In [ ]:
```
%config InlineBackend.figure_format = 'retina'
```

Testing Enviornment

In [ ]:
```
import pandas as pd
from pandas import DataFrame
import numpy as np
```

Running tests

The following section unit-tests most of the code written for the proof of concept prototype.

In [ ]:
```
%run ../test/test_algorithms.py
```

In [ ]:
```
%run ../test/test_dataloader.py
```

In [ ]:
```
%run ../test/test_predictorsI.py
```

In [ ]:
```
%run ../test/test_predictorsII.py
```

In [ ]:
```
%run ../test/test_activate.py
```

Running example of the system

In [ ]:
```
%run ../consensus/algorithms.py
```

In [ ]:
```
%run ../tools/dataloader.py
```

In [ ]:
```
test = DataLoader('aapl', '2009-01-01', '2010-02-10')
```

In [ ]:
```
prices = test.get_close()
```

In [ ]:
```
prices
```

In [ ]:
```
#prices = np.array(prices)
#len(prices)
```

In [ ]:
```
prices
```

In [ ]:
```
%run ../tools/predictorsI.py
```

In [ ]:
```
op0 = BasicUnivariatePredictor(prices, 24, 30)
op1 = BasicUnivariatePredictor(prices, 25, 7)
op2 = BasicUnivariatePredictor(prices, 25, 7)
op3 = BasicUnivariatePredictor(prices, 25, 7)
```

In [ ]:
```
op0.create_bilstm()
```

In [ ]:
```
op0.model_blueprint()
```

In [ ]:
```
op0.fit_model(150)
```

In [ ]:
```
op0.show_performance()
```

In [ ]:
```
op0.save_model()
```

In [ ]:
```
oyea = prices[-26:-1]
#oyea = X[-1]
#oyea
```


In [ ]:
```
nice = op0.predict(oyea)
nice
```

In [ ]:
```
op1.create_lstm()
```

In [ ]:
```
op1.model_blueprint()
```

In [ ]:
```
op1.fit_model(10)
```

In [ ]:
```
op1.show_performance()
```

In [ ]:
```
nice = op1.predict(oyea)
nice
```

In [ ]:
```
op2.create_cnn()
```

In [ ]:
```
op2.model_blueprint()
```

In [ ]:
```
op2.fit_model(10)
```

In [ ]:
```
op2.show_performance()
```

In [ ]:
```
nice = op2.predict(oyea)
nice
```

In [ ]:
```
op3.create_mlp()
```

In [ ]:
```
op3.model_blueprint()
```

In [ ]:
```
op3.fit_model(100)
```

In [ ]:
```
op3.show_performance()
```

In [ ]:
```
oyea = prices[-26:-1]
#oyea = X[-1]
#oyea
```

In [ ]:
```
nice = op3.predict(oyea)
nice
```

In [ ]:
```
%run ../tools/predictorsII.py
```

In [ ]:
```
oo = UnivariatePredictorII(prices, 7)
```

In [ ]:
```
oo.fit_neural_model(100,"D")
```

In [ ]:
```
oo.show_performance_neural()
```

In [ ]:
```
oo.predict_neural()
```

In [ ]:
```
oo.fit_prophet_model()
```

In [ ]:
```
oo.show_performance_prophet()
```

In [ ]:
```
oo.predict_prophet()
```

In [ ]:
```
%run ../tools/predictorsIII.py
```

In [ ]:
```
len(prices)
```

In [ ]:
```
op4 = HybridUnivariatePredictor(prices,2, 24, 7)
```

In [ ]:
```
op4.create_cnnlstm()
```

In [ ]:
```
op4.model_blueprint()
```

In [ ]:
```
op4.fit_model(10)
```

In [ ]:
```
op4.show_performance()
```

In [ ]:
```
oyea = prices[-25:-1]
#oyea = X[-1]
#oyea
```

In [ ]:
```
nice = op4.predict(oyea)
nice
#noice = pd.DataFrame(nice, columns=['yea'])
#noice = nice.reshape(20, 1)
#noice = pd.DataFrame(noice, columns=['yea'])
```

In [ ]:
```
nice.plot()
```

Whole system test - I am alive v.2

In [1]:
```
%run ../tools/dataloader.py
%run ../system/activate.py
```

In [5]:
```
training = DataLoader('aapl', '2009-01-01', '2010-05-01')
```

In [6]:
```
training = training.get_close()
```

In [ ]:
```
#real = DataLoader('aapl', '2010-05-01', '2010-05-28')
```

In [ ]:
```
#real = real.get_close()
```

In [ ]:
```
#len(real)
```

In [2]:
```
predict = DataLoader('aapl', '2010-06-01', '2010-09-01')
```

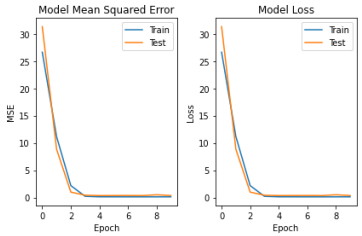
In [3]:
```
predict = predict.get_close()
```

In [4]:
```
predict_req, real = data_prep(predict, 24, 30)
```

In [ ]:
```
#real
```

In [7]:
```
final_df1 = individual_predictors1(training, predict_req, 30)
```

```
ared_error: 0.1170 - val_loss: 0.3637 - val_mean_squared_error: 0.3637
```



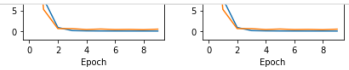
In [8]:
```
final_df2 = individual_predictors2(training, predict_req, 30)
```

```
unction repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), p
lease define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes a
rgument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/tutorials/customization/p
erformance#python_or_tensor_args (https://www.tensorflow.org/tutorials/customization/performance#python_or_tensor_args) and http
s://www.tensorflow.org/api_docs/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/function) for  more details.
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000020B0A451
CA0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.f
unction repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), p
lease define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes a
rgument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/tutorials/customization/p
erformance#python_or_tensor_args (https://www.tensorflow.org/tutorials/customization/performance#python_or_tensor_args) and http
s://www.tensorflow.org/api_docs/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/function) for  more details.
WARNING:tensorflow:7 out of the last 7 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000020B0A5D7
9D0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.f
unction repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), p
lease define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes a
rgument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/tutorials/customization/p
erformance#python_or_tensor_args (https://www.tensorflow.org/tutorials/customization/performance#python_or_tensor_args) and http
s://www.tensorflow.org/api_docs/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/function) for  more details.
```

In [9]:
```
final_df3 = individual_predictors3(training, predict_req, 30)
```

```
f.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For
 (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that re
laxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/tutorials/customiz
ation/performance#python_or_tensor_args (https://www.tensorflow.org/tutorials/customization/performance#python_or_tensor_args) and
https://www.tensorflow.org/api_docs/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/function) for  more details.
WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000020B160
8F040> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @t
f.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For
 (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that re
laxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/tutorials/customiz
ation/performance#python_or_tensor_args (https://www.tensorflow.org/tutorials/customization/performance#python_or_tensor_args) and
https://www.tensorflow.org/api_docs/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/function) for  more details.
WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000020B160
8FC10> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @t
f.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For
 (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that re
laxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/tutorials/customiz
ation/performance#python_or_tensor_args (https://www.tensorflow.org/tutorials/customization/performance#python_or_tensor_args) and
https://www.tensorflow.org/api_docs/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/function) for  more details.
```

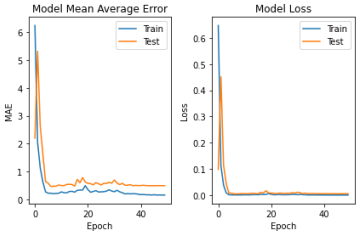
In [10]:
```
final_df4 = individual_predictors4(training, predict_req, 30)
```




```
WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000020B0AA
998B0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @t
f.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For
(1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that rel
axes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/tutorials/customiza
tion/performance#python_or_tensor_args (https://www.tensorflow.org/tutorials/customization/performance#python_or_tensor_args) and
https://www.tensorflow.org/api_docs/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/function) for  more details.
WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000020B01D
41B80> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @t
f.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For
 (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that re
laxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/tutorials/customiz
ation/performance#python_or_tensor_args (https://www.tensorflow.org/tutorials/customization/performance#python_or_tensor_args) and
https://www.tensorflow.org/api_docs/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/function) for  more details.
```

In [11]:
```
final_df5 = individual_predictors5(training, 20)
```

```
INFO: nprophet.utils - set_auto_seasonalities: Disabling yearly seasonality. Run NeuralProphet with yearly_seasonality=True to override
this.
INFO: nprophet.utils - set_auto_seasonalities: Disabling daily seasonality. Run NeuralProphet with daily_seasonality=True to override t
his.
INFO: nprophet.config - set_auto_batch_epoch: Auto-set batch_size to 8

96%                                                    96/100 [00:00<00:00, 125.97it/s]

INFO: nprophet - _lr_range_test: learning rate range test found optimal lr: 2.31E-01
Epoch[50/50]: 100%|| 50/50 [00:02<00:00, 20.38it/s, SmoothL1Loss=0.000612, MAE=0.16, RegLoss=0, MAE_val=0.497, SmoothL
```



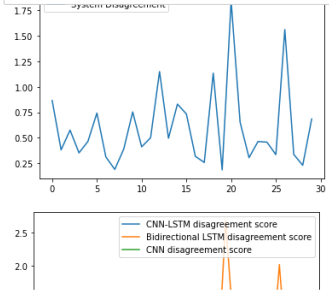

```
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
INFO:fbprophet:Making 20 forecasts with cutoffs between 2009-10-02 00:00:00 and 2010-04-10 00:00:00

100%                                                    20/20 [00:29<00:00, 1.56s/it]
```

In [12]:
```
final_df6 = individual_predictors_pretrained1(predict_req, 30)
```

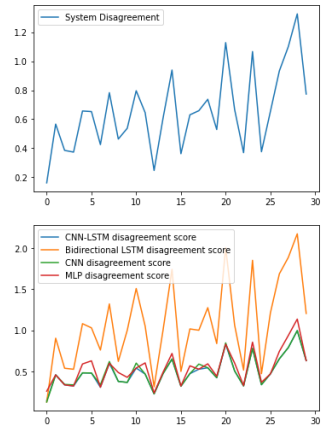
```
WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000020B22183CA0
> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define
your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes tha
t can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/tutorials/customization/performance#python_or_te
nsor_args (https://www.tensorflow.org/tutorials/customization/performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/function) for  more details.

WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000020B22183CA0
> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define
your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes tha
t can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/tutorials/customization/performance#python_or_te
nsor_args (https://www.tensorflow.org/tutorials/customization/performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/function) for  more details.

WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000020B0A7E3C10
> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define
your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes tha
t can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/tutorials/customization/performance#python_or_te
nsor_args (https://www.tensorflow.org/tutorials/customization/performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/function) for  more details.

WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000020B0A7E3C10
> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define
your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes tha
t can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/tutorials/customization/performance#python_or_te
nsor_args (https://www.tensorflow.org/tutorials/customization/performance#python_or_tensor_args) and https://www.tensorflow.org/api_doc
s/python/tf/function (https://www.tensorflow.org/api_docs/python/tf/function) for  more details.
```

System Disagreement

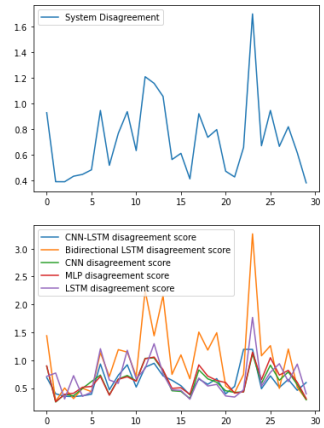
In [13]:
```
system_disagreement(final_df1)
```



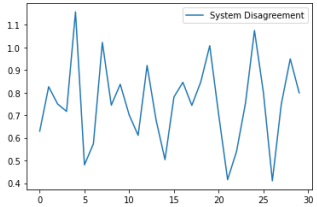
In [14]:
```
system_disagreement(final_df2)
```

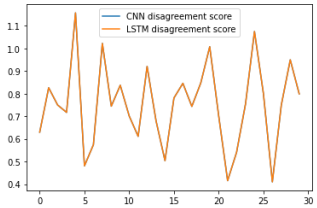


In [15]:
```
system_disagreement(final_df3)
```

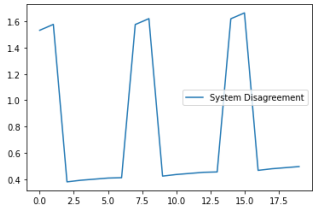


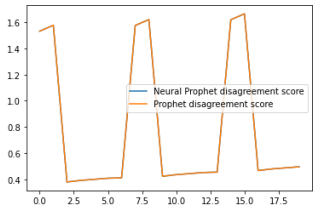
In [16]:
```
system_disagreement(final_df4)
```



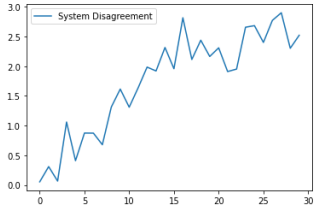


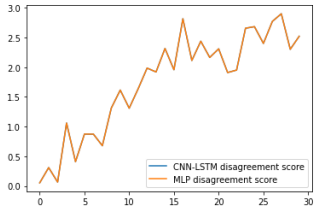
In [17]:
```
system_disagreement(final_df5)
```





In [18]:
```
system_disagreement(final_df6)
```





System consensus

In [19]:
```
algos1 = consensus(final_df1, real)
```

In [20]:
```
algos2 = consensus(final_df2, real)
```

In [21]:
```
algos3 = consensus(final_df3, real)
```

In [22]:
```
algos4 = consensus(final_df4, real)
```

In [23]:
```
algos5 = consensus(final_df5, real)
```

In [24]:
```
algos6 = consensus(final_df6, real)
```

In [25]:
```
ui1 = combined_frame(final_df1, algos1, real)
```

In [26]:
```
ui2 = combined_frame(final_df2, algos2, real)
```

In [27]:
```
ui3 = combined_frame(final_df3, algos3, real)
```

In [28]:
```
ui4 = combined_frame(final_df4, algos4, real)
```

In [30]:
```
#ui5 = combined_frame(final_df5, algos5, real)
```

In [31]:
```
ui6 = combined_frame(final_df6, algos6, real)
```

In [36]:
```
yu = all_stats_frame(ui3, final_df3)
```

In [37]:
```
yu
```

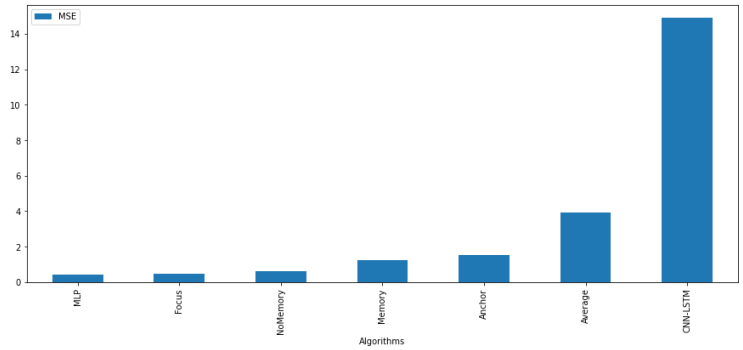
Out[37]:

Focus	Anchor	Real Value	CNN-LSTM	Bidirectional LSTM	CNN	MLP	LSTM	System Disagreement	CNN-LSTM disagreement score	Bidirectional LSTM disagreement score	CNN disagreement score	MLP disagreement score	disagr
8.688638	8.778287	8.879643	8.567571	7.249403	9.577195	9.586743	8.462280	0.926335	0.690451	1.439235	0.892376	0.898105	0.
9.184317	9.670227	9.238214	9.184317	9.409862	9.469923	9.496254	10.353127	0.387842	0.398380	0.263053	0.251041	0.256307	0.
8.810283	8.989960	9.217500	8.810283	8.562825	9.448236	9.454889	9.051553	0.387533	0.354257	0.502732	0.385340	0.389332	0.
8.450292	9.194567	9.272143	9.078054	9.251715	9.507317	9.581393	8.450292	0.430634	0.346805	0.312073	0.363193	0.407639	0.
8.615701	9.159352	9.188929	8.841496	8.615701	9.594939	9.629401	8.888203	0.444935	0.362770	0.498247	0.494776	0.515453	0.
8.710784	9.177057	8.992857	8.904978	8.697761	9.752643	9.604809	8.710784	0.480606	0.389781	0.436434	0.618448	0.529748	0.
8.632515	9.463546	9.026071	8.632515	8.292269	9.846837	9.735095	10.634358	0.943760	0.931798	1.135946	0.733631	0.711282	1.
9.178944	9.432972	8.980357	9.178944	8.780439	9.691327	9.664295	10.135436	0.515580	0.470546	0.709649	0.378882	0.373476	0.
8.700507	9.047548	8.925000	8.700507	7.934389	9.787399	9.777500	9.422422	0.765282	0.730384	1.190055	0.662956	0.657016	0.
8.637691	9.473903	8.770714	8.637691	8.258598	9.829446	9.713176	10.582926	0.934465	0.918313	1.145769	0.726470	0.703216	1.
9.244512	9.079181	8.996071	9.244512	8.431995	9.777229	9.764645	8.513307	0.630689	0.519314	0.714342	0.630892	0.623341	0.
8.874205	8.805569	9.080000	8.874205	6.583148	9.865089	9.855217	8.981299	1.207183	0.874010	2.248643	1.033298	1.027374	0.
7.593911	8.911449	9.250714	9.345822	7.356065	9.825441	9.851751	7.593911	1.155665	0.945443	1.438533	1.041367	1.057153	1.
9.416009	9.117242	9.283571	9.416009	6.912010	9.863962	9.895929	9.248715	1.053294	0.719833	2.155315	0.809424	0.828604	0.
9.923914	9.541223	9.260000	8.903001	8.726488	9.841742	9.968306	9.923914	0.560728	0.640294	0.746202	0.452546	0.495616	0.
9.358685	9.498684	9.431429	9.358685	8.437756	9.867372	10.039862	9.961917	0.609191	0.542805	1.095362	0.441068	0.506744	0.
9.395229	9.582307	9.320000	9.395229	8.914496	9.847772	9.968861	9.783707	0.409803	0.379077	0.667517	0.314194	0.386848	0.
9.199203	9.245677	9.218214	9.199203	7.742396	10.013658	10.164110	9.116761	0.918452	0.663722	1.504829	0.826613	0.916884	0.
9.145348	9.317163	9.187500	9.145348	8.121027	9.936973	10.019570	9.295599	0.734194	0.568084	1.182676	0.666308	0.715867	0.
9.283673	9.385106	9.351786	9.283673	7.911933	9.988925	10.042387	9.798404	0.794586	0.670087	1.493131	0.605245	0.637323	0.
9.207683	9.558914	9.354643	9.207683	8.972844	9.867282	10.110354	9.400574	0.469539	0.398000	0.538903	0.452764	0.598607	0.
9.738857	9.653956	9.392143	9.157172	9.346651	10.112705	10.091066	9.738857	0.424877	0.532118	0.418431	0.423415	0.410431	0.
10.631354	9.862649	9.346429	8.691995	10.631354	10.096802	10.116220	9.898582	0.655417	1.194996	0.744364	0.431400	0.435283	0.
11.254105	9.497908	9.288929	9.673720	6.219878	10.070283	10.214292	11.254105	1.697444	1.194272	3.266577	1.114960	1.143761	1.
9.576543	9.524735	9.348214	9.576543	8.436949	10.056212	10.170460	9.344705	0.668565	0.489004	1.080025	0.584938	0.653486	0.
8.822781	9.323080	9.264643	9.144053	8.026108	10.110650	10.332362	8.822781	0.944061	0.718825	1.261083	0.912144	1.045172	0.
9.336094	9.561226	8.935357	9.336094	9.414968	10.140548	10.296932	8.624962	0.663743	0.511060	0.495285	0.640401	0.734231	0.
9.382923	9.410616	8.992500	9.185884	8.278503	10.251716	10.297830	9.382923	0.816718	0.656440	1.200868	0.790790	0.818459	0.
9.549059	9.462631	8.896429	9.549059	9.287271	10.056882	10.127342	8.592072	0.614424	0.460976	0.513334	0.562541	0.604817	0.
9.738330	9.908859	8.844286	9.399212	10.283153	10.263792	10.310456	9.738330	0.378770	0.599777	0.295086	0.291213	0.311468	0.

In [38]:
```
mse_score(ui6)
```

Out[38]:

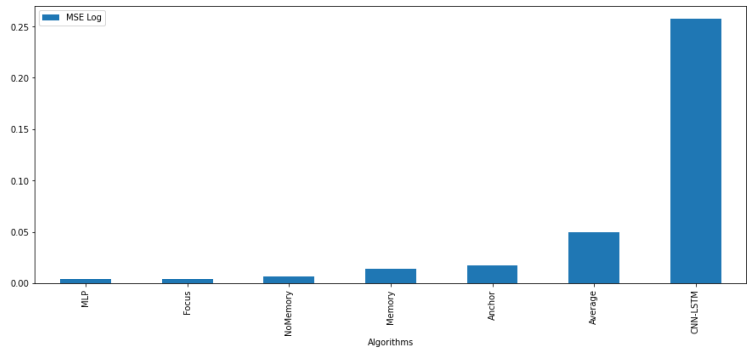
	Algorithms	MSE
0	Average	3.909294
1	NoMemory	0.622769
2	Memory	1.254143
3	Focus	0.455512
4	Anchor	1.531899
5	CNN-LSTM	14.893194
6	MLP	0.425638



In [39]:
```
mse_log_score(ui6)
```

Out[39]:

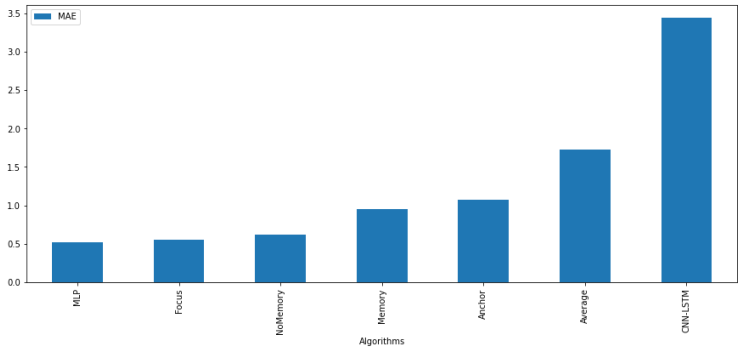
	Algorithms	MSE Log
0	Average	0.049263
1	NoMemory	0.006790
2	Memory	0.014065
3	Focus	0.004447
4	Anchor	0.017361
5	CNN-LSTM	0.257301
6	MLP	0.004128



In [40]:
```
mae_score(ui6)
```

Out[40]:

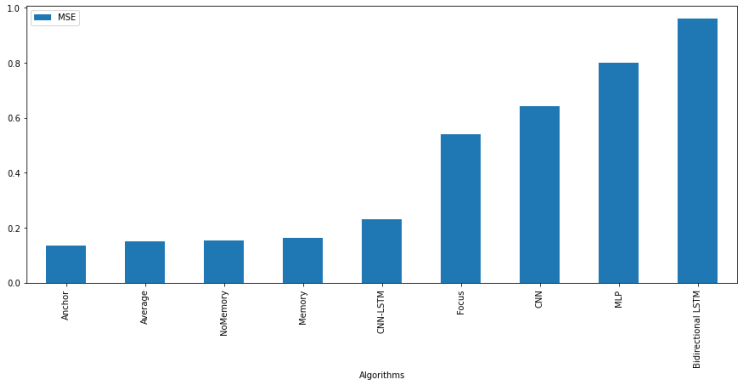
	Algorithms	MAE
0	Average	1.720874
1	NoMemory	0.615720
2	Memory	0.947032
3	Focus	0.547369
4	Anchor	1.067112
5	CNN-LSTM	3.436043
6	MLP	0.519092



In [41]:
```
mse_score(ui2)
```

Out[41]:

	Algorithms	MSE
0	Average	0.151567
1	NoMemory	0.153678
2	Memory	0.164208
3	Focus	0.540058
4	Anchor	0.134033
5	CNN-LSTM	0.231376
6	Bidirectional LSTM	0.961046
7	CNN	0.642561
8	MLP	0.798883



In [ ]:
```
mse_log_score(ui1)
```

In [ ]:
```
mae_score(ui2)
```


In [ ]:
```
mse_score(ui3)
```

In [ ]:
```
mse_log_score(ui3)
```

In [ ]:
```
mae_score(ui3)
```

In [ ]:
```
plot_performance(ui1)
```

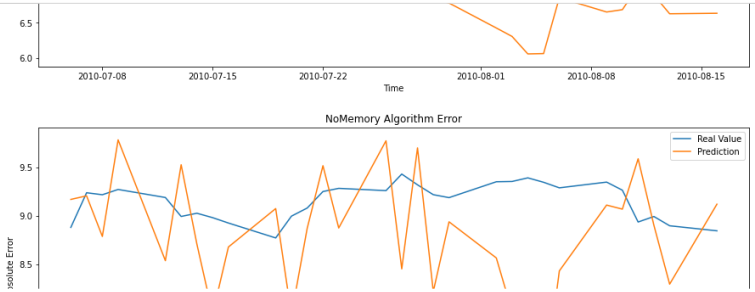
In [ ]:
```
plot_performance(ui2)
```

In [ ]:
```
plot_performance(ui3)
```

In [ ]:
```
plot_performance(ui4)
```

In [ ]:
```
plot_performance(ui5)
```

In [42]:
```
plot_performance(ui6)
```



In [43]:
```
training = DataLoader('aapl', '2000-01-01', '2010-05-01')
```

In [44]:
```
training = training.get_close()
```

In [46]:
```
motest = HybridUnivariatePredictor(2, 24, 30, training)
```

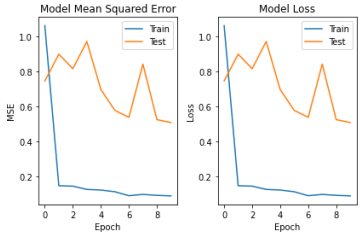
In [47]:
```
motest.create_cnnlstm()
```

In [49]:
```
motest.fit_model(10)
```

```
Epoch 1/10
204/204 [==============================] - 1s 7ms/step - loss: 1.0612 - mean_squared_error: 1.0612 - val_loss: 0.7455 - val_mean_square
d_error: 0.7455
Epoch 2/10
204/204 [==============================] - 1s 5ms/step - loss: 0.1476 - mean_squared_error: 0.1476 - val_loss: 0.8986 - val_mean_square
d_error: 0.8986
Epoch 3/10
204/204 [==============================] - 1s 4ms/step - loss: 0.1453 - mean_squared_error: 0.1453 - val_loss: 0.8154 - val_mean_square
d_error: 0.8154
Epoch 4/10
204/204 [==============================] - 1s 4ms/step - loss: 0.1267 - mean_squared_error: 0.1267 - val_loss: 0.9708 - val_mean_square
d_error: 0.9708
Epoch 5/10
204/204 [==============================] - 1s 4ms/step - loss: 0.1232 - mean_squared_error: 0.1232 - val_loss: 0.6966 - val_mean_square
d_error: 0.6966
Epoch 6/10
204/204 [==============================] - 1s 4ms/step - loss: 0.1133 - mean_squared_error: 0.1133 - val_loss: 0.5776 - val_mean_square
d_error: 0.5776
Epoch 7/10
204/204 [==============================] - 1s 5ms/step - loss: 0.0908 - mean_squared_error: 0.0908 - val_loss: 0.5381 - val_mean_square
d_error: 0.5381
Epoch 8/10
204/204 [==============================] - 1s 4ms/step - loss: 0.0983 - mean_squared_error: 0.0983 - val_loss: 0.8414 - val_mean_square
d_error: 0.8414
Epoch 9/10
204/204 [==============================] - 1s 5ms/step - loss: 0.0928 - mean_squared_error: 0.0928 - val_loss: 0.5250 - val_mean_square
d_error: 0.5250
Epoch 10/10
204/204 [==============================] - 1s 5ms/step - loss: 0.0898 - mean_squared_error: 0.0898 - val_loss: 0.5080 - val_mean_square
d_error: 0.5080
```

Out[49]:
```
<tensorflow.python.keras.callbacks.History at 0x20b11c376a0>
```

In [50]:
```
motest.show_performance()
```



In [ ]: