Index

Bayesian Cone

In the Bayesian approach we do not get a single estimate for our model parameters as we would with maximum likelihood estimation. Instead, we get a complete posterior distribution for each model parameter, which quantifies how likely different values are for that model parameter. For example, with few data points our estimation uncertainty will be high reflected by a wide posterior distribution. As we gather more data, our uncertainty about the model parameters will decrease and we will get an increasingly narrower posterior distribution. There are many more benefits to the Bayesian approach, such as the ability to incorporate prior knowledge.

There are two Bayesian models that can be used for prediction: Normal model and T-model. The Normal model assumes that daily returns are sampled from a normal distribution whose mean and standard deviation are accordingly sampled from a normal distribution and a halfcauchy distribution. The T-model is very much similar to the normal model except that it assumes that daily returns are sampled from a Student-T distribution. The T distribution is very much like a normal distribution, but it has heavier tails, which makes it a better distribution to capture data points that are far away from the center of data distribution. It is well known that daily returns are in fact not normally distributed as they have heavy tails.
This is the code used to implement the T-model in PyMC3:

```
import pymc3 as pm
with pm.Model():
    mu = pm.Normal('mean returns', mu=0, sd=.01)
    sigma = pm.HalfCauchy('volatility', beta=1)
    nu = pm.Exponential('nu_minus_two', 1. / 10.)

    returns = pm.StudentT('returns', nu=nu + 2, mu=mu, sd=sigma,
                          observed=data)

    # Fit model to data
    start = pm.find_MAP(fmin=sp.optimize.fmin_powell)
    step = pm.NUTS(scaling=start)
    trace = pm.sample(samples, step, start=start)
```

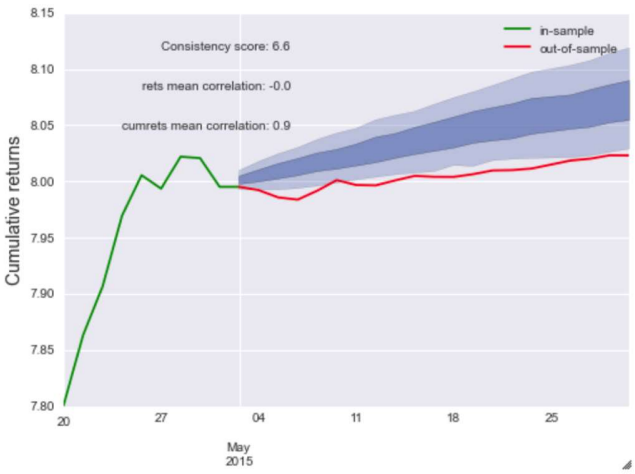
One of the ways by which the Bayesian cone can be useful is detecting the overfit algorithms with good backtest results. In order to be able to numerically measure by how much a strategy is overfit, there is the Bayesian consistency score. This score is a numerical measure to report the level of consistency between the model predictions and the actual live trading results. To calculate the Bayesian consistency score, we compute the average percentile score of the paper-trading returns to the predictions and normalize to yield a value between 100 (perfect fit) and 0 (completely outside of cone).

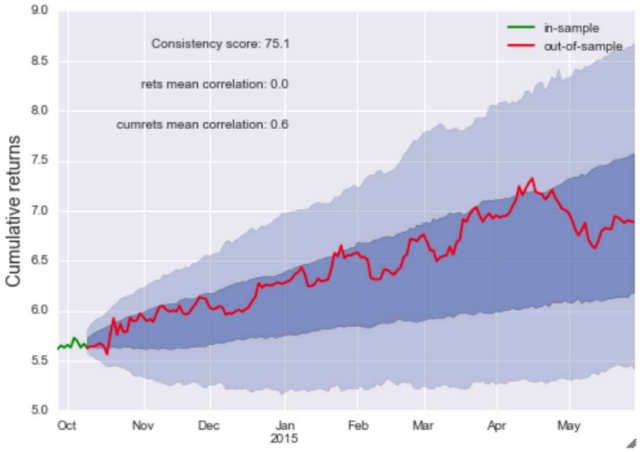
Computational modelling always comes with some risks such as estimation uncertainty, model misspecifications and implementation limitations and errors. According to such risk factors, model predictions are not always perfect and 100% reliable. However, model predictions still can be used to extract useful information about algorithms, even if the predictions are not perfect.

Estimation uncertainty is one of the risk factors, which becomes relevant with modelling and it is reflected on the width of the prediction cone. The more uncertain our predictions, the wider the cone would be. There are two ways by which we may get uncertain predictions from our model:
1) little data,

2) high volatility in the daily returns.

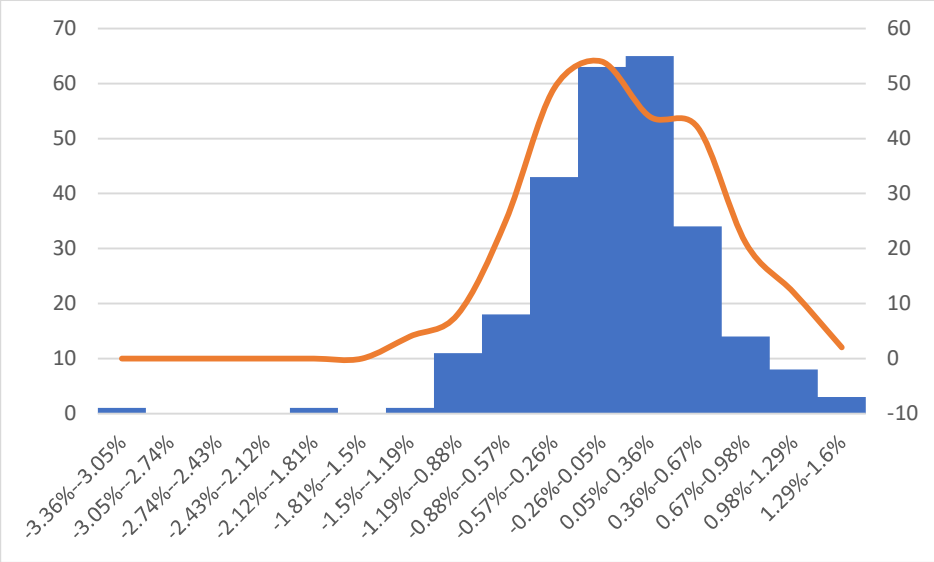
The cone on the right shows an algorithm whose live trading results are pretty much within our prediction area and to be more accurate even in high confidence interval of our prediction area. This basically means that the algorithm is performing in line with our predictions. On the other hand, the cone on the left shows an algorithm whose live trading results are pretty much outside of our prediction area, which would prompt us to take a closer look as to why the algorithm is behaving according to specifications and potentially turn it off if it is used for real-money live trading.



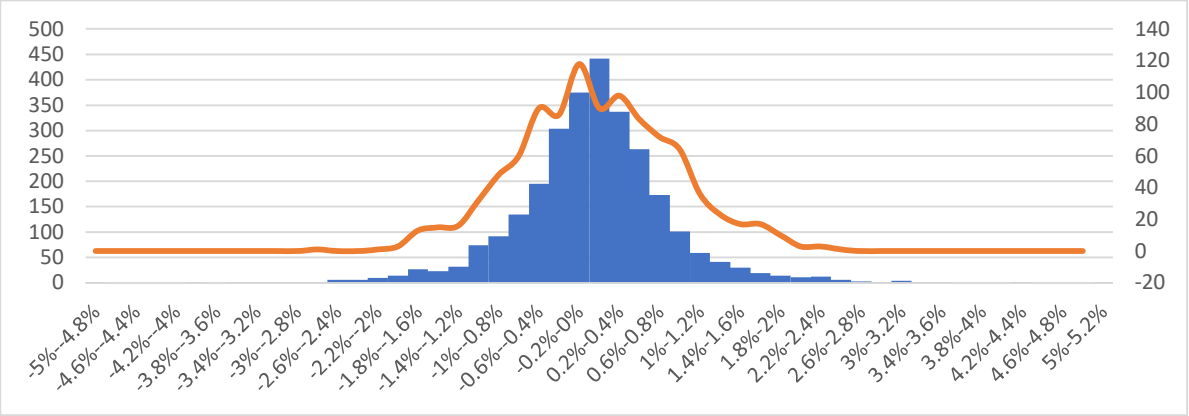


Empirical Results on sample trading strategy – F54

Percentages from the Daily NAV Comm sheet plotted against a normal distribution:

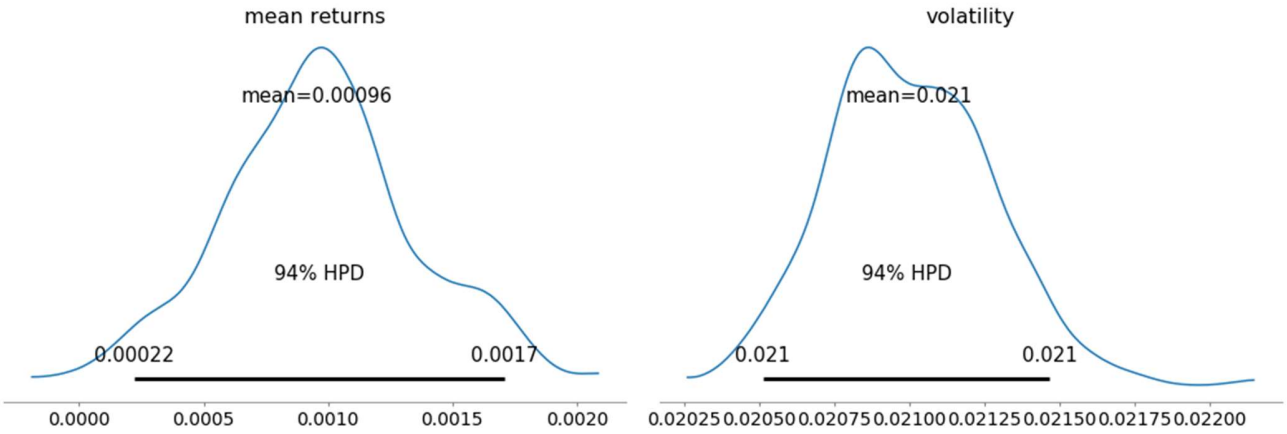


Percentages from the historical backtests plotted against a normal distribution:

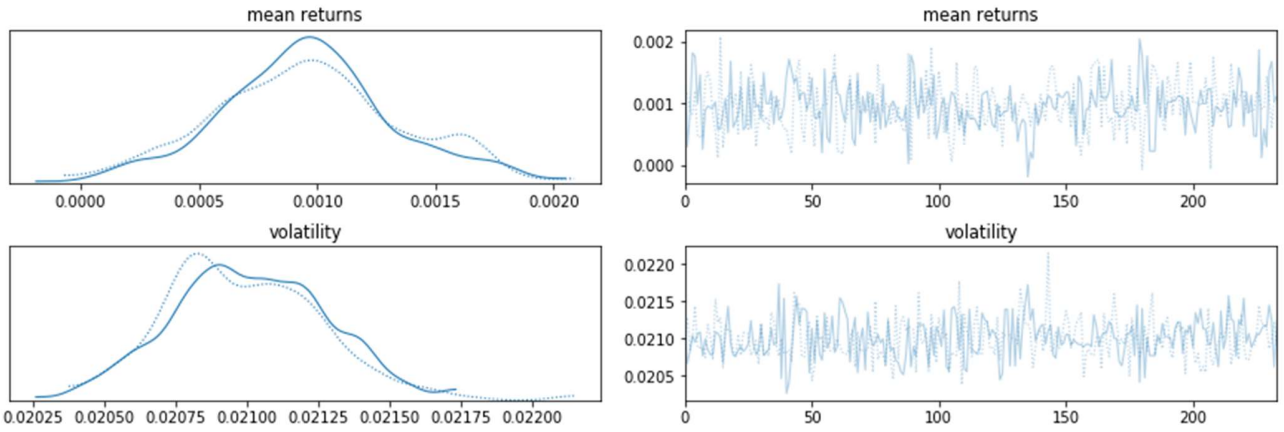


I ran the code in python assuming the normal model fits the data and got the model. I also plotted the posterior distributions of the model parameters.

Posterior distribution for the model parameters:



HPD – Highest Posterior Density

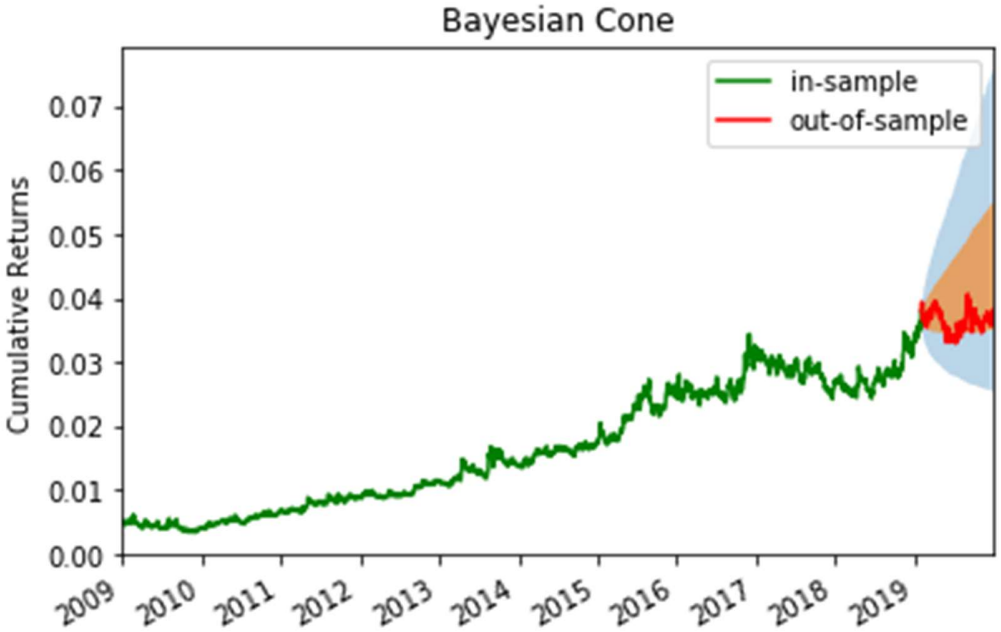


Bayesian Cone

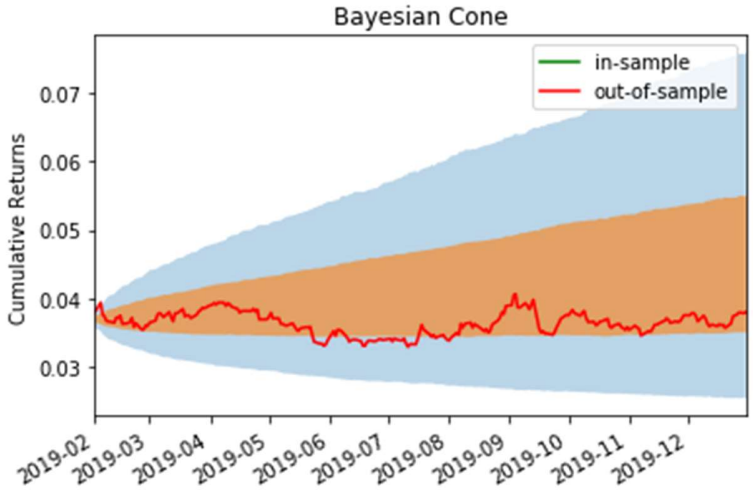
Attempt 1:
Assume that the daily returns are normally distributed.
The code in python –

```
import pymc3 as pm
import scipy as sp
import numpy as np
with pm.Model() as model:
    mu = pm.Normal('mean returns', mu=0, sd=.01, testval=np.mean(data))
    sigma = pm.HalfCauchy('volatility', beta=1, testval=np.std(data))
    returns = pm.Normal('returns', mu=mu, sd=sigma, observed=data)
    start = pm.find_MAP()
    step = pm.NUTS(scaling=start)
    trace = pm.sample(234, step, start=start)
ppc = pm.sample_posterior_predictive(trace, samples = 10000, model = model)
```

The above code trains the model to generate 10000 predicted returns for the next 234 days.
Using this we can create the Bayesian cone using pyfolio.



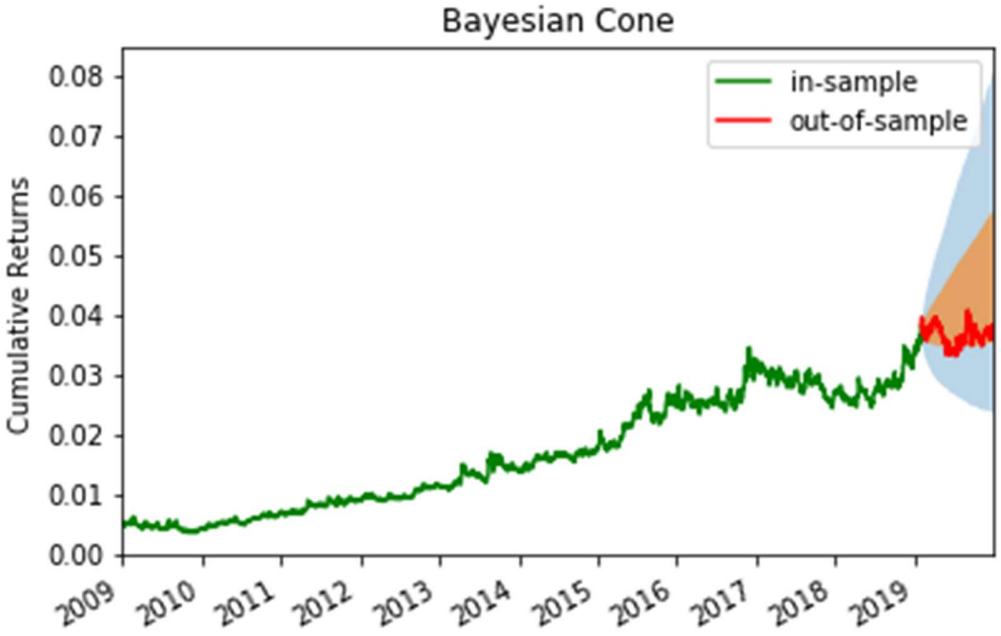
Bayesian Consistency Score – 69.16



Attempt 2:
Assume that the daily returns are Student-T distributed.
The code in python –

```
import pymc3 as pm
import scipy as sp
import numpy as np
with pm.Model() as model:
    mu = pm.Normal('mean returns', mu=0, sd=.01)
    sigma = pm.HalfCauchy('volatility', beta=1)
    nu = pm.Exponential('nu_minus_two', 1. / 10.)
    returns = pm.StudentT('returns', nu=nu + 2, mu=mu, sd=sigma, observed=data)
    start = pm.find_MAP(fmin=sp.optimize.fmin_powell)
    step = pm.NUTS(scaling=start)
    trace = pm.sample(234, step, start=start)
ppc = pm.sample_posterior_predictive(trace, samples = 10000, model = model)
```

The above code trains the model to generate 10000 predicted returns for the next 234 days.
Using this we can create the Bayesian cone using pyfolio.

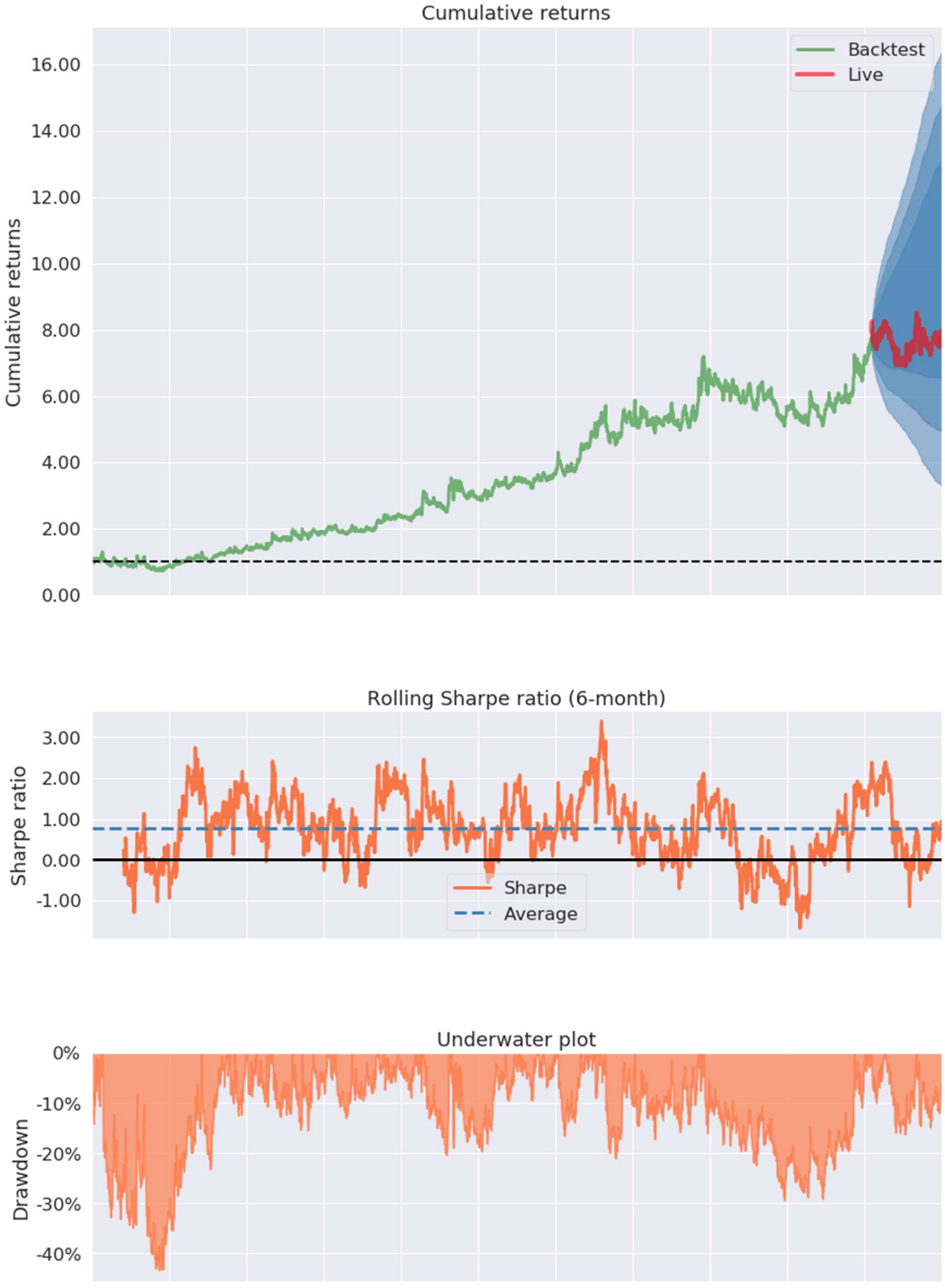


Bayesian Consistency Score – 67.61

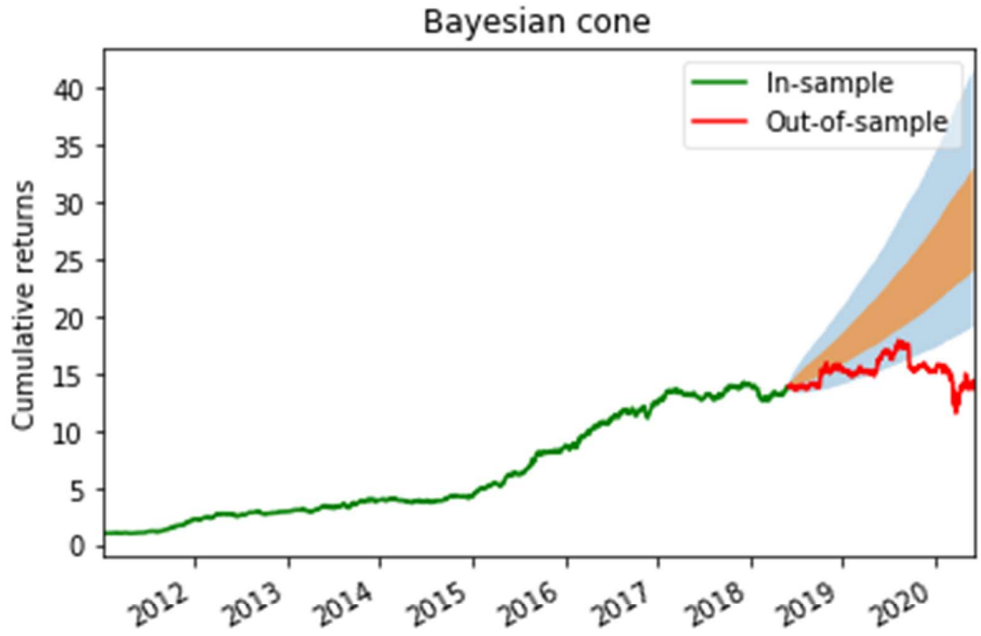
Attempt 3:
For this attempt I used pyfolio again but I used their create_simple_tear_sheet function which gives back a detailed analysis of the daily returns.

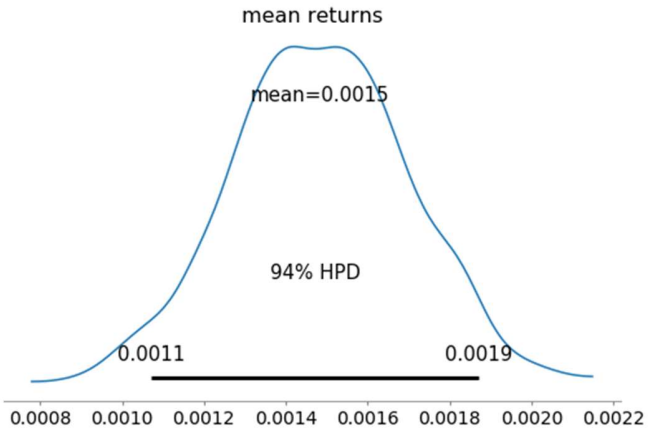
Start date	2009-01-01
End date	2019-12-31
In-sample months	134
Out-of-sample months	11

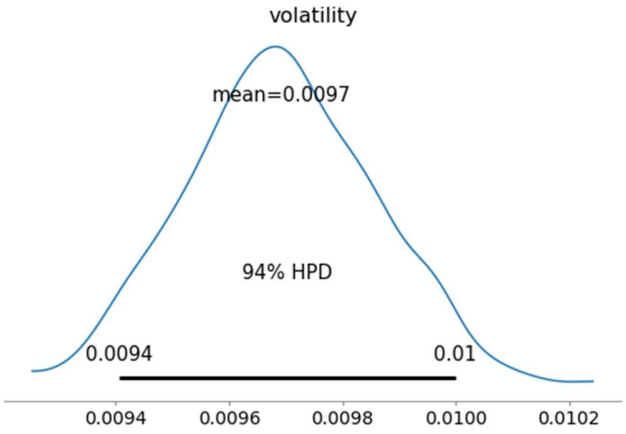
	All	In-sample	Out-of-sample
Annual return	18.6%	20.0%	3.4%
Cumulative returns	697.6%	673.4%	3.1%
Annual volatility	32.9%	33.3%	26.6%
Sharpe ratio	0.68	0.71	0.26
Calmar ratio	0.43	0.46	0.21
Stability	0.96	0.96	0.01
Max drawdown	-43.3%	-43.3%	-16.4%
Omega ratio	1.14	1.14	1.05
Sortino ratio	1.00	1.04	0.34
Skew	0.04	0.09	-1.14
Kurtosis	6.85	6.77	6.17
Tail ratio	0.99	0.99	0.91
Daily value at risk	-4.1%	-4.1%	-3.3%

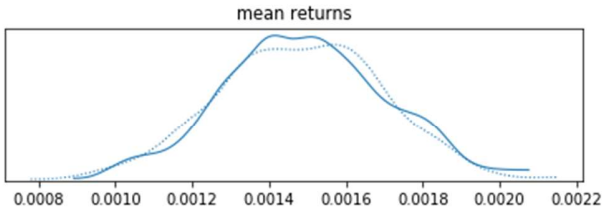

Cumulative returns
Backtest
Live
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
Cumulative returns
Rolling Sharpe ratio (6-month)
3.00
2.00
1.00
0.00
-1.00
Sharpe ratio
Sharpe
Average
Underwater plot
0%
-10%
-20%
-30%
-40%
Drawdown

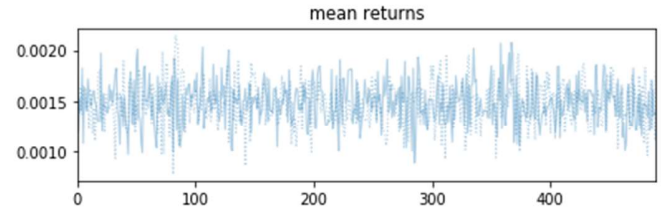
Model 1

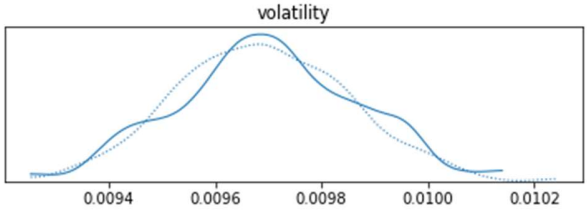


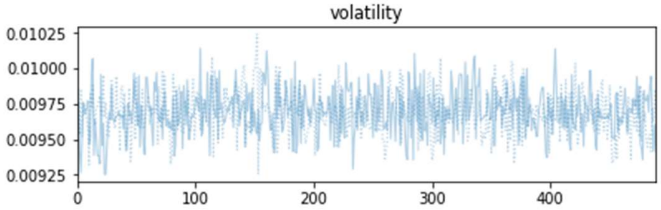




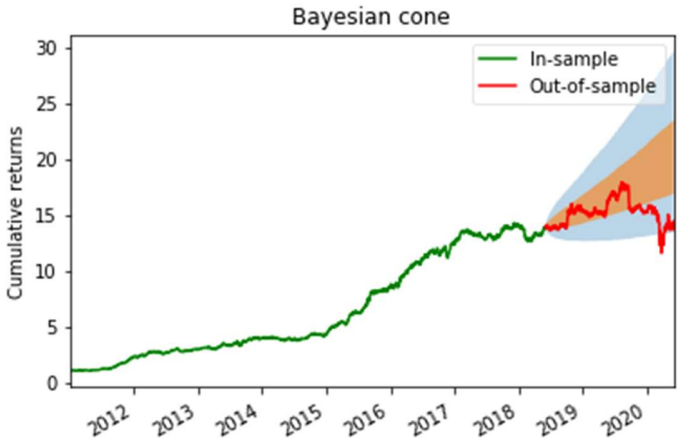


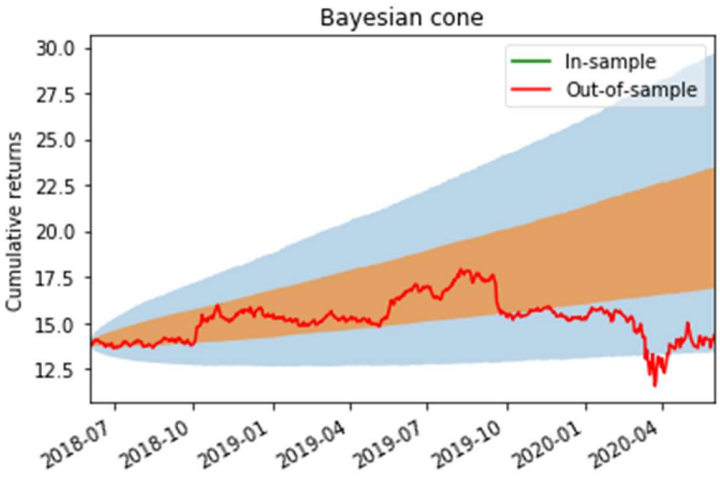






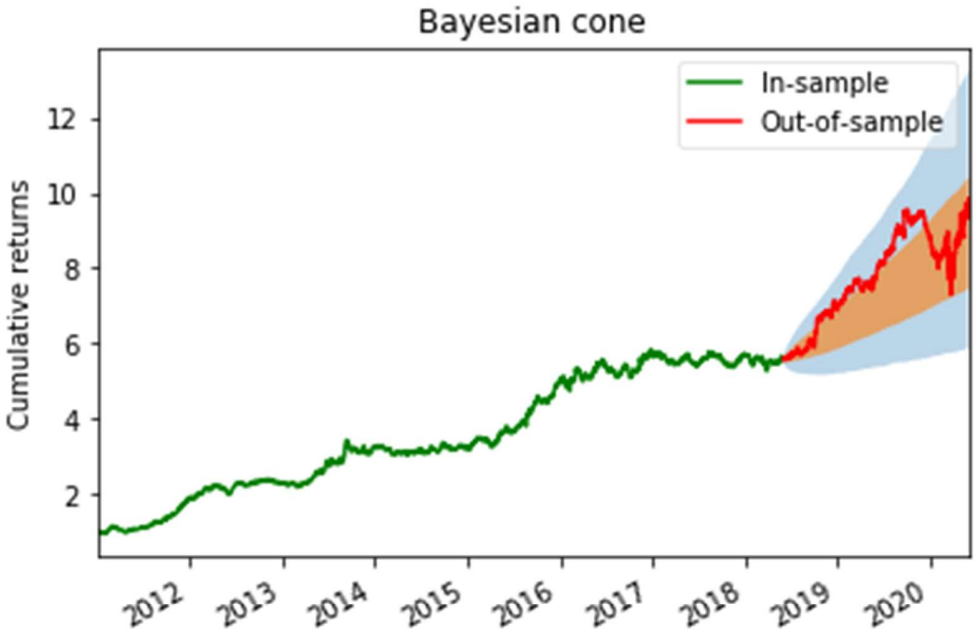
Bayesian Score – 21.4

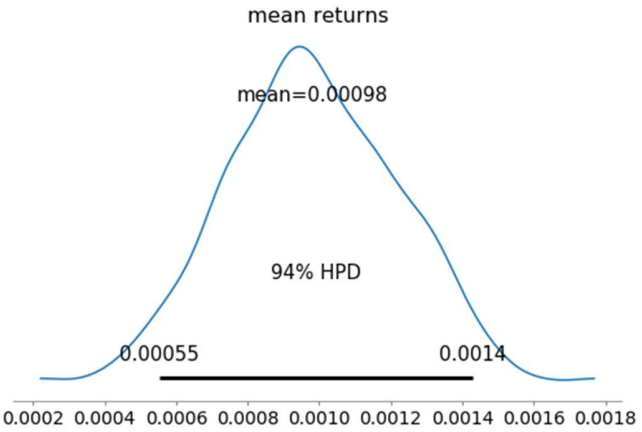


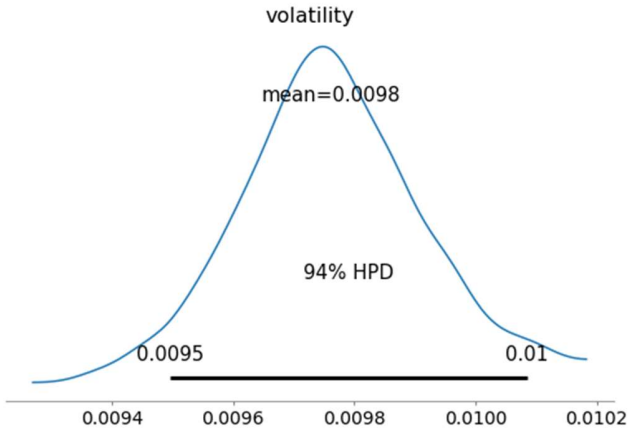


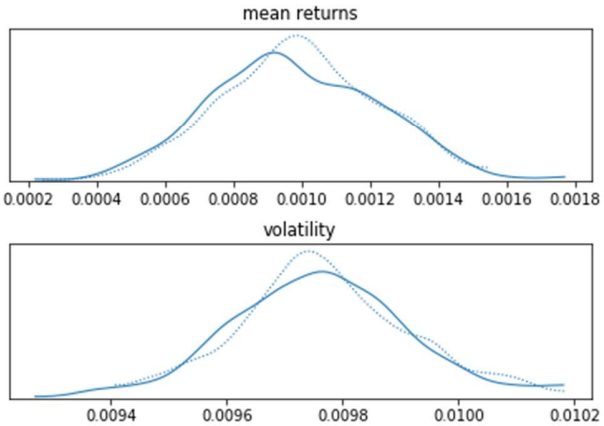
Bayesian Score – 66.24

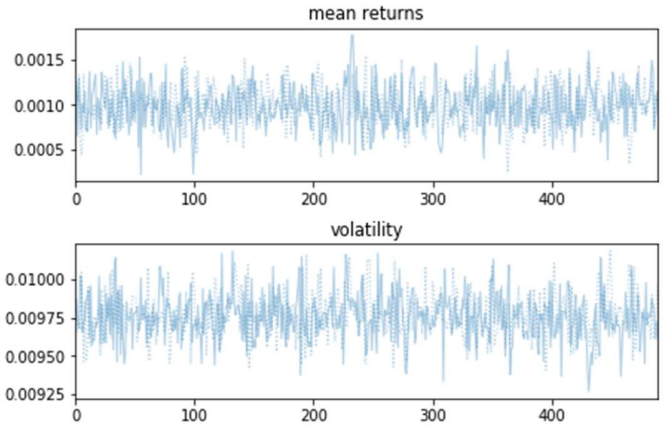
Model 2



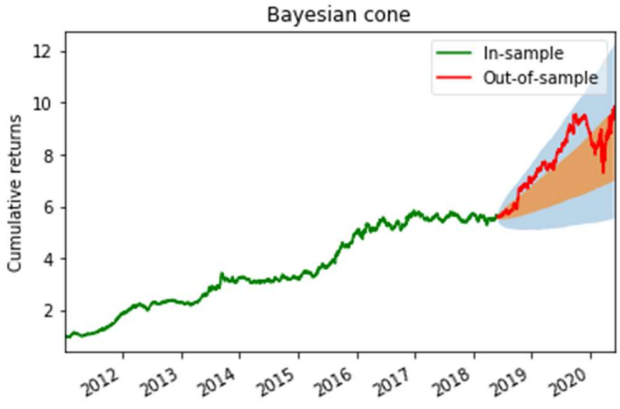


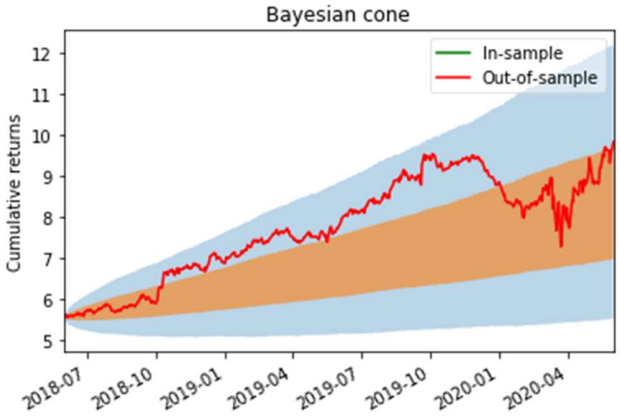






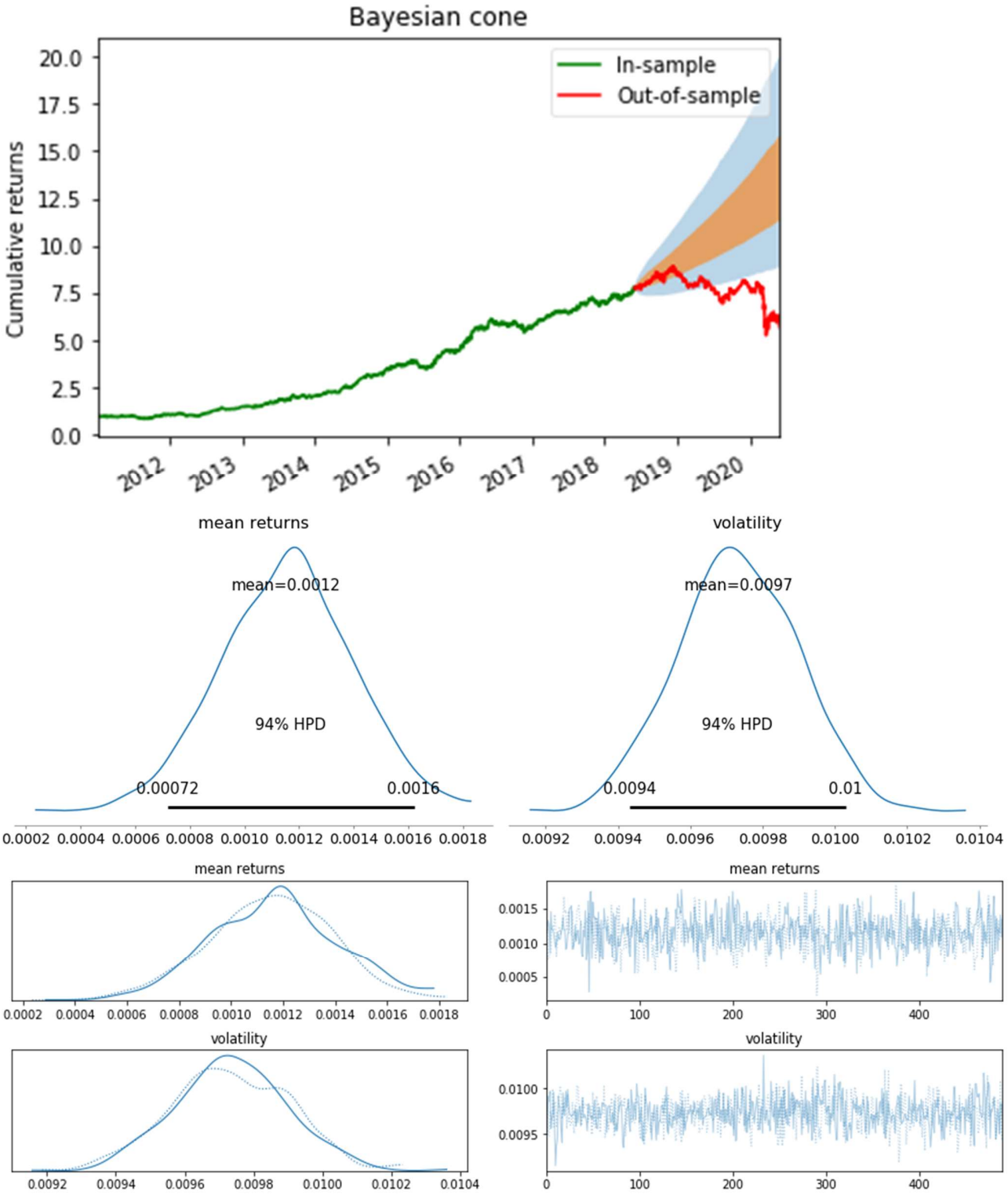
Bayesian Score – 62.85



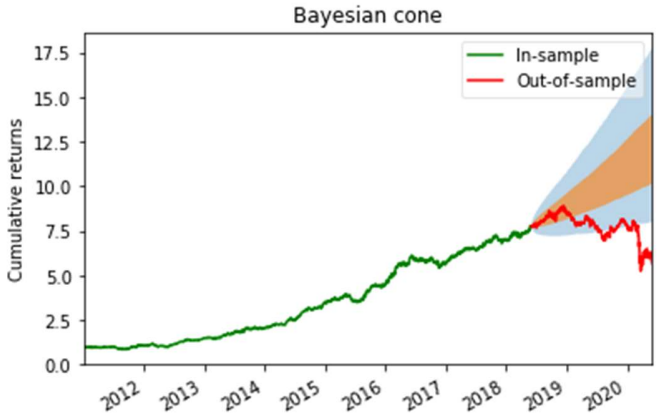


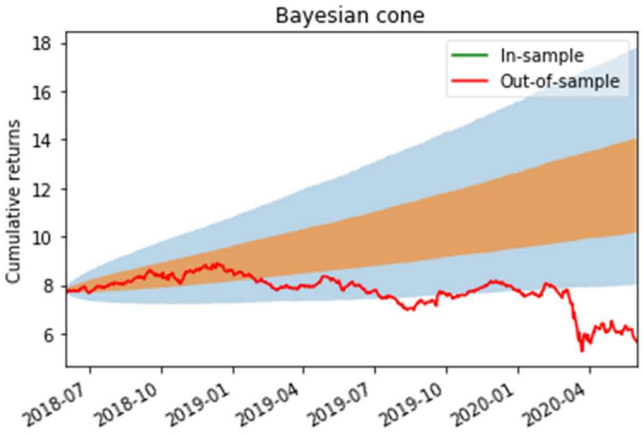
Bayesian Score – 50.47

Model 3



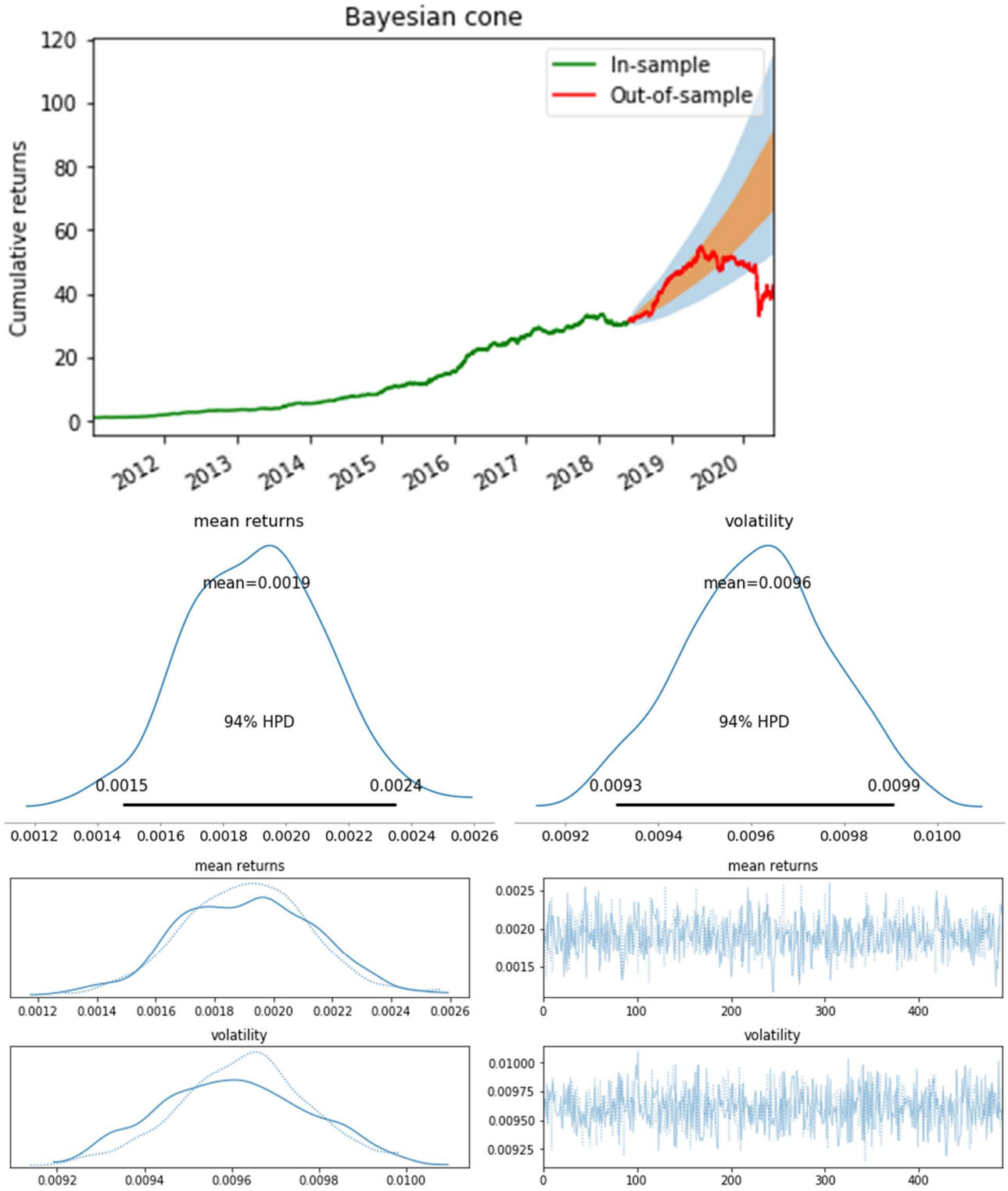
Bayesian Score – 30.09





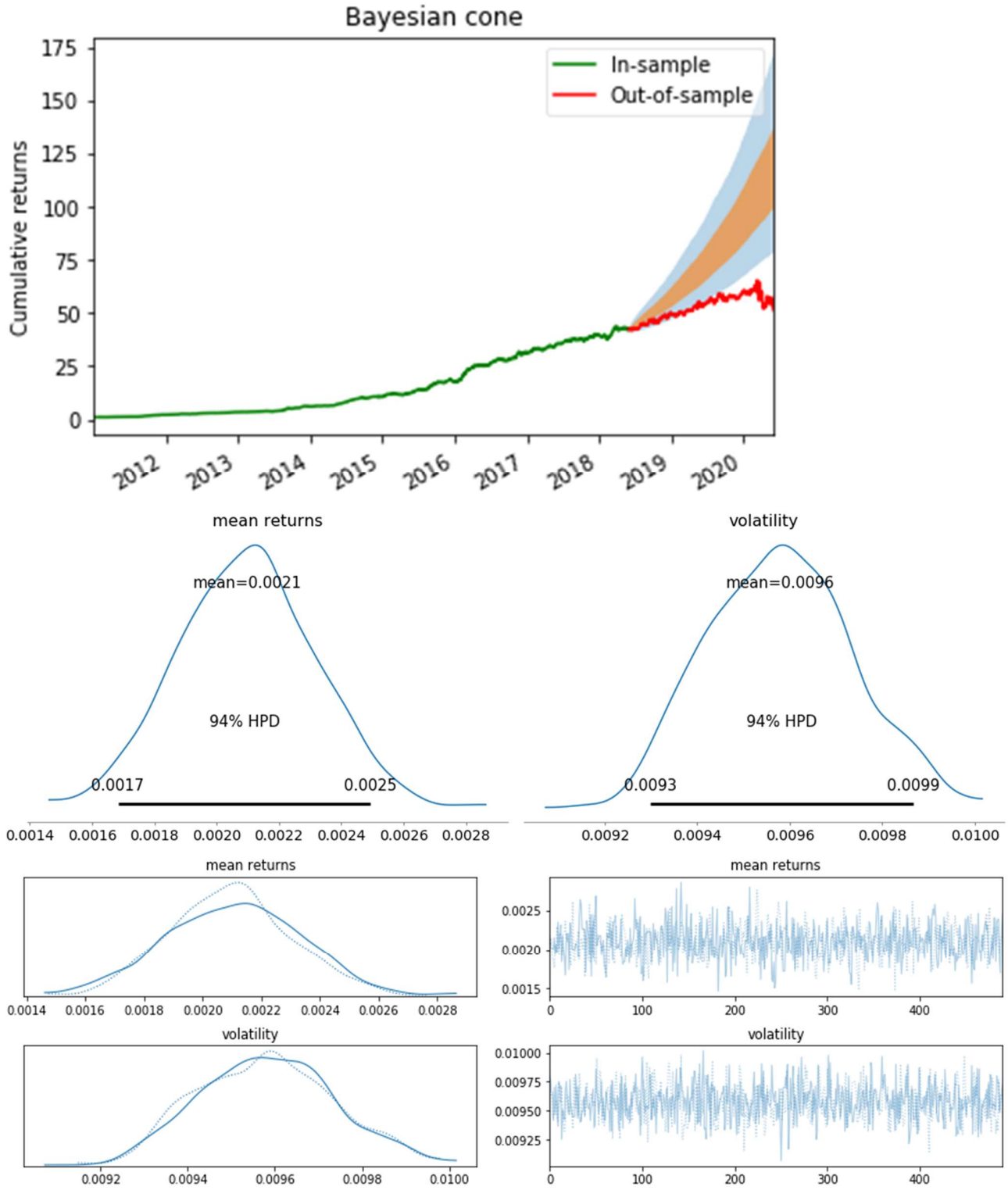
Bayesian Score – 38.89

Model 4



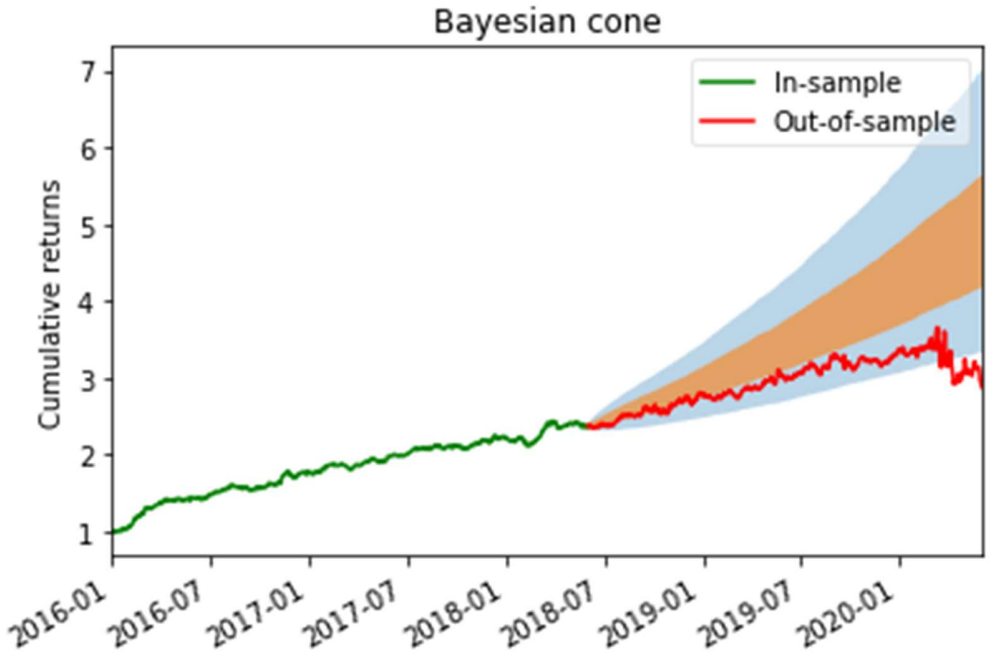
Bayesian Score – 79.24

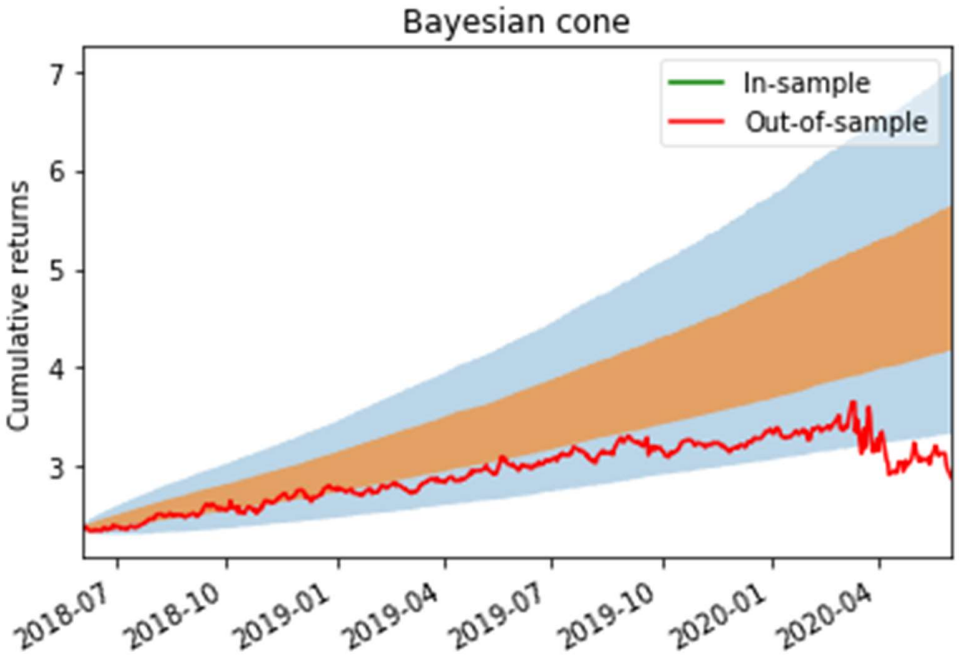
Model 5



Bayesian Score – 14.16

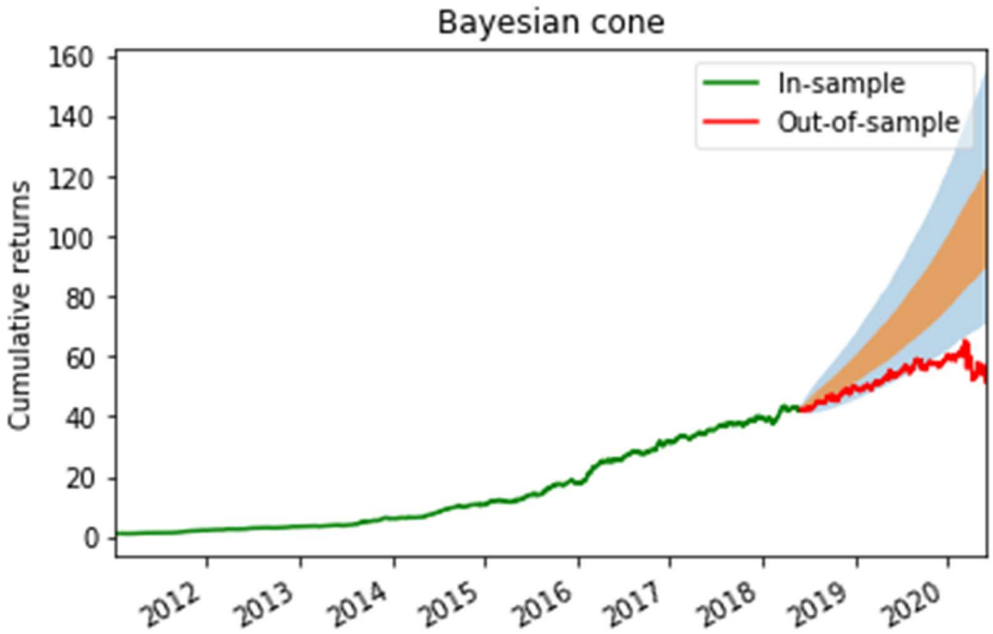
Training model from 2016:



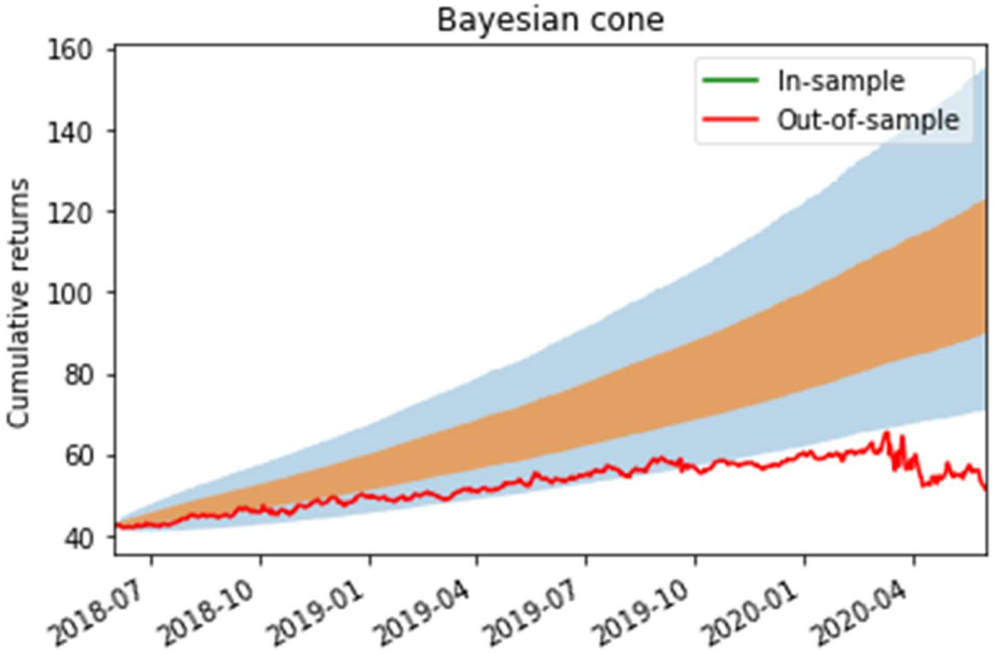


Bayesian Score – 36.08

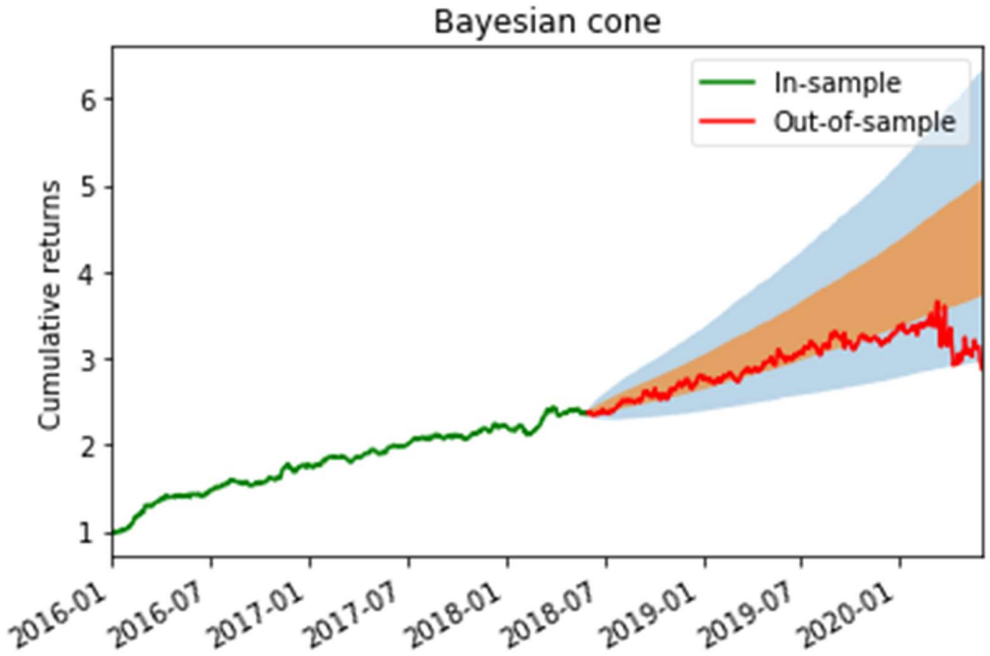
Run with t model:



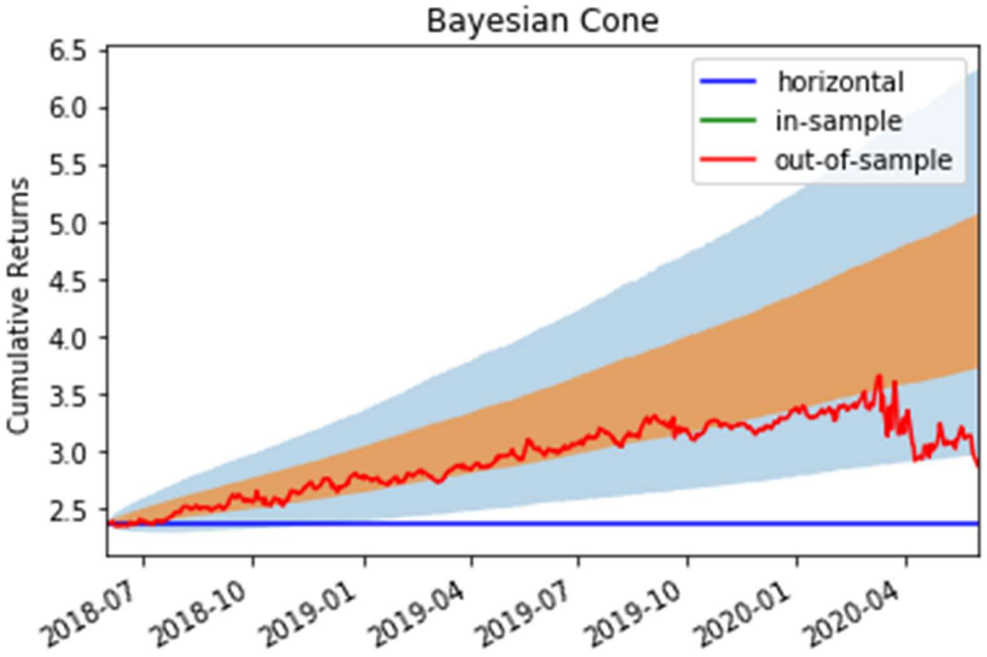
Bayesian Score – 20.62



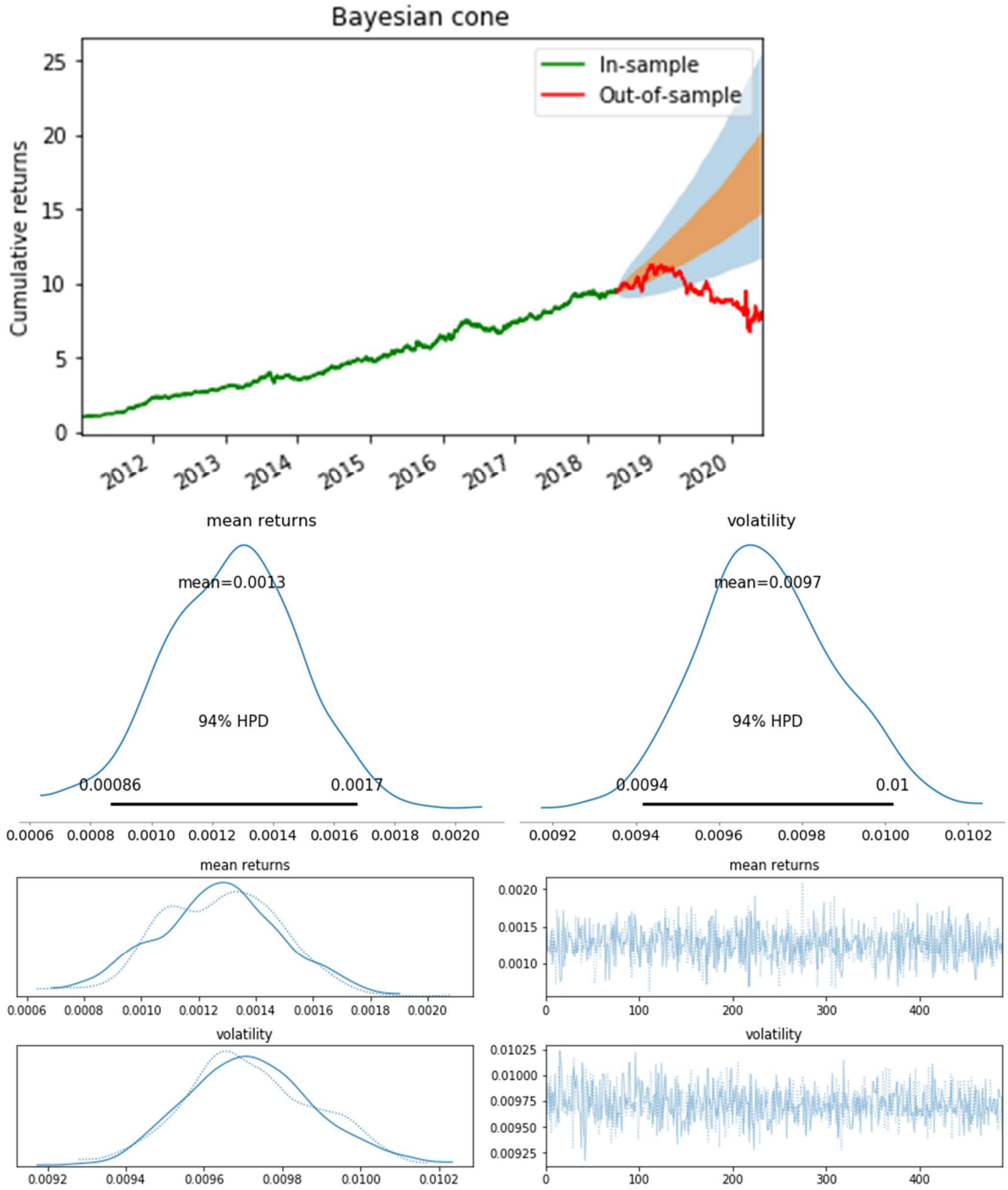
Training model from 2016:



Bayesian Score – 57.17

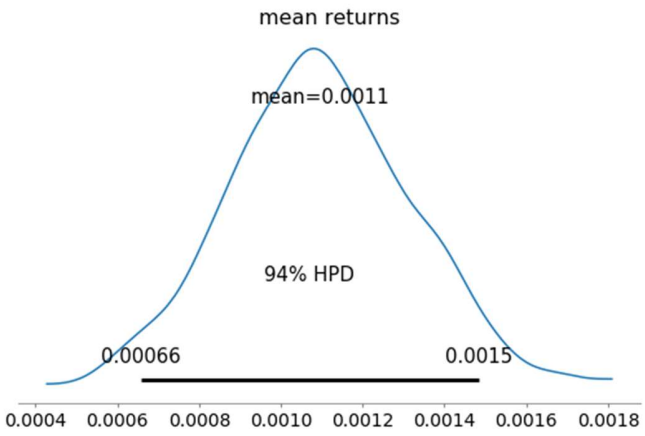


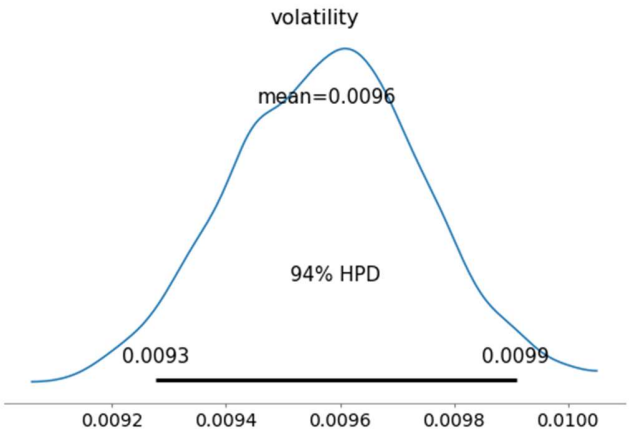
Model 6

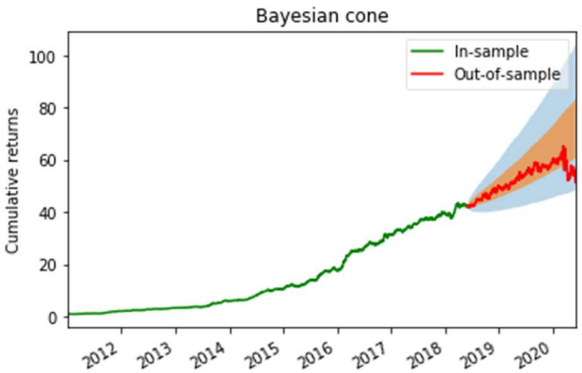


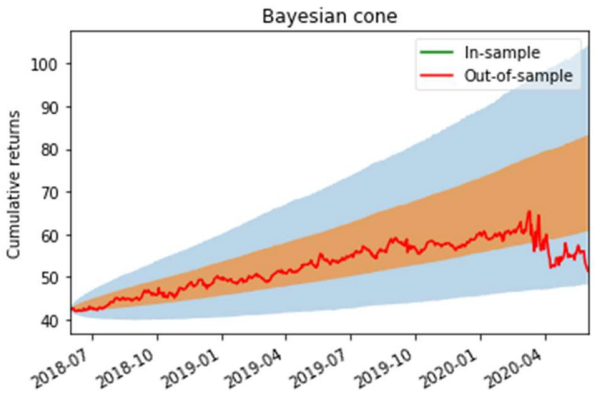
Bayesian Score – 35.93

More on Model 5
After changing mean returns:

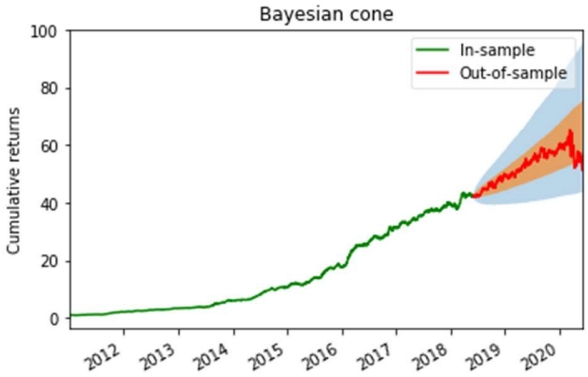


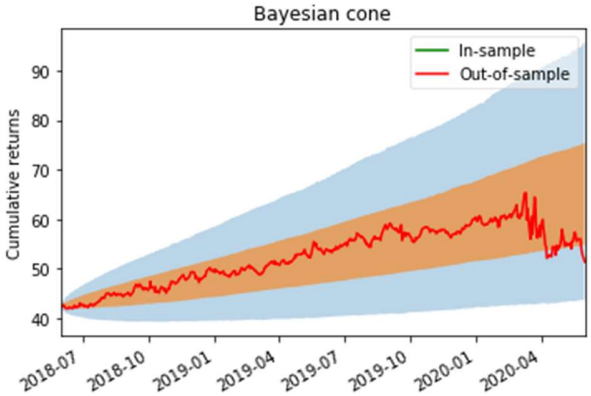






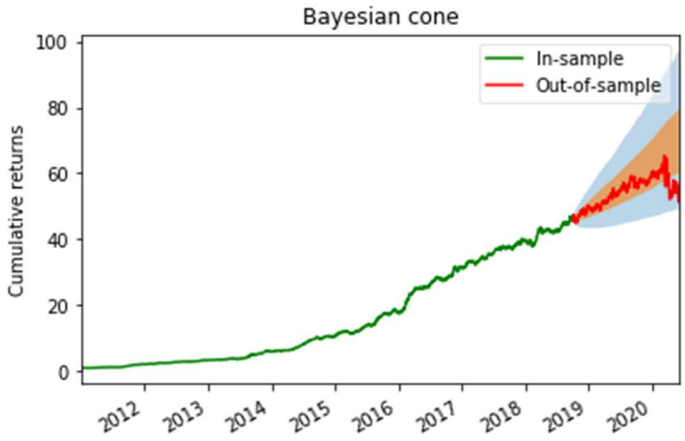
Bayesian Score – 78.08

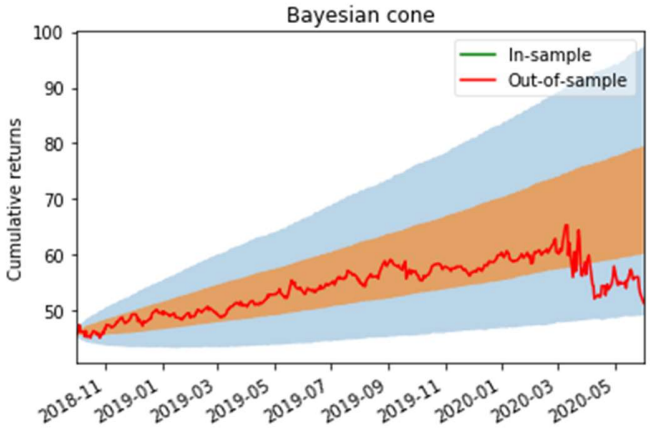




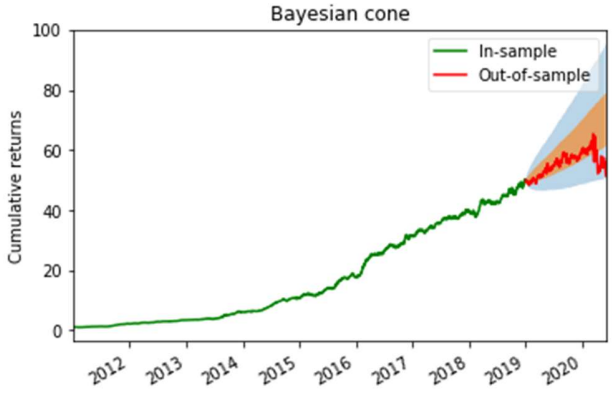
Bayesian Score – 99.77

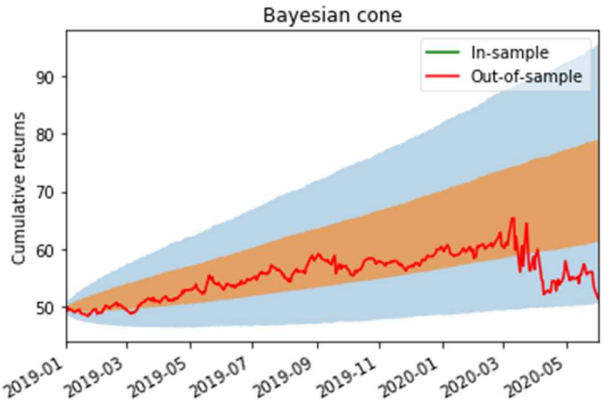
Cones after each qtr:



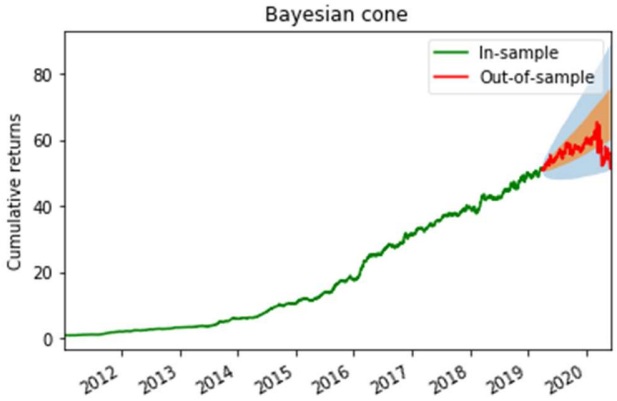


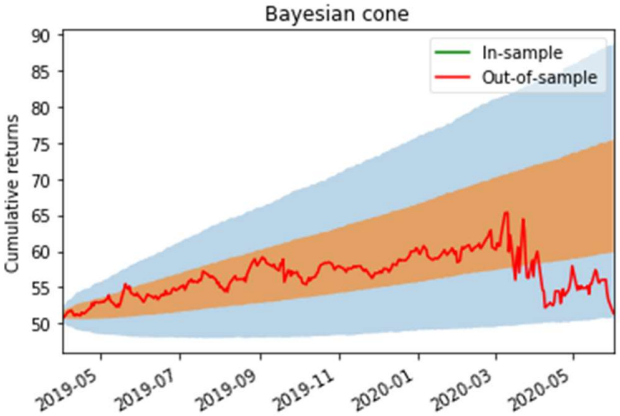
Bayesian Score – 80.31



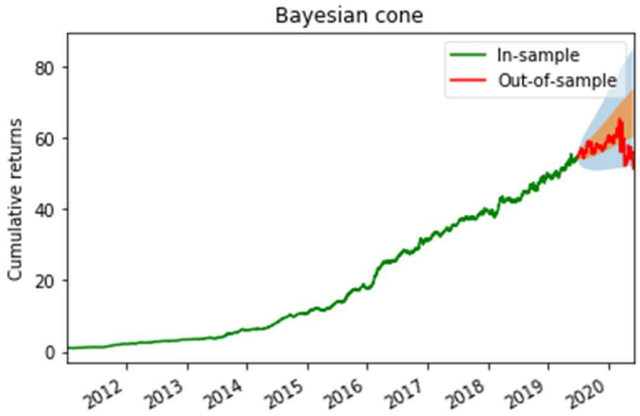


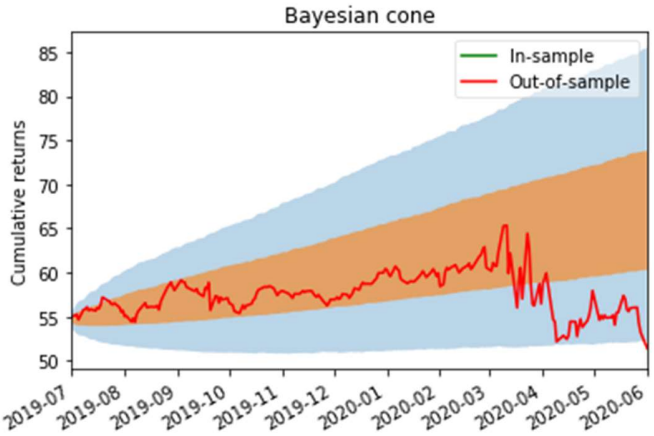
Bayesian Score – 67.80



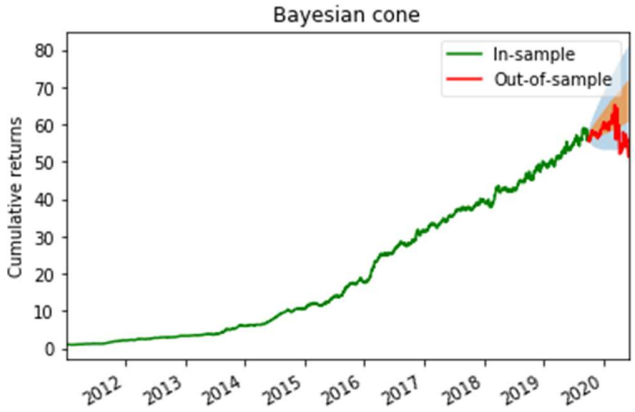


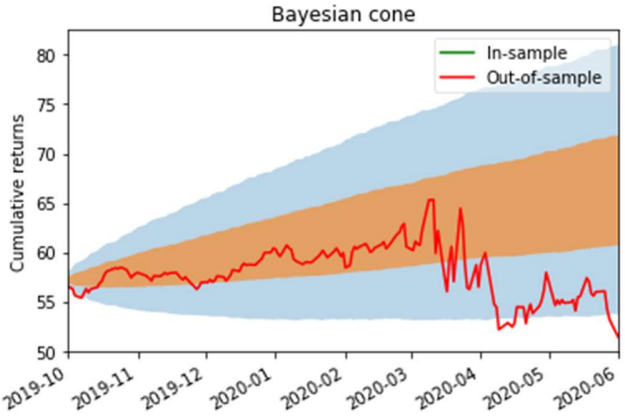
Bayesian Score – 87.16



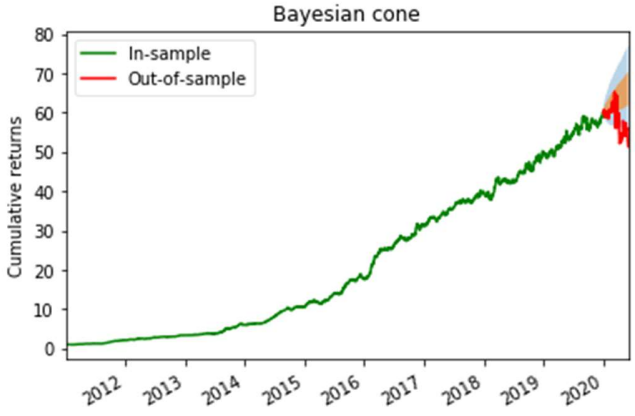


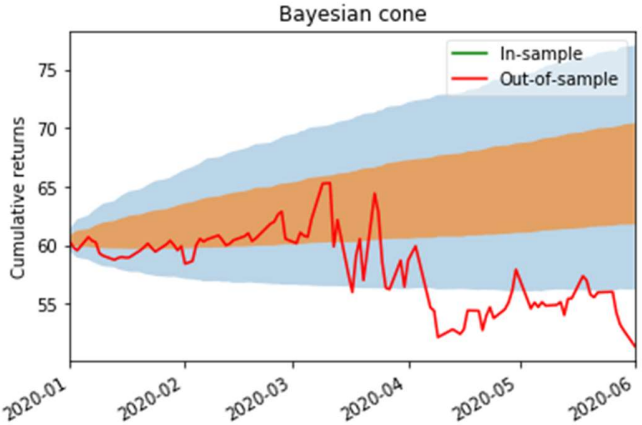
Bayesian Score – 77.59



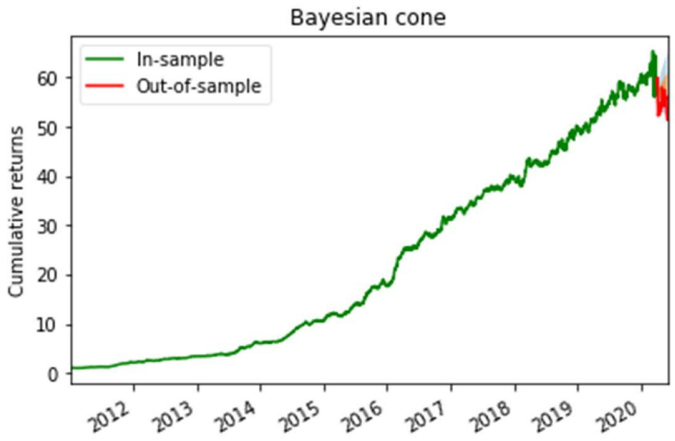


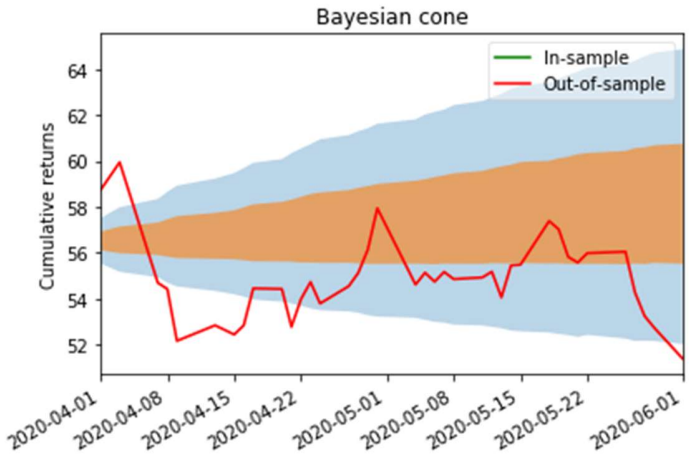
Bayesian Score – 59.89





Bayesian Score – 38.31





Bayesian Score – 42.56