03-RNN-Exercise-Solutions

October 19, 2022

For more information, visit us at www.pieriandata.com

1 RNN Example for Time Series

TASK: IMPORT THE BASIC LIBRARIES YOU THINK YOU WILL USE

[1]:
```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
```

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]:
```
df = pd.read_csv('../Data/Miles_Traveled.csv',index_col='DATE',parse_dates=True)
df.index.freq = 'MS'
```

[4]:
```
df.head()
```

[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
```

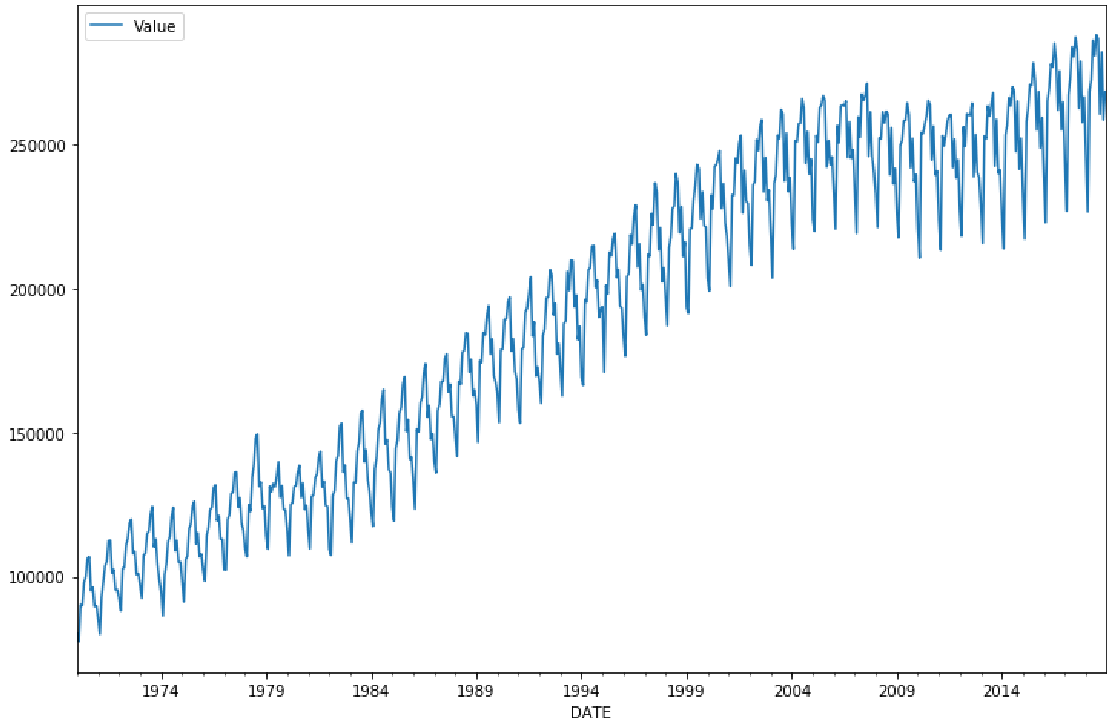
Task: Change the column names to Value

[5]:
```
df.columns = ['Value']
```

TASK: Plot out the time series

[6]:
```
df.plot(figsize=(12,8))
```

[6]: <matplotlib.axes._subplots.AxesSubplot at 0x27987751240>



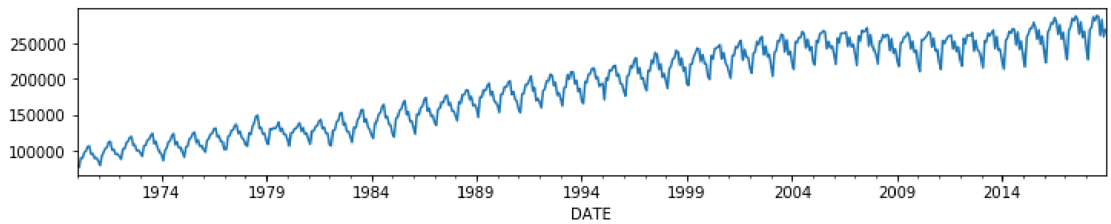
TASK: Perform a Seasonal Decomposition on the model and plot out the ETS components

[7]:
```
# CODE HERE
```

[8]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
```

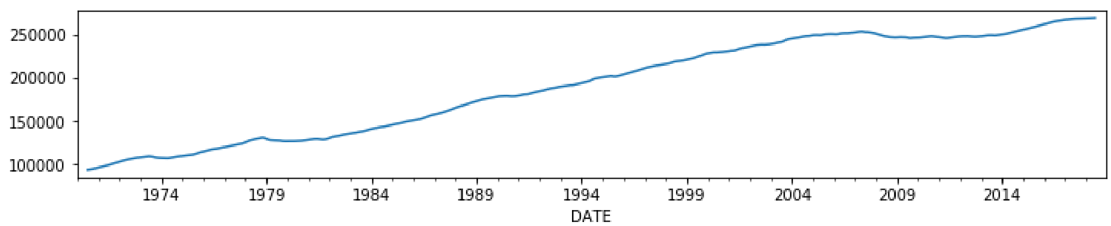
[9]:
```
results = seasonal_decompose(df['Value'])
results.observed.plot(figsize=(12,2))
```

[9]: <matplotlib.axes._subplots.AxesSubplot at 0x279898815c0>



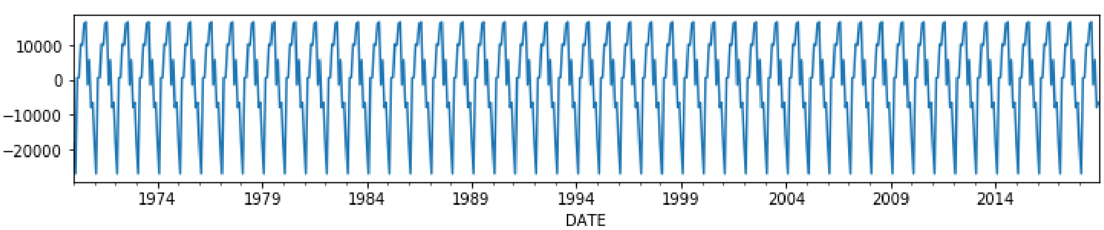
[10]:
```
results.trend.plot(figsize=(12,2))
```

[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27987a4eac8>
```



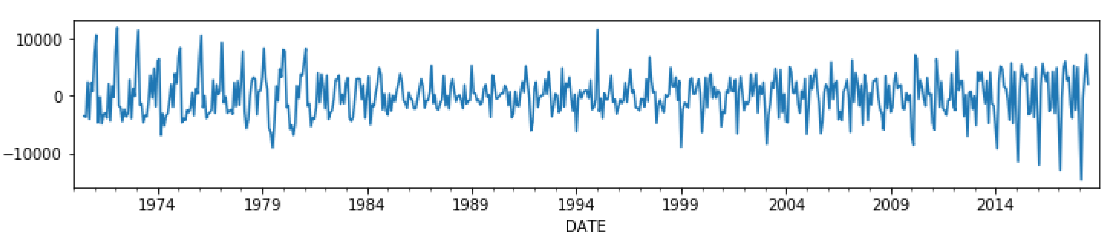
[11]:
```
results.seasonal.plot(figsize=(12,2))
```

[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x279899b64e0>
```



[12]:
```
results.resid.plot(figsize=(12,2))
```

[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x279899a9828>
```



1.2 Train Test Split

TASK: Figure out the length of the data set

[13]:
```
len(df)
```

[13]:
```
588
```

[14]:
```
train_len = len(df)-12
```

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

[15]:
```
train = df.iloc[:train_len]
test = df.iloc[train_len:]
```

[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]:
```
from sklearn.preprocessing import MinMaxScaler
```

[19]:
```
scaler = MinMaxScaler()
```

[20]:
```
# IGNORE WARNING ITS JUST CONVERTING TO FLOATS
# WE ONLY FIT TO TRAININ DATA, OTHERWISE WE ARE CHEATING ASSUMING INFO ABOUT␣
↪TEST SET
scaler.fit(train)
```

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

[21]:
```
scaled_train = scaler.transform(train)
scaled_test = scaler.transform(test)
```

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]:
```
from keras.preprocessing.sequence import TimeseriesGenerator
```

```
Using TensorFlow backend.
```

[24]:
```
n_input = 24
n_features=1
generator = TimeseriesGenerator(scaled_train, scaled_train, length=n_input,␣
 ↪batch_size=1)
```

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]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
```

[26]:
```
# define model
model = Sequential()
model.add(LSTM(150, activation='relu', input_shape=(n_input, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
```

[27]:
```
model.summary()
```

```
_________________________________________________________________
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
```

[29]:
```
# fit model
model.fit_generator(generator,epochs=30)
```

```
Epoch 1/30
552/552 [==============================] - 18s 33ms/step - loss: 0.0139
Epoch 2/30
552/552 [==============================] - 17s 30ms/step - loss: 0.0070
Epoch 3/30
552/552 [==============================] - 16s 30ms/step - loss: 0.0056
Epoch 4/30
552/552 [==============================] - 16s 29ms/step - loss: 0.0050
Epoch 5/30
552/552 [==============================] - 17s 30ms/step - loss: 0.0036
Epoch 6/30
552/552 [==============================] - 16s 30ms/step - loss: 0.0025
Epoch 7/30
552/552 [==============================] - 16s 30ms/step - loss: 0.0025
Epoch 8/30
552/552 [==============================] - 17s 30ms/step - loss: 0.0018
Epoch 9/30
552/552 [==============================] - 18s 32ms/step - loss: 0.0016
Epoch 10/30
552/552 [==============================] - 19s 34ms/step - loss: 0.0017
Epoch 11/30
552/552 [==============================] - 18s 33ms/step - loss: 0.0016
Epoch 12/30
552/552 [==============================] - 18s 33ms/step - loss: 0.0013
Epoch 13/30
552/552 [==============================] - 18s 33ms/step - loss: 0.0014
Epoch 14/30
552/552 [==============================] - 18s 33ms/step - loss: 0.0012 0s - l
Epoch 15/30
552/552 [==============================] - 18s 33ms/step - loss: 0.0013
Epoch 16/30
552/552 [==============================] - 19s 34ms/step - loss: 0.0011
Epoch 17/30
552/552 [==============================] - 17s 32ms/step - loss: 0.0011
Epoch 18/30
552/552 [==============================] - 18s 33ms/step - loss: 9.6872e-04
Epoch 19/30
552/552 [==============================] - 18s 33ms/step - loss: 0.0011
Epoch 20/30
552/552 [==============================] - 18s 33ms/step - loss: 0.0011
Epoch 21/30
552/552 [==============================] - 18s 33ms/step - loss: 9.6653e-04
```

```
Epoch 22/30
552/552 [==============================] - 18s 33ms/step - loss: 9.4804e-04
Epoch 23/30
552/552 [==============================] - 18s 33ms/step - loss: 8.5220e-04
Epoch 24/30
552/552 [==============================] - 18s 33ms/step - loss: 7.8977e-04
Epoch 25/30
552/552 [==============================] - 18s 33ms/step - loss: 7.9855e-04
Epoch 26/30
552/552 [==============================] - 18s 33ms/step - loss: 8.0307e-04
Epoch 27/30
552/552 [==============================] - 18s 33ms/step - loss: 7.8170e-04
Epoch 28/30
552/552 [==============================] - 18s 33ms/step - loss: 7.8733e-04
Epoch 29/30
552/552 [==============================] - 18s 33ms/step - loss: 7.7737e-04
Epoch 30/30
552/552 [==============================] - 18s 33ms/step - loss: 7.7677e-04
```

[29]:
```
<keras.callbacks.History at 0x2798f3695f8>
```

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

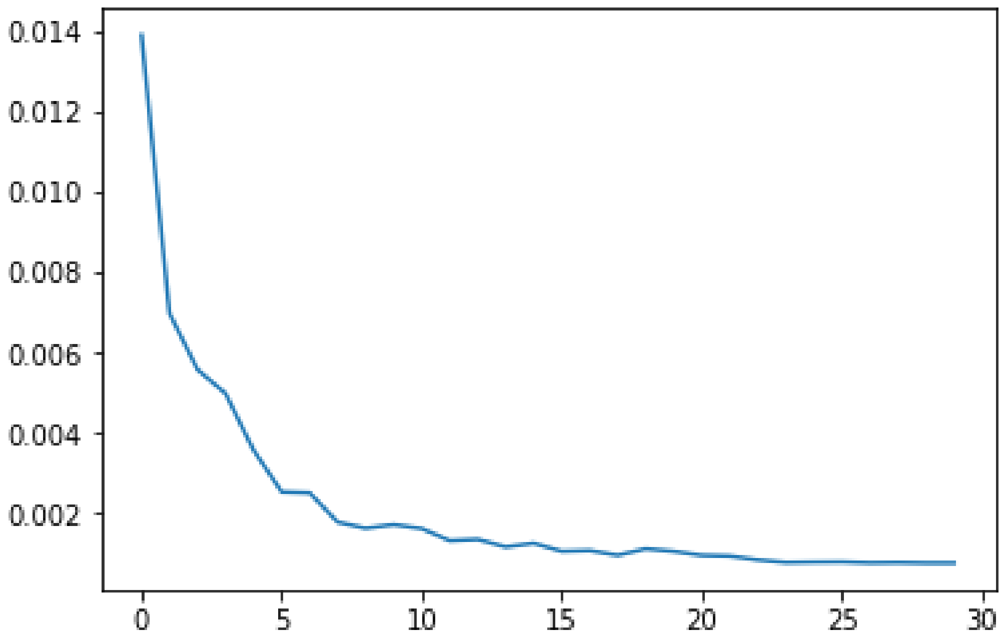
[30]:
```
# CODE HERE
```

[31]:
```
model.history.history.keys()
```

[31]:
```
dict_keys(['loss'])
```

[32]:
```
loss_per_epoch = model.history.history['loss']
plt.plot(range(len(loss_per_epoch)),loss_per_epoch)
```

[32]:
```
[<matplotlib.lines.Line2D at 0x27bf44532e8>]
```



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.

[33]:
```
first_eval_batch = scaled_train[-12:]
```

[34]:
```
first_eval_batch
```

[34]:
```
array([[0.79630397],
       [0.71226435],
       [0.90477416],
       [0.93121043],
       [0.98386382],
       [0.96757519],
       [1.        ],
       [0.9801859 ],
       [0.8824684 ],
       [0.95995255],
       [0.85883345],
       [0.90086755]])
```

[35]:
```
first_eval_batch = first_eval_batch.reshape((1, 12, n_features))
```

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

[36]:
```
# CODE HERE
```

[37]:
```
test_predictions = []

first_eval_batch = scaled_train[-n_input:]
current_batch = first_eval_batch.reshape((1, n_input, n_features))

for i in range(len(test)):

    # get prediction 1 time stamp ahead ([0] is for grabbing just the number␣
 ↪instead of [array])
    current_pred = model.predict(current_batch)[0]

    # store prediction
    test_predictions.append(current_pred)

    # update batch to now include prediction and drop first value
    current_batch = np.append(current_batch[:,1:,:],[[current_pred]],axis=1)
```

2.2 Inverse Transformations and Compare

TASK: Inverse Transform your new forecasted predictions.

[38]:
```
#CODE HERE
```

[39]:
```
true_predictions = scaler.inverse_transform(test_predictions)
```

[40]:
```
true_predictions
```

[40]:
```
array([[244790.4583751 ],
       [233844.90710074],
       [270457.81608635],
       [272201.29687929],
       [279289.6588499 ],
       [279346.60926121],
       [286210.98546427],
       [281885.08126384],
       [268614.50942963],
       [275487.78100669],
       [258639.72996229],
       [262481.15531212]])
```

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

[41]:
```
# IGNORE WARNINGS
test['Predictions'] = true_predictions
```

```
C:\Users\Marcial\Anaconda3\lib\site-packages\ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
```

[42]:
```
test
```

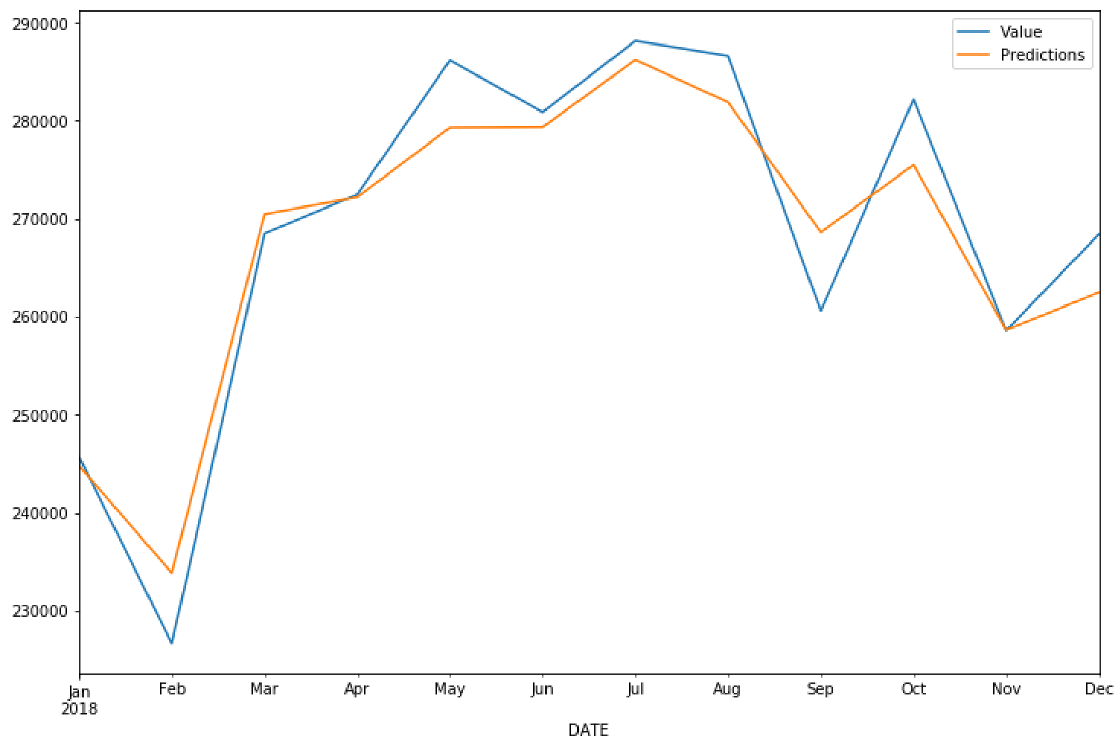
[42]:

	Value	Predictions
DATE		
2018-01-01	245695.0	244790.458375
2018-02-01	226660.0	233844.907101
2018-03-01	268480.0	270457.816086
2018-04-01	272475.0	272201.296879
2018-05-01	286164.0	279289.658850
2018-06-01	280877.0	279346.609261
2018-07-01	288145.0	286210.985464
2018-08-01	286608.0	281885.081264
2018-09-01	260595.0	268614.509430
2018-10-01	282174.0	275487.781007
2018-11-01	258590.0	258639.729962
2018-12-01	268413.0	262481.155312

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

[43]:
```
test.plot(figsize=(12,8))
```

[43]: <matplotlib.axes._subplots.AxesSubplot at 0x279bcda4588>



3 Saving Models

TASK: Optional, Save your model!

[44]:
```
model.save('solutions_model.h5')
```