Experiment Results

Results for Boston

In [2]:
```
import warnings
warnings.filterwarnings('ignore')

%run helper_functions.py
%matplotlib inline
```

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

In [3]:
```
city='Boston'
results_top_10 = get_results(city=city, top_hm_results=10)
results_top_10
```

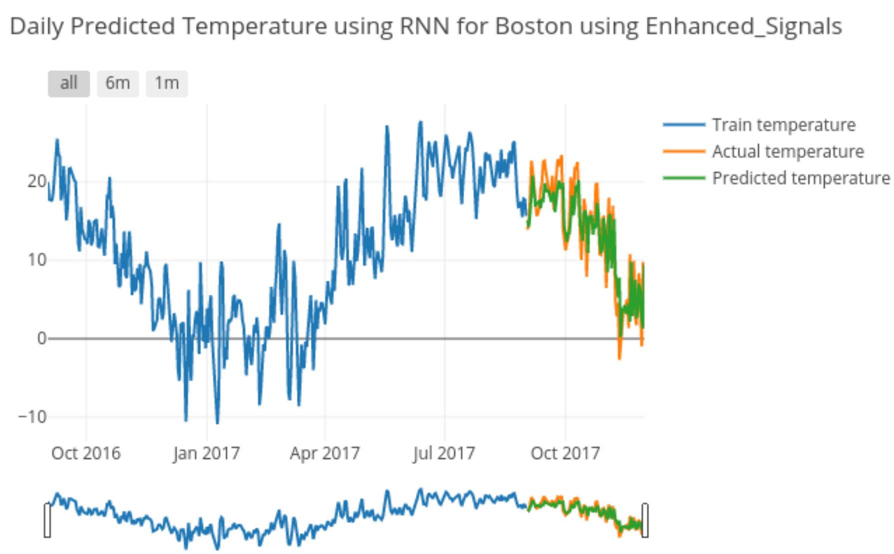
Out[3]:

	RUN_ID	DATETIME	MODEL_NAME	CITY	FEATURE_TYPE	HOST_MACHINE	MODEL_PARAMETERS	MODEL_RESULTS	MEAN_SQUARED_ERROR
1	51	2018-08-14 20:39:52.178626	RNN	Boston	Enhanced_Signals	DESKTOP-KN40C32	{'epochs': 50, 'Info': {'feature_set_type': 'E...	{'features': ['temperature_lag1', 'temperature...	11.645510
2	45	2018-08-14 20:37:08.003250	RNN	Boston	Inc_Signals	DESKTOP-KN40C32	{'epochs': 50, 'Info': {'feature_set_type': 'I...	{'features': ['temperature_lag1', 'temperature...	11.682041
3	39	2018-08-14 20:34:39.550435	RNN	Boston	Basic	DESKTOP-KN40C32	{'epochs': 50, 'Info': {'feature_set_type': 'B...	{'features': ['temperature_lag1', 'temperature...	11.869203
4	21	2018-08-14 19:54:48.599391	RANDOM FOREST	Boston	Basic	DESKTOP-KN40C32	{'max_depth': 8, 'n_estimators': 50, 'Info': {...	{'features': ['temperature_lag1', 'temperature...	11.877780
5	27	2018-08-14 19:55:39.652542	RANDOM FOREST	Boston	Inc_Signals	DESKTOP-KN40C32	{'max_depth': 8, 'n_estimators': 50, 'Info': {...	{'features': ['temperature_lag1', 'temperature...	12.735153
6	33	2018-08-14 19:56:30.584670	RANDOM FOREST	Boston	Enhanced_Signals	DESKTOP-KN40C32	{'max_depth': 8, 'n_estimators': 50, 'Info': {...	{'features': ['temperature_lag1', 'temperature...	12.735153
7	3	2018-08-14 19:42:23.328525	DECISION TREE	Boston	Basic	DESKTOP-KN40C32	{'max_depth': 8, 'Info': {'feature_set_type': ...	{'features': ['temperature_lag1', 'temperature...	16.504857
8	9	2018-08-14 19:43:20.383683	DECISION TREE	Boston	Inc_Signals	DESKTOP-KN40C32	{'max_depth': 8, 'Info': {'feature_set_type': ...	{'features': ['temperature_lag1', 'temperature...	22.967065
9	15	2018-08-14 19:44:14.731752	DECISION TREE	Boston	Enhanced_Signals	DESKTOP-KN40C32	{'max_depth': 8, 'Info': {'feature_set_type': ...	{'features': ['temperature_lag1', 'temperature...	25.109055

RNN has done better than Random forest but the results are close so there is no clear winner, lets take a look at that chart.

In [4]:
```
display_results(results_top_10.head(1), chart_type='predict')
```

Experiment #51 - run by DESKTOP-KN40C32 on 2018-08-14 20:39:52.178626



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.

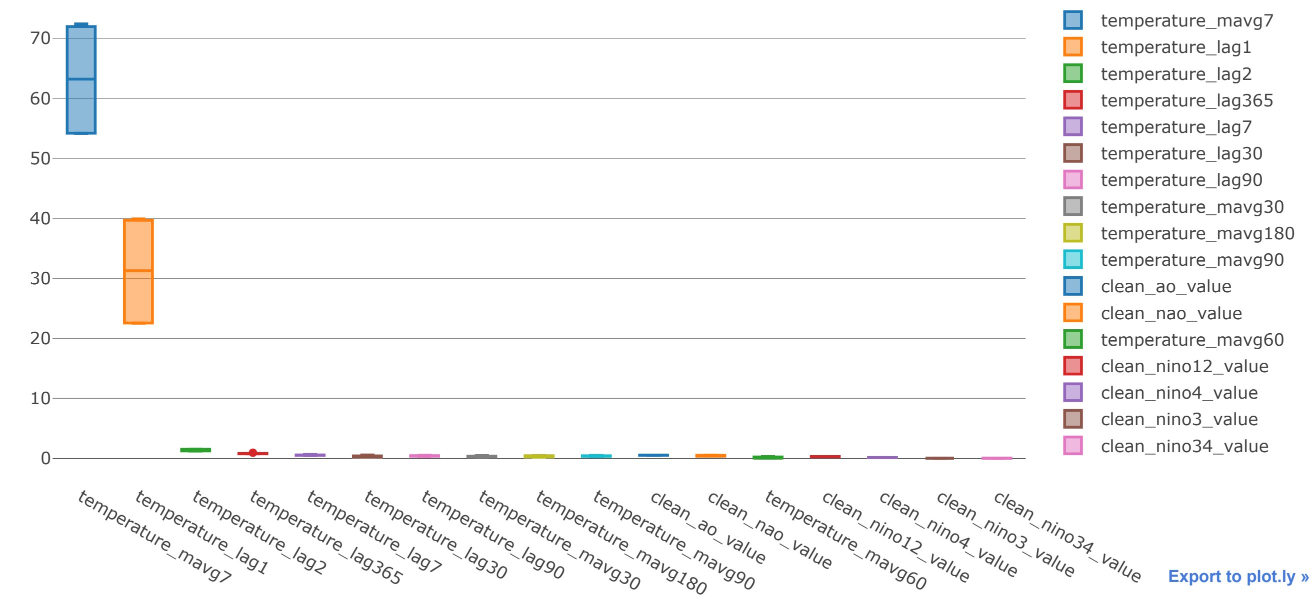
In [5]:
```python
features_df = get_feature_importances(results_top_10)
features_df.pivot_table(index=['FEATURE'], columns=['MODEL_NAME','FEATURE_TYPE'], values="IMPORTANCE")
```

Out[5]:

MODEL_NAME	DECISION TREE			RANDOM FOREST		
FEATURE_TYPE	Basic	Enhanced_Signals	Inc_Signals	Basic	Enhanced_Signals	Inc_Signals
FEATURE						
clean_ao_value	NaN	0.494070	0.547344	NaN	0.519692	0.519692
clean_nao_value	NaN	0.365090	0.362958	NaN	0.509393	0.509393
clean_nino12_value	NaN	0.287805	0.302207	NaN	NaN	NaN
clean_nino34_value	NaN	0.021185	0.006980	NaN	NaN	NaN
clean_nino3_value	NaN	0.012564	0.012307	NaN	NaN	NaN
clean_nino4_value	NaN	0.146795	0.132922	NaN	NaN	NaN
temperature_lag1	22.853599	22.547786	22.555122	39.903638	39.676008	39.676008
temperature_lag2	1.512019	1.278805	1.293871	1.508032	1.242142	1.242142
temperature_lag30	0.261568	0.202941	0.197705	0.572096	0.415173	0.415173
temperature_lag365	0.821443	0.820219	0.772308	0.938950	0.762632	0.762632
temperature_lag7	0.475197	0.497790	0.543488	0.657224	0.573611	0.573611
temperature_lag90	0.296958	0.351368	0.361960	0.514342	0.442467	0.442467
temperature_mavg180	0.472520	0.218633	0.242651	0.417609	NaN	NaN
temperature_mavg30	0.255383	0.256516	0.265126	0.472850	0.361452	0.361452
temperature_mavg60	0.291385	0.051112	0.079570	NaN	NaN	NaN
temperature_mavg7	72.406163	71.954276	71.946289	54.442588	54.181522	54.181522
temperature_mavg90	0.353767	0.493047	0.377192	0.295478	NaN	NaN

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

In [6]:
```
traces = create_boxplot_traces_for_features(features_df)
iplot(traces)
```



Export to plot.ly »

Lets review the means for the features

In [7]:
```
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[7]:

	mean
	IMPORTANCE
FEATURE	
clean_ao_value	0.520199
clean_nao_value	0.436709
clean_nino12_value	0.295006
clean_nino34_value	0.014083
clean_nino3_value	0.012436
clean_nino4_value	0.139858
temperature_lag1	31.202027
temperature_lag2	1.346168
temperature_lag30	0.344109
temperature_lag365	0.813031
temperature_lag7	0.553487
temperature_lag90	0.401594
temperature_mavg180	0.337853
temperature_mavg30	0.328797
temperature_mavg60	0.140689
temperature_mavg7	63.185393
temperature_mavg90	0.379871

Now lets review the top 5 only

In [8]:
```
pivot_df.columns = pivot_df.columns.get_level_values(0)
pivot_df.sort_values(['mean'], ascending=False).head(5)
```

Out[8]:

	mean
FEATURE	
temperature_mavg7	63.185393
temperature_lag1	31.202027
temperature_lag2	1.346168
temperature_lag365	0.813031
temperature_lag7	0.553487

This is different than the prior cases as we see the moving average taking a lead in the predictions, vs. the temperature of the prior day.