02-RNN-Exercise

October 19, 2022

---

1 RNN Example for Time Series

TASK: IMPORT THE BASIC LIBRARIES YOU THINK YOU WILL USE

[1]:
```
# IMPORTS HERE
```

1.1 Data

Info about this data set: https://fred.stlouisfed.org/series/TRFVOLUSM227NFWA

Read in the data set "Miles_Traveled.csv" from the Data folder. Figure out how to set the date to a datetime index columns

[2]:
```
# CODE HERE
```

[3]:
```
```

[4]:
```
```

[4]:
```
            TRFVOLUSM227NFWA
DATE
1970-01-01           80173.0
1970-02-01           77442.0
1970-03-01           90223.0
1970-04-01           89956.0
1970-05-01           97972.0
```

Task: Change the column names to Value

[ ]:
```
# CODE HERE
```

[5]:

```

```

TASK: Plot out the time series

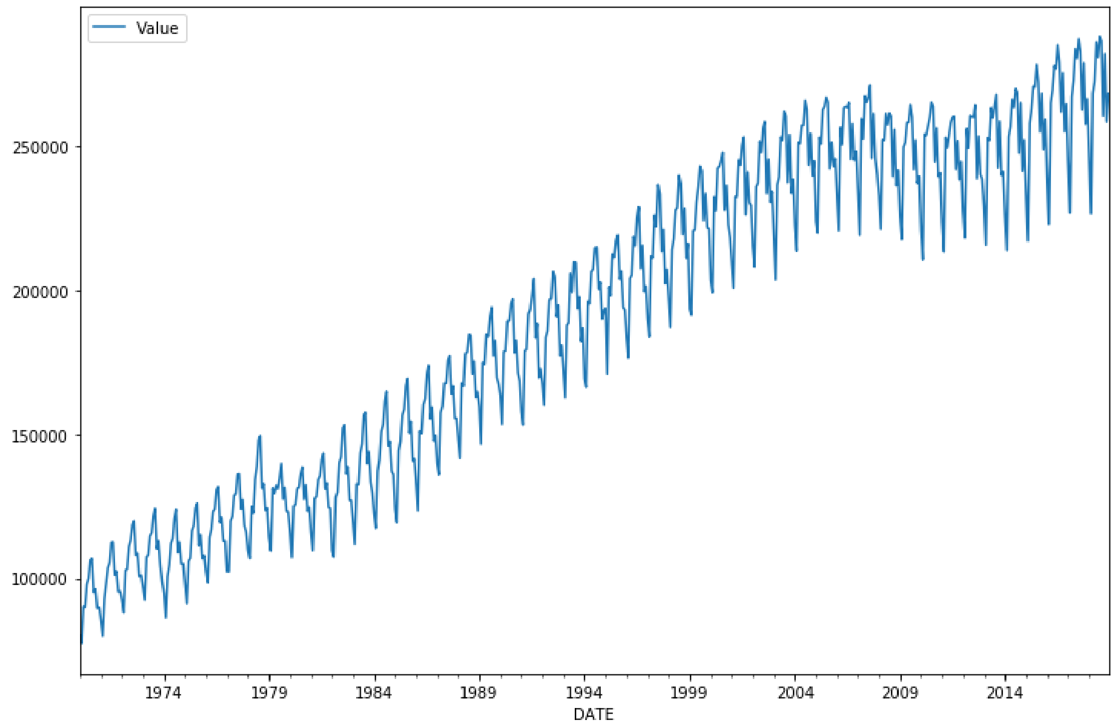
[ ]:

```
# CODE HERE
```

[6]:

```

```

[6]: <matplotlib.axes._subplots.AxesSubplot at 0x12cc3a4d2e8>



TASK: Perform a Seasonal Decomposition on the model and plot out the ETS components

[7]:

```
# CODE HERE
```

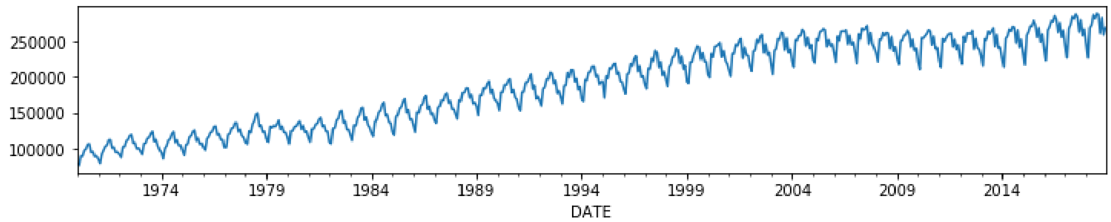
[8]:

```

```

[9]:

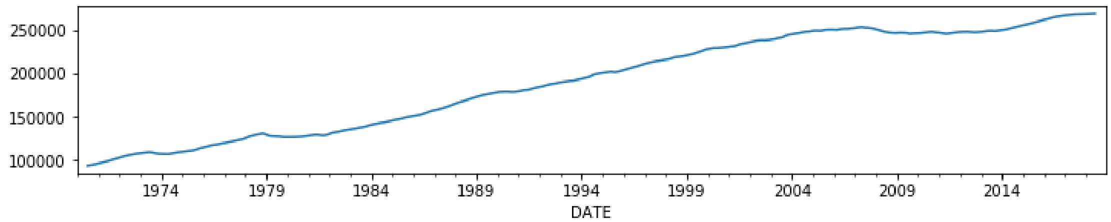
```

```

[9]: <matplotlib.axes._subplots.AxesSubplot at 0x12cc5b5d588>



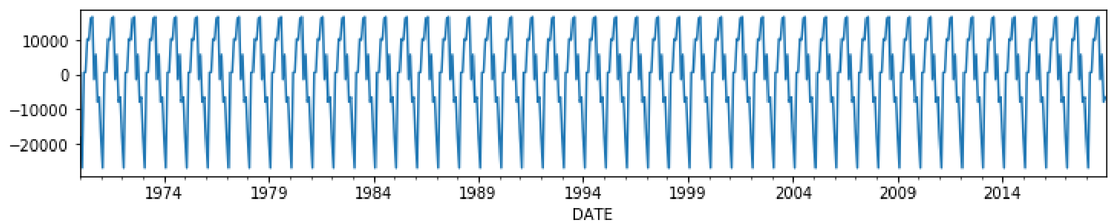
[10]:

[10]: <matplotlib.axes._subplots.AxesSubplot at 0x12cc5b89588>



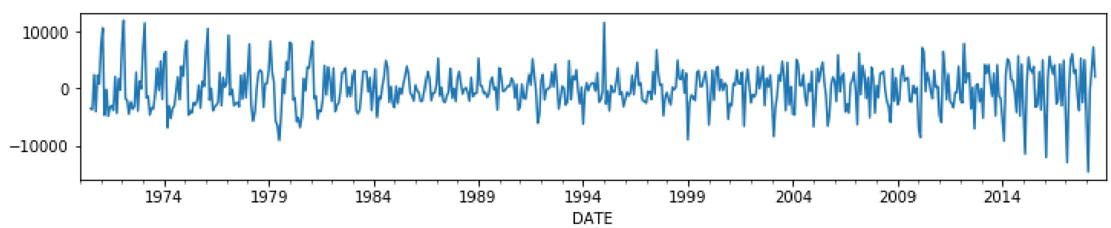
[11]:

[11]: <matplotlib.axes._subplots.AxesSubplot at 0x12cc3fe6e80>



[12]:

[12]: <matplotlib.axes._subplots.AxesSubplot at 0x12cc5def780>



1.2 Train Test Split

TASK: Figure out the length of the data set

[ ]:
```
# CODE HERE
```

[13]:
```
```

[13]: 588

[14]:
```
```

TASK: Split the data into a train/test split where the test set is the last 12 months of data.

[15]:
```
# CODE HERE
```

[16]:
```
len(test)
```

[16]: 12

1.3 Scale Data

TASK: Use a MinMaxScaler to scale the train and test sets into scaled versions.

[17]:
```
# CODE HERE
```

[18]:
```
```

[19]:
```
```

[20]:
```
```

[20]: MinMaxScaler(copy=True, feature_range=(0, 1))

[21]:
```
```

2 Time Series Generator

TASK: Create a TimeSeriesGenerator object based off the scaled_train data. The n_input is up to you, but at a minimum it should be at least 12.

[22]:
```
#CODE HERE
```

[23]:
```
```

```
Using TensorFlow backend.
```

[24]:
```
```

2.0.1 Create the Model

TASK: Create a Keras Sequential Model with as many LSTAM units you want and a final Dense Layer.

[25]:
```
```

[26]:
```
```

[27]:
```
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 150)               91200
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 151
=================================================================
Total params: 91,351
Trainable params: 91,351
Non-trainable params: 0
_________________________________________________________________
```

TASK: Fit the model to the generator (it should be a lot of epochs, but do as many as you have the patience for! :)

[28]:
```
# CODE HERE
```

[46]:
```
```

```
Epoch 1/10
552/552 [==============================] - 17s 30ms/step - loss: 0.0010
Epoch 2/10
552/552 [==============================] - 17s 30ms/step - loss: 0.0011
Epoch 3/10
552/552 [==============================] - 16s 30ms/step - loss: 9.5115e-04
Epoch 4/10
552/552 [==============================] - 17s 30ms/step - loss: 8.8495e-04
Epoch 5/10
```

```
552/552 [==============================] - 17s 30ms/step - loss: 8.4229e-04
Epoch 6/10
552/552 [==============================] - 16s 30ms/step - loss: 0.0012
Epoch 7/10
552/552 [==============================] - 17s 30ms/step - loss: 8.6496e-04
Epoch 8/10
552/552 [==============================] - 17s 30ms/step - loss: 7.5506e-04
Epoch 9/10
552/552 [==============================] - 17s 30ms/step - loss: 0.0010
Epoch 10/10
552/552 [==============================] - 16s 30ms/step - loss: 0.0010
```

[46]:
```
<keras.callbacks.History at 0x12ef492b128>
```

TASK: Plot the history of the loss that occured during training.

[47]:
```
# CODE HERE
```

[48]:
```

```

[48]:
```
dict_keys(['loss'])
```

[ ]:
```

```

2.1 Evaluate on Test Data

TASK: Based on your test data and input size, create an appropriate;y sized "first evaluation batch" like we did in the lecture.

[50]:
```
# CODE HERE
```

[51]:
```

```

[51]:
```
array([[0.79630397],
       [0.71226435],
       [0.90477416],
       [0.93121043],
       [0.98386382],
       [0.96757519],
       [1.        ],
       [0.9801859 ],
       [0.8824684 ],
       [0.95995255],
       [0.85883345],
       [0.90086755]])
```

[52]:
```

```

TASK: Generate predictions into the same time stamps as the test set

[53]:
```
# CODE HERE
```

[54]:
```
```

2.2 Inverse Transformations and Compare

TASK: Inverse Transform your new forecasted predictions.

[55]:
```
#CODE HERE
```

[56]:
```
```

[57]:
```
```

[57]:
```
array([[246787.65124869],
       [235267.94174141],
       [258981.00705367],
       [269320.52187717],
       [280305.23281485],
       [283555.27218211],
       [288203.31152987],
       [283559.36330348],
       [271973.99567699],
       [276999.53167695],
       [261872.87937891],
       [264047.44175631]])
```

TASK: Create a new dataframe that has both the original test values and your predictions for them.

[1]:
```
# CODE HERE
```

[59]:
```
```

[59]:
```
               Value    Predictions
DATE
2018-01-01  245695.0  246787.651249
2018-02-01  226660.0  235267.941741
2018-03-01  268480.0  258981.007054
2018-04-01  272475.0  269320.521877
2018-05-01  286164.0  280305.232815
2018-06-01  280877.0  283555.272182
2018-07-01  288145.0  288203.311530
2018-08-01  286608.0  283559.363303
2018-09-01  260595.0  271973.995677
```

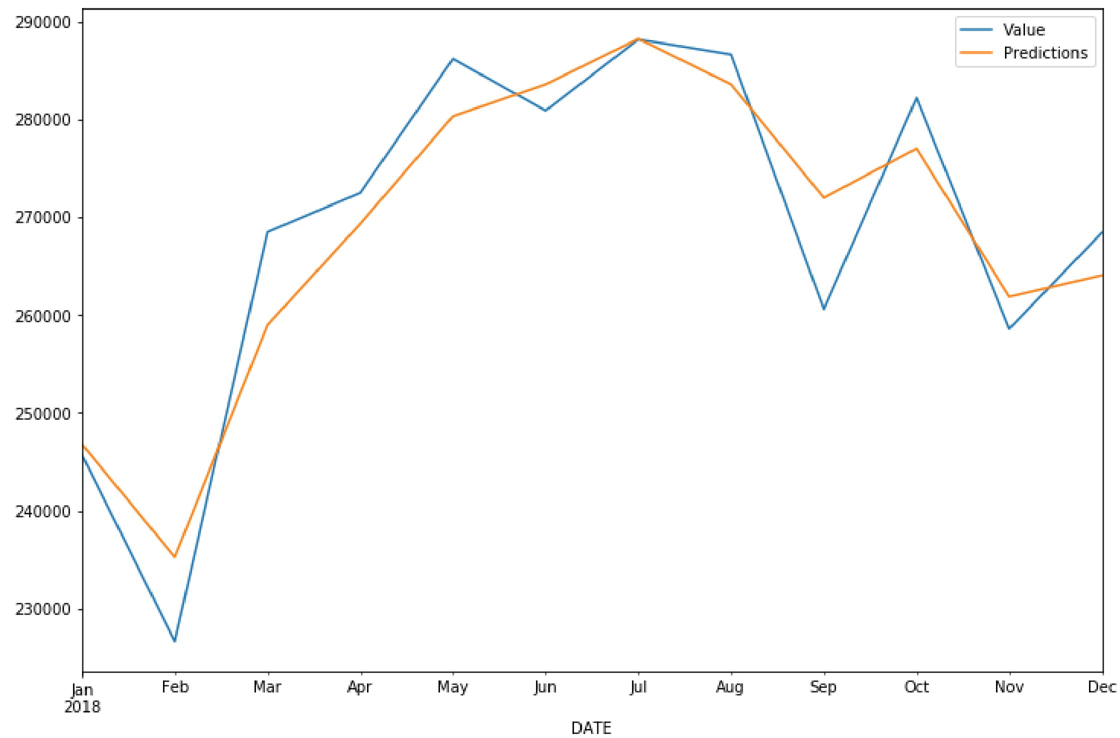
```
2018-10-01  282174.0  276999.531677
2018-11-01  258590.0  261872.879379
2018-12-01  268413.0  264047.441756
```

TASK: Plot out the test set against your own predicted values.

[2]:
```
# CODE HERE
```

[60]:
```
```

[60]: <matplotlib.axes._subplots.AxesSubplot at 0x12ef97a50f0>



3 Saving Models

TASK: Optional, Save your model!

[44]:
```
```