Out[1]: Click here to toggle on/off theraw code.

1. Task

This MLP is designed with the goal of predicting stock price for each day, each day's previous 20 closing stock prices.

2. Data

The following implementation uses Boeing (BA) stock.

```
[*********************100%***********************]  1 of 1 completed
```

```
C:\Users\Jasmine\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: Future
Warning: The signature of `Series.to_csv` was aligned to that of `DataFrame.
to_csv`, and argument 'header' will change its default value from False to T
rue: please pass an explicit value to suppress this warning.
  """Entry point for launching an IPython kernel.
```

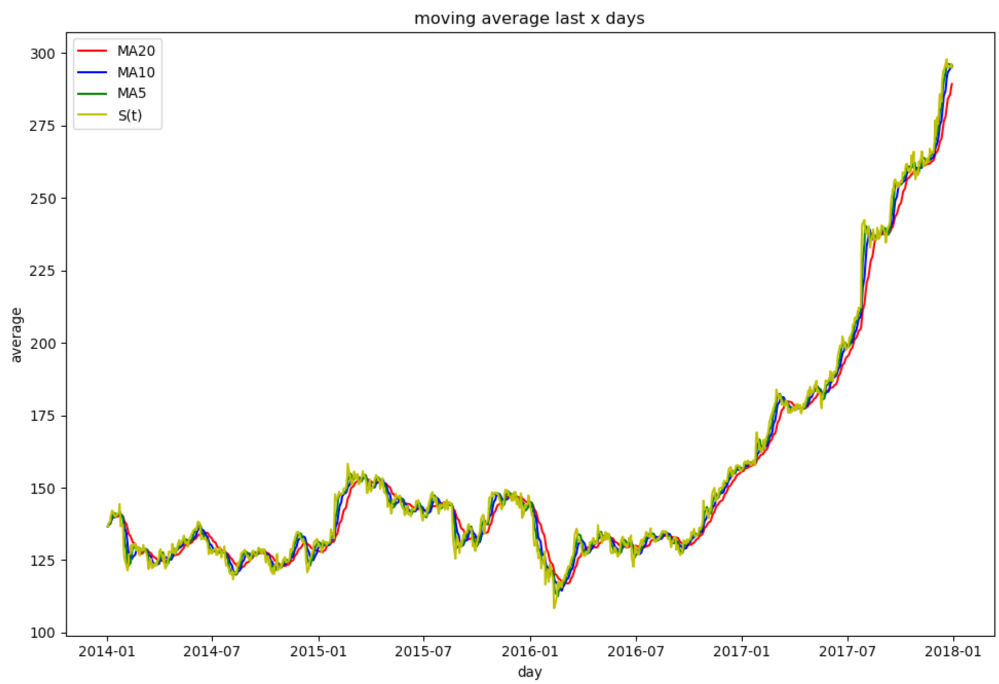
2. Preprocessing

Checking for nulls and evaluating means

Out[4]:
```
Date
2014-01-02    136.669998
2014-01-03    137.619995
2014-01-06    138.410004
2014-01-07    140.509995
2014-01-08    140.820007
                 ...
2017-12-22    295.100006
2017-12-26    295.359985
2017-12-27    295.619995
2017-12-28    296.350006
2017-12-29    294.910004
Name: Close, Length: 1007, dtype: float64
```

Out[5]:
```
Date
2014-01-02    136.669998
2014-01-03    137.144997
2014-01-06    137.566666
2014-01-07    138.302498
2014-01-08    138.806000
                 ...
2017-12-22    293.418002
2017-12-26    294.638000
2017-12-27    295.206000
2017-12-28    295.657001
2017-12-29    295.760001
Name: Close, Length: 1007, dtype: float64
```

A comparison of the daily price,5 day moving average,10 day moving average,and 20 day moving average



Out[144]:
```
<matplotlib.legend.Legend at 0x2b87ac35a88>
```

comments: The actual price seems to stay above the moving averages. 5 day moving average is closest to the actual price followed by the 10 day, then the 20 day. In this case, the smaller the window of calculation, the better the estimates is.

3. Training & Test Set

Combine Data sets of the 3 Moving averages and 15 days recent histories

Out[10]:

	0	1	2	3	4	5	6	
0	138.740999	137.542999	133.481998	126.529999	129.779999	137.089996	137.360001	136.(
1	138.170499	136.047999	131.203999	125.260002	126.529999	129.779999	137.089996	137.:
2	137.443499	134.309998	128.348000	123.080002	125.260002	126.529999	129.779999	137.(
3	136.624999	132.346999	125.338000	122.040001	123.080002	125.260002	126.529999	129.7
4	135.669500	130.049999	123.662001	121.400002	122.040001	123.080002	125.260002	126.5
...	...	...	...	...	...	...	...	
982	282.782999	292.498001	296.052002	295.029999	297.899994	297.250000	296.140015	293.9
983	284.243999	293.418002	296.284003	295.100006	295.029999	297.899994	297.250000	296.1
984	285.732999	294.638000	296.127997	295.359985	295.100006	295.029999	297.899994	297.2
985	287.114499	295.206000	295.801996	295.619995	295.359985	295.100006	295.029999	297.8
986	288.467000	295.657001	295.491998	296.350006	295.619995	295.359985	295.100006	295.(

987 rows × 18 columns

Split of target feature and split into train and test.

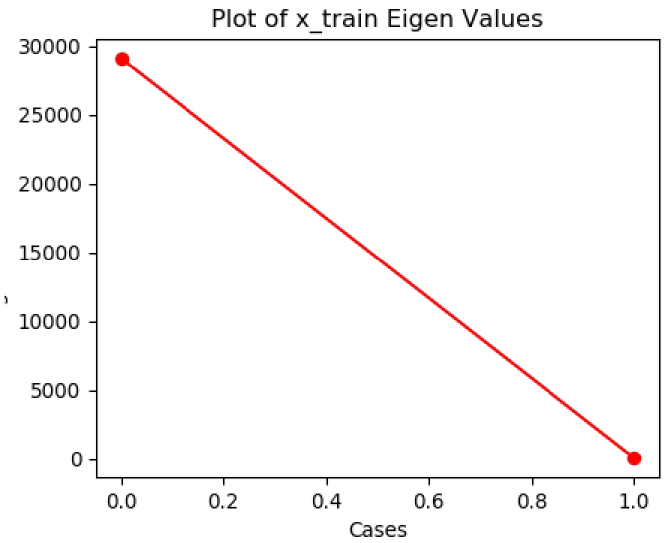
Out[11]: ((987, 18), (987,))

Out[12]: ((888, 18), (99, 18))

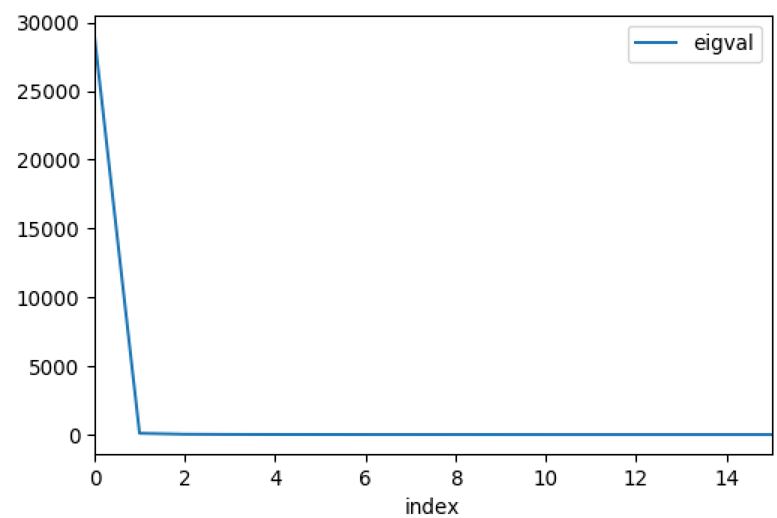
PCA

Perform PCA to choose the number of nodes in the hidden layer

```
Number of components, h90: 2
eigen values from x_train:
 [29074.59, 97.08]
```

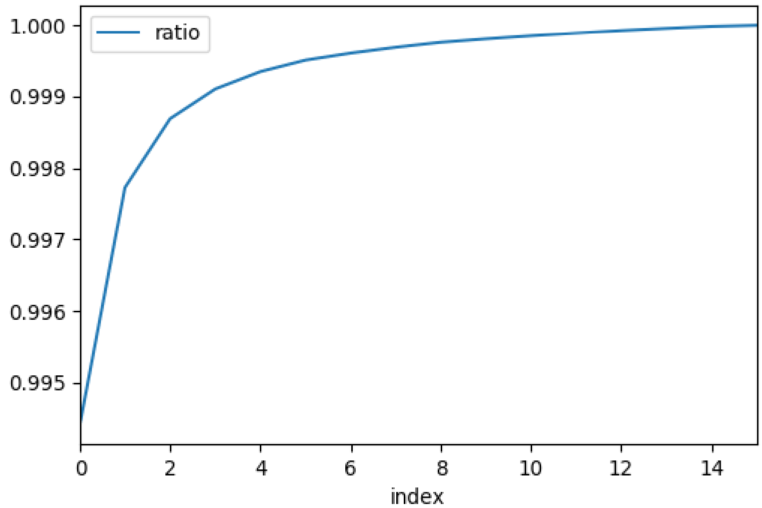


Out[145]: `Text(0.5, 1.0, 'Plot of x_train Eigen Values')`



Out[146]: `<matplotlib.axes._subplots.AxesSubplot at 0x2b879fd9dc8>`

Comments: Most of the eigenvalues are near zero



Out[147]: <matplotlib.axes._subplots.AxesSubplot at 0x2b87c8503c8>

Comments:Very close to 100% explained varience with only 2 Principle components

Out[17]: 0.9977289532253927

num weights and biases

Compute the number w of weights and thresholds in this MLP, and compare w to the number of informations provided by the training set.

TotInfos=total # infos for for 987x1 cases=987X1X1=987

Ratio=987/41 weights and biases or 987/38 just weights

Ratio=24.07 total infos per weight and bias comboniation

Ratio=25.97 total infos per weight

These are considered very reasonable ratios!

4. MLP Predictor

architecture:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 2)                 38
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 3
=================================================================
Total params: 41
Trainable params: 41
Non-trainable params: 0
_________________________________________________________________
```

5. Training of MLP predictor

Comments: Selected model optimizer to be Adam after getting better results, also made made the learning rate 10 times higher.

mlpMonitor = mlp.fit(x_train, y_train, epochs=5000, batch_size=10000, callbacks = [mlpMyMonitor, es], validation_data = (x_test, y_test), verbose = 2)

Train Tuning Report(losses): Epoch 100, BS 32: 40.233

Epoch 100, BS 64: 40.28

Epoch 100, BS 128: 11.97

Epoch 100, BS 256: 6.629

Epoch 50, BS 32: 2.7036

Epoch 50, BS 64: 2.619

Epoch 50, BS 128: 1.5109

Epoch 50, BS 256: 1.476

Epoch 10, BS 32: 2.597

Epoch 10, BS 64: 1.437

Epoch 10, BS 128: 1.3026

Epoch 10, BS 256: 1.31516

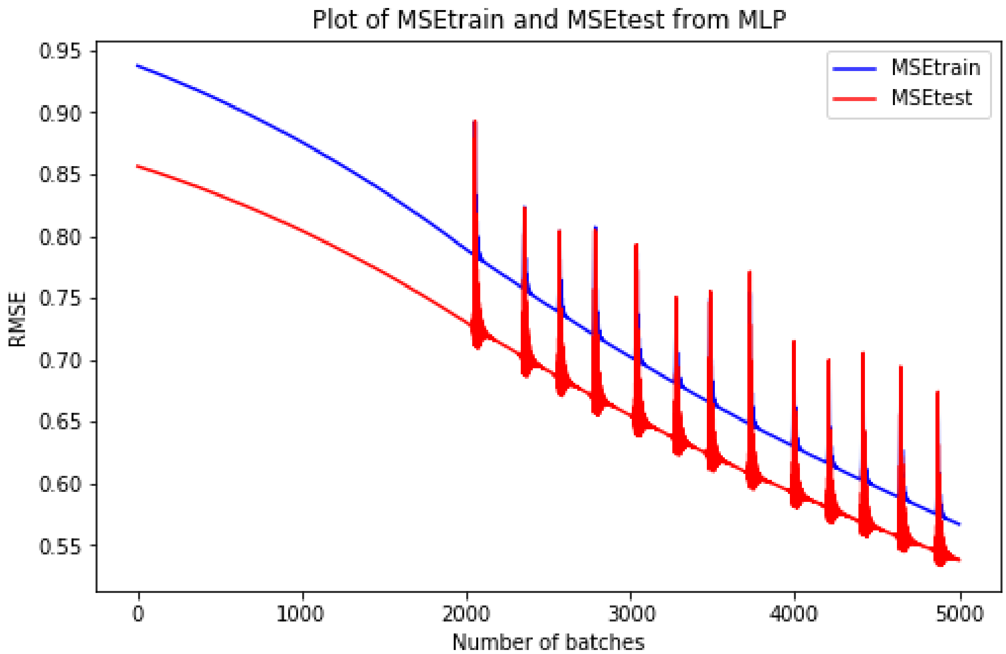
***After observing a trend in decrease of loss when increasing Epoch and Batch size the hyperparameters of epochs=5000, batch_size=10000 were chosen. this resulted in error of .322**runtime was 33 mins

```
888/888 [==============================] - 0s 42us/sample - loss: 0.3219
```

Out[108]:
```
0.3218624987849244
```

RMSE Plots

```
Out[114]: Text(0.5, 1.0, 'Plot of MSEtrain and MSEtest from MLP')
```

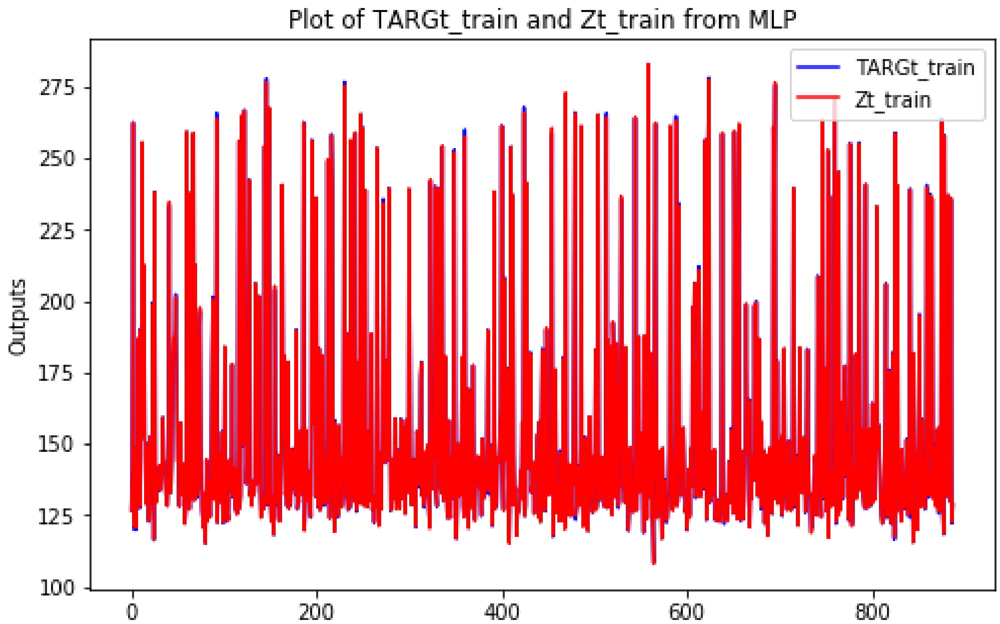


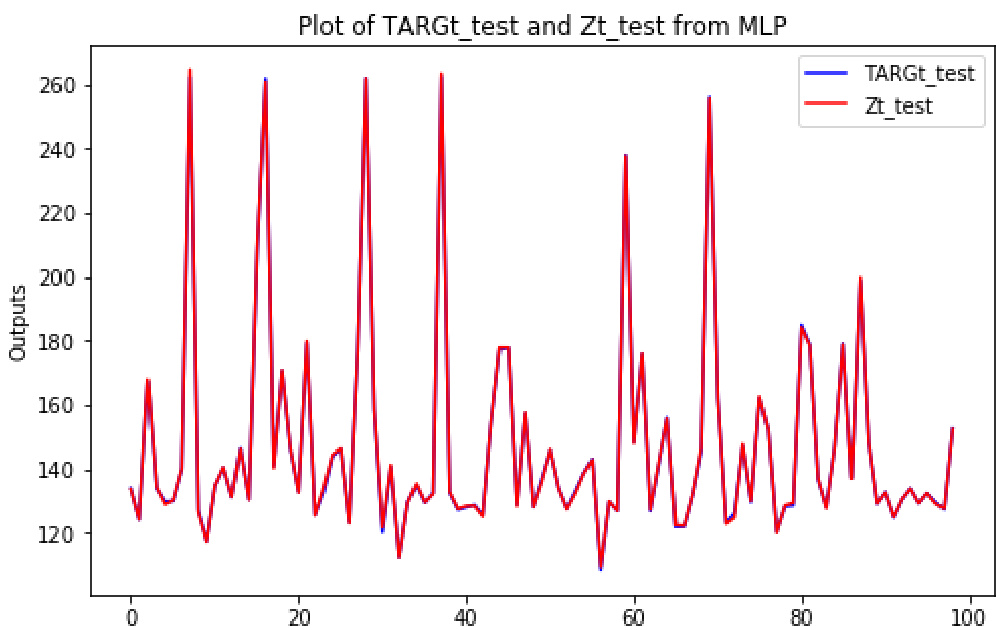
comments: This plot illustrates a gradual decrease in RMSE for both test and train. However, train has a steeper decrease which means it is learning bettern, yet test has lower RMSE better. After seeing about 2000 batches both sets show sudden rises in error that quickly dissipate, this continues throughout the rest of training.

Target V Predicted Plots

Out[115]:
```
Text(0.5, 1.0, 'Plot of TARGt_test and Zt_test from MLP')
```

```
<Figure size 432x288 with 0 Axes>
```





comments:The plots indicate that the model is predicting stock prices that are very close to the actual prices. The training comparison plot shows that the error is commonly a result of the model underestimating the actual price. With the test set, there is no observable trend in cause of error.

MREP's

train MREP

Out[122]:
```
0.0027
```

test MREP

Out[121]:
```
0.0026
```

comments: The MREP is very low for both models this is an indication of extremely accurate performance. Another observation is that the MREP for both sets is almost identical there is only a difference of .0001. This is very important, because a common problem with the MLP model is overfitting. With both sets having nearly identical MREP, it can be concluded that the model has problem with generalizing.

6. Hidden Layer Node Analysis

```
WARNING:tensorflow:Layer dense is casting an input tensor from dtype float64
to the layer's dtype of float32, which is new behavior in TensorFlow 2.  The
layer has dtype float32 because it's dtype defaults to floatx.

If you intended to run this layer in float32, you can safely ignore this war
ning. If in doubt, this warning is likely only an issue if you are porting a
TensorFlow 1.X model to TensorFlow 2.

To change all layers to have dtype float64 by default, call `tf.keras.backen
d.set_floatx('float64')`. To change just this layer, pass dtype='float64' to
the layer constructor. If you are the author of this layer, you can disable
autocasting by passing autocast=False to the base Layer constructor.
```

Out[123]:
```
array([[143.7786 ,   0.     ],
       [163.25966,   0.     ],
       [317.93805,   0.     ],
       ...,
       [284.48108,   0.     ],
       [138.47762,   0.     ],
       [146.39401,   0.     ]], dtype=float32)
```

Weights

Out[124]:
```
array([-19.052961,   5.      ], dtype=float32)
```

Y1 (node 1 mean activity)

```
Out[125]: 71.8893
```

Y2 (node 2 mean activity)

```
Out[126]: 81.62983
```

IMP1

```
Out[127]: -1369.704
```

IMP2

```
Out[128]: 408.14914
```

comments: node 1 had over 3 times the impact that node 2 did

7. Input Layer node Analysis

Un (weights)

Out[129]:
```
array([[ 0.03180819, -0.00138187],
       [ 0.0597933 ,  0.34437972],
       [-0.32013065, -0.35270762],
       [ 0.01332817,  0.01572883],
       [ 0.02285013, -0.40769258],
       [ 0.24521337,  0.14818501],
       [-0.09524061, -0.49049821],
       [ 0.08580012, -0.44354504],
       [-0.01636629,  0.15898448],
       [ 0.0777326 ,  0.4087631 ],
       [-0.09147263,  0.05419827],
       [-0.06120602,  0.43324876],
       [ 0.22592591, -0.42854345],
       [-0.32335147,  0.4599594 ],
       [ 0.23842485, -0.03919548],
       [-0.19482055,  0.05586547],
       [ 0.29518902,  0.19086069],
       [ 1.0912493 , -0.3986366 ]], dtype=float32)
```

Input mean activities

Out[130]:
```
0     153.672931
1     154.434193
2     154.829935
3     155.157457
4     154.988693
5     154.828075
6     154.667994
7     154.507457
8     154.351712
9     154.196160
10    154.038531
11    153.880801
12    153.725046
13    153.569767
14    153.415937
15    153.264732
16    153.121611
17    152.975947
dtype: float64
```

Fs (impacts)

Out[138]:
```
0       4.888058
1       9.234130
2      49.565807
3       2.067965
4       3.541511
5      37.965915
6      14.730674
7      13.256758
8       2.526165
9      11.986068
10     14.090309
11      9.418432
12     34.730470
13     49.657010
14     36.578172
15     29.859120
16     45.199819
17    166.934903
dtype: float64
```

top 5 input impacts

Out[139]:
```
17    166.934903
13     49.657010
2      49.565807
16     45.199819
5      37.965915
dtype: float64
```

comments: 17,13,2,16, and 5 had the highest impacts. All of these inputs were direct closing prices, as opposed to the other type of inputs that were moving averages of prices. It can be noted that the furthest day from the target prediction had the most meaning. But, aside from that there was no clear trend as far as how time plays a role in prediction. It would be helpful to investigate that further, if recent or historic observations are more meaningful.