volume: the daily number of shares traded in a particular stock, index, or other investment.

DATA MANIPULATION

TA-lib library and machine learning models require clean data to work.

The main steps to make the data as usable as possible are:

1. Removing Nan values
2. Transform the "Date" column in DateTime format and set it as index
3. Turn the remaining columns into float values
4. Computed new columns from the initial ones: spread open/close, spread high/low, daily returns (and their log)

LUISS

INDICATORS

TA-Lib is an open-source python library that is used in analyzing the stock market's historical data like share price, volume, etc.

OUR INDICATORS:

STOCH D	STOCH K
EMA	RSI
TEMA	OBV
BOLLINGER BANDS	TSF
MOM	WILLR
MOVING AVERAGE	ROC
CCI	MACD

RULES

Rules for each indicator have been created. In this way for each observation we obtained three different values as output:

- 1 if the indicator value was translatable as a long signal
- -1 if the indicator value was translatable as a short signal
- 0 if the value of the indicator assumed a neutral value

Example

Regarding RSI, a value > 70 indicates an *overbought* situation, so the value of the stock will probably fall in the following days.
Differently, an RSI value < 30 indicates an *oversold* situation, so the stock price is likely to rise in the coming days.

LUISS

Index

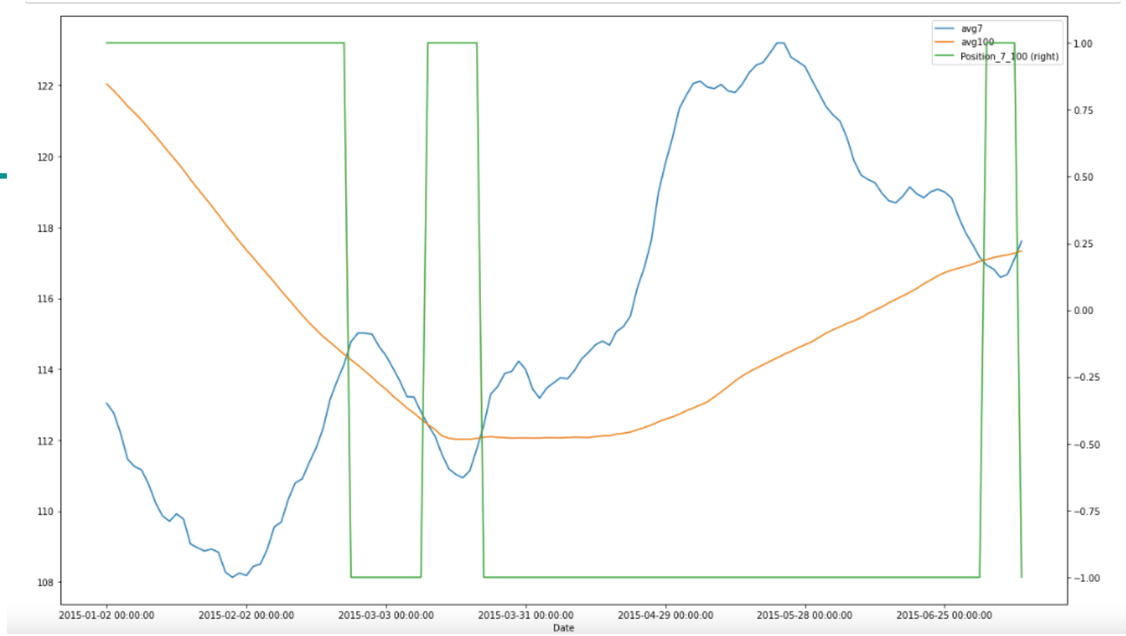
Algorithms

Simple Moving Average

What is SMA?

In its simplest form, this strategy is expressed as buying (or selling) when the short-period moving averages rises above (or falls below) the long period moving averages

an example of avg(7)-avg(100)



Calculated on:

Average 7
Average 10
Average 50
Average 100
Average 200
Average 253
*rolling average of the Adjusted Close

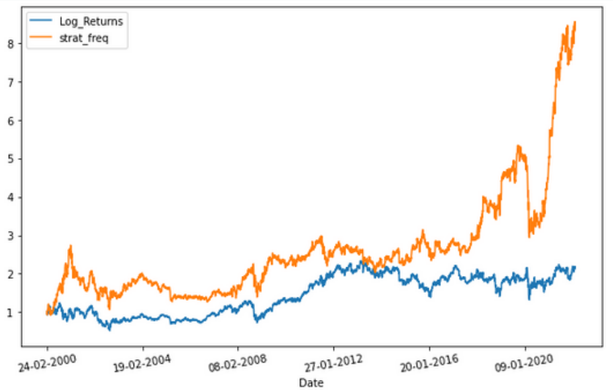
Ensembling the Positions

Then a Random Forest has ben runned on the Positions to have an ensemble return of them.

Frequency Approach

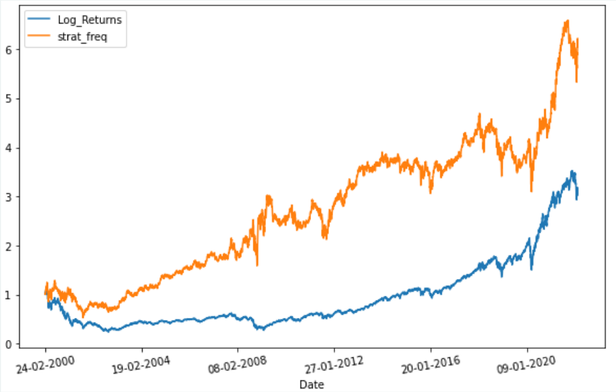
- Digitizes the feature values given the bins parameter (Lags)
- shows the digitized features values and label values
- Shows the frequency of the possible movements conditional on the feature value combinations.
- It is calculated on the signals of the two lags, and if for the two consecutive days the signal is positive (so if the sum is equal to two), it will return a -1, otherwise a 1.

IBM

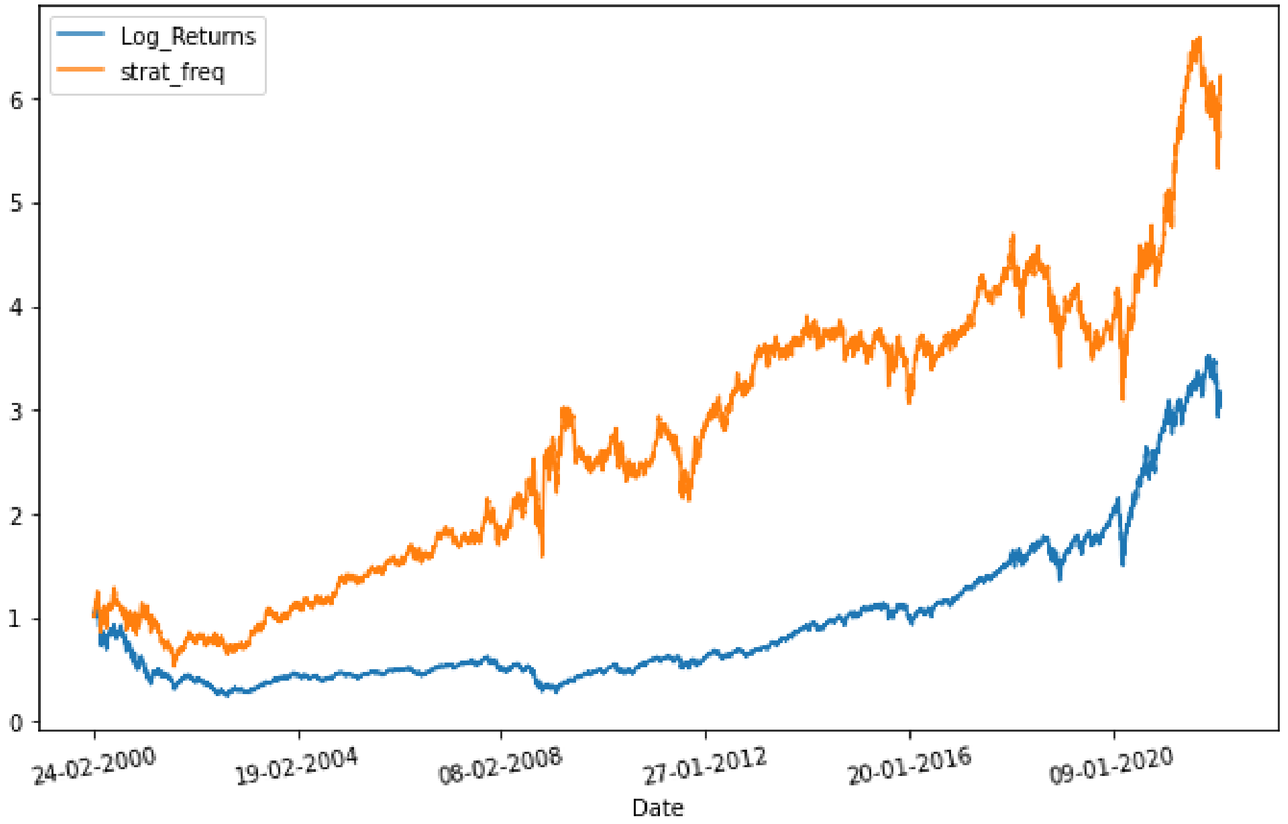


WTI

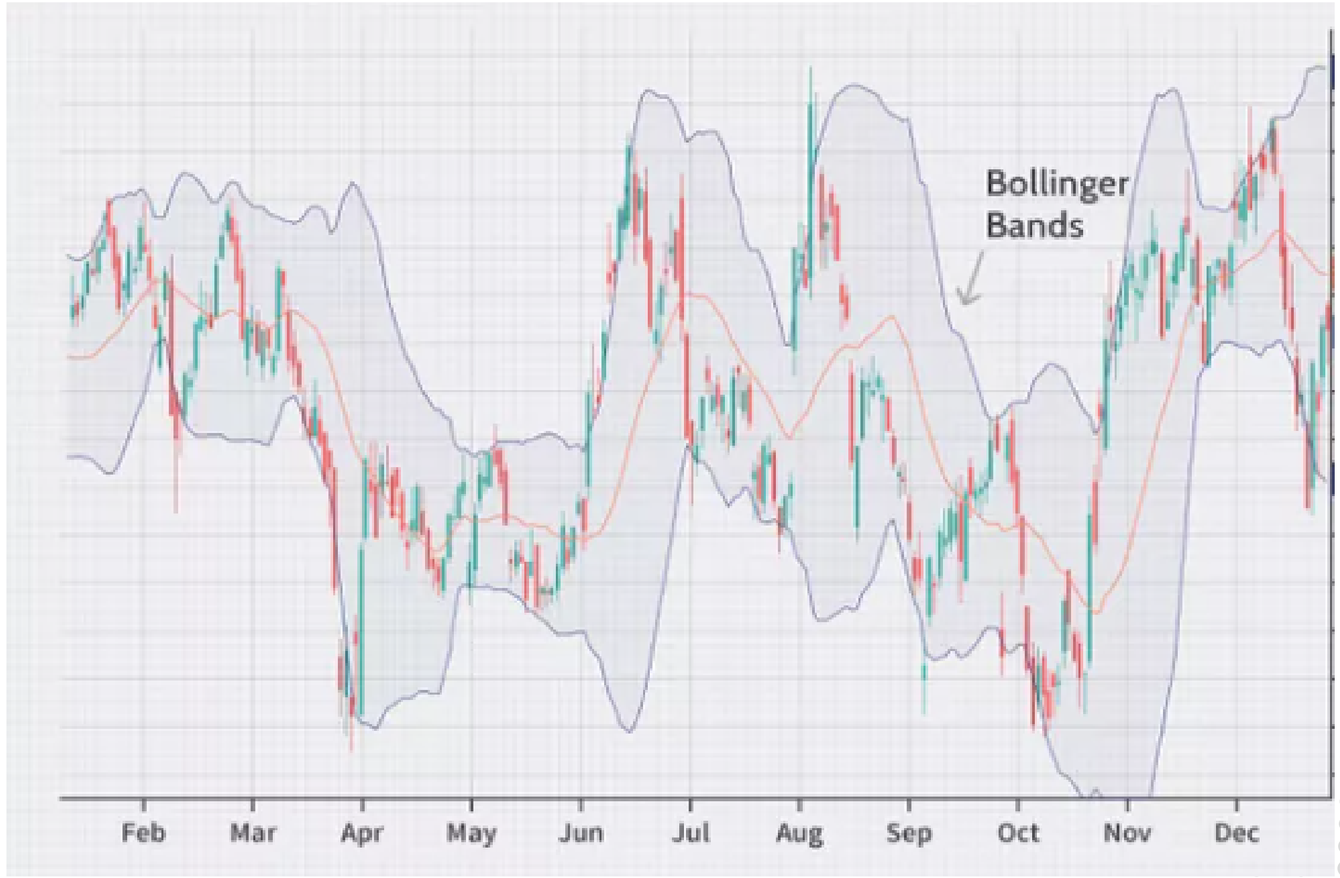
NASDAQ



GOLD


Log_Returns
strat_freq
6
5
4
3
2
1
0
24-02-2000
19-02-2004
08-02-2008
27-01-2012
20-01-2016
09-01-2020
Date

Bolinger Bands



1.Created an "empty" column as placeholder for the position signals

2. Fill the newly created position column - to sell (-1) when the price hits the upper band, and set to buy (1) when it hits the lower band

3. Forward fill the position column to replace the "None" values with the correct long/short positions to represent the "holding" of our position

LUISS

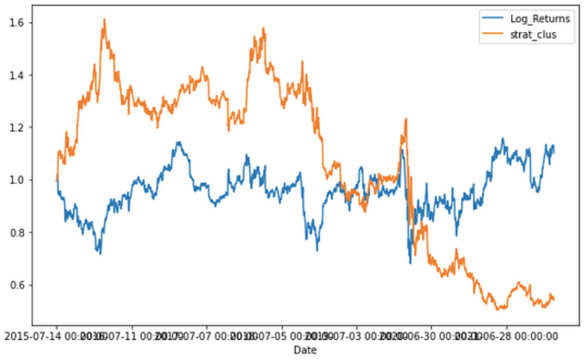
K-means

Features: K-means algorithm was run on the signal of 2 lags (log returns shifted respectively by one and two days).

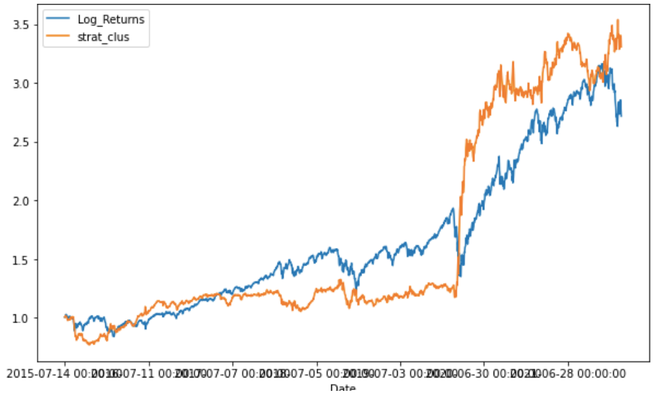
K = 2

	TRAIN	TEST
IBM	0.487	0.486
NASDAQ	0.498	0.537
GLD	0.480	0.493
WTI	0.514	0.506

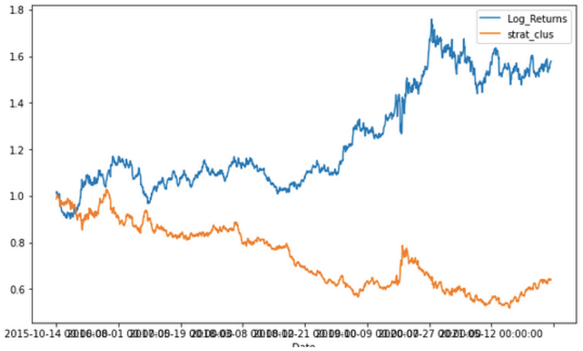
IBM



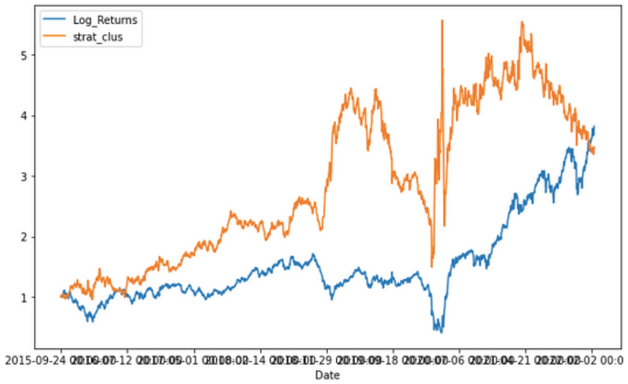
NASDAQ



GOLD



WTI



	On two lag of log return singals		On five lag of log return signals		On five lag of log returns digitized	
	train	**test**	**train**	**test**	**train**	**test**
IBM	0.513	0.532	0.513	0.518	0.516	0.526
NASDQ	0.534	0.566	0.534	0.566	0.535	0.567
GLD	0.531	0.532	0.528	0.528	0.536	0.517
WTI	0.521	0.517	0.526	0.524	0.535	0.513

[μ-σ, μ, μ+σ]

Baseline

LUISS

Logistic Regression

Variables

Two Lag of Log Return signals: the signal of 2 lags (Log Returns shifted respectively by one and two days).
Five Lag of Log Return Signals: the signal of 5 lags (Log Returns shifted respectively by one, two, three, four and five days).
Five Lag of Log Returns Digitized: defined an interval with [mu-v,mu,mu+v] where mu is the mean and v is the standard deviation of the log returns. To each lag a number between 0 and 3 was assigned depending if the lag is in the interval or not.

Target variable: direction (signal of Log Returns)

Naive Bayes Classifier

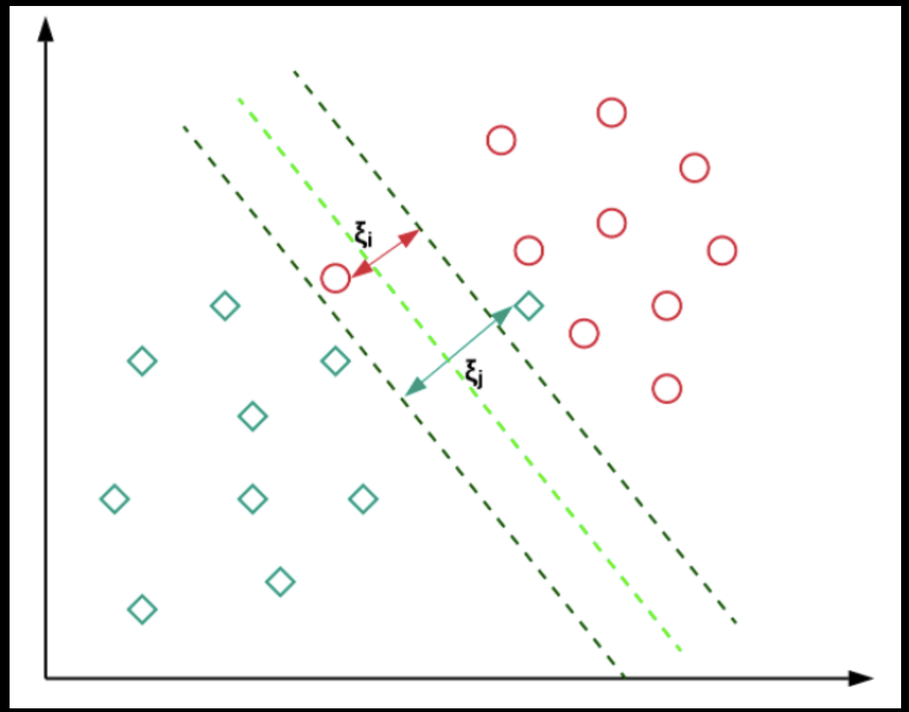
	On two lag of log return singals		On five lag of log return signals		On five lag of log returns digitized	
	train	**test**	**train**	**test**	**train**	**test**
IBM	0.514	0.533	0.513	0.519	0.514	0.529
NASDAQ	0.534	0.567	0.534	0.567	0.537	0.532
GOLD	0.532	0.533	0.529	0.529	0.540	0.517
WTI	0.522	0.517	0.527	0.525	0.530	0.511

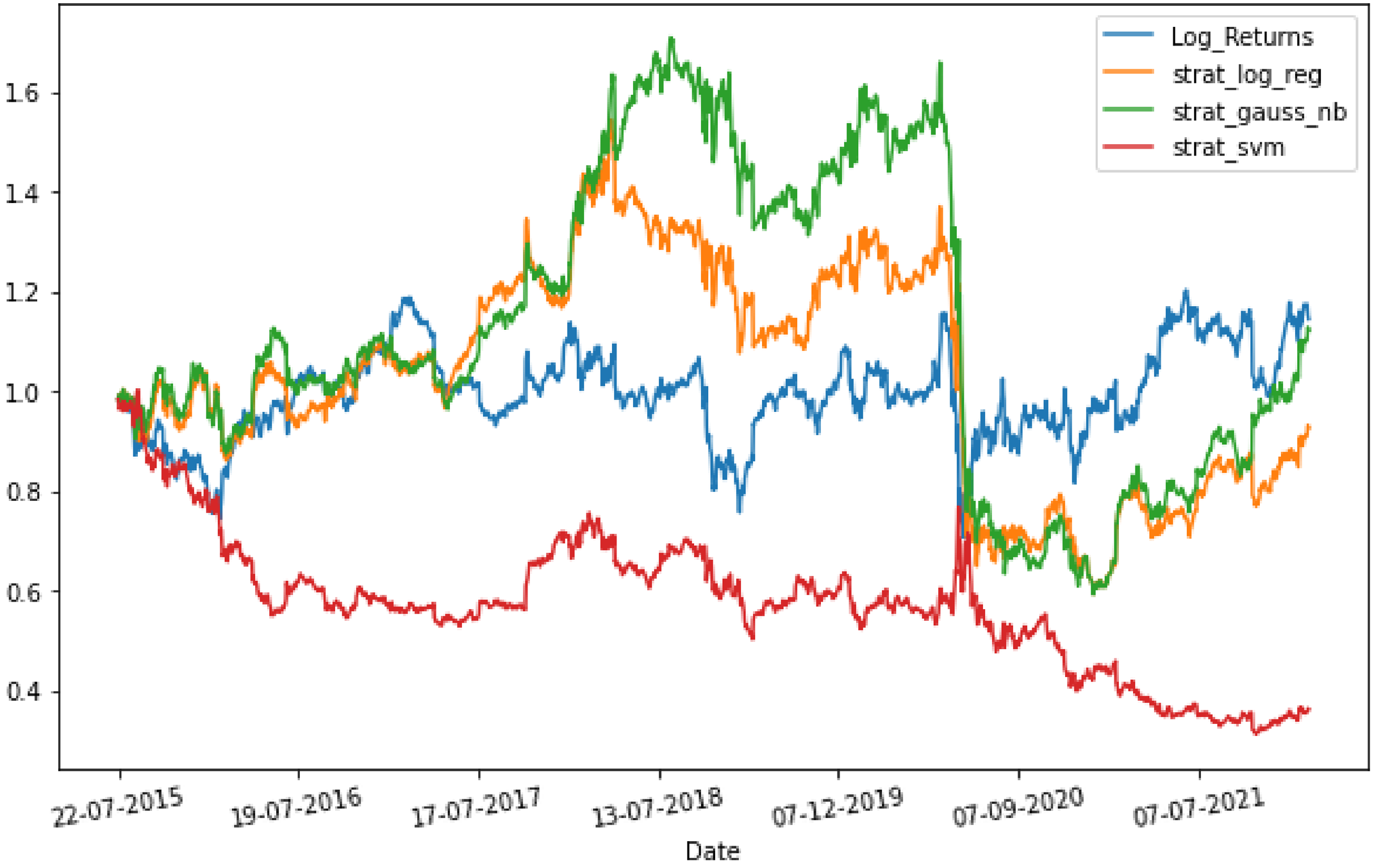
Naive Bayes is a classifier which uses the Bayes Theorem. It predicts membership probabilities for each class such as the probability that given record or data point belongs to a particular class. The class with the highest probability is considered as the most likely class.

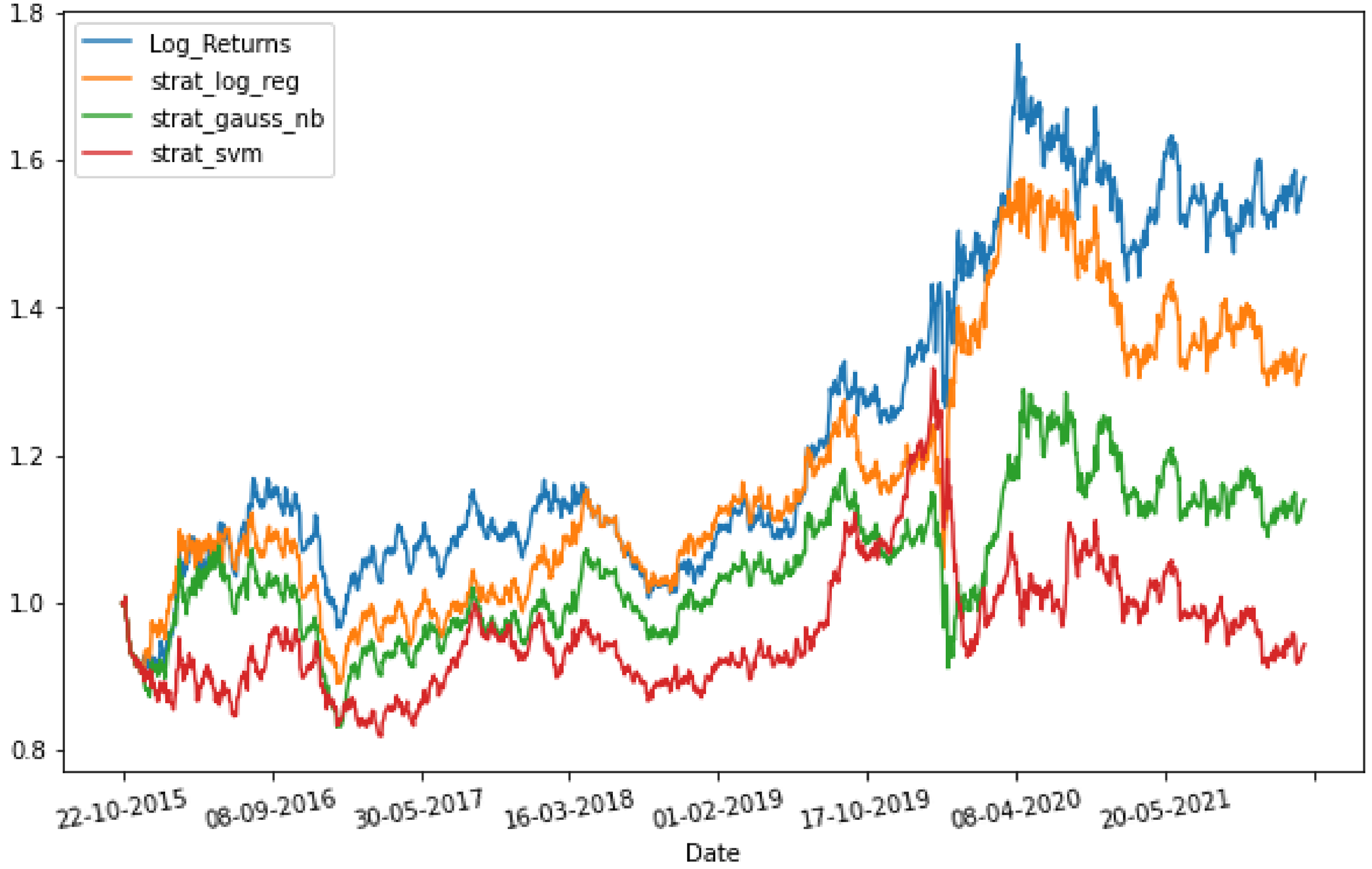
Support Vector Machine classifier

	On two lag of log return singals		On five lag of log return signals		On five lag of log returns digitized	
	Train	Test	Train	Test	Train	Test
IBM	0.514	0.533	0.533	0.497	0.572	0.497
NASDAQ	0.534	0.566	0.544	0.541	0.579	0.549
GOLD	0.532	0.533	0.553	0.517	0.585	0.529
WTI	0.522	0.517	0.543	0.520	0.576	0.534

- Kernel = Radial Basis Function
- C = 2




Log_Returns
strat_log_reg
strat_gauss_nb
strat_svm
1.6
1.4
1.2
1.0
0.8
0.6
0.4
22-07-2015
19-07-2016
17-07-2017
13-07-2018
07-12-2019
07-09-2020
07-07-2021
Date


Log_Returns
strat_log_reg
strat_gauss_nb
strat_svm
1.8
1.6
1.4
1.2
1.0
0.8
22-10-2015
08-09-2016
30-05-2017
16-03-2018
01-02-2019
17-10-2019
08-04-2020
20-05-2021
Date

DNN with scikit-learn

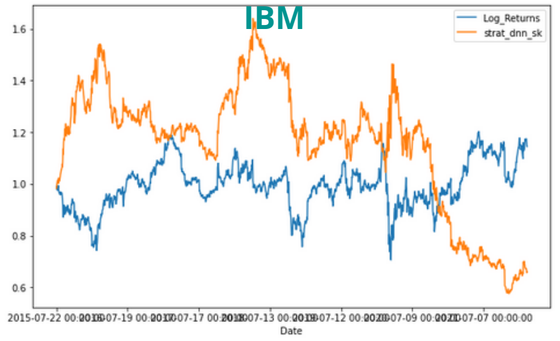
2 layers with 250 neurons

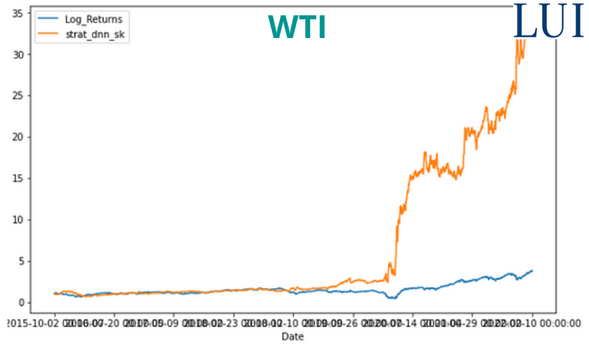
	IBM	NASDAQ	WTI	GOLD
Train	0.626	0.633	0.629	0.646
Test	0.498	0.519	0.542	0.509

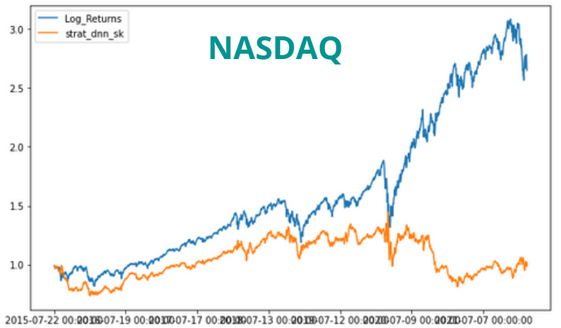
DNN with TensorFlow

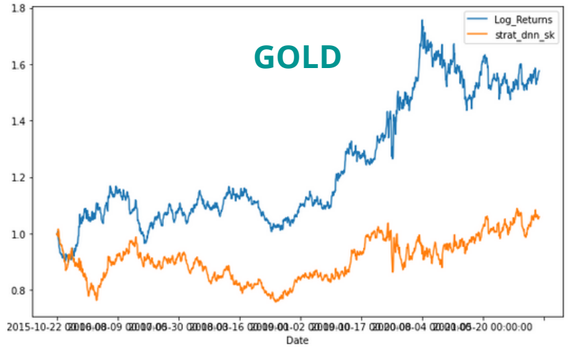
3 layers with 500 neurons

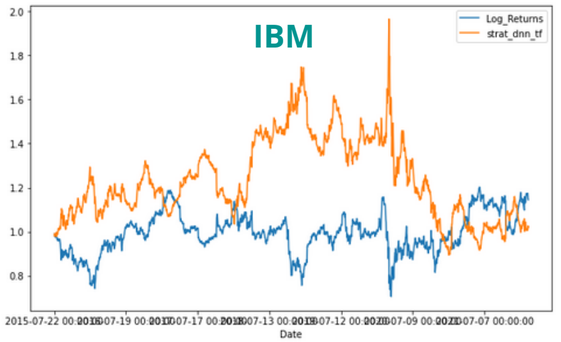
	IBM	NASDAQ	WTI	GOLD
Train	0.522	0.472	0.518	0.532
Test	0.494	0.432	0.466	0.533

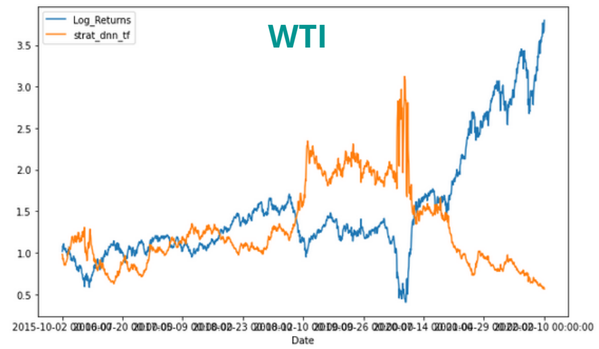


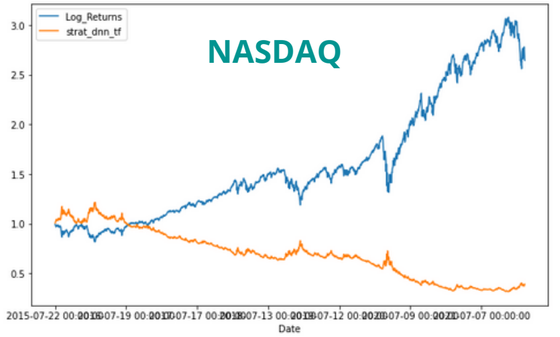












Extreme Learning Machine

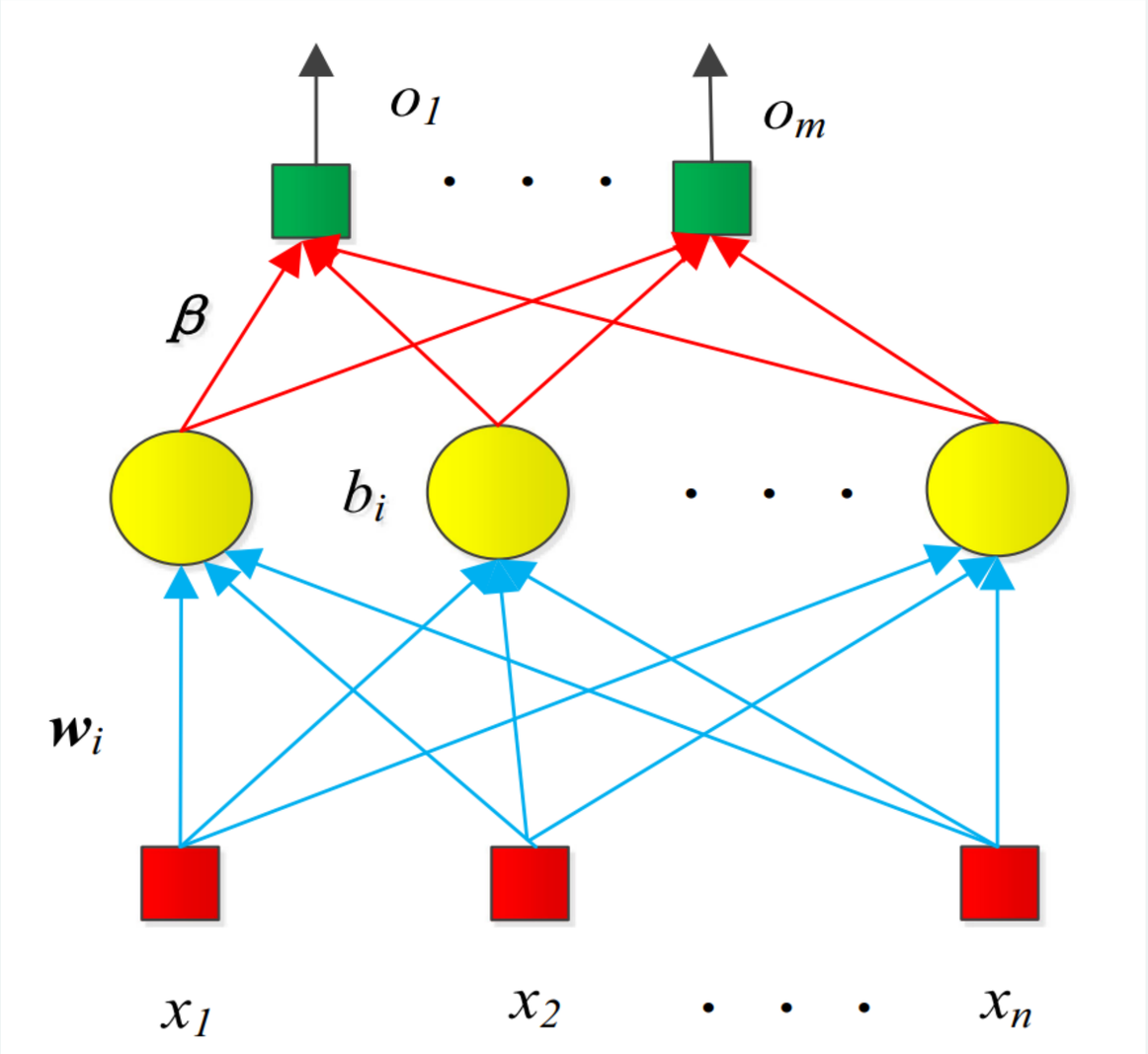
Single hidden layer feedforward neural network. ELM assigns random values to the weights between input and hideen layer, frozen during training.

Advantages:

- Converge much faster than traditional neural network because it learns without iteration.
- More likely to find a global extrema

Drawbacks:

- Since the weights are randomized it is not constant in the results



Extreme learning machine

LUISS

Target: log returns direction

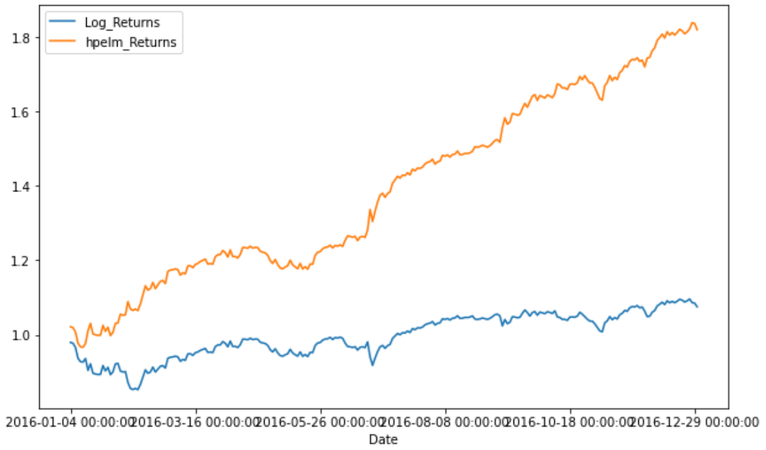
Features*:

1. Simple Moving Average (10 days)
2. Moving Average Convergence and Divergence
3. Stochastic K
4. Stochastic D
5. Relative Strength Index
6. Larry William's R%

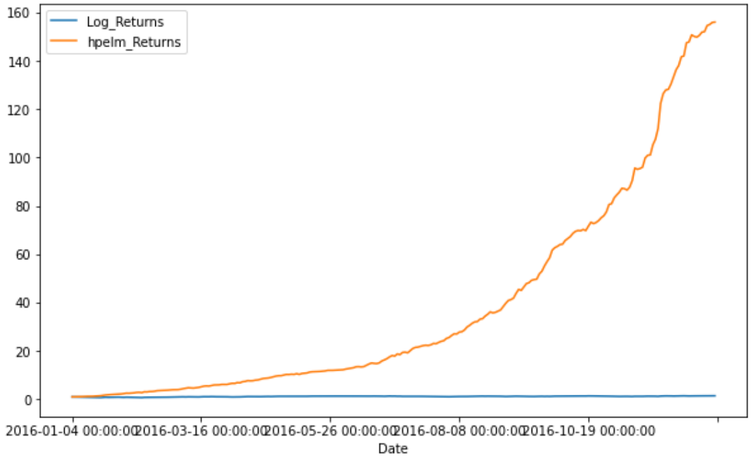
Output: output trading signal between 0 and 1.

	Train	Test
IBM	0.846	0.837
NASDAQ	0.837	0.623
GOLD	0.842	0.805
WTI	0.848	0.826

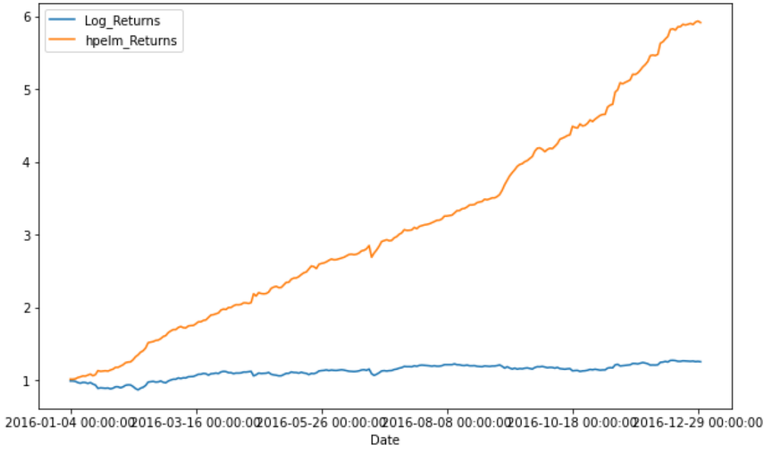
NASDAQ



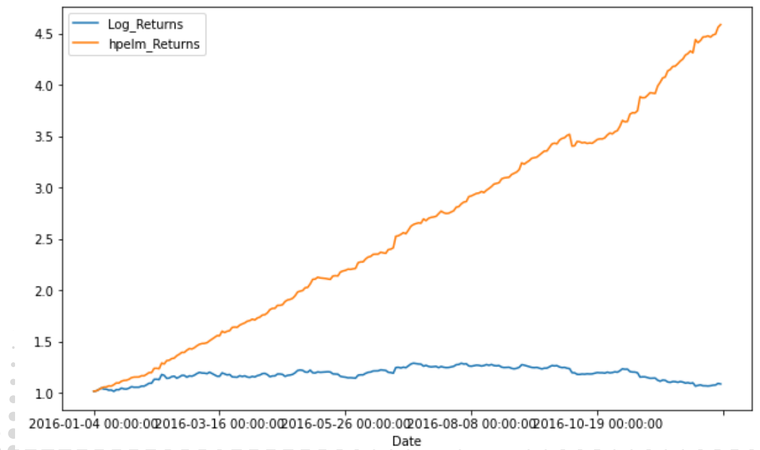
WTI



IBM



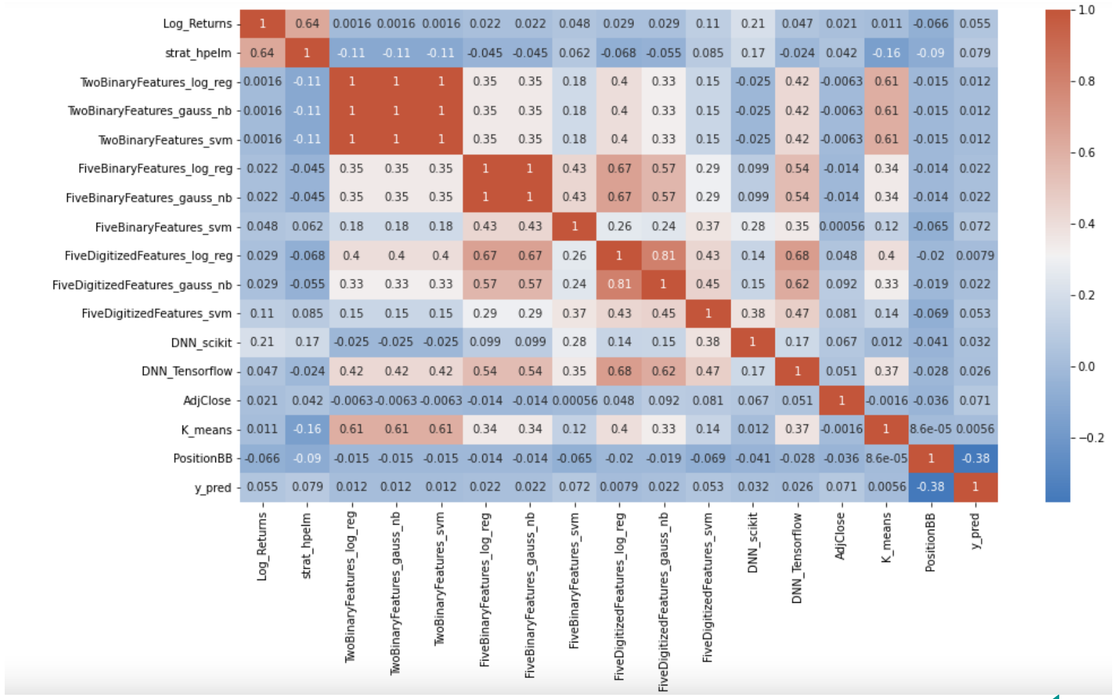
GOLD



*Simerjot Kaur

ENSEMBLE MODEL WITH RANDOM FOREST

Correlations



Correlations between the outputs of each algorithm have been computed, in order to remove those correlated above 0.4. For each dataset, the following features have been removed

IBM

Two Binary Log reg, Two Binary Gauss_naive Bayes, Five Binary Log Reg, Five Binary SVM, Five digitized Log Reg

NASDAQ

Two Binary Log Reg, Two Binary Naive Bayer, Two Binary SVM, Five Binary Log reg, Five Binary Gauss Naive Bayes, Five digitized Gauss Naive Bayes

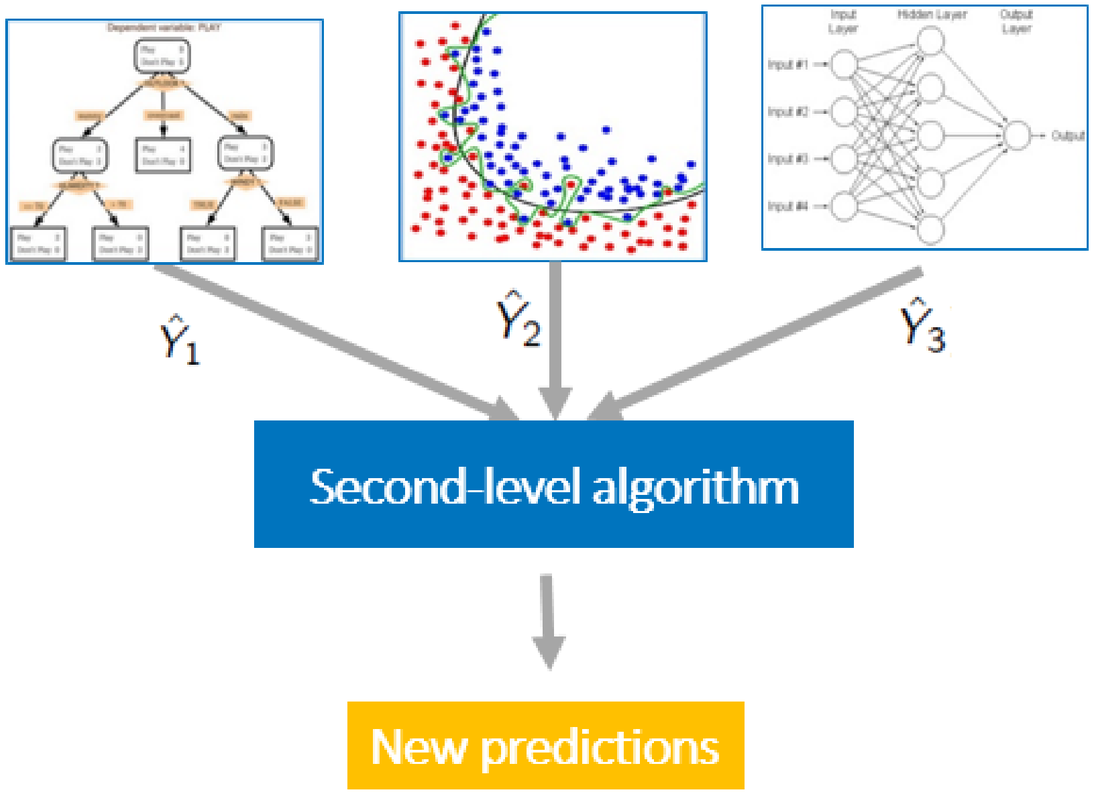
GLD

Two Binary Log Reg, Two Binary Naive Bayer, Two Binary SVM, Five Binary Log reg, Five digitized Gauss Naive Bayes, Five digitized Gauss Naive Bayes

WTI

Two Binary Log Reg, Two Binary Naive Bayer, Five Binary Log reg, Five digitized Gauss Naive Bayes, Five digitized Gauss Naive Bayes, Five digitized svm

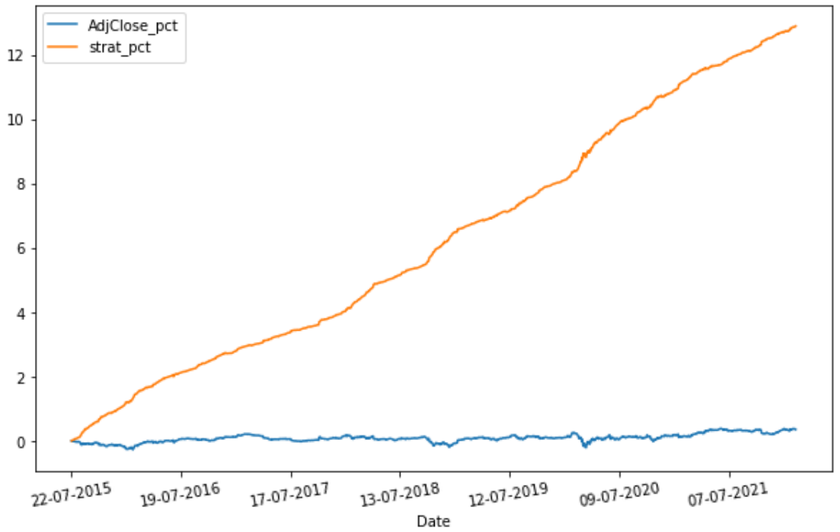
Ensemble Modelling with Random Forest Classifier



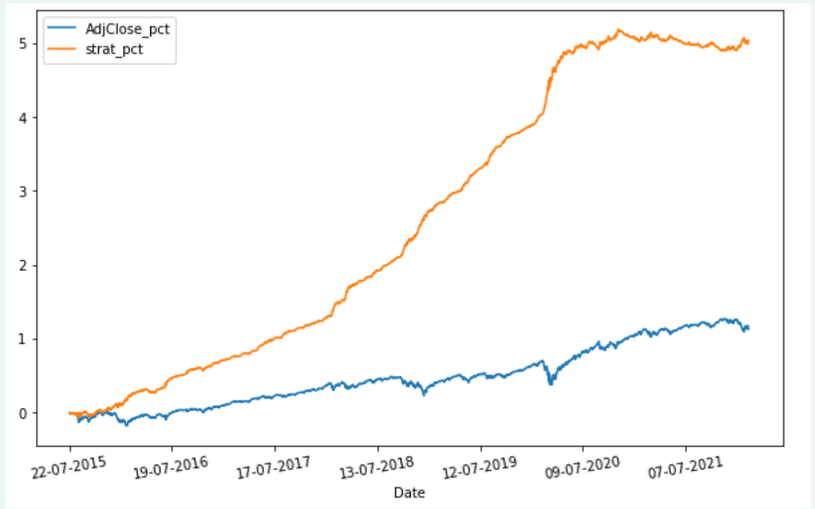
We used the previously decorrelated outputs to feed them to the new ensemble model that combines the previous output to create a new prediction

	TRAIN	TEST
IBM	0.857	0.798
NASDAQ	0.835	0.596
GLD	0.855	0.796
WTI	0.859	0.814

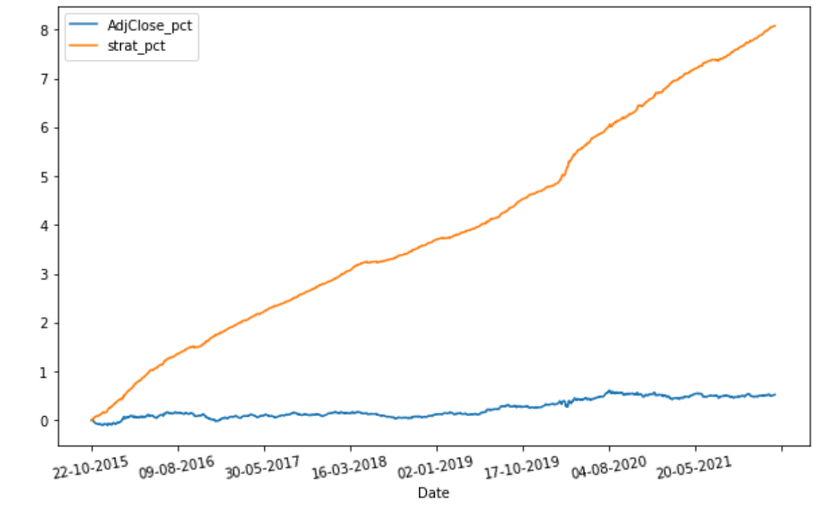
IBM



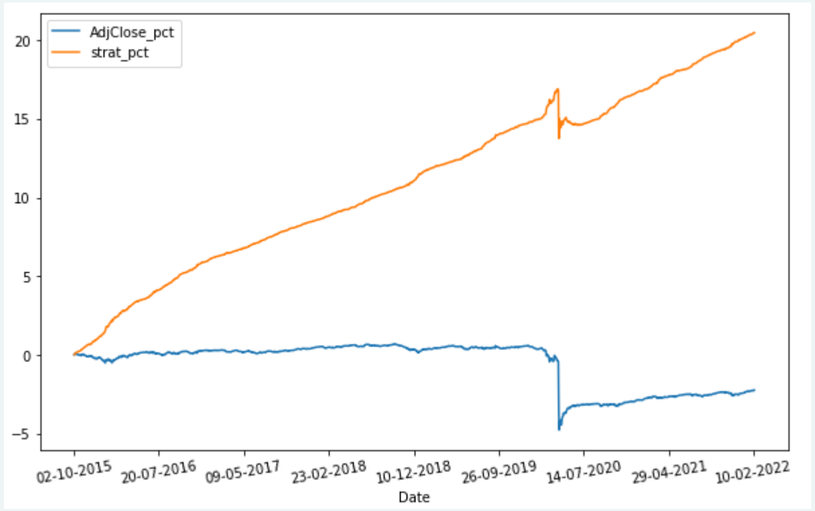
NASDAQ



GOLD

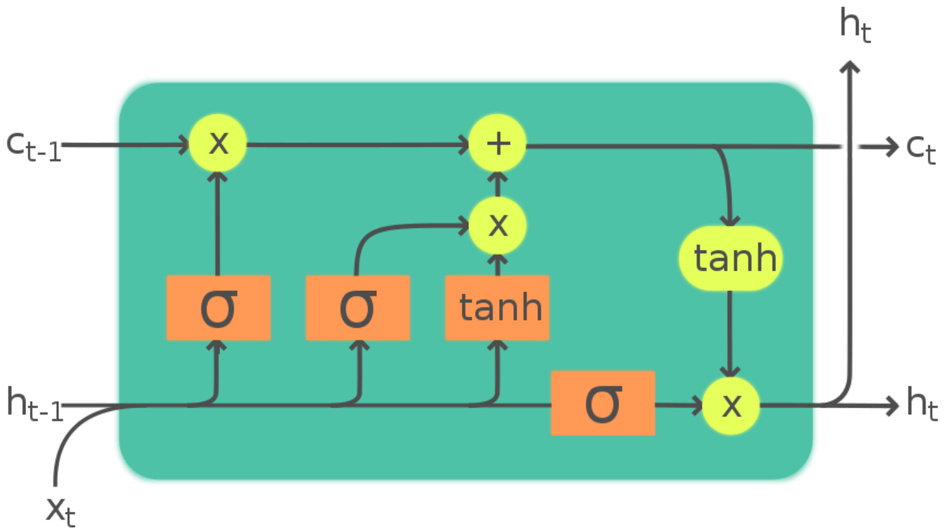


WTI

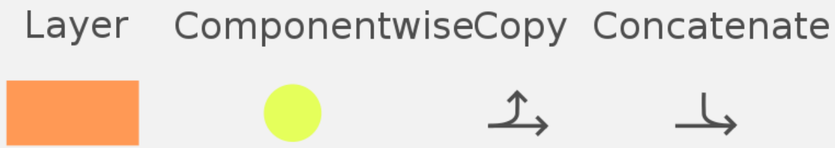


LONG-SHORT TERM MEMORY MODEL

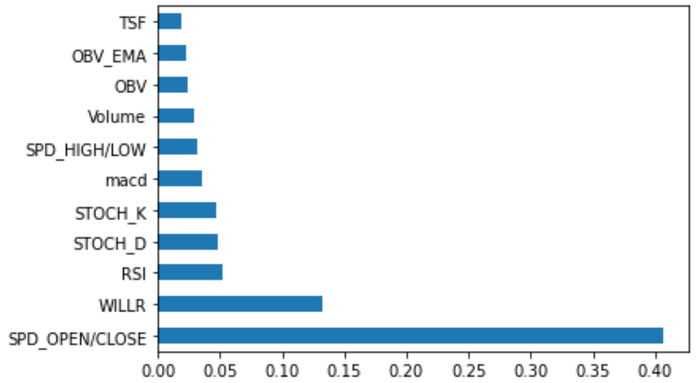
LSTM



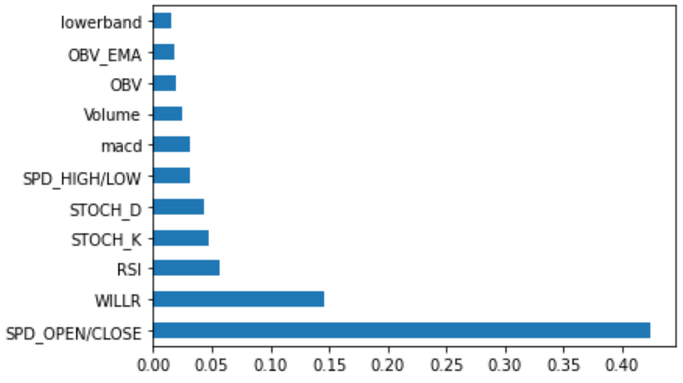
Legend:



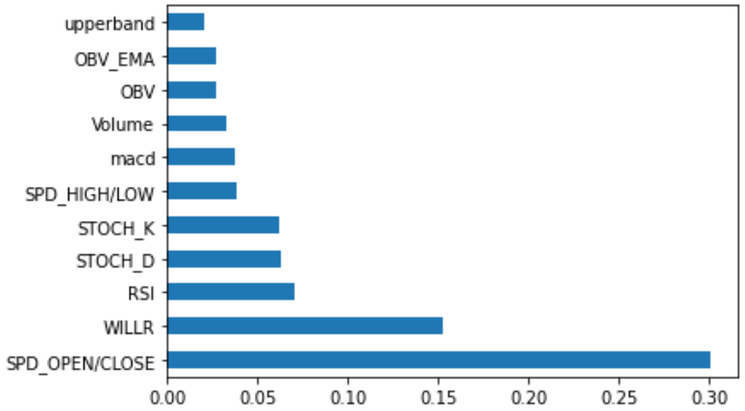
Random Forest importances



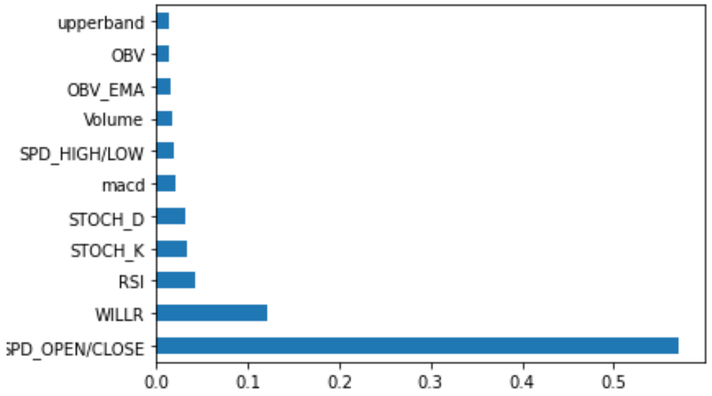
IBM



NASDAQ



GOLD



WTI

Before running the LSTM model a Random Forest Classifier was used to select the best features

We considered the 14 days before to predict the next day's movement.

This formula was used to find the number of hidden neurons

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

N_i = number of input neurons.
N_o = number of output neurons.
N_s = number of samples in training data set.
α = an arbitrary scaling factor usually 2-10.

LUISS

```
Model: "sequential_36"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 cu_dnnlstm_63 (CuDNNLSTM)    (None, 14, 256)           268288

 dropout_63 (Dropout)         (None, 14, 256)           0

 batch_normalization_63 (Bat  (None, 14, 256)           1024
 chNormalization)

 cu_dnnlstm_64 (CuDNNLSTM)    (None, 128)               197632

 dropout_64 (Dropout)         (None, 128)               0

 batch_normalization_64 (Bat  (None, 128)               512
 chNormalization)

 dense_37 (Dense)             (None, 1)                 129

=================================================================
Total params: 467,585
Trainable params: 466,817
Non-trainable params: 768
```

IBM

```
Model: "sequential_37"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 cu_dnnlstm_65 (CuDNNLSTM)    (None, 14, 256)           268288

 dropout_65 (Dropout)         (None, 14, 256)           0

 batch_normalization_65 (Bat  (None, 14, 256)           1024
 chNormalization)

 cu_dnnlstm_66 (CuDNNLSTM)    (None, 128)               197632

 dropout_66 (Dropout)         (None, 128)               0

 batch_normalization_66 (Bat  (None, 128)               512
 chNormalization)

 dense_38 (Dense)             (None, 1)                 129

=================================================================
Total params: 467,585
Trainable params: 466,817
Non-trainable params: 768
```

NASDAQ

```
Model: "sequential_38"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 cu_dnnlstm_67 (CuDNNLSTM)    (None, 14, 256)           270336

 dropout_67 (Dropout)         (None, 14, 256)           0

 batch_normalization_67 (Bat  (None, 14, 256)           1024
 chNormalization)

 cu_dnnlstm_68 (CuDNNLSTM)    (None, 128)               197632

 dropout_68 (Dropout)         (None, 128)               0

 batch_normalization_68 (Bat  (None, 128)               512
 chNormalization)

 dense_39 (Dense)             (None, 1)                 129

=================================================================
Total params: 469,633
Trainable params: 468,865
Non-trainable params: 768
_________________________________________________________________
```

GOLD

```
Model: "sequential_33"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 cu_dnnlstm_58 (CuDNNLSTM)    (None, 256)               268288

 dropout_58 (Dropout)         (None, 256)               0

 batch_normalization_58 (Bat  (None, 256)               1024
 chNormalization)

 dense_34 (Dense)             (None, 1)                 257

=================================================================
Total params: 269,569
Trainable params: 269,057
Non-trainable params: 512
```

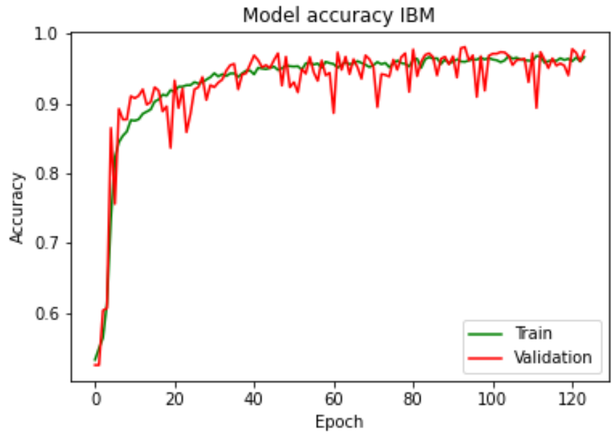
WTI

The prediction for the next day is based on the 14 days before.

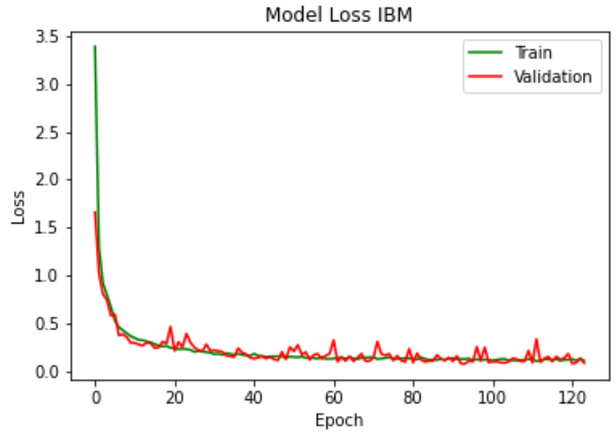
The most important features found through the Random Forest Classifier as features for our LSTM model. We used a CUDNN LSTM that exploit the powerful capabilities of the Nvidia GPU, with different configurations for the 4 securities

LSTM Performance

IBM



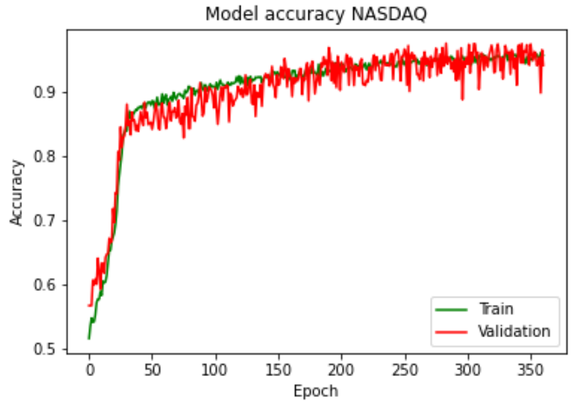
Train accuracy: 96.01%
Test accuracy: 90.10%



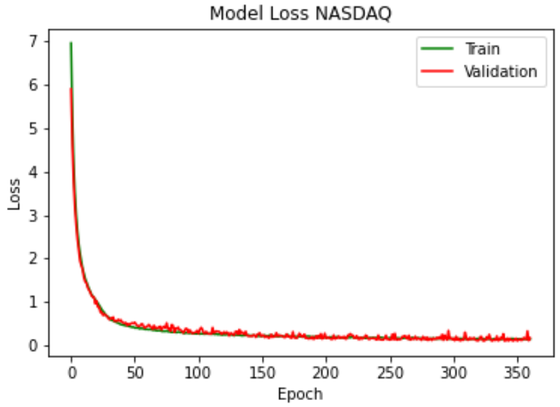
Binary Cross-Entropy train: 0.119
Binary Cross-Entropy test: 0.2967

```
Epoch 115/1000
161/161 [==============================] - 6s 35ms/step - loss: 0.1111 - accuracy: 0.9647 - val_loss: 0.1717 - val_accuracy:
0.9393
Epoch 116/1000
161/161 [==============================] - 5s 33ms/step - loss: 0.1190 - accuracy: 0.9609 - val_loss: 0.0875 - val_accuracy:
0.9763
Epoch 117/1000
161/161 [==============================] - 5s 33ms/step - loss: 0.1242 - accuracy: 0.9565 - val_loss: 0.2398 - val_accuracy:
0.9107
Epoch 118/1000
161/161 [==============================] - 5s 34ms/step - loss: 0.1236 - accuracy: 0.9598 - val_loss: 0.1115 - val_accuracy:
0.9593
Epoch 119/1000
161/161 [==============================] - 5s 33ms/step - loss: 0.1171 - accuracy: 0.9606 - val_loss: 0.1072 - val_accuracy:
0.9587
Epoch 120/1000
161/161 [==============================] - 5s 33ms/step - loss: 0.1199 - accuracy: 0.9601 - val_loss: 0.2967 - val_accuracy:
0.9010
Epoch 120: early stopping
```

LUISS
NASDAQ



Train accuracy: 93.16%
Test accuracy: 91.44%

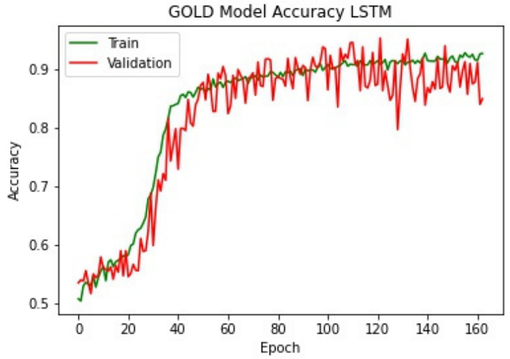


Binary Cross-Entropy train: 0.194
Binary Cross-Entropy test: 0.2335

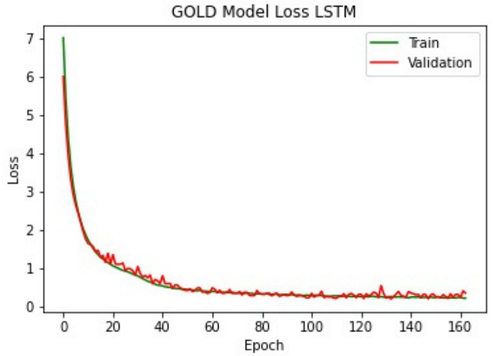
```
Epoch 200/1000
161/161 [==============================] - 5s 30ms/step - loss: 0.1919 - accuracy: 0.9331 - val_loss: 0.2790 - val_accuracy:
0.9016
Epoch 201/1000
161/161 [==============================] - 5s 29ms/step - loss: 0.1899 - accuracy: 0.9334 - val_loss: 0.2175 - val_accuracy:
0.9223
Epoch 202/1000
161/161 [==============================] - 5s 33ms/step - loss: 0.1885 - accuracy: 0.9375 - val_loss: 0.1879 - val_accuracy:
0.9338
Epoch 203/1000
161/161 [==============================] - 5s 33ms/step - loss: 0.1870 - accuracy: 0.9365 - val_loss: 0.1729 - val_accuracy:
0.9375
Epoch 204/1000
161/161 [==============================] - 5s 33ms/step - loss: 0.1776 - accuracy: 0.9448 - val_loss: 0.2601 - val_accuracy:
0.8980
Epoch 205/1000
161/161 [==============================] - 5s 32ms/step - loss: 0.1943 - accuracy: 0.9316 - val_loss: 0.2335 - val_accuracy:
0.9144
Epoch 205: early stopping
```

LSTM Performance

GOLD



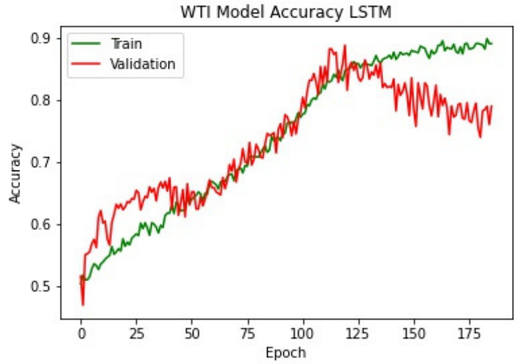
Train accuracy: 92.6%
Test accuracy: 84.87%



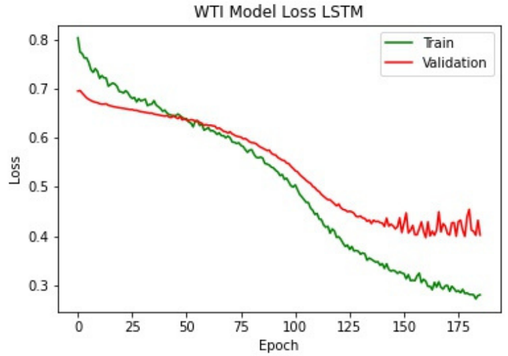
Binary Cross-Entropy train: 0.2138
Binary Cross-Entropy test: 0.347

```
Epoch 158/1000
155/155 [==============================] - 5s 34ms/step - loss: 0.2214 - accuracy: 0.9198 - val_loss: 0.2310 - val_accuracy:
0.9089
Epoch 159/1000
155/155 [==============================] - 6s 36ms/step - loss: 0.2191 - accuracy: 0.9246 - val_loss: 0.3102 - val_accuracy:
0.8741
Epoch 160/1000
155/155 [==============================] - 5s 35ms/step - loss: 0.2338 - accuracy: 0.9152 - val_loss: 0.3087 - val_accuracy:
0.8772
Epoch 161/1000
155/155 [==============================] - 5s 34ms/step - loss: 0.2394 - accuracy: 0.9146 - val_loss: 0.2284 - val_accuracy:
0.9108
Epoch 162/1000
155/155 [==============================] - 6s 36ms/step - loss: 0.2143 - accuracy: 0.9254 - val_loss: 0.4047 - val_accuracy:
0.8392
Epoch 163/1000
155/155 [==============================] - 6s 37ms/step - loss: 0.2138 - accuracy: 0.9260 - val_loss: 0.3470 - val_accuracy:
0.8487
Epoch 163: early stopping
```

WTI



Train accuracy: 93.16%
Test accuracy: 79%



Binary Cross-Entropy train: 0.194
Binary Cross-Entropy test: 0.2335

```
Epoch 181/1000
156/156 [==============================] - 4s 23ms/step - loss: 0.2812 - accuracy: 0.8897 - val_loss: 0.4546 - val_accuracy:
0.7398
Epoch 182/1000
156/156 [==============================] - 4s 23ms/step - loss: 0.2810 - accuracy: 0.8887 - val_loss: 0.4123 - val_accuracy:
0.7824
Epoch 183/1000
156/156 [==============================] - 4s 23ms/step - loss: 0.2807 - accuracy: 0.8820 - val_loss: 0.4099 - val_accuracy:
0.7843
Epoch 184/1000
156/156 [==============================] - 4s 27ms/step - loss: 0.2722 - accuracy: 0.8988 - val_loss: 0.4018 - val_accuracy:
0.7900
Epoch 185/1000
156/156 [==============================] - 4s 23ms/step - loss: 0.2789 - accuracy: 0.8908 - val_loss: 0.4323 - val_accuracy:
0.7605
Epoch 186/1000
156/156 [==============================] - 4s 23ms/step - loss: 0.2806 - accuracy: 0.8908 - val_loss: 0.4021 - val_accuracy:
0.7900
Epoch 186: early stopping
```

RESULTS

EUKLID

Which algorithm is recommended?

IBM

LSTM

96% accuracy on train
90.10% accuracy on test

NASDAQ

LSTM

93.16% accuracy on train
91.44% accuracy on test

GOLD

LSTM

92.6% accuracy on train
84.87% accuracy on test

WTI

ELM

84.8% accuracy on train
82.6% accuracy on test

THANK YOU!

Hanna Carucci Viterbi

Lorenzo Antolini

Cosimo Poccianti

Miro Confalone

References

- Simerjot Kaur, *Algorithmic Trading using Sentiment Analysis and Reinforcement Learning*
- Yves Hilpisch, *Python for Finance, 2019*
- Jian Wang, Siyuan Lu, Shui-Hua Wang, Yu-Dong Zhang, *A review on extreme learning machine,* 2021
- Ta-lib library : https://mrjbq7.github.io/ta-lib/