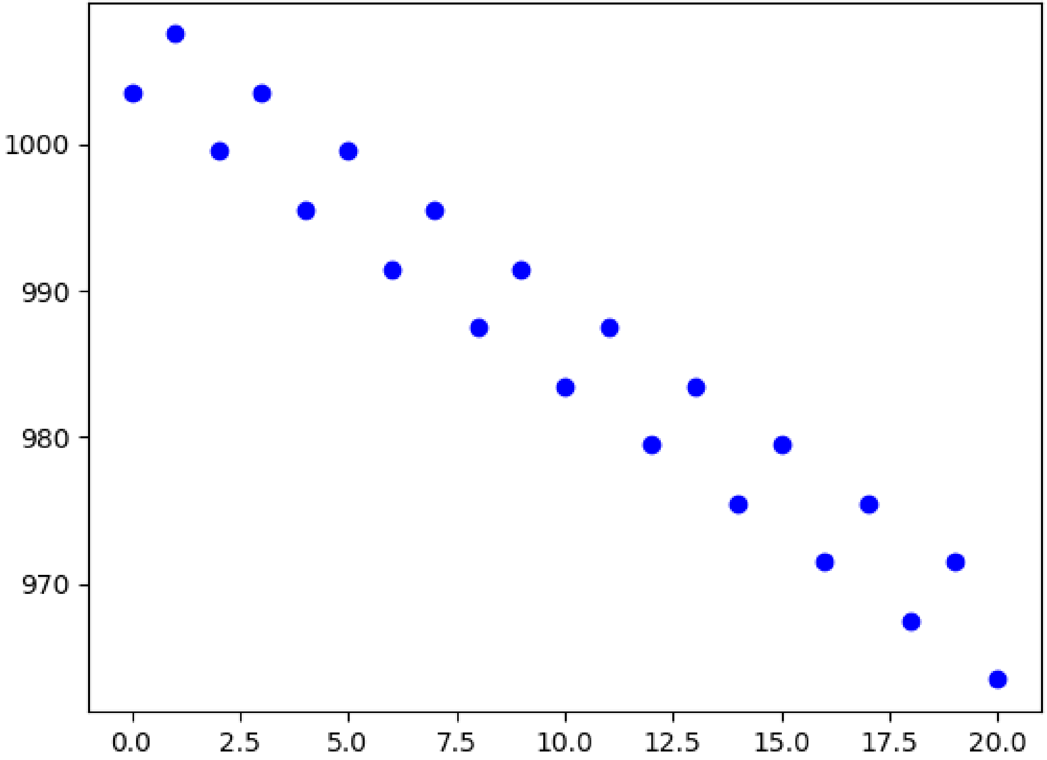
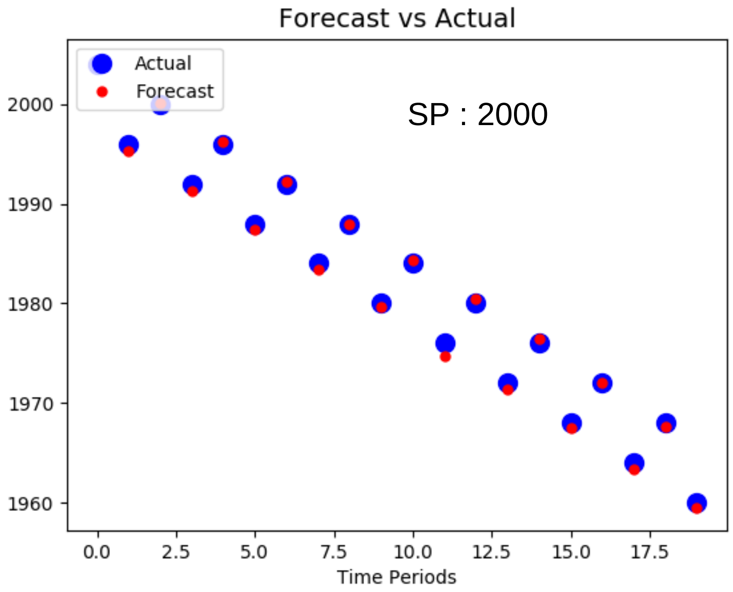
Case study 1 : overview

- Time_series_forecasting.py
- Training set
  - Alternation of +4 and -8
  - starting at 2000
  - Size : 200
- Testing set identical
  - Generated separately
  - Different starting point or not
  - Size : 20

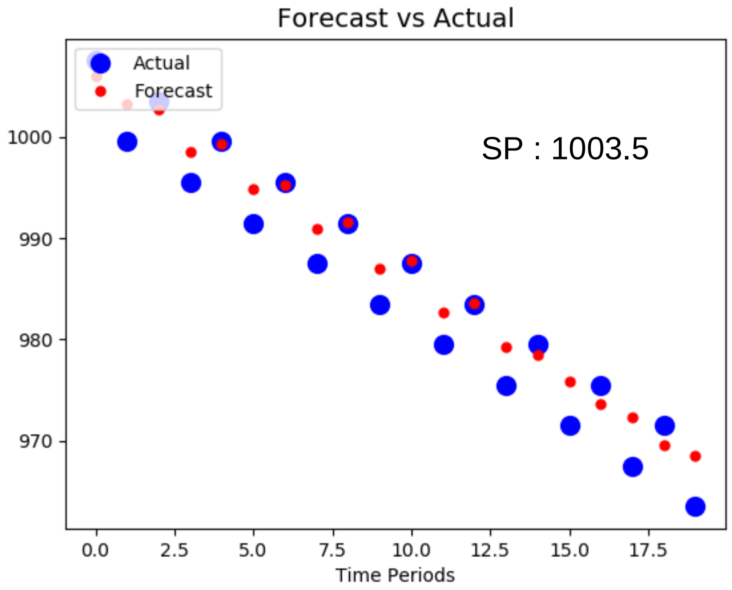


Case study 1 : experimental part



Forecast vs Actual

SP : 2000



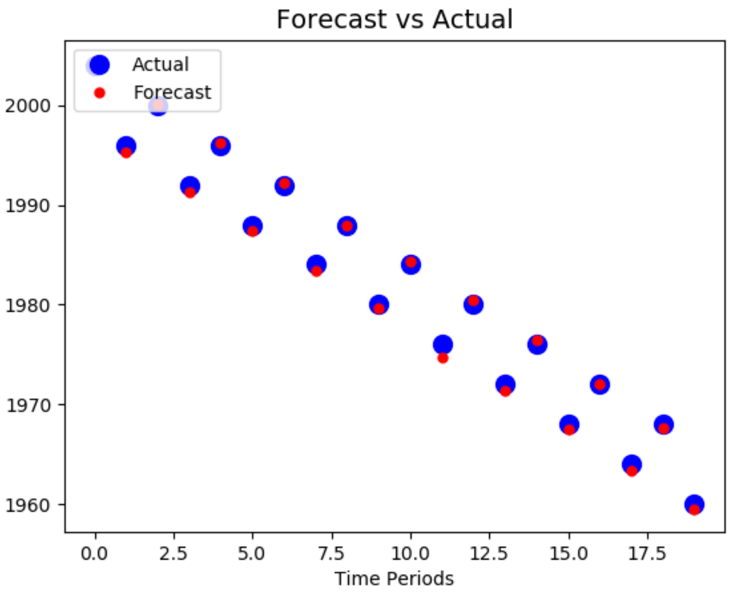
Forecast vs Actual

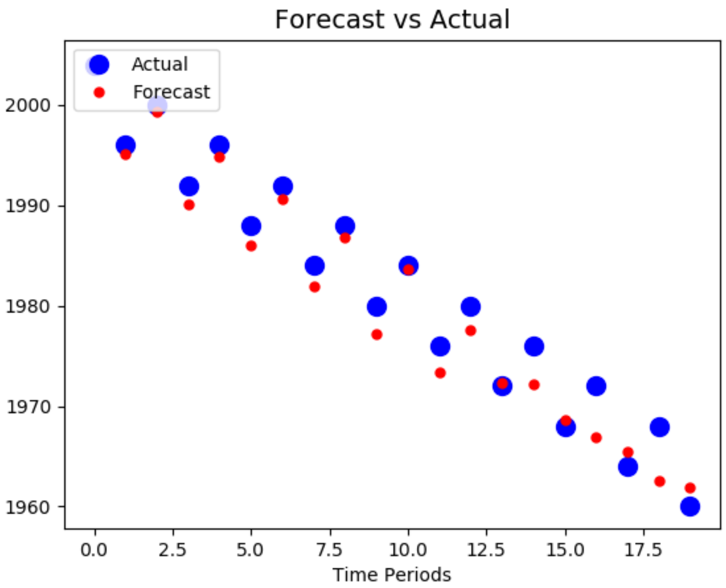
SP : 1003.5

After slight adaptation of tutorial’s script, rather good result. However,

- Sometimes, tendency to « draw a line between dots »
- Sensitive to starting point (overfitting?)

Case study 1 : experimental part

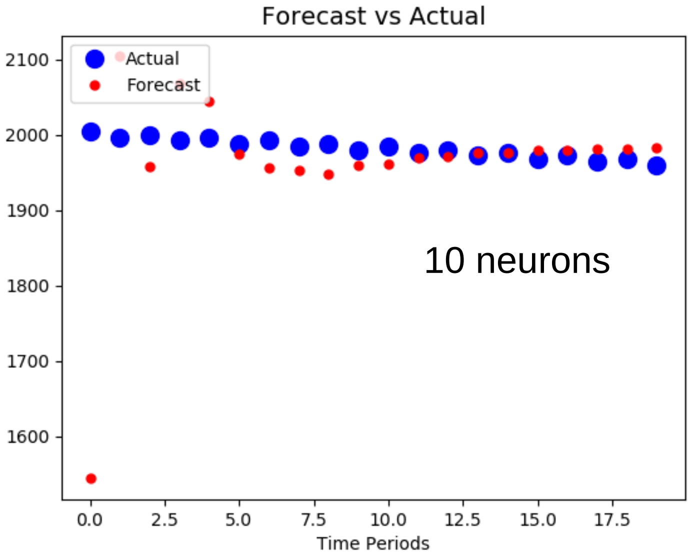




Two successive runs of same script leads to different results…
- model optimization for lower error

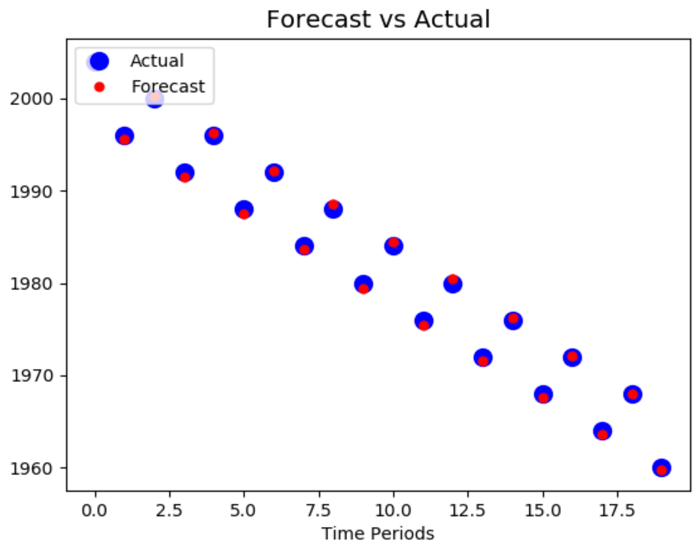
Model architecture

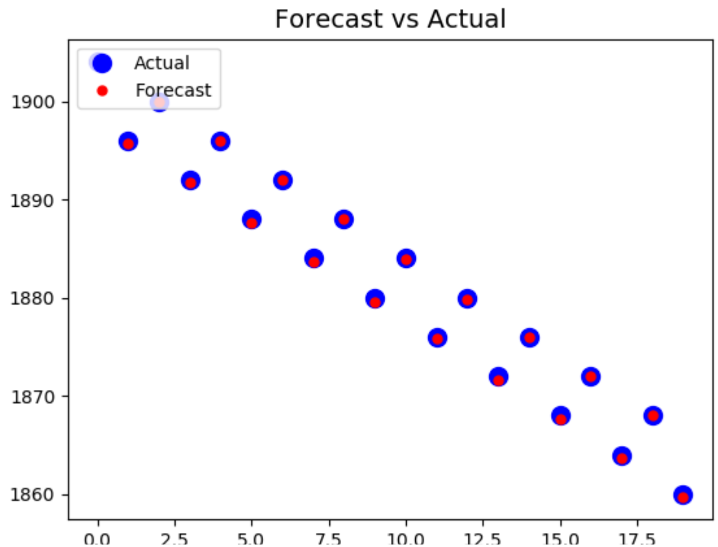
- Recurrent neural network
- 1 input = 20 period-long vector
- 1 hidden layer (100 neurons)
- 1 output = forecasting at t+1

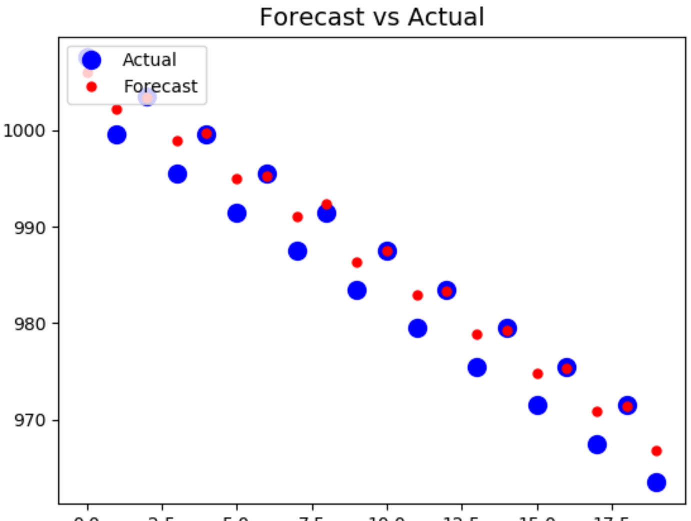

Forecast vs Actual
Actual
Forecast
2100
2000
1900
1800
1700
1600
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Time Periods

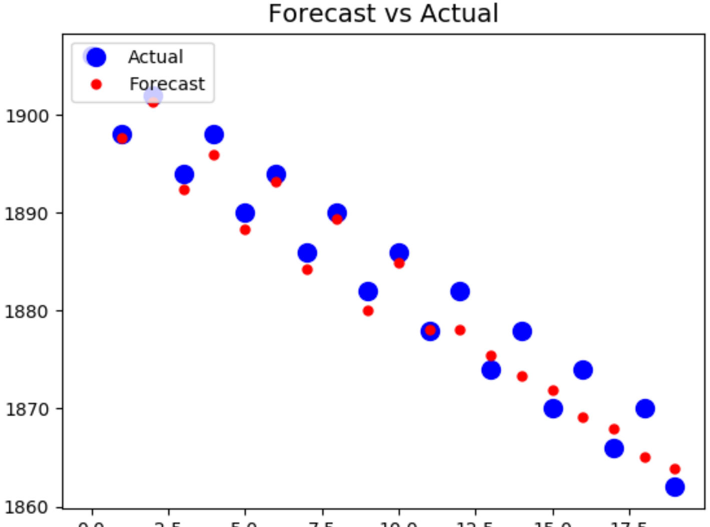
10 neurons

Case study 1 : experimental part


Forecast vs Actual

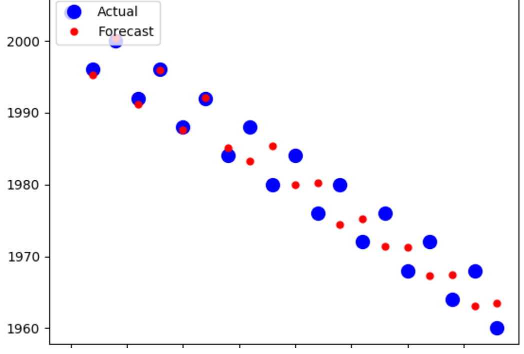

Forecast vs Actual

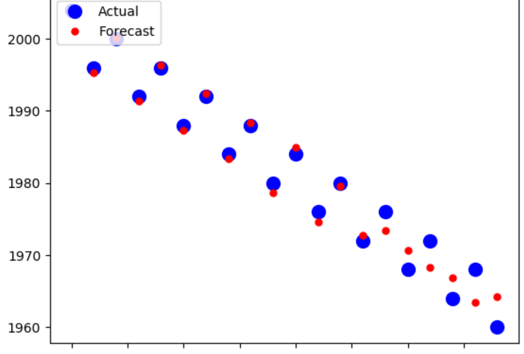

Forecast vs Actual

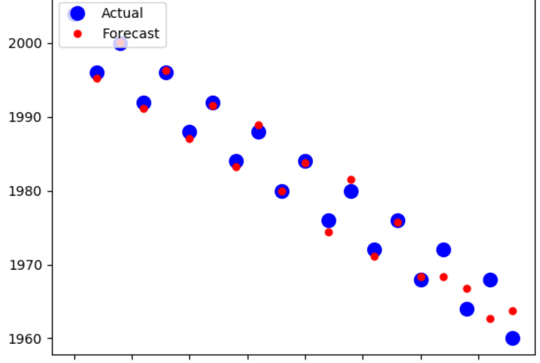

Forecast vs Actual

2000 and 1900 belongs to the training dataset… 1003.5 and 1902 do not → clearly overvitting

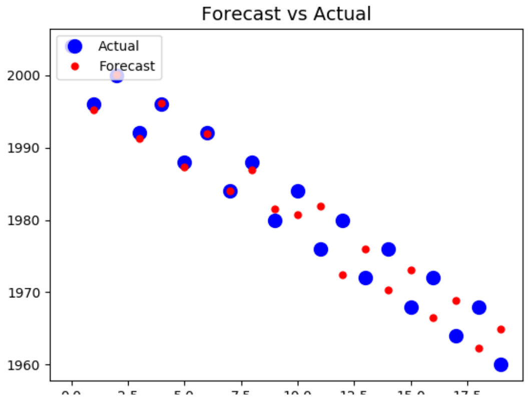
50 neurons

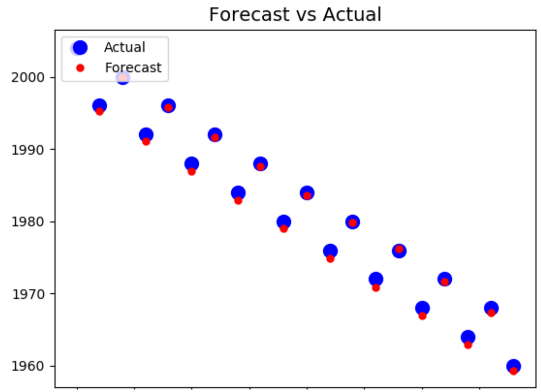


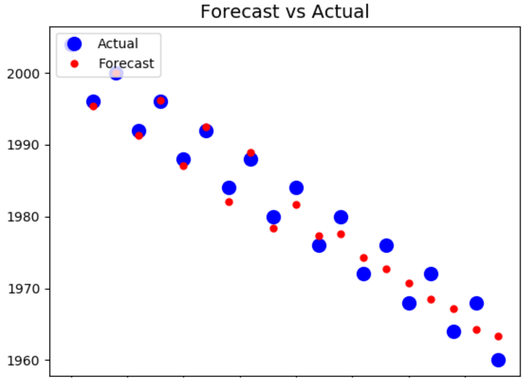




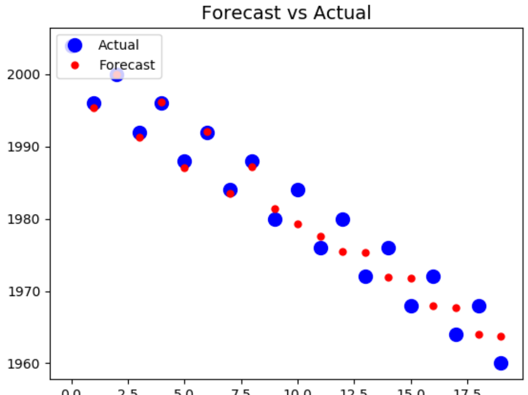
100 neurons

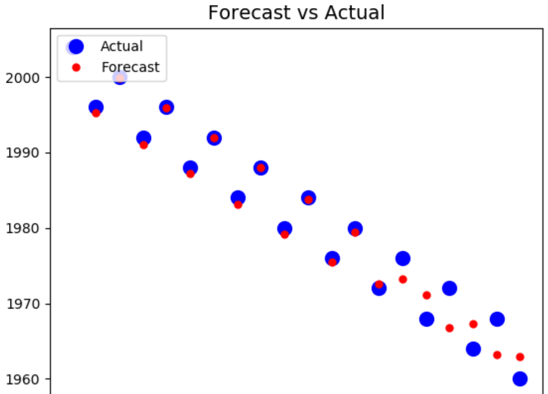

Forecast vs Actual

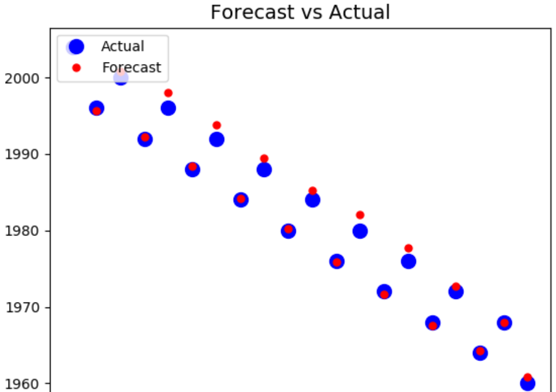

Forecast vs Actual

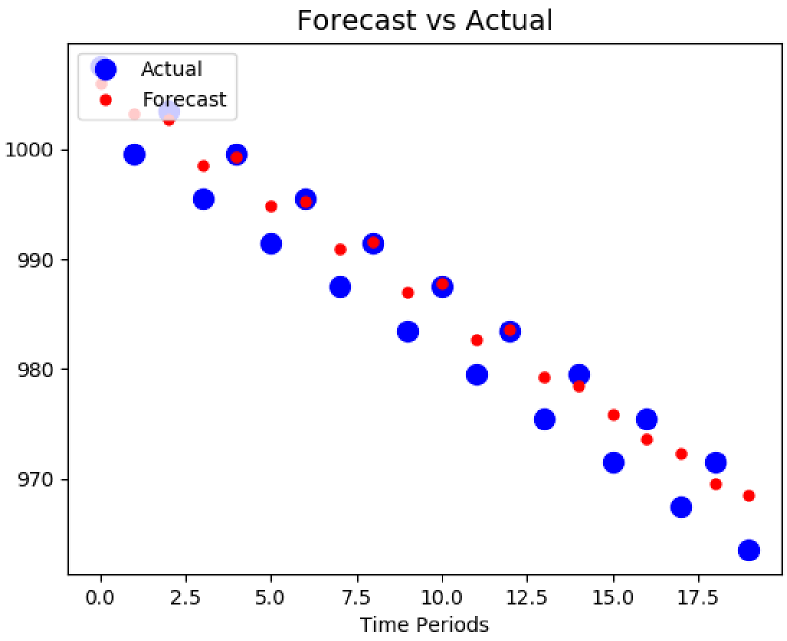

Forecast vs Actual

200 neurons


Forecast vs Actual

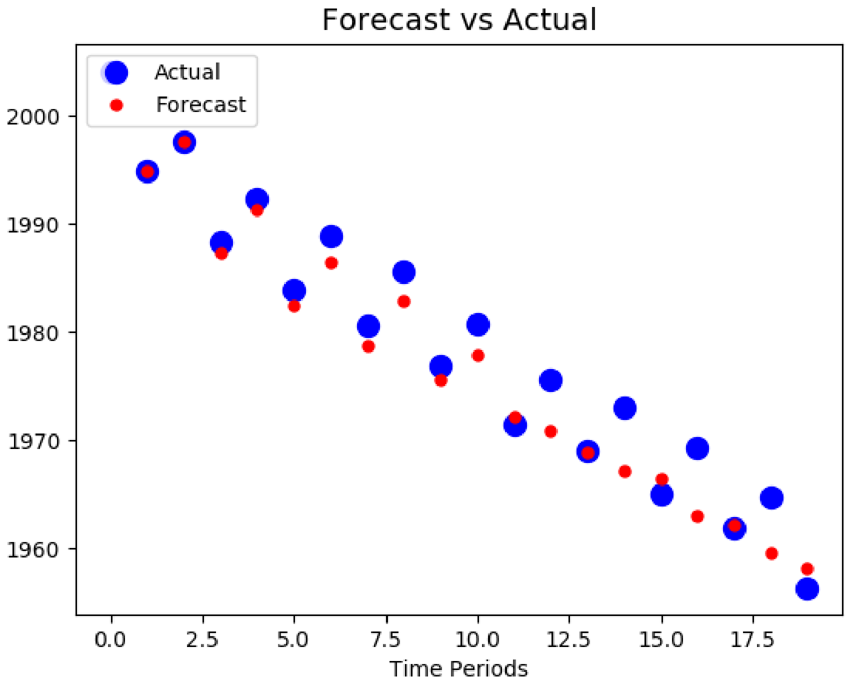

Forecast vs Actual

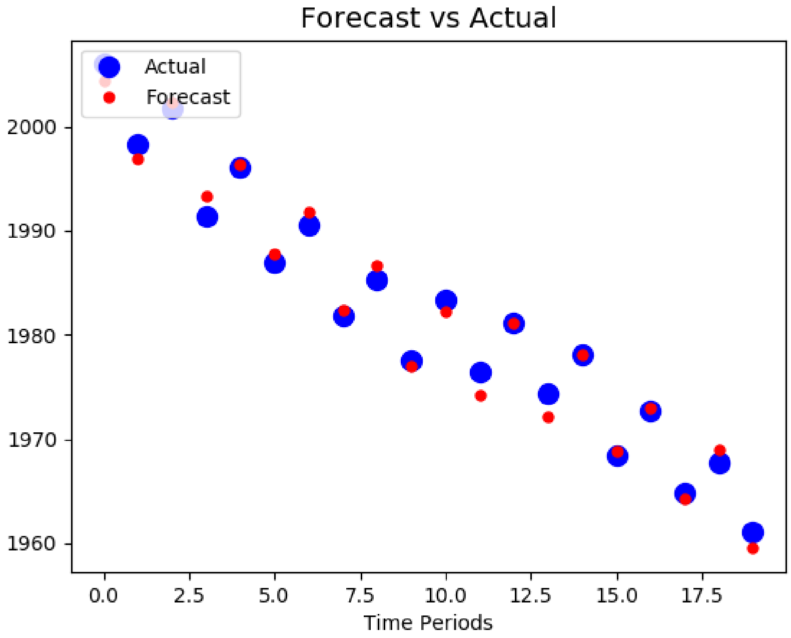

Forecast vs Actual



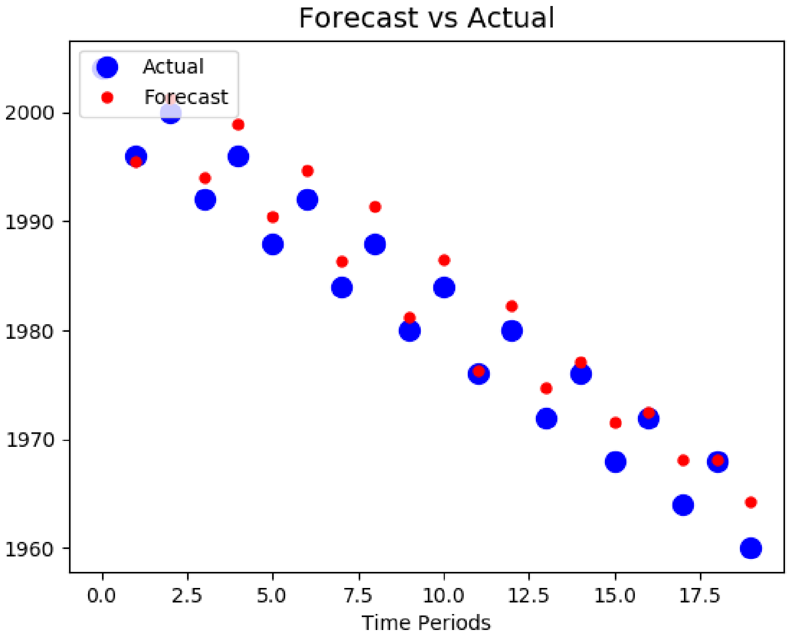
Training from 2000
Starting from 1003.5...

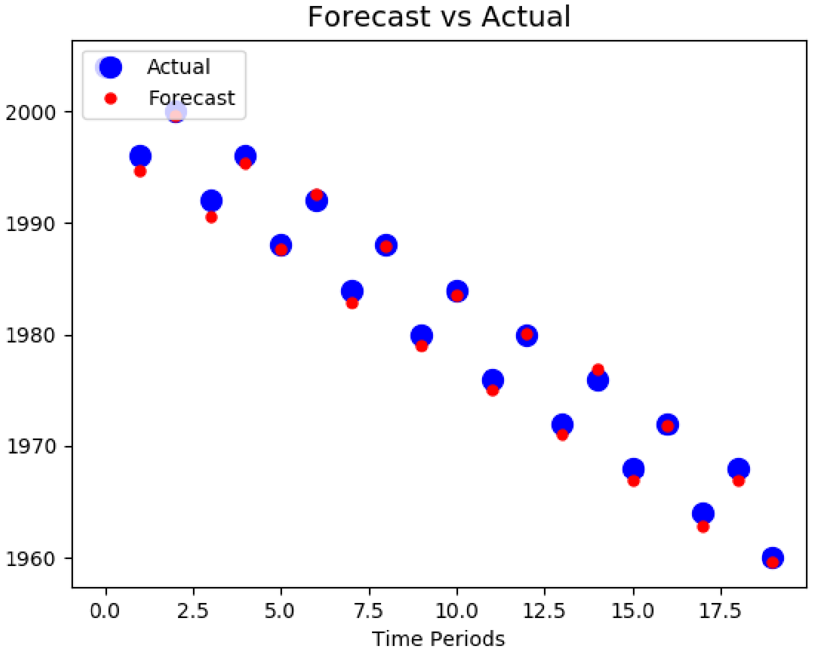
Noise applied to test data, apply noise to prevent overfit





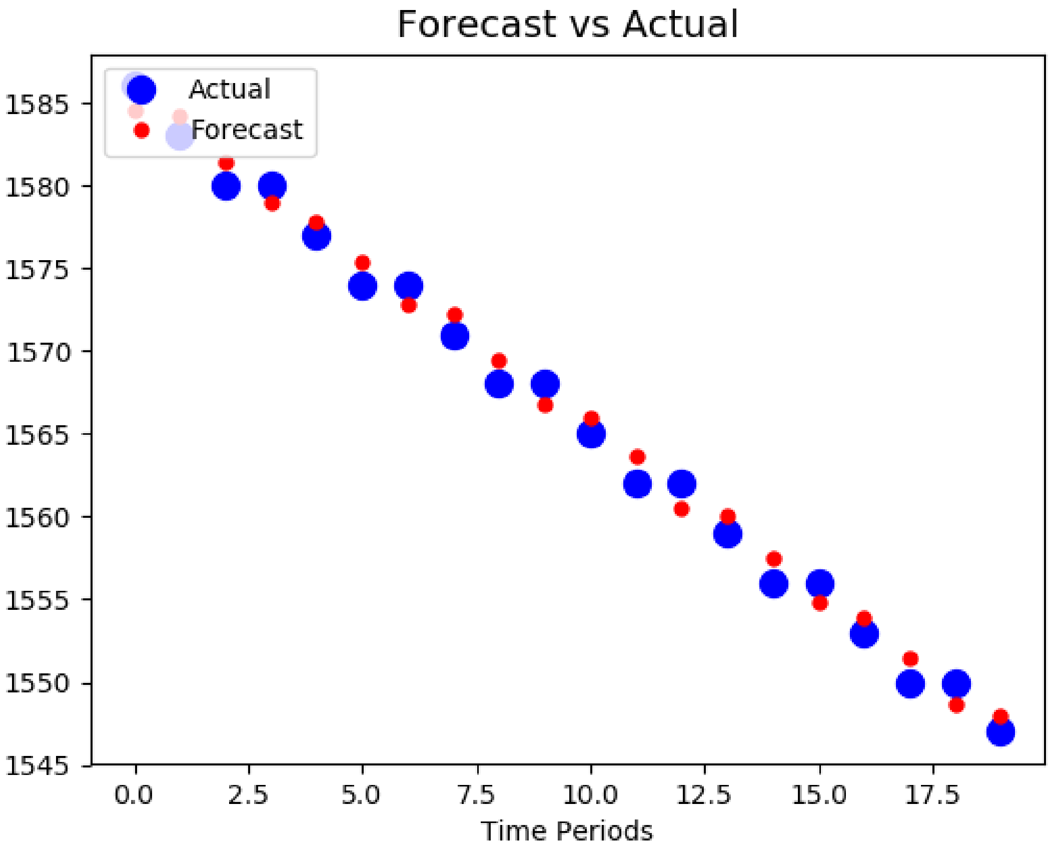
Noise applied to train data


Forecast vs Actual

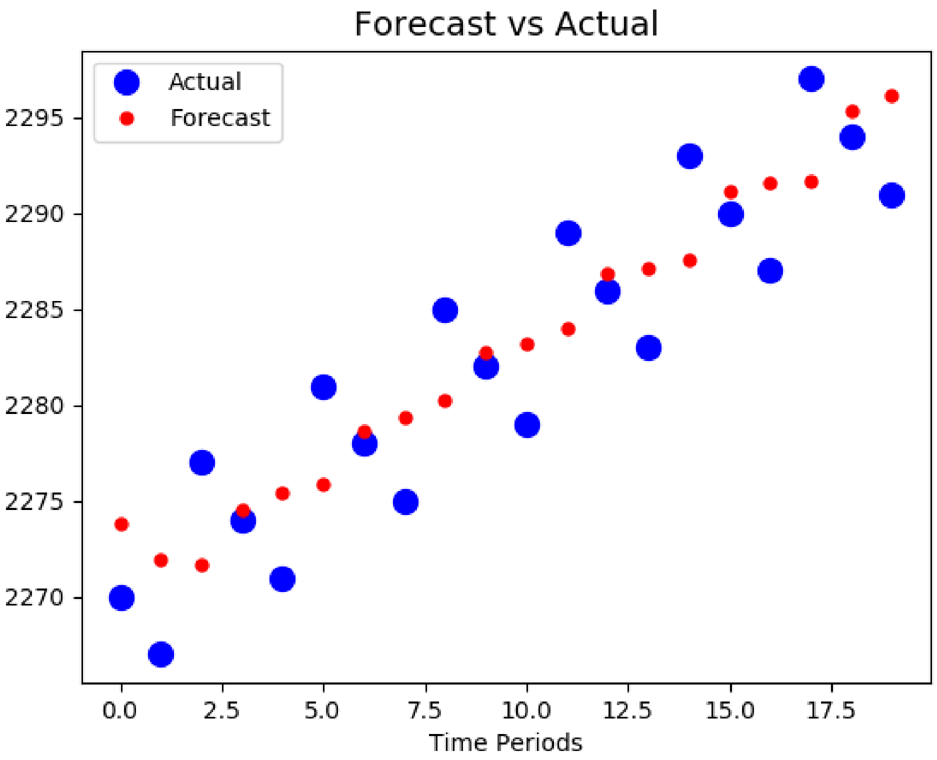

Forecast vs Actual

Mmmh, is the first point around 2000 more noised on the

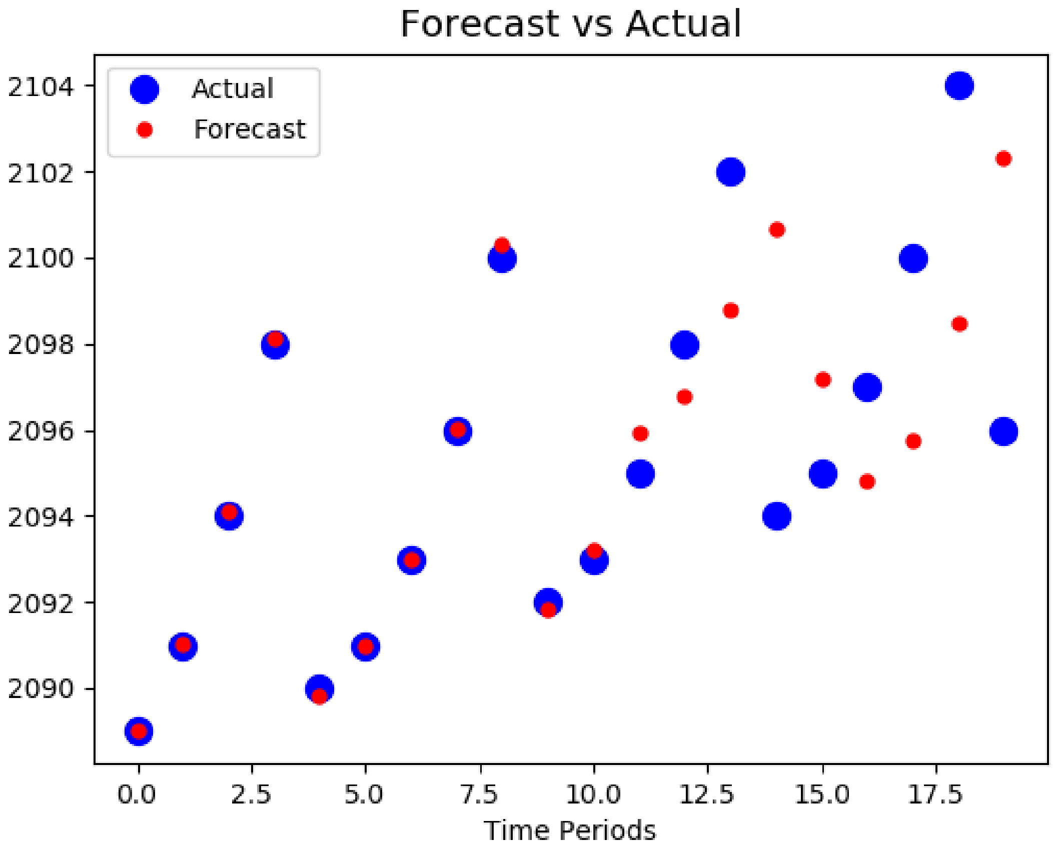
Other patterns...



Other patterns...

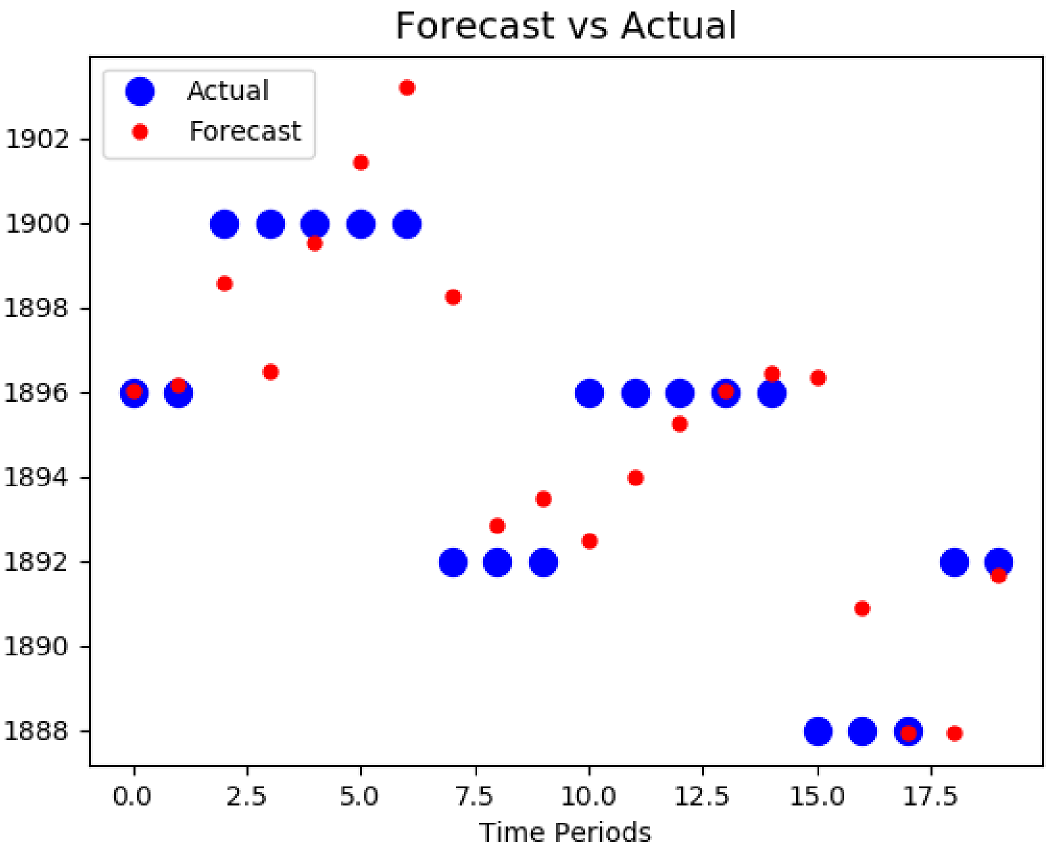


Other patterns...



```
if np.mod(y,5) == 0:
                price[y] = price[y-1] - 8
        else:
                price[y] = price[y-1] + np.mod(y,5)
```

Other patterns...



```
if np.mod(y,8) == 0:
        price[y] = price[y-1] - 8
    elif np.mod(y,8) == 3:
        price[y] = price[y-1] + 4
    else:
        price[y] = price[y-1]
```