Time Series to observe DAILY temperature variations

Daily temperature prediction using RNN

Moving from predictive Machine Learning classifier to Unpredicitve Neural Nets, we use Sequential **Recurrent Neural Net (RNN)** in the notebook below.

You need to ensure that you have the right environment installed on top of your python3 to run Keras and Tensorflow. Two libraries needed to successfully run (RNN).

Just like in the other models, we begin by loading all necessary libraries and paths to read the "pickles" as well as store image for the graph towards the end of our code. The pickles are read and the data is fed into an RNN model. Finally, we have two graphs showing the DT results vs. the fitted model as well as predicted results vs. actuals and test data

Here we are importing the train and test Data from pickle files created through the EDA file

In [274]:
```
import warnings
warnings.filterwarnings('ignore')

%run helper_functions.py
%matplotlib inline
```

Create a folder for every run of the RNN to store images

In [275]:
```
city='Miami' # New_York Atlanta Boston Dallas Houston Miami
analysis_type = 'Enhanced_Signals' # Basic, Inc_Signals, Enhanced_Signals
```

In [276]:
```
EXPERIMENT_DIR, EXPERIMENT_ID  = create_results_perrun()
print("Path of the results directory",EXPERIMENT_DIR )
```

```
Path of the results directory ../experiment_results/RUN-54
```

In [277]:
```
#EXPERIMENT_DIR = '../experiment_results/RUN-37'
#EXPERIMENT_ID = 37
```

In [278]:
```
X_train = pd.read_pickle(f'{PICKLE_PATH}/X_train_{city}_{analysis_type}.pkl')
Y_train = pd.read_pickle(f'{PICKLE_PATH}/Y_train_{city}_{analysis_type}.pkl')

X_test  = pd.read_pickle(f'{PICKLE_PATH}/X_test_{city}_{analysis_type}.pkl')
Y_test  = pd.read_pickle(f'{PICKLE_PATH}/Y_test_{city}_{analysis_type}.pkl')

print("Shape of Training Dataset " , X_train.shape)
print("Shape of Testing Dataset " , X_test.shape)
```

```
Shape of Training Dataset  (1399, 17)
Shape of Testing Dataset  (90, 17)
```

In [279]:
```
# Function to fit a sequential rnn with loss estimate being mean squared error
def train_model(X_train, y_train, X_test, y_test, epochs):
    model = Sequential(
        [
            Dense(10, activation="relu", input_shape=(X_train.shape[1],)),
            Dense(10, activation="relu"),
            Dense(10, activation="relu"),
            Dense(1, activation="linear")
        ]
    )
    model.compile(optimizer=Adam(lr=0.001), loss="mean_squared_error")

    history = model.fit(X_train, y_train, epochs=epochs, shuffle=False)
    return model, history
```

In [280]:
```
# Function to fit a sequential rnn with epochs = 50
epochs = 50
model_encoded, encoded_hist = train_model(
    X_train,
    Y_train,
    X_test,
    Y_test,
    epochs=epochs
)
```

```
Epoch 1/50
1399/1399 [==============================] - 1s 956us/step - loss: 780.4398
Epoch 2/50
1399/1399 [==============================] - 0s 74us/step - loss: 565.0433
Epoch 3/50
1399/1399 [==============================] - 0s 72us/step - loss: 346.5662
Epoch 4/50
1399/1399 [==============================] - 0s 84us/step - loss: 45.1367
Epoch 5/50
1399/1399 [==============================] - 0s 85us/step - loss: 7.4433
Epoch 6/50
1399/1399 [==============================] - 0s 78us/step - loss: 6.7492
Epoch 7/50
1399/1399 [==============================] - 0s 74us/step - loss: 6.6773
Epoch 8/50
1399/1399 [==============================] - 0s 65us/step - loss: 6.6082
Epoch 9/50
1399/1399 [==============================] - 0s 69us/step - loss: 6.5385
Epoch 10/50
```

In [281]:
```
# Run the model on the training dataset
Y_train_pred = model_encoded.predict(X_train)
# Calculate mean squared error for the predicted values
mse_train = mean_squared_error(Y_train, model_encoded.predict(X_train))
print('Mean Squared Error for the training dataset: %.3f' % mse_train)
```

```
Mean Squared Error for the training dataset: 3.901
```

In [282]:
```
# Run the model on the testing dataset
Y_test_pred = model_encoded.predict(X_test)
# Calculate mean squared error for the test vs predicted values
mse_test = mean_squared_error(Y_test, model_encoded.predict(X_test))
print('Mean Squared Error for the testing dataset: %.3f' % mse_test)
```

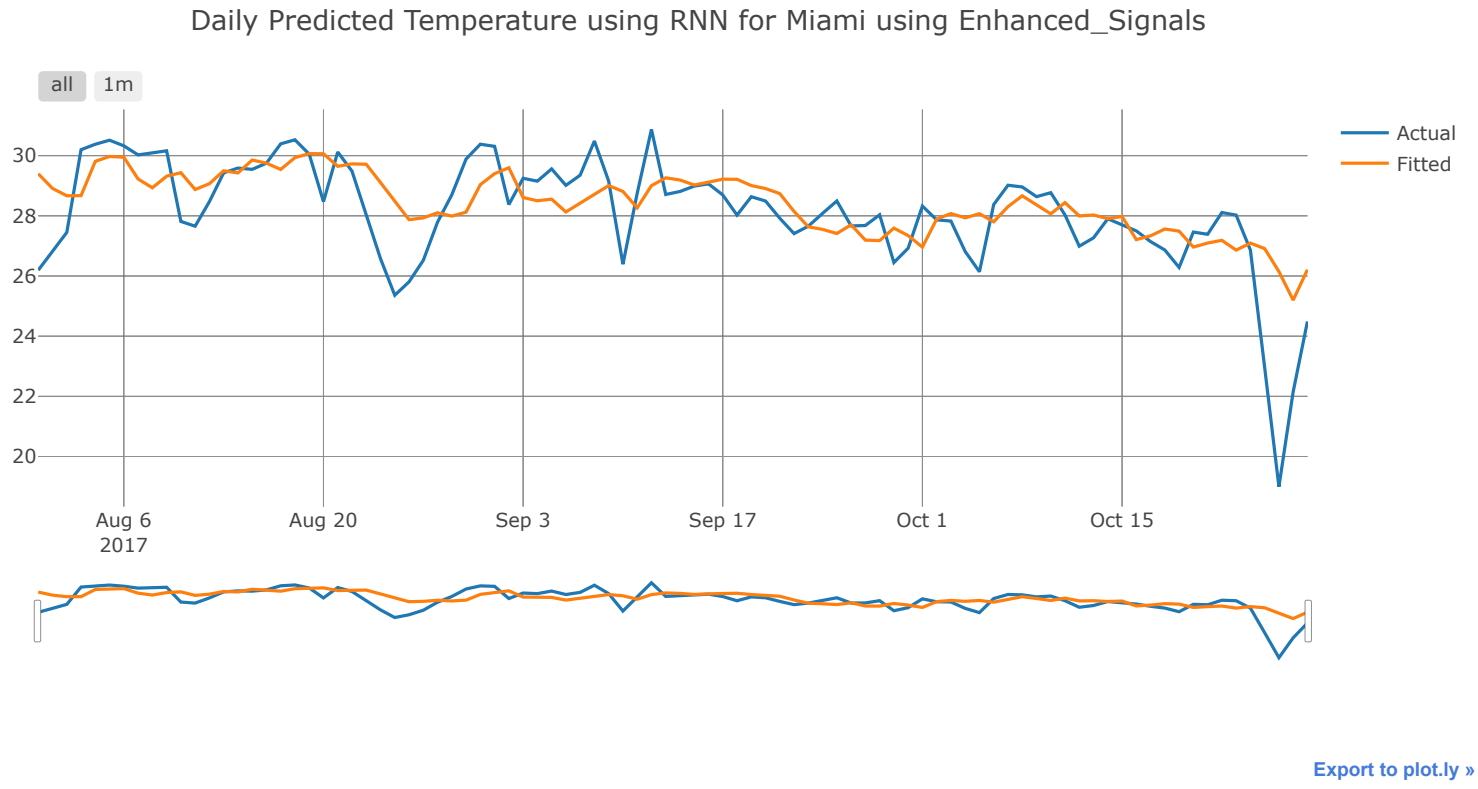
```
Mean Squared Error for the testing dataset: 2.013
```

In [283]:
```
# Creating a dataframe for predicted/fitted values
future_forecast = pd.DataFrame(Y_test_pred,index = Y_test.index,columns=['Fitted'])

# Concatenate the predicted/fitted values with actual values to display graphs
predictions = pd.concat([Y_test,future_forecast],axis=1)
predictions.columns = ["Actual","Fitted"]

# Displaying few of the predicted values
predictions.head(10)
```

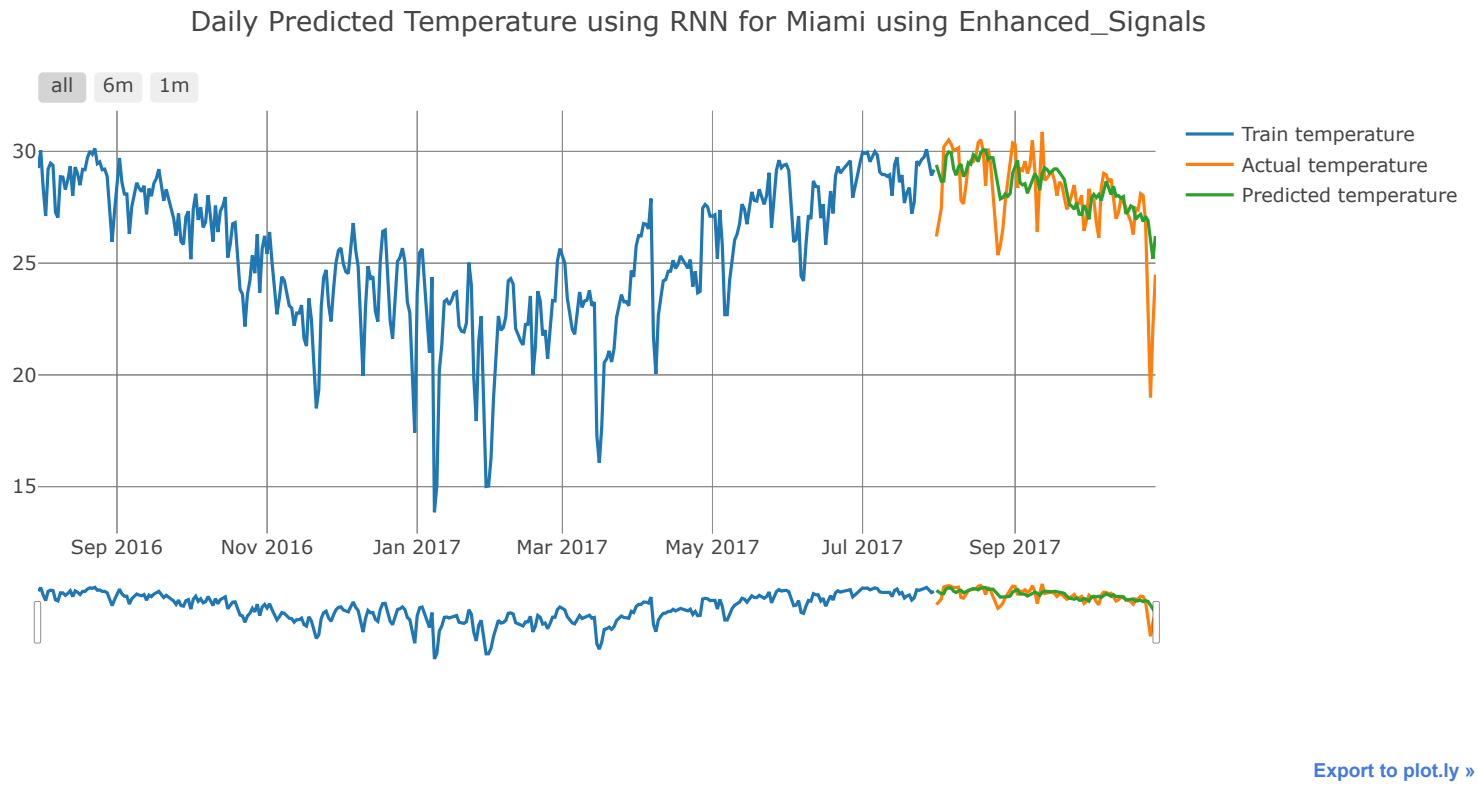
Out[283]:

datetime	Actual	Fitted
2017-07-31	26.192917	29.396700
2017-08-01	26.835000	28.911028
2017-08-02	27.449583	28.666040
2017-08-03	30.198333	28.672266
2017-08-04	30.380000	29.811632
2017-08-05	30.513333	29.973490
2017-08-06	30.326667	29.948656
2017-08-07	30.024583	29.224321
2017-08-08	30.094583	28.931959
2017-08-09	30.160833	29.316683

In [284]:
```
city = city.replace('_',' ')
# Plotting the daily predicted temperature vs Actual Temperature - RNN
fig = charter_helper_fitted(f"Daily Predicted Temperature using RNN for {city} using {analysis_type}", predictions)
iplot(fig)

py.image.save_as(fig, f'{EXPERIMENT_DIR}/Daily_actual_vs_predict.png')
```



In [285]:
```
 # Plotting the training data for past year, Actual/test data and predicted temperature - Decision Tree
fig = charter_helper_prediction(f"Daily Predicted Temperature using RNN for {city} using {analysis_type}",
                      X_train,Y_train,X_test,Y_test,future_forecast)

iplot(fig)

py.image.save_as(fig, f'{EXPERIMENT_DIR}/Daily_predict.png')
```



In [286]:
```
results = update_results_function(EXPERIMENT_ID, 'RNN',city,analysis_type,
                                  {'epochs': epochs,'Info': X_train._metadata},
                                  {'features' : X_train.columns.values.tolist(),
                                   'importances':'NA',
                                   'mse_train' : mse_train},
                                    mse_test)
```

In [287]:
```
results.tail(1)
```

Out[287]:

	RUN_ID	DATETIME	MODEL_NAME	CITY	FEATURE_TYPE	HOST_MACHINE	MODEL_PARAMETERS	MODEL_RESULTS	MEAN_SQUARED_ERROR
53	54	2018-08-14 20:40:54.336628	RNN	Miami	Enhanced_Signals	DESKTOP-KN40C32	{'epochs': 50, 'Info': {'feature_set_type': 'E...	{'features': ['temperature_lag1', 'temperature...	2.01286

In [288]:
```
results = pd.read_pickle('../pickles/results.pkl')
results
```

Out[288]:

	RUN_ID	DATETIME	MODEL_NAME	CITY	FEATURE_TYPE	HOST_MACHINE	MODEL_PARAMETERS	MODEL_RESULTS	MEAN_SQUARED_ERROR
0	1	2018-08-14 19:41:45.275297	DECISION TREE	New York	Basic	DESKTOP-KN40C32	{'max_depth': 8, 'Info': {'feature_set_type': ...	{'features': ['temperature_lag1', 'temperature...	5.799872
1	2	2018-08-14 19:42:13.654060	DECISION TREE	Atlanta	Basic	DESKTOP-KN40C32	{'max_depth': 8, 'Info': {'feature_set_type': ...	{'features': ['temperature_lag1', 'temperature...	8.436149
2	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
3	4	2018-08-14 19:42:30.715058	DECISION TREE	Dallas	Basic	DESKTOP-KN40C32	{'max_depth': 8, 'Info': {'feature_set_type': ...	{'features': ['temperature_lag1', 'temperature...	12.523335
4	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
		2018-08-14	DECISION			DESKTOP-	{'max_depth': 8, 'Info':	{'features':	

In [ ]:
```
```