Experiment Results

Results for Houston

In [8]:
```
import warnings
warnings.filterwarnings('ignore')

%run helper_functions.py
%matplotlib inline
```

Lets review the top 10 results, ordered by Mean Square Error (ascending)

In [2]:
```
city='Houston'
results_top_10 = get_results(city=city, top_hm_results=10)
results_top_10
```

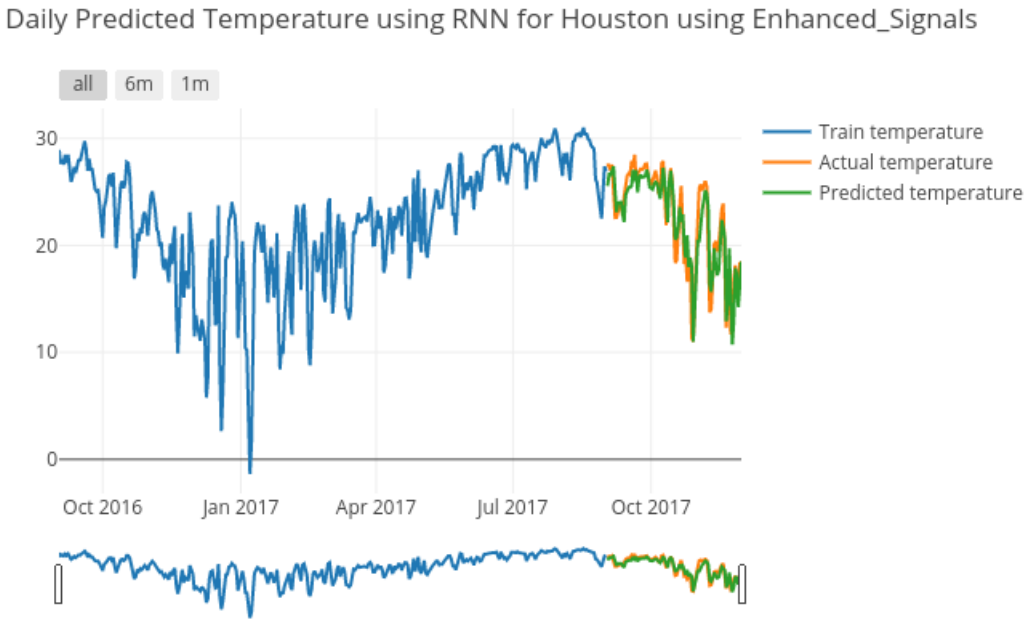
Out[2]:

	RUN_ID	DATETIME	MODEL_NAME	CITY	FEATURE_TYPE	HOST_MACHINE	MODEL_PARAMETERS	MODEL_RESULTS	MEAN_SQUARED_ERROR
1	53	2018-08-14 20:40:23.868627	RNN	Houston	Enhanced_Signals	DESKTOP-KN40C32	{'epochs': 50, 'Info': {'feature_set_type': 'E...	{'features': ['temperature_lag1', 'temperature...	4.217506
2	47	2018-08-14 20:37:47.827488	RNN	Houston	Inc_Signals	DESKTOP-KN40C32	{'epochs': 50, 'Info': {'feature_set_type': 'I...	{'features': ['temperature_lag1', 'temperature...	4.588875
3	41	2018-08-14 20:35:12.061515	RNN	Houston	Basic	DESKTOP-KN40C32	{'epochs': 50, 'Info': {'feature_set_type': 'B...	{'features': ['temperature_lag1', 'temperature...	5.581542
4	29	2018-08-14 19:55:55.798404	RANDOM FOREST	Houston	Inc_Signals	DESKTOP-KN40C32	{'max_depth': 8, 'n_estimators': 50, 'Info': {...	{'features': ['temperature_lag1', 'temperature...	6.041626
5	35	2018-08-14 19:56:44.111200	RANDOM FOREST	Houston	Enhanced_Signals	DESKTOP-KN40C32	{'max_depth': 8, 'n_estimators': 50, 'Info': {...	{'features': ['temperature_lag1', 'temperature...	6.041626
6	23	2018-08-14 19:55:04.570078	RANDOM FOREST	Houston	Basic	DESKTOP-KN40C32	{'max_depth': 8, 'n_estimators': 50, 'Info': {...	{'features': ['temperature_lag1', 'temperature...	6.637421
7	5	2018-08-14 19:42:41.506483	DECISION TREE	Houston	Basic	DESKTOP-KN40C32	{'max_depth': 8, 'Info': {'feature_set_type': ...	{'features': ['temperature_lag1', 'temperature...	7.705077
8	18	2018-08-14 19:48:11.736850	DECISION TREE	Houston	Enhanced_Signals	DESKTOP-KN40C32	{'max_depth': 8, 'Info': {'feature_set_type': ...	{'features': ['temperature_lag1', 'temperature...	10.205057
9	11	2018-08-14 19:43:39.374706	DECISION TREE	Houston	Inc_Signals	DESKTOP-KN40C32	{'max_depth': 8, 'Info': {'feature_set_type': ...	{'features': ['temperature_lag1', 'temperature...	10.519872

RNN has done much better than Random forest so in this instance it is the winner, lets take a look at that chart.

In [3]:
```
display_results(results_top_10.head(1), chart_type='predict')
```

Experiment #53 - run by DESKTOP-KN40C32 on 2018-08-14 20:40:23.868627



We can see the forecast follows the trend and it does a better job but like the prior example of Miami it has some difficulty following the peaks and troughs of the actual temperature pattern experiences.

Lets examine the top 10 features across the model runs for this location in a pivot table form.

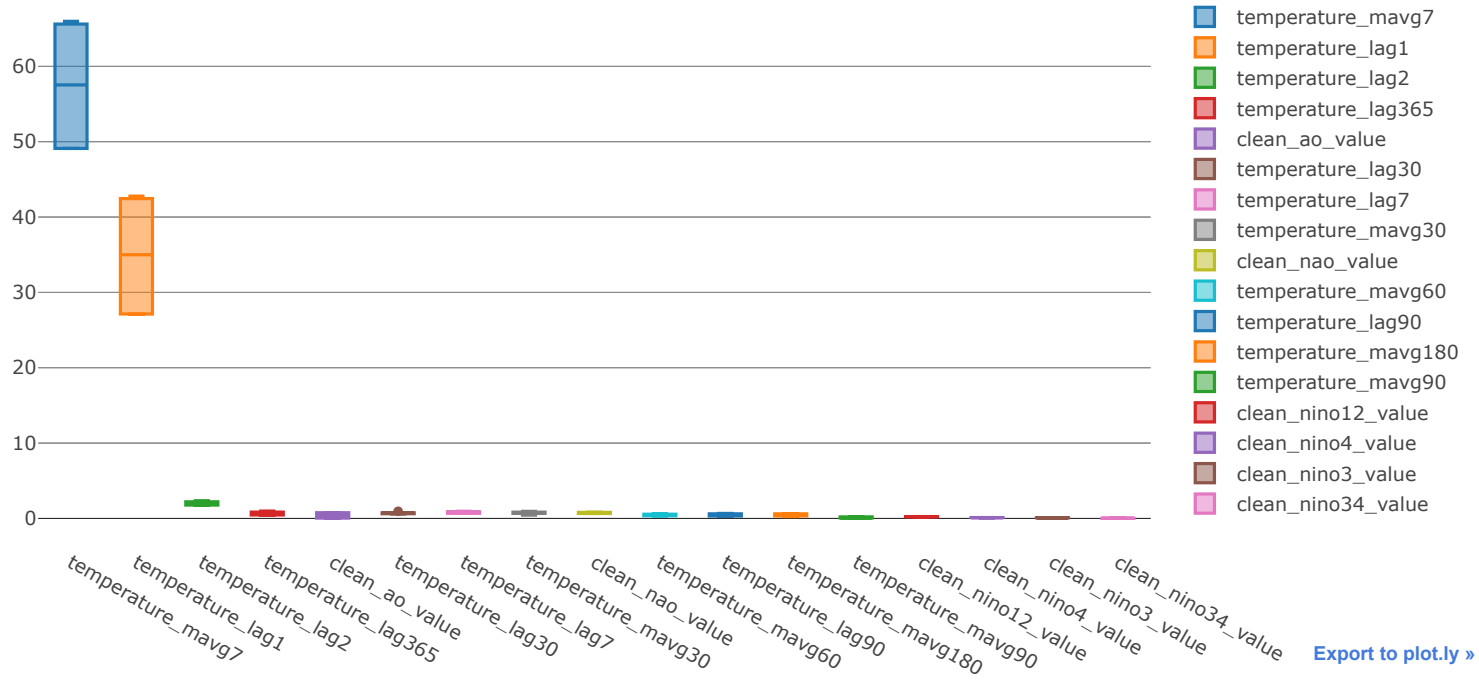
In [4]:
```
features_df = get_feature_importances(results_top_10)
features_df.pivot_table(index=['FEATURE'], columns=['MODEL_NAME','FEATURE_TYPE'], values="IMPORTANCE")
```

Out[4]:

MODEL_NAME	DECISION TREE			RANDOM FOREST		
FEATURE_TYPE	Basic	Enhanced_Signals	Inc_Signals	Basic	Enhanced_Signals	Inc_Signals
FEATURE						
clean_ao_value	NaN	0.043560	0.041918	NaN	0.756749	0.756749
clean_nao_value	NaN	0.816215	0.794028	NaN	0.656157	0.656157
clean_nino12_value	NaN	0.235922	0.244526	NaN	NaN	NaN
clean_nino34_value	NaN	0.003483	0.018493	NaN	NaN	NaN
clean_nino3_value	NaN	0.087301	0.090106	NaN	NaN	NaN
clean_nino4_value	NaN	0.109168	0.112347	NaN	NaN	NaN
temperature_lag1	27.548477	27.136183	27.082286	42.775543	42.447002	42.447002
temperature_lag2	2.202069	1.773779	1.774231	2.347156	2.064882	2.064882
temperature_lag30	0.609669	0.646106	0.621661	0.944930	0.742389	0.742389
temperature_lag365	0.753962	0.440651	0.455842	0.975389	0.840931	0.840931
temperature_lag7	0.954081	0.859909	0.875849	0.863763	0.675088	0.675088
temperature_lag90	0.662133	0.330657	0.305711	0.556063	NaN	NaN
temperature_mavg180	0.241951	0.605832	0.616706	0.407410	NaN	NaN
temperature_mavg30	0.437132	0.722921	0.940497	0.831117	0.664705	0.664705
temperature_mavg60	0.558999	0.354695	0.328816	0.598374	0.469204	0.469204
temperature_mavg7	65.979684	65.591639	65.612385	49.480824	49.105328	49.105328
temperature_mavg90	0.051843	0.241981	0.084600	NaN	NaN	NaN

Lets look a boxplot of this data to see how the distributions look across our experiments for this location.

In [5]:
```
traces = create_boxplot_traces_for_features(features_df)
iplot(traces)
```



Lets review the means for the features

In [6]:
```
features_df = get_feature_importances(results_top_10)
pivot_df = features_df.pivot_table(index=['FEATURE'], columns=[], values="IMPORTANCE", aggfunc= [np.mean])
pivot_df
```

Out[6]:

FEATURE	mean IMPORTANCE
clean_ao_value	0.399744
clean_nao_value	0.730639
clean_nino12_value	0.240224
clean_nino34_value	0.010988
clean_nino3_value	0.088704
clean_nino4_value	0.110757
temperature_lag1	34.906082
temperature_lag2	2.037833
temperature_lag30	0.717857
temperature_lag365	0.717951
temperature_lag7	0.817296
temperature_lag90	0.463641
temperature_mavg180	0.467975
temperature_mavg30	0.710179
temperature_mavg60	0.463216
temperature_mavg7	57.479198
temperature_mavg90	0.126141

Now lets review the top 5 only

In [7]:
```
pivot_df.columns = pivot_df.columns.get_level_values(0)
pivot_df.sort_values(['mean'], ascending=False).head(5)
```

Out[7]:

FEATURE	mean
temperature_mavg7	57.479198
temperature_lag1	34.906082
temperature_lag2	2.037833
temperature_lag7	0.817296
clean_nao_value	0.730639

This is different than the prior cases as the moving average takes a clear lead in the predictions vs. the prior lags of temperature.

In [ ]: