Data Science Project: LSTM

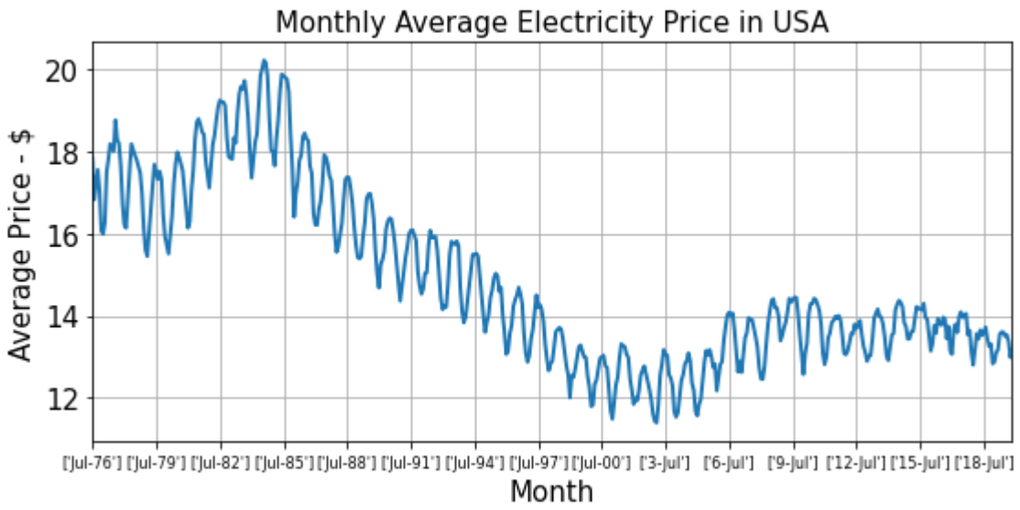
Setting up system

In [1]:
```
import numpy
import pandas
import matplotlib.pyplot as plt
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
```

In [2]:
```
# fix random seed for reproducibility
numpy.random.seed(7)
```

Loading and pre-processing data

In [7]:
```
# load the dataset
dataframe = pandas.read_csv('price.csv', usecols=[2], engine='python'
my_xticks = pandas.read_csv('price.csv', usecols=[1], engine='python'

dataset = dataframe.values
dataset = dataset.astype('float32')
my_xticks = my_xticks.values
my_xticks = my_xticks.astype('str')

#print(plt.rcParams.get('figure.figsize'))
fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 9
fig_size[1] = 4
plt.rcParams["figure.figsize"] = fig_size

plt.plot(dataset,label='Real data',lw=2)

x = numpy.linspace(0,504,15)
x = x.astype(int)
my_xticks = [my_xticks[y] for y in x]
plt.xticks(x, my_xticks)

plt.xlabel('Month',fontsize=15)
plt.ylabel('Average Price - $',fontsize=15)
plt.title('Monthly Average Electricity Price in USA',fontsize=15)
plt.grid(b=None, which='major', axis='both')
plt.xlim([0,520])
plt.xticks(fontsize=8)
plt.yticks(fontsize=15)

plt.savefig('data.png', dpi=600)
plt.show()
```



In [4]:
```
# normalize the dataset
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)
```

In [5]:
```
# split into train and test sets
train_size = int(len(dataset) * 0.794)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset
```

In [6]:
```
# convert an array of values into a dataset matrix
def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])
    return numpy.array(dataX), numpy.array(dataY)
```

In [7]:
```
# reshape into X=t and Y=t+1
look_back = 1
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)
```

In [8]:
```
# reshape input to be [samples, time steps, features]
trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
```