01_univariate_time_series_regression

September 29, 2021

1 Recurrent Neural Networks

1.1 Univariate Time Series Regression

This notebook demonstrates how to forecast the S&P 500 index using a Recurrent Neural Network.

1.2 Run inside docker container for GPU acceleration

See tensorflow guide and more detailed instructions

```
docker run -it -p 8889:8888 -v /path/to/machine-learning-for-trading/18_recurrent_neural_nets:/
--name tensorflow tensorflow/tensorflow:latest-gpu-py3 bash
```

Inside docker container: `jupyter notebook --ip 0.0.0.0 --no-browser --allow-root`

1.3 Imports & Settings

[1]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pandas_datareader.data as web
from datetime import datetime, date
from sklearn.metrics import mean_squared_error, roc_auc_score
from sklearn.preprocessing import minmax_scale
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.models import Sequential
from keras.layers import Dense, LSTM
import keras
```

```
Using TensorFlow backend.
```

[2]:
```
sns.set_style('whitegrid')
np.random.seed(42)
```

1.4 Get Data

We obtain data for 2010-2018 from the Federal Reserve Bank's Data Service FRED using the pandas_datareader library in introduced in Chapter 2 on Market and Fundamental Data.

[3]:
```
sp500 = web.DataReader('SP500', 'fred', start='2010', end='2019').dropna()
sp500.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2264 entries, 2010-01-04 to 2018-12-31
Data columns (total 1 columns):
SP500    2264 non-null float64
dtypes: float64(1)
memory usage: 35.4 KB
```

1.5 Preprocessing

[4]:
```
sp500_scaled = sp500.apply(minmax_scale)
sp500_scaled.describe()
```

[4]:

	SP500
count	2264.000000
mean	0.437175
std	0.272843
min	0.000000
25%	0.172811
50%	0.462480
75%	0.600596
max	1.000000

1.6 Generating recurrent sequences from our time series

Our time series is a sequence of numbers indexed by time:

$$x_0, x_1, x_2, ..., x_T$$

where $\{x_t\}$ is the numerical value in period t and T is the total length of the series.

To apply a RNN for regression of classification, we use a sliding window to construct a rolling set of input/output pairs for our model to learn from as animated below.

We will generate sequences of 63 trading days, approximately three months, and use a single LSTM layer with 20 hidden units to predict the index value one timestep ahead. The input to every LSTM layer must have three dimensions, namely: - **Samples**: One sequence is one sample. A batch contains one or more samples. - **Time Steps**: One time step is one point of observation in the sample. - **Features**: One feature is one observation at a time step.

Our S&P 500 sample has 2,264 observations or time steps. We will create overlapping sequences using a window of 63 observations each. For a simpler window of size T = 5, we obtain input-output pairs as shown in the following table:

Input	Output
$\langle x_1, x_2, x_3, x_4, x_5 \rangle$	x_6
$\langle x_2, x_3, x_4, x_5, x_6 \rangle$	x_7
$\vdots$	$\vdots$
$\langle x_{T-5}, x_{T-4}, x_{T-3}, x_{T-2}, x_{T-1} \rangle$	x_T

Generally speaking, for window size S, the relationship takes the form

$$x_t = f(x_{t-1}, x_{t-2}, ..., x_{t-S}) \quad \forall t = S, S+1, ..., T$$

Each of the $T - S$ lagged input sequence or vector is of length S with a corresponding scalar output.

We can use the function create_univariate_rnn_data() to stack sequences selected using a rolling windows:

[8]:
```
def create_univariate_rnn_data(data, window_size):
    y = data[window_size:]
    data = data.values.reshape(-1, 1) # make 2D
    n = data.shape[0]
    X = np.hstack(tuple([data[i: n-j, :] for i, j in␣
 ↪enumerate(range(window_size, 0, -1))]))
    return pd.DataFrame(X, index=y.index), y
```

We apply this function to the rescaled stock index for a window_size=63 to obtain a two-dimensional dataset of shape number of samples x number of timesteps:

[9]:
```
window_size = 63
```

[10]:
```
X, y = create_univariate_rnn_data(sp500_scaled, window_size=window_size)
```

[11]:
```
X.head()
```

[11]:
```
                   0         1         2         3         4         5  \
DATE
2010-04-06  0.057862  0.059712  0.060037  0.062421  0.064145  0.065193
2010-04-07  0.059712  0.060037  0.062421  0.064145  0.065193  0.059554
2010-04-08  0.060037  0.062421  0.064145  0.065193  0.059554  0.064512
2010-04-09  0.062421  0.064145  0.065193  0.059554  0.064512  0.065969
2010-04-12  0.064145  0.065193  0.059554  0.064512  0.065969  0.059455

                   6         7         8         9  …        53        54  \
DATE                                                …
2010-04-06  0.059554  0.064512  0.065969  0.059455  …  0.075061  0.079443
2010-04-07  0.064512  0.065969  0.059455  0.066897  …  0.079443  0.076062
2010-04-08  0.065969  0.059455  0.066897  0.060508  …  0.076062  0.075020
2010-04-09  0.059455  0.066897  0.060508  0.049209  …  0.075020  0.075470
2010-04-12  0.066897  0.060508  0.049209  0.036255  …  0.075470  0.078945
```

```
                   55        56        57        58        59        60  \
DATE
2010-04-06  0.076062  0.075020  0.075470  0.078945  0.078971  0.076959
2010-04-07  0.075020  0.075470  0.078945  0.078971  0.076959  0.081502
2010-04-08  0.075470  0.078945  0.078971  0.076959  0.081502  0.086397
2010-04-09  0.078945  0.078971  0.076959  0.081502  0.086397  0.087445
2010-04-12  0.078971  0.076959  0.081502  0.086397  0.087445  0.083782

                   61        62
DATE
2010-04-06  0.081502  0.086397
2010-04-07  0.086397  0.087445
2010-04-08  0.087445  0.083782
2010-04-09  0.083782  0.085873
2010-04-12  0.085873  0.090029

[5 rows x 63 columns]
```

[12]:
```python
y.head()
```

[12]:
```
               SP500
DATE
2010-04-06  0.087445
2010-04-07  0.083782
2010-04-08  0.085873
2010-04-09  0.090029
2010-04-12  0.091134
```

[13]:
```python
X.shape
```

[13]: (2201, 63)

[14]:
```python
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2201 entries, 2010-04-06 to 2018-12-31
Data columns (total 63 columns):
0      2201 non-null float64
1      2201 non-null float64
2      2201 non-null float64
3      2201 non-null float64
4      2201 non-null float64
5      2201 non-null float64
6      2201 non-null float64
7      2201 non-null float64
8      2201 non-null float64
```

```
9     2201 non-null float64
10    2201 non-null float64
11    2201 non-null float64
12    2201 non-null float64
13    2201 non-null float64
14    2201 non-null float64
15    2201 non-null float64
16    2201 non-null float64
17    2201 non-null float64
18    2201 non-null float64
19    2201 non-null float64
20    2201 non-null float64
21    2201 non-null float64
22    2201 non-null float64
23    2201 non-null float64
24    2201 non-null float64
25    2201 non-null float64
26    2201 non-null float64
27    2201 non-null float64
28    2201 non-null float64
29    2201 non-null float64
30    2201 non-null float64
31    2201 non-null float64
32    2201 non-null float64
33    2201 non-null float64
34    2201 non-null float64
35    2201 non-null float64
36    2201 non-null float64
37    2201 non-null float64
38    2201 non-null float64
39    2201 non-null float64
40    2201 non-null float64
41    2201 non-null float64
42    2201 non-null float64
43    2201 non-null float64
44    2201 non-null float64
45    2201 non-null float64
46    2201 non-null float64
47    2201 non-null float64
48    2201 non-null float64
49    2201 non-null float64
50    2201 non-null float64
51    2201 non-null float64
52    2201 non-null float64
53    2201 non-null float64
54    2201 non-null float64
55    2201 non-null float64
56    2201 non-null float64
```

```
57     2201 non-null float64
58     2201 non-null float64
59     2201 non-null float64
60     2201 non-null float64
61     2201 non-null float64
62     2201 non-null float64
dtypes: float64(63)
memory usage: 1.1 MB
```

1.7 Train-test split

To respect the time series nature of the data, we set aside the data at the end of the sample as hold-out or test set. More specifically, we'll use the data for 2018.

[15]:
```
sp500_scaled.plot(lw=2);
```

[16]:
```
X_train = X[:'2017'].values.reshape(-1, window_size, 1)
y_train = y[:'2017']

# keep the last year for testing
X_test = X['2018'].values.reshape(-1, window_size, 1)
y_test = y['2018']
```

[17]:
```
n_obs, window_size, n_features = X_train.shape
```

[18]:
```
y_train.shape
```

[18]: (1950, 1)

1.8 Keras LSTM Layer

Keras has several built-in RNN layers with various configuration options described in detail in the documentation.

[ ]:
```
LSTM(units,
      activation='tanh',
      recurrent_activation='hard_sigmoid',
      use_bias=True,
      kernel_initializer='glorot_uniform',
      recurrent_initializer='orthogonal',
      bias_initializer='zeros',
      unit_forget_bias=True,
      kernel_regularizer=None,
      recurrent_regularizer=None,
      bias_regularizer=None,
      activity_regularizer=None,
      kernel_constraint=None,
      recurrent_constraint=None,
      bias_constraint=None,
      dropout=0.0,
      recurrent_dropout=0.0,
      implementation=1,
      return_sequences=False,
      return_state=False,
      go_backwards=False,
      stateful=False,
      unroll=False)
```

1.9 Define the Model Architecture

Having created input/output pairs out of our time series and cut this into training/testing sets, we can now begin setting up our RNN. We use Keras to quickly build a two hidden layer RNN of the following specifications

- layer 1 uses an LSTM module with 20 hidden units (note here the input_shape = (window_size,1))
- layer 2 uses a fully connected module with one unit
- the 'mean_squared_error' loss should be used (remember: we are performing regression here)

This can be constructed using just a few lines - see e.g., the general Keras documentation and the LTSM documentation in particular for examples of how to quickly use Keras to build neural network models. Make sure you are initializing your optimizer given the keras-recommended approach for RNNs

[19]:
```
rnn = Sequential([
    LSTM(units=20,
```

```
        input_shape=(window_size, n_features), name='LSTM'),
    Dense(1, name='Output')
])
```

```
WARNING:tensorflow:From
/home/stefan/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.6/site-
packages/tensorflow/python/framework/op_def_library.py:263: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Colocations handled automatically by placer.
```

The summary shows that the model has 1,781 parameters:

[20]:
```
rnn.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
LSTM (LSTM)                  (None, 20)                1760
_________________________________________________________________
Output (Dense)               (None, 1)                 21
=================================================================
Total params: 1,781
Trainable params: 1,781
Non-trainable params: 0
_________________________________________________________________
```

1.10 Train the Model

We train the model using the RMSProp optimizer recommended for RNN with default settings and compile the model with mean squared error for this regression problem:

[21]:
```
optimizer = keras.optimizers.RMSprop(lr=0.001,
                                     rho=0.9,
                                     epsilon=1e-08,
                                     decay=0.0)
```

[22]:
```
rnn.compile(loss='mean_squared_error', optimizer=optimizer)
```

[23]:
```
rnn_path = 'models/sp500.lstm5.weights.best.hdf5'
checkpointer = ModelCheckpoint(filepath=rnn_path,
                              monitor='val_loss',
                              save_best_only=True,
                              save_weights_only=True,
                              period=5)
```

We define an EarlyStopping callback and train the model for 500 episodes.

[24]:
```
early_stopping = EarlyStopping(monitor='val_loss',
                              patience=25,
                              restore_best_weights=True)
```

[25]:
```
result = rnn.fit(X_train,
                 y_train,
                 epochs=500,
                 batch_size=20,
                 validation_data=(X_test, y_test),
                 callbacks=[checkpointer, early_stopping],
                 verbose=1)
```

```
WARNING:tensorflow:From
/home/stefan/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.6/site-
packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from
tensorflow.python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.cast instead.
Train on 1950 samples, validate on 251 samples
Epoch 1/500
1950/1950 [==============================] - 4s 2ms/step - loss: 0.0259 -
val_loss: 0.0022
Epoch 2/500
1950/1950 [==============================] - 3s 1ms/step - loss: 6.4320e-04 -
val_loss: 0.0014
Epoch 3/500
1950/1950 [==============================] - 3s 1ms/step - loss: 4.0963e-04 -
val_loss: 0.0043
Epoch 4/500
1950/1950 [==============================] - 3s 1ms/step - loss: 3.6222e-04 -
val_loss: 8.3634e-04
Epoch 5/500
1950/1950 [==============================] - 3s 1ms/step - loss: 3.2014e-04 -
val_loss: 0.0017
Epoch 6/500
1950/1950 [==============================] - 3s 1ms/step - loss: 2.8507e-04 -
val_loss: 6.9610e-04
Epoch 7/500
1950/1950 [==============================] - 3s 1ms/step - loss: 2.7760e-04 -
val_loss: 6.6384e-04
Epoch 8/500
1950/1950 [==============================] - 3s 1ms/step - loss: 2.5975e-04 -
val_loss: 6.3310e-04
Epoch 9/500
1950/1950 [==============================] - 3s 1ms/step - loss: 2.4940e-04 -
val_loss: 6.0271e-04
```

```
Epoch 10/500
1950/1950 [==============================] - 3s 1ms/step - loss: 2.3685e-04 -
val_loss: 8.4301e-04
Epoch 11/500
1950/1950 [==============================] - 3s 1ms/step - loss: 2.2719e-04 -
val_loss: 7.6135e-04
Epoch 12/500
1950/1950 [==============================] - 3s 1ms/step - loss: 2.1947e-04 -
val_loss: 0.0019
Epoch 13/500
1950/1950 [==============================] - 3s 1ms/step - loss: 2.1333e-04 -
val_loss: 5.3785e-04
Epoch 14/500
1950/1950 [==============================] - 3s 2ms/step - loss: 2.0020e-04 -
val_loss: 6.2005e-04
Epoch 15/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.8950e-04 -
val_loss: 0.0015
Epoch 16/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.8961e-04 -
val_loss: 5.1533e-04
Epoch 17/500
1950/1950 [==============================] - 3s 2ms/step - loss: 1.8438e-04 -
val_loss: 4.3401e-04
Epoch 18/500
1950/1950 [==============================] - 3s 2ms/step - loss: 1.7802e-04 -
val_loss: 0.0017
Epoch 19/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.7023e-04 -
val_loss: 4.1096e-04
Epoch 20/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.6940e-04 -
val_loss: 7.7811e-04
Epoch 21/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.6567e-04 -
val_loss: 4.2575e-04
Epoch 22/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.6778e-04 -
val_loss: 5.9410e-04
Epoch 23/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.6142e-04 -
val_loss: 3.4634e-04
Epoch 24/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.5385e-04 -
val_loss: 3.3708e-04
Epoch 25/500
1950/1950 [==============================] - 3s 2ms/step - loss: 1.5047e-04 -
val_loss: 3.5181e-04
```

```
Epoch 26/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.5033e-04 -
val_loss: 4.0619e-04
Epoch 27/500
1950/1950 [==============================] - 3s 2ms/step - loss: 1.4270e-04 -
val_loss: 4.9892e-04
Epoch 28/500
1950/1950 [==============================] - 3s 2ms/step - loss: 1.4054e-04 -
val_loss: 5.5422e-04
Epoch 29/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.4403e-04 -
val_loss: 3.2917e-04
Epoch 30/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.3060e-04 -
val_loss: 3.1522e-04
Epoch 31/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.3969e-04 -
val_loss: 3.8150e-04
Epoch 32/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.2551e-04 -
val_loss: 3.9315e-04
Epoch 33/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.2725e-04 -
val_loss: 3.1077e-04
Epoch 34/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.3154e-04 -
val_loss: 9.7449e-04
Epoch 35/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.2289e-04 -
val_loss: 6.6856e-04
Epoch 36/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.2949e-04 -
val_loss: 3.7870e-04
Epoch 37/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.2081e-04 -
val_loss: 2.8874e-04
Epoch 38/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.1083e-04 -
val_loss: 3.3363e-04
Epoch 39/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.1815e-04 -
val_loss: 2.8503e-04
Epoch 40/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.1974e-04 -
val_loss: 2.7384e-04
Epoch 41/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.1099e-04 -
val_loss: 2.7397e-04
```

```
Epoch 42/500
1950/1950 [==============================] - 3s 2ms/step - loss: 1.1578e-04 -
val_loss: 3.0450e-04
Epoch 43/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.1036e-04 -
val_loss: 3.4770e-04
Epoch 44/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.0236e-04 -
val_loss: 6.2072e-04
Epoch 45/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.1311e-04 -
val_loss: 3.3050e-04
Epoch 46/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.0739e-04 -
val_loss: 4.8290e-04
Epoch 47/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.0477e-04 -
val_loss: 2.8105e-04
Epoch 48/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.9790e-05 -
val_loss: 2.6678e-04
Epoch 49/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.0286e-04 -
val_loss: 4.1710e-04
Epoch 50/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.0573e-04 -
val_loss: 3.1239e-04
Epoch 51/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.0260e-04 -
val_loss: 2.6442e-04
Epoch 52/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.0153e-04 -
val_loss: 9.4587e-04
Epoch 53/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.0048e-04 -
val_loss: 4.5468e-04
Epoch 54/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.6916e-05 -
val_loss: 2.6880e-04
Epoch 55/500
1950/1950 [==============================] - 3s 1ms/step - loss: 1.0439e-04 -
val_loss: 4.1931e-04
Epoch 56/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.7635e-05 -
val_loss: 3.1589e-04
Epoch 57/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.4119e-05 -
val_loss: 3.1625e-04
```

```
Epoch 58/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.5449e-05 -
val_loss: 4.9882e-04
Epoch 59/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.6122e-05 -
val_loss: 2.7764e-04
Epoch 60/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.4437e-05 -
val_loss: 2.6602e-04
Epoch 61/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.2302e-05 -
val_loss: 5.5054e-04
Epoch 62/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.6147e-05 -
val_loss: 2.4944e-04
Epoch 63/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.0018e-05 -
val_loss: 4.9056e-04
Epoch 64/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.7334e-05 -
val_loss: 3.5986e-04
Epoch 65/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.9901e-05 -
val_loss: 2.6305e-04
Epoch 66/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.3220e-05 -
val_loss: 3.1781e-04
Epoch 67/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.3404e-05 -
val_loss: 5.4133e-04
Epoch 68/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.1018e-05 -
val_loss: 3.5198e-04
Epoch 69/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.1941e-05 -
val_loss: 2.3394e-04
Epoch 70/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.8822e-05 -
val_loss: 3.5617e-04
Epoch 71/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.8030e-05 -
val_loss: 3.9396e-04
Epoch 72/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.5764e-05 -
val_loss: 6.6562e-04
Epoch 73/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.5432e-05 -
val_loss: 4.6716e-04
```

```
Epoch 74/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.7653e-05 -
val_loss: 2.4659e-04
Epoch 75/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.9781e-05 -
val_loss: 2.3857e-04
Epoch 76/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.8396e-05 -
val_loss: 3.5092e-04
Epoch 77/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.5004e-05 -
val_loss: 2.3115e-04
Epoch 78/500
1950/1950 [==============================] - 3s 1ms/step - loss: 9.1321e-05 -
val_loss: 4.1456e-04
Epoch 79/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.7536e-05 -
val_loss: 3.9532e-04
Epoch 80/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.7761e-05 -
val_loss: 4.0047e-04
Epoch 81/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.6047e-05 -
val_loss: 2.3446e-04
Epoch 82/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.0903e-05 -
val_loss: 5.7490e-04
Epoch 83/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.5997e-05 -
val_loss: 3.4693e-04
Epoch 84/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.4650e-05 -
val_loss: 2.4621e-04
Epoch 85/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.4828e-05 -
val_loss: 7.6684e-04
Epoch 86/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.4825e-05 -
val_loss: 2.4294e-04
Epoch 87/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.2162e-05 -
val_loss: 2.8051e-04
Epoch 88/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.7945e-05 -
val_loss: 2.9734e-04
Epoch 89/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.4162e-05 -
val_loss: 2.3595e-04
```

```
Epoch 90/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.5688e-05 -
val_loss: 2.2958e-04
Epoch 91/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.3299e-05 -
val_loss: 4.1246e-04
Epoch 92/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.0612e-05 -
val_loss: 8.1742e-04
Epoch 93/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.6531e-05 -
val_loss: 3.3606e-04
Epoch 94/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.4190e-05 -
val_loss: 2.5197e-04
Epoch 95/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.2990e-05 -
val_loss: 2.9431e-04
Epoch 96/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.2887e-05 -
val_loss: 2.3025e-04
Epoch 97/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.2982e-05 -
val_loss: 3.2498e-04
Epoch 98/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.4710e-05 -
val_loss: 2.5490e-04
Epoch 99/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.2775e-05 -
val_loss: 2.6047e-04
Epoch 100/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.3247e-05 -
val_loss: 3.5049e-04
Epoch 101/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.2596e-05 -
val_loss: 2.5096e-04
Epoch 102/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.2268e-05 -
val_loss: 3.7958e-04
Epoch 103/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.3137e-05 -
val_loss: 2.2946e-04
Epoch 104/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.2451e-05 -
val_loss: 2.6558e-04
Epoch 105/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.1882e-05 -
val_loss: 2.2816e-04
```

```
Epoch 106/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.3570e-05 -
val_loss: 2.3123e-04
Epoch 107/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.8835e-05 -
val_loss: 3.5646e-04
Epoch 108/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.9922e-05 -
val_loss: 2.5957e-04
Epoch 109/500
1950/1950 [==============================] - 3s 2ms/step - loss: 8.0363e-05 -
val_loss: 3.6044e-04
Epoch 110/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.3008e-05 -
val_loss: 2.4002e-04
Epoch 111/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.1985e-05 -
val_loss: 6.5130e-04
Epoch 112/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.1532e-05 -
val_loss: 2.3963e-04
Epoch 113/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.1931e-05 -
val_loss: 2.5075e-04
Epoch 114/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.1507e-05 -
val_loss: 2.2866e-04
Epoch 115/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.0922e-05 -
val_loss: 2.2727e-04
Epoch 116/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.1009e-05 -
val_loss: 4.5465e-04
Epoch 117/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.7969e-05 -
val_loss: 4.4201e-04
Epoch 118/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.9101e-05 -
val_loss: 2.6725e-04
Epoch 119/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.9465e-05 -
val_loss: 4.1787e-04
Epoch 120/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.3085e-05 -
val_loss: 3.0253e-04
Epoch 121/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.0912e-05 -
val_loss: 2.3440e-04
```

```
Epoch 122/500
1950/1950 [==============================] - 3s 2ms/step - loss: 8.2004e-05 -
val_loss: 5.3592e-04
Epoch 123/500
1950/1950 [==============================] - 3s 2ms/step - loss: 8.2487e-05 -
val_loss: 2.6276e-04
Epoch 124/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.7115e-05 -
val_loss: 3.0590e-04
Epoch 125/500
1950/1950 [==============================] - 3s 2ms/step - loss: 8.4017e-05 -
val_loss: 5.3031e-04
Epoch 126/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.1123e-05 -
val_loss: 4.2408e-04
Epoch 127/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.9488e-05 -
val_loss: 3.7248e-04
Epoch 128/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.3095e-05 -
val_loss: 2.2539e-04
Epoch 129/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.7840e-05 -
val_loss: 4.3200e-04
Epoch 130/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.1818e-05 -
val_loss: 2.3462e-04
Epoch 131/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.9268e-05 -
val_loss: 3.3856e-04
Epoch 132/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.6852e-05 -
val_loss: 2.4157e-04
Epoch 133/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.9114e-05 -
val_loss: 2.4436e-04
Epoch 134/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.9573e-05 -
val_loss: 3.5230e-04
Epoch 135/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.0803e-05 -
val_loss: 2.8551e-04
Epoch 136/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.8001e-05 -
val_loss: 2.2527e-04
Epoch 137/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.6306e-05 -
val_loss: 2.4812e-04
```

```
Epoch 138/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.9999e-05 -
val_loss: 2.9246e-04
Epoch 139/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.8573e-05 -
val_loss: 2.2760e-04
Epoch 140/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.8896e-05 -
val_loss: 3.0829e-04
Epoch 141/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.8071e-05 -
val_loss: 2.9816e-04
Epoch 142/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.1856e-05 -
val_loss: 3.2728e-04
Epoch 143/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.0268e-05 -
val_loss: 6.5046e-04
Epoch 144/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.7013e-05 -
val_loss: 6.2298e-04
Epoch 145/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.8551e-05 -
val_loss: 6.0778e-04
Epoch 146/500
1950/1950 [==============================] - 3s 2ms/step - loss: 7.8303e-05 -
val_loss: 6.7426e-04
Epoch 147/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.5538e-05 -
val_loss: 2.2505e-04
Epoch 148/500
1950/1950 [==============================] - 3s 1ms/step - loss: 8.2132e-05 -
val_loss: 2.3028e-04
Epoch 149/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.6928e-05 -
val_loss: 2.2649e-04
Epoch 150/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.8825e-05 -
val_loss: 2.6291e-04
Epoch 151/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.9428e-05 -
val_loss: 2.3031e-04
Epoch 152/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.7981e-05 -
val_loss: 2.2469e-04
Epoch 153/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.8535e-05 -
val_loss: 3.3456e-04
```

```
Epoch 154/500
1950/1950 [==============================] - 3s 2ms/step - loss: 7.5944e-05 -
val_loss: 2.2601e-04
Epoch 155/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.7516e-05 -
val_loss: 2.2957e-04
Epoch 156/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.8716e-05 -
val_loss: 2.5327e-04
Epoch 157/500
1950/1950 [==============================] - 3s 2ms/step - loss: 7.5919e-05 -
val_loss: 3.3461e-04
Epoch 158/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.7729e-05 -
val_loss: 3.8942e-04
Epoch 159/500
1950/1950 [==============================] - 3s 2ms/step - loss: 7.9647e-05 -
val_loss: 2.2755e-04
Epoch 160/500
1950/1950 [==============================] - 3s 2ms/step - loss: 7.7441e-05 -
val_loss: 2.2691e-04
Epoch 161/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.8661e-05 -
val_loss: 2.2745e-04
Epoch 162/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.5917e-05 -
val_loss: 2.2520e-04
Epoch 163/500
1950/1950 [==============================] - 3s 2ms/step - loss: 7.9040e-05 -
val_loss: 2.2528e-04
Epoch 164/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.7964e-05 -
val_loss: 2.3334e-04
Epoch 165/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.4630e-05 -
val_loss: 4.4796e-04
Epoch 166/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.8689e-05 -
val_loss: 4.2133e-04
Epoch 167/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.7186e-05 -
val_loss: 3.9237e-04
Epoch 168/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.4509e-05 -
val_loss: 4.4618e-04
Epoch 169/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.7624e-05 -
val_loss: 3.2093e-04
```

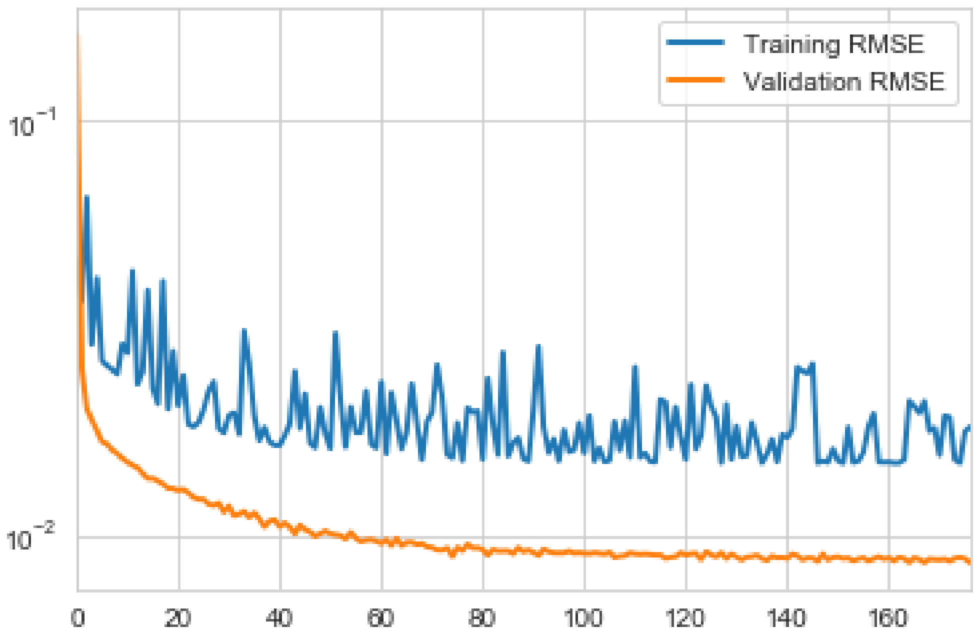
```
Epoch 170/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.7089e-05 -
val_loss: 3.3168e-04
Epoch 171/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.6876e-05 -
val_loss: 2.4541e-04
Epoch 172/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.7408e-05 -
val_loss: 3.7961e-04
Epoch 173/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.5646e-05 -
val_loss: 3.7333e-04
Epoch 174/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.8253e-05 -
val_loss: 2.3300e-04
Epoch 175/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.8868e-05 -
val_loss: 2.2529e-04
Epoch 176/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.7858e-05 -
val_loss: 3.1906e-04
Epoch 177/500
1950/1950 [==============================] - 3s 1ms/step - loss: 7.4459e-05 -
val_loss: 3.3510e-04
```

Training stops after 177 epochs and we reload the weights for the best model:

[26]:
```
rnn.load_weights(rnn_path)
```

The loss history shows how the model's validation error converges to an error level that illustrates the noise inherent in predicting stock prices:

[27]:
```
loss_history = pd.DataFrame(result.history).pow(.5)
loss_history.columns=['Training RMSE', 'Validation RMSE']
loss_history.plot(logy=True, lw=2);
```



1.11 Evaluate model performance

The following charts illustrate the out-of-sample forecast performance that generally track the index development in 2018 well with a test RMSE of 0.015 on the rescaled price series. The test IC is 95.85%.

[28]:
```
def eval_result():
    test_predict = pd.Series(rnn.predict(X_test).squeeze(), index=y_test.index)
    train_predict = pd.Series(rnn.predict(X_train).squeeze(), index=y_train.
 ↪index)
    rmse = np.sqrt(mean_squared_error(test_predict, y_test))
    return test_predict, train_predict, rmse
```

[29]:
```
test_predict, train_predict, rmse = eval_result()
```

[30]:
```
predictions = (test_predict.to_frame('prediction').assign(data='test')
               .append(train_predict.to_frame('prediction').
 ↪assign(data='train')))
predictions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2201 entries, 2018-01-02 to 2017-12-29
Data columns (total 2 columns):
prediction    2201 non-null float32
data          2201 non-null object
dtypes: float32(1), object(1)
```

```
memory usage: 43.0+ KB
```

[31]:
```
results = sp500_scaled.join(predictions).dropna()
results.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2201 entries, 2010-04-06 to 2018-12-31
Data columns (total 3 columns):
SP500         2201 non-null float64
prediction    2201 non-null float32
data          2201 non-null object
dtypes: float32(1), float64(1), object(1)
memory usage: 60.2+ KB
```

[32]:
```
corr = {}
for run, df in results.groupby('data'):
    corr[run] = df.SP500.corr(df.prediction)
```

[33]:
```
sp500_scaled['Train Prediction'] = pd.Series(train_predict.squeeze(),␣
 ↪index=y_train.index)
sp500_scaled['Test Prediction'] = pd.Series(test_predict.squeeze(),␣
 ↪index=y_test.index)
```

[34]:
```
training_error = np.sqrt(rnn.evaluate(X_train, y_train, verbose=0))
testing_error = np.sqrt(rnn.evaluate(X_test, y_test, verbose=0))
print('Training Error: {:.4f} | Test Error: {:.4f}'.format(training_error,␣
 ↪testing_error))
```

```
Training Error: 0.0076 | Test Error: 0.0150
```

[35]:
```
fig=plt.figure(figsize=(14,7))
ax1 = plt.subplot(221)
ax2 = plt.subplot(222)
ax3 = plt.subplot(223)
ax4 = plt.subplot(224, sharex = ax2, sharey=ax2)

sp500_scaled.loc['2015':, 'SP500'].plot(lw=4, ax=ax1, c='k')
sp500_scaled.loc['2015':, ['Test Prediction', 'Train Prediction']].plot(lw=1,␣
 ↪ax=ax1, ls='--')
ax1.set_title('In- and Out-of-sample Predictions')

sns.scatterplot(x='SP500', y='prediction', data=results, hue='data', ax=ax3)
ax3.text(x=.02, y=.95, s='Test IC ={:.2%}'.format(corr['test']), transform=ax3.
 ↪transAxes)
ax3.text(x=.02, y=.87, s='Train IC={:.2%}'.format(corr['train']), transform=ax3.
 ↪transAxes)
ax3.set_title('Correlation')
```

```python
ax3.legend(loc='lower right')

sns.distplot(train_predict.squeeze()-y_train.squeeze(), ax=ax2)
ax2.set_title('Train Error')
ax2.text(x=.03, y=.9, s='Train RMSE ={:.4f}'.format(training_error), transform=ax2.transAxes)
sns.distplot(test_predict.squeeze()-y_test.squeeze(), ax=ax4)
ax4.set_title('Test Error')
ax4.text(x=.03, y=.9, s='Test RMSE ={:.4f}'.format(testing_error), transform=ax4.transAxes)

fig.tight_layout()
fig.savefig('images/rnn_sp500_regression', dpi=300);
```