Knapp, Stephen

Deep Learning Assignment 7

In [1]:
```
# Python ≥3.5 is required
import sys
assert sys.version_info >= (3, 5)

# Scikit-Learn ≥0.20 is required
import sklearn
assert sklearn.__version__ >= "0.20"

try:
    # %tensorflow_version only exists in Colab.
    %tensorflow_version 2.x
    IS_COLAB = True
except Exception:
    IS_COLAB = False

# TensorFlow ≥2.0 is required
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dropout
from tensorflow.keras.models import load_model
assert tf.__version__ >= "2.0"

if not tf.test.is_gpu_available():
    print("No GPU was detected. LSTMs and CNNs can be very slow without a GP
U.")
    if IS_COLAB:
        print("Go to Runtime > Change runtime and select a GPU hardware accele
rator.")

# Common imports
import numpy as np
import os
import pandas as pd

#Scale import
from sklearn.preprocessing import MinMaxScaler

# to make this notebook's output stable across runs
np.random.seed(42)
tf.random.set_seed(42)

# To plot pretty figures
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)

# Where to save the figures
PROJECT_ROOT_DIR = "."
CHAPTER_ID = "rnn"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "images", CHAPTER_ID)
```

```
os.makedirs(IMAGES_PATH, exist_ok=True)

def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    print("Saving figure", fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)
```

```
WARNING:tensorflow:From <ipython-input-1-d7ef6e8ac660>:25: is_gpu_available
(from tensorflow.python.framework.test_util) is deprecated and will be remove
d in a future version.
Instructions for updating:
Use `tf.config.list_physical_devices('GPU')` instead.
```

Problem 1

In [0]:
```
def generate_time_series(batch_size, n_steps):
    freq1, freq2, offsets1, offsets2 = np.random.rand(4, batch_size, 1)
    t = np.linspace(0, 15, n_steps)
    series = t / 3 #
    series = series * np.sin((t - offsets1) * (freq1 * 2 + 2)) # wave 1
    series = series + 2 * np.sin((5 * t - offsets2) * (freq2 * 3 + 3)) # wave
 2
    series = series + 0.5 * (np.random.rand(batch_size, n_steps) + 0.5) # nois
e
    return series[..., np.newaxis].astype(np.float32)
```

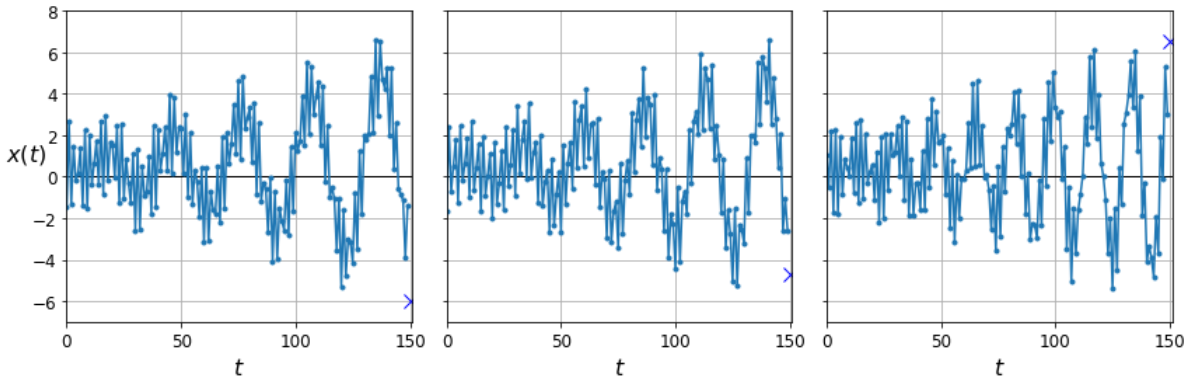
In [0]:
```
np.random.seed(42)

n_steps = 150
series = generate_time_series(10000, n_steps + 1)
X_train, y_train = series[:7000, :n_steps], series[:7000, -1]
X_valid, y_valid = series[7000:9000, :n_steps], series[7000:9000, -1]
X_test, y_test = series[9000:, :n_steps], series[9000:, -1]
```

In [126]:
```
X_train.shape, y_train.shape, X_valid.shape, y_valid.shape, X_test.shape, y_te
st.shape
```

Out[126]:
```
((7000, 150, 1),
 (7000, 1),
 (2000, 150, 1),
 (2000, 1),
 (1000, 150, 1),
 (1000, 1))
```

In [127]:
```
def plot_series(series, y=None, y_pred=None, x_label="$t$", y_label="$x(t)$"):
    plt.plot(series, ".-")
    if y is not None:
        plt.plot(n_steps, y, "bx", markersize=10)
    if y_pred is not None:
        plt.plot(n_steps, y_pred, "ro")
    plt.grid(True)
    if x_label:
        plt.xlabel(x_label, fontsize=16)
    if y_label:
        plt.ylabel(y_label, fontsize=16, rotation=0)
    plt.hlines(0, 0, 200, linewidth=1)
    plt.axis([0, n_steps + 1, -7, 8])

fig, axes = plt.subplots(nrows=1, ncols=3, sharey=True, figsize=(12, 4))
for col in range(3):
    plt.sca(axes[col])
    plot_series(X_valid[col, :, 0], y_valid[col, 0],
                y_label=("$x(t)$" if col==0 else None))
save_fig("time_series_plot")
plt.show()
```

```
Saving figure time_series_plot
```



In [128]:
```
np.random.seed(42)
tf.random.set_seed(42)

model = keras.models.Sequential([
    keras.layers.SimpleRNN(40, return_sequences=True, input_shape=[None, 1]),
    keras.layers.SimpleRNN(40),
    keras.layers.Dense(20)
])

model.compile(loss="mse", optimizer="adam")
history = model.fit(X_train, y_train, epochs=10,
                    validation_data=(X_valid, y_valid))
```

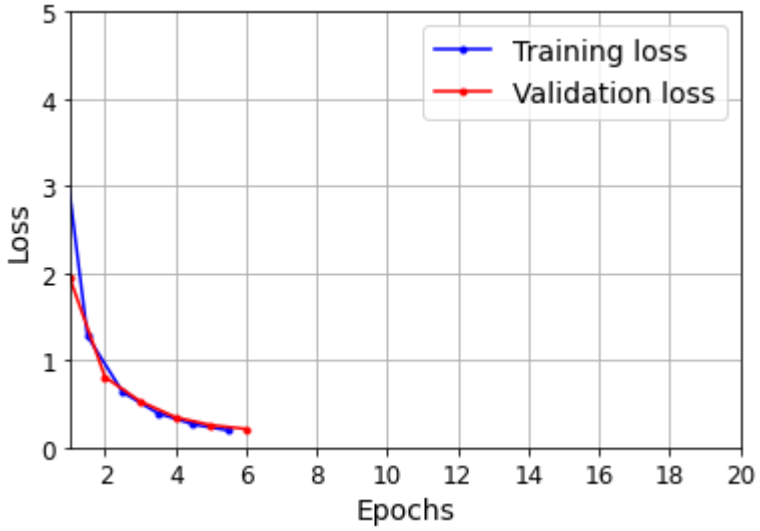
```
Epoch 1/6
219/219 [==============================] - 39s 178ms/step - loss: 4.5474 - va
l_loss: 1.9470
Epoch 2/6
219/219 [==============================] - 39s 178ms/step - loss: 1.2797 - va
l_loss: 0.8076
Epoch 3/6
219/219 [==============================] - 38s 174ms/step - loss: 0.6335 - va
l_loss: 0.5281
Epoch 4/6
219/219 [==============================] - 39s 176ms/step - loss: 0.3922 - va
l_loss: 0.3460
Epoch 5/6
219/219 [==============================] - 38s 175ms/step - loss: 0.2640 - va
l_loss: 0.2558
Epoch 6/6
219/219 [==============================] - 38s 176ms/step - loss: 0.1953 - va
l_loss: 0.2122
```

In [129]:
```
model.evaluate(X_valid, y_valid)
```

```
63/63 [==============================] - 1s 18ms/step - loss: 0.2122
```

Out[129]:
```
0.21222138404846191
```

In [130]:
```
def plot_learning_curves(loss, val_loss):
    plt.plot(np.arange(len(loss)) + 0.5, loss, "b.-", label="Training loss")
    plt.plot(np.arange(len(val_loss)) + 1, val_loss, "r.-", label="Validation
 loss")
    plt.gca().xaxis.set_major_locator(mpl.ticker.MaxNLocator(integer=True))
    plt.axis([1, 10, 0, 5])
    plt.legend(fontsize=14)
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.grid(True)

plot_learning_curves(history.history["loss"], history.history["val_loss"])
plt.show()
```



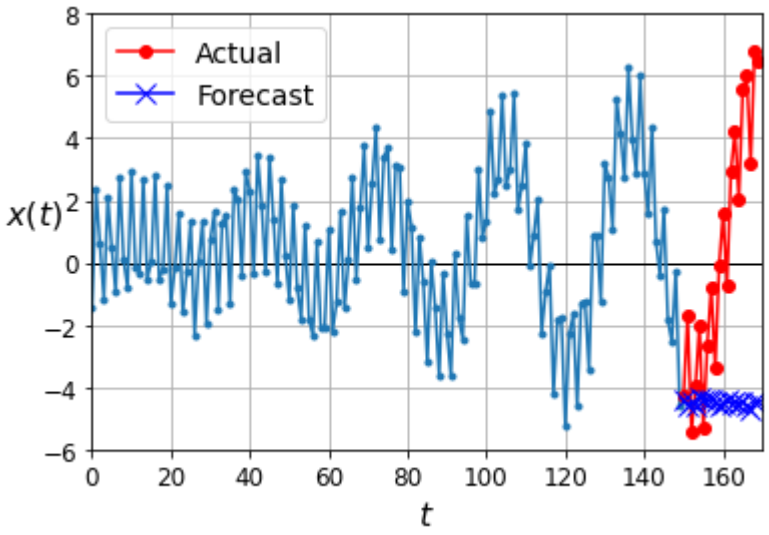
In [0]:
```
np.random.seed(43)

series = generate_time_series(1, 150 + 20)
X_new, Y_new = series[:, :150, :], series[:, -20:, :]
Y_pred = model.predict(X_new)[..., np.newaxis]
```

In [0]:
```
def plot_multiple_forecasts(X, Y, Y_pred):
    n_steps = X.shape[1]
    ahead = Y.shape[1]
    plot_series(X[0, :, 0])
    plt.plot(np.arange(n_steps, n_steps + ahead), Y[0, :, 0], "ro-", label="Ac
tual")
    plt.plot(np.arange(n_steps, n_steps + ahead), Y_pred[0, :, 0], "bx-", labe
l="Forecast", markersize=10)
    plt.axis([0, n_steps + ahead, -6, 8])
    plt.legend(fontsize=14)
```

In [133]:
```
X_new.shape, Y_new.shape, Y_pred.shape
```

Out[133]:
```
((1, 150, 1), (1, 20, 1), (1, 20, 1))
```

In [134]:
```
plot_multiple_forecasts(X_new, Y_new, Y_pred)
plt.show()
```



Problem 2

In [0]:
```
# read in 5 years of data (ending 3/27/2020)
apple_training_complete = pd.read_csv(r'AAPL.csv')
```

In [242]:
```
# plot loaded data
apple_training_complete.plot(x='Date',y='Open')
```

Out[242]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3e22743780>

In [0]:
```
# trim data to only what is needed
apple_training_processed = apple_training_complete.iloc[:, 1:2].values
```

In [244]:
```
print(apple_training_processed)
print("length of the dataset: ", apple_training_processed.size)
```

```
[[124.050003]
 [126.089996]
 [124.82    ]
 ...
 [250.75    ]
 [246.520004]
 [252.75    ]]
length of the dataset:  1259
```

In [0]:
```
# scale data from 0 to 1
scaler = MinMaxScaler(feature_range = (0, 1))

apple_training_scaled = scaler.fit_transform(apple_training_processed)
```

In [0]:
```
# separate features and labels
features_set = []
labels = []
# predictions will be based on 60 days worth of previous data
for i in range(60, 1259):
    features_set.append(apple_training_scaled[i-60:i, 0])
    labels.append(apple_training_scaled[i, 0])
```

In [0]:
```
# turn features and labels into arrays
features_set, labels = np.array(features_set), np.array(labels)
```

In [0]:
```
# reshape features into a tensor shape that the model can accept
features_set = np.reshape(features_set, (features_set.shape[0], features_set.s
hape[1], 1))
```

In [249]:
```
print(features_set.shape[0], features_set.shape[1], 1)
```

```
1199 60 1
```

In [0]:
```
# create sequential LSTM model with dropout layers (to prevent overfitting)
model = Sequential()

model.add(LSTM(units=50, return_sequences=True, input_shape=(features_set.shape[1], 1),unroll=False))
model.add(Dropout(0.2))

model.add(LSTM(units=50, return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(units=50, return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(units=50))
model.add(Dropout(0.2))

# add dense layer with units equal to the number of predicted days needed
model.add(Dense(units = 1))
```

In [0]:
```
# compile model and set optimization method and loss measurement
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

In [252]:
```
#train the model
history = model.fit(features_set, labels, epochs = 100, batch_size = 32)
```

```
Epoch 1/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0185
Epoch 2/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0041
Epoch 3/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0033
Epoch 4/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0037
Epoch 5/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0033
Epoch 6/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0030
Epoch 7/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0026
Epoch 8/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0024
Epoch 9/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0030
Epoch 10/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0023
Epoch 11/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0031
Epoch 12/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0021
Epoch 13/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0020
Epoch 14/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0020
Epoch 15/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0021
Epoch 16/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0020
Epoch 17/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0019
Epoch 18/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0022
Epoch 19/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0018
Epoch 20/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0020
Epoch 21/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0015
Epoch 22/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0017
Epoch 23/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0019
Epoch 24/100
38/38 [==============================] - 1s 14ms/step - loss: 0.0018
Epoch 25/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0018
Epoch 26/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0018
Epoch 27/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0017
Epoch 28/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0017
Epoch 29/100
```

```
38/38 [==============================] - 1s 13ms/step - loss: 0.0019
Epoch 30/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0018
Epoch 31/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0017
Epoch 32/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0015
Epoch 33/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0015
Epoch 34/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0015
Epoch 35/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0017
Epoch 36/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0015
Epoch 37/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0015
Epoch 38/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0015
Epoch 39/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0015
Epoch 40/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0015
Epoch 41/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0021
Epoch 42/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0016
Epoch 43/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0014
Epoch 44/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0015
Epoch 45/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0014
Epoch 46/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0016
Epoch 47/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0015
Epoch 48/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0013
Epoch 49/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0012
Epoch 50/100
38/38 [==============================] - 1s 14ms/step - loss: 0.0016
Epoch 51/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0017
Epoch 52/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0013
Epoch 53/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0011
Epoch 54/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0011
Epoch 55/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0013
Epoch 56/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0010
Epoch 57/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0014
```

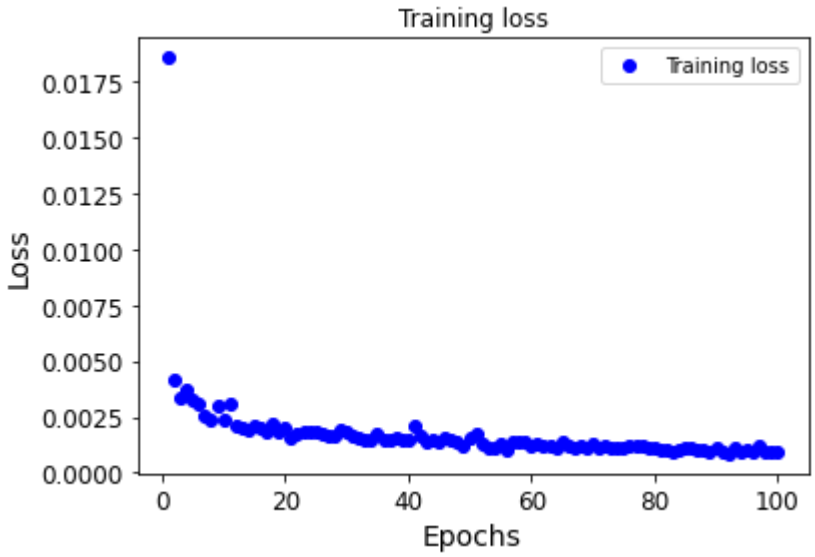
```
Epoch 58/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0014
Epoch 59/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0014
Epoch 60/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0012
Epoch 61/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0013
Epoch 62/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0012
Epoch 63/100
38/38 [==============================] - 1s 14ms/step - loss: 0.0012
Epoch 64/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0011
Epoch 65/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0013
Epoch 66/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0012
Epoch 67/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0011
Epoch 68/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0012
Epoch 69/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0011
Epoch 70/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0013
Epoch 71/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0011
Epoch 72/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0012
Epoch 73/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0011
Epoch 74/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0011
Epoch 75/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0011
Epoch 76/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0012
Epoch 77/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0012
Epoch 78/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0012
Epoch 79/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0011
Epoch 80/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0012
Epoch 81/100
38/38 [==============================] - 0s 13ms/step - loss: 9.9341e-04
Epoch 82/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0011
Epoch 83/100
38/38 [==============================] - 0s 13ms/step - loss: 9.6369e-04
Epoch 84/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0010
Epoch 85/100
38/38 [==============================] - 1s 13ms/step - loss: 0.0011
Epoch 86/100
```

```
38/38 [==============================] - 0s 13ms/step - loss: 0.0011
Epoch 87/100
38/38 [==============================] - 0s 13ms/step - loss: 9.9958e-04
Epoch 88/100
38/38 [==============================] - 0s 13ms/step - loss: 9.9458e-04
Epoch 89/100
38/38 [==============================] - 0s 13ms/step - loss: 9.4945e-04
Epoch 90/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0011
Epoch 91/100
38/38 [==============================] - 0s 13ms/step - loss: 9.3111e-04
Epoch 92/100
38/38 [==============================] - 1s 13ms/step - loss: 8.5288e-04
Epoch 93/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0011
Epoch 94/100
38/38 [==============================] - 0s 13ms/step - loss: 9.1870e-04
Epoch 95/100
38/38 [==============================] - 0s 13ms/step - loss: 9.9460e-04
Epoch 96/100
38/38 [==============================] - 1s 13ms/step - loss: 9.4695e-04
Epoch 97/100
38/38 [==============================] - 0s 13ms/step - loss: 0.0012
Epoch 98/100
38/38 [==============================] - 1s 13ms/step - loss: 9.3619e-04
Epoch 99/100
38/38 [==============================] - 0s 13ms/step - loss: 9.1814e-04
Epoch 100/100
38/38 [==============================] - 1s 13ms/step - loss: 9.6444e-04
```

In [253]:
```
#save the model
model.save('apple.h5')
#place holder for reloading the model
#model = load_model('apple.h5')
#store the model loss history for plotting
history_dict = history.history
history_dict.keys()
```

Out[253]:
```
dict_keys(['loss'])
```

In [254]:
```
# plot the loss during training
loss = history.history['loss']

epochs = range(1, len(loss) + 1)

# "bo" is for "blue dot"
plt.plot(epochs, loss, 'bo', label='Training loss')

plt.title('Training loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



In [255]:
```
# Load test data for target week 3/30-4/3
apple_testing_complete = pd.read_csv(r'AAPL_test2.csv')
apple_testing_processed = apple_testing_complete.iloc[:, 1:2].values
apple_testing_processed
print("Number of data points March 30 - April 3: ", apple_testing_processed.si
ze)
```

```
Number of data points March 30 - April 3:  5
```

In [0]:
```
# combine training and test data
apple_total = pd.concat((apple_training_complete['Open'], apple_testing_comple
te['Open']), axis=0)
```

In [0]:
```
# trimming down data to previous 60 days before the prediction week and the va
lues of the prediction week
test_inputs = apple_total[len(apple_total) - len(apple_testing_complete) - 60
:].values
```

In [0]:
```
# reshape and scale test_inputs so that the model will have the correct tensor
shape and to ensure the same input scale
test_inputs = test_inputs.reshape(-1,1)
test_inputs = scaler.transform(test_inputs)
```

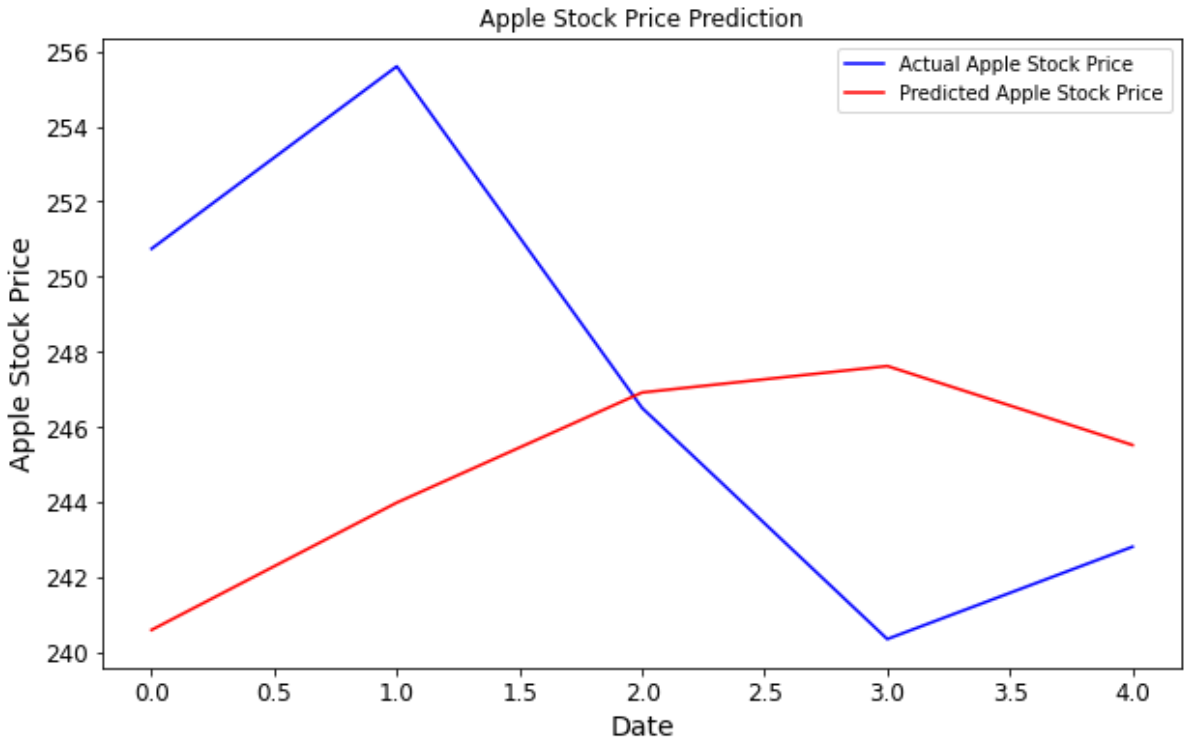
In [0]:
```
# break test inputs into separate 60 day long features for each of the 5 days
 to be predicted
test_features = []
for i in range(60, 65):
    test_features.append(test_inputs[i-60:i, 0])
```

In [0]:
```
# turn test feature into an array and reshape it so that the model will have t
he correct tensor shape input
test_features = np.array(test_features)
test_features = np.reshape(test_features, (test_features.shape[0], test_featur
es.shape[1], 1))
```

In [0]:
```
# generate predictions on each test feature (5 in total)
predictions = model.predict(test_features)
```

In [0]:
```
# untransform the predictions back to actual scale
predictions = scaler.inverse_transform(predictions)
```

In [263]:
```
# plot prediction versus actual
plt.figure(figsize=(10,6))
plt.plot(apple_testing_processed, color='blue', label='Actual Apple Stock Pric
e')
plt.plot(predictions , color='red', label='Predicted Apple Stock Price')
plt.title('Apple Stock Price Prediction')
plt.xlabel('Date')
plt.ylabel('Apple Stock Price')
plt.legend()
plt.show()
```



Problem 3

In [0]:
```
# read in data
jena_weather = pd.read_csv(r'jena_climate_2009_2016.csv')
```

In [0]:
```
jena_weather = jena_weather.drop(columns=['p (mbar)','Tpot (K)','Tdew (degC)',
'rh (%)','VPmax (mbar)','VPact (mbar)','VPdef (mbar)','sh (g/kg)','H2OC (mmol/
mol)','rho (g/m**3)','wv (m/s)','max. wv (m/s)','wd (deg)'])
```

In [0]:
```
jena_weather = jena_weather.rename(columns= {"Date Time":"DateTime", "T (deg
C)":"T"})
```

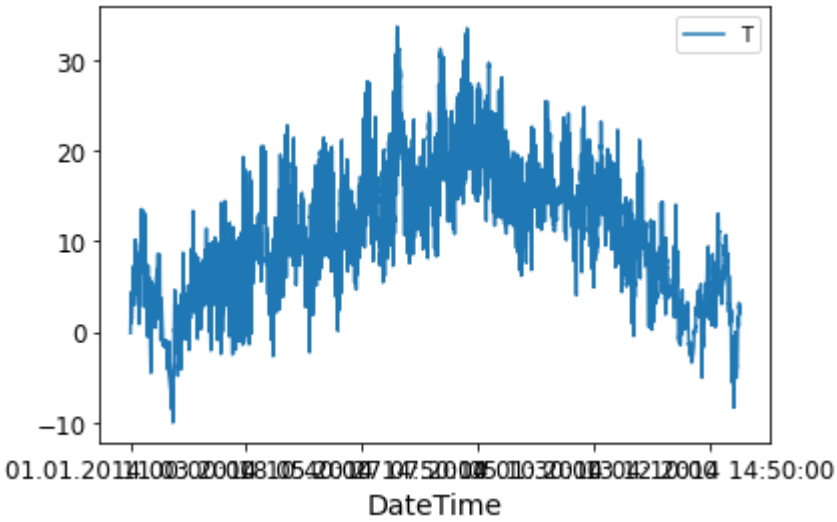
In [0]:
```
jena_weather = jena_weather[jena_weather.DateTime.str.contains(".2014")]
```

In [57]:
```
jena_weather.shape
```

Out[57]:
```
(52647, 2)
```

In [58]:
```
# plot loaded data
jena_weather.plot(x='DateTime',y='T')
```

Out[58]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fadf36eaba8>
```



In [0]:
```
#Scale Temperatures
scaler = MinMaxScaler()
jena_weather[['T']] = scaler.fit_transform(jena_weather[['T']])
```

In [60]:
```
jena_weather.shape
```

Out[60]:
```
(52647, 2)
```

In [0]:
```
TRAIN_SPLIT = 40000
```

In [0]:
```
tf.random.set_seed(13)
```

In [0]:
```
temp_original = jena_weather['T'] # select the 'T (degC)' colums
temp_original.index = jena_weather['DateTime'] # Assign 'Date Time' as index
```

In [64]:
```
type(temp_original)
```

Out[64]:
```
pandas.core.series.Series
```

In [65]:
```
temp_original.head()
```

Out[65]:
```
DateTime
01.01.2014 00:00:00    0.226234
01.01.2014 00:10:00    0.226463
01.01.2014 00:20:00    0.227377
01.01.2014 00:30:00    0.227377
01.01.2014 00:40:00    0.233547
Name: T, dtype: float64
```

In [66]:
```
# convert temperatures to numpy
temp_original = temp_original.values
print(type(temp_original))
```

```
<class 'numpy.ndarray'>
```

In [0]:
```
def univariate_data(dataset, start_index, end_index, history_size, target_size
):
    """
    dataset: this is dataset we are using
    start_index: we use to select training and validation set
    - for training data, we set start_index = 0 and set end_index = TRAIN_SPLI
T = 30000(in this case)
    end_index: this is the end of the data we want
    - for training data, we set end_index = TRAIN_SPLIT = 30000(in this case)
    - for validation data, we set this to NONE
    history size many data points we want to use to make predictions
    target_size: how many predictions do we want to make in future
    """
    data = [] # data will be the historical values
    labels = [] # labels are the y values for the data
    start_index = start_index + history_size
    if end_index is None: # this is none in the case of validation data
        end_index = len(dataset) - target_size # we set out index to end

    #The for loop allows us to select the historical values we need to pick
    # the correspoding lables(y_train)

    for i in range(start_index, end_index):
        # example: when loop start select dataset[0: history_size]
        # then select dataset[1: history_size+1]
        indices = range(i-history_size, i)
        # Reshape data from (history_size,) to (history_size, 1)
        data.append(np.reshape(dataset[indices], (history_size, 1)))
        labels.append(dataset[i+target_size])  # labels are the ith item + tar
get_size
    return np.array(data), np.array(labels)
```

In [0]:
```
temp_past_history = 432 #use 3 days worth of data
temp_future_target = 1 #1 10min interval
```

In [0]:
```
x_train, y_train = univariate_data(dataset = temp_original,
                                   start_index = 0,
                                   end_index = TRAIN_SPLIT,
                                   history_size = temp_past_history,
                                   target_size = temp_future_target)
```

In [73]:
```
print('x_shape: ', x_train.shape)
print('y_shape: ', y_train.shape)
```

```
x_shape:  (39568, 432, 1)
y_shape:  (39568,)
```

In [0]:
```
x_val, y_val = univariate_data(dataset=temp_original,
                               start_index = TRAIN_SPLIT,
                               end_index = None,
                               history_size = temp_past_history,
                               target_size = temp_future_target)
```

In [75]:
```
print('x_val: ', x_val.shape)
print('y_val: ', y_val.shape)
```

```
x_val:  (12214, 432, 1)
y_val:  (12214,)
```

In [0]:
```
def create_time_steps(length):
    return list(range(-length, 0))
```

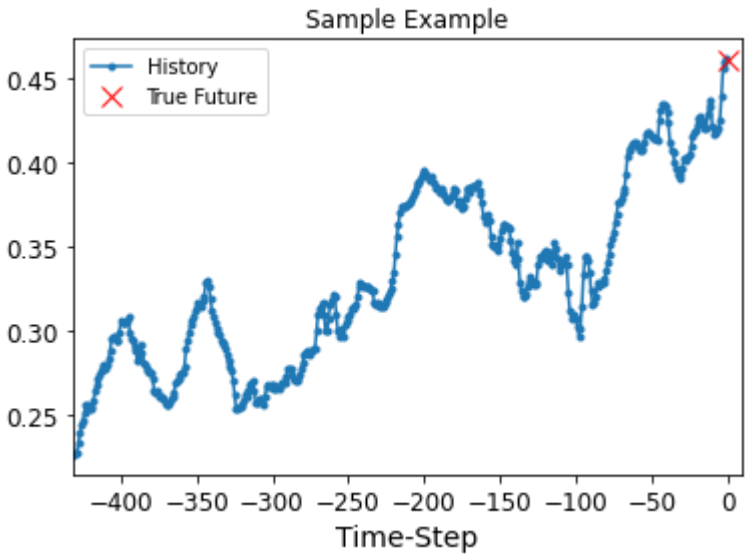
In [77]:
```
create_time_steps(length=3)
```

Out[77]:
```
[-3, -2, -1]
```

In [0]:
```
def show_plot(plot_data, delta, title):
    labels = ['History', 'True Future', 'Model Prediction']
    marker = ['.-', 'rx', 'go']
    time_steps = create_time_steps(plot_data[0].shape[0])
    if delta:
        future = delta
    else:
        future = 0

    plt.title(title)
    for i, x in enumerate(plot_data):
        if i:
            plt.plot(future, plot_data[i], marker[i], markersize=10,
               label=labels[i])
        else:
            plt.plot(time_steps, plot_data[i].flatten(), marker[i], label=labe
ls[i])
    plt.legend()
    plt.xlim([time_steps[0], (future+5)*2])
    plt.xlabel('Time-Step')
    return plt
```

In [79]:
```
show_plot(plot_data = [x_train[0], y_train[0]],  delta = 0, title = 'Sample Example')
```

Out[79]:
```
<module 'matplotlib.pyplot' from '/usr/local/lib/python3.6/dist-packages/matplotlib/pyplot.py'>
```



In [0]:
```
BATCH_SIZE = 256
BUFFER_SIZE = 10000
```

In [0]:
```
train = tf.data.Dataset.from_tensor_slices((x_train, y_train))
```

In [0]:
```
train = train.cache().shuffle(BUFFER_SIZE).batch(BATCH_SIZE).repeat()
```

In [83]:
```
for element in train:
    print('x-train:\n', element[0].shape)
    print('y-train:\n', element[1].shape)
    break
```

```
x-train:
 (256, 432, 1)
y-train:
 (256,)
```

In [0]:
```
val = tf.data.Dataset.from_tensor_slices((x_val, y_val))
```

In [0]:
```
val = val.batch(BATCH_SIZE).repeat()
```

In [86]:
```
for element in val:
    print('x-train:\n', element[0].shape)
    print('y-train:\n', element[1].shape)
    break
```

```
x-train:
 (256, 432, 1)
y-train:
 (256,)
```

In [87]:
```
x_train.shape[-2:]
```

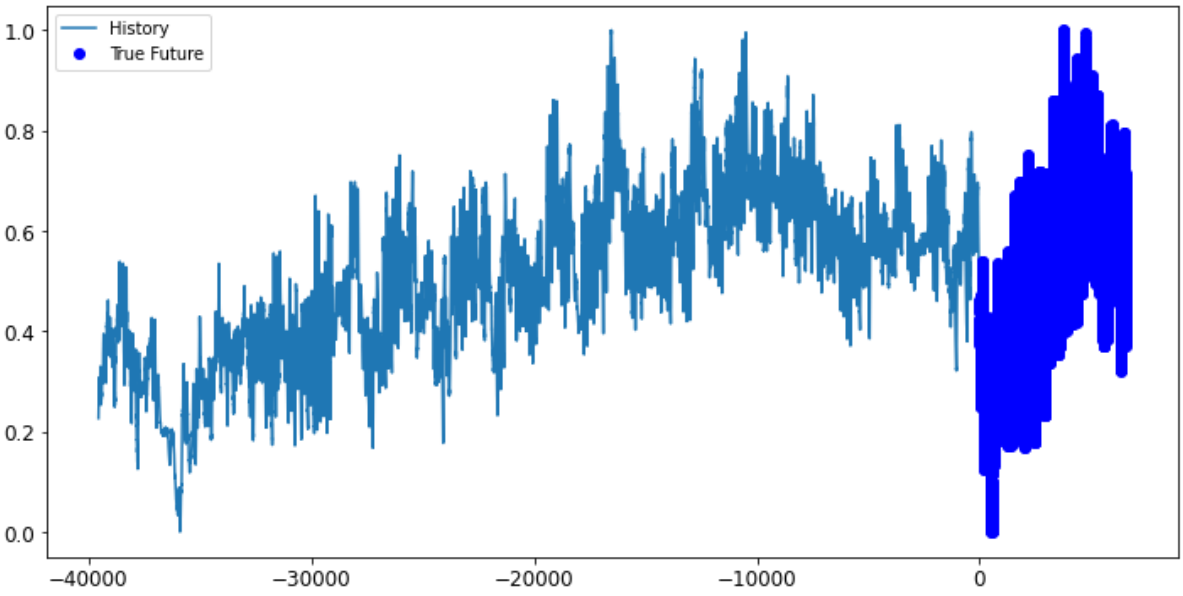
Out[87]:
```
(432, 1)
```

In [88]:
```
x_train.shape
```

Out[88]:
```
(39568, 432, 1)
```

In [0]:
```
def multi_step_plot(history, true_future, prediction):
    plt.figure(figsize=(12, 6))
    num_in = create_time_steps(len(history))
    num_out = len(true_future)

    plt.plot(num_in, np.array(history[:, 1]), label='History')

    plt.plot(np.arange(num_out)/STEP, np.array(true_future), 'bo',
           label='True Future')

    if prediction.any():
        plt.plot(np.arange(num_out)/STEP, np.array(prediction), 'ro',
             label='Predicted Future')
    plt.legend(loc='upper left')
    plt.show()
```

In [95]:
```
for x, y in train.take(1):
    multi_step_plot(x_train, y_train, np.array([0]))
```



In [96]:
```
# use functional programming

# step 1: define layers
inputs = tf.keras.layers.Input(shape=x_train.shape[-2:],name='model_input')
lstm_01 = tf.keras.layers.LSTM(units = 8)
dense_01 =tf.keras.layers.Dense(units = 1, activation='tanh', name='dense_01')

# connect layers
x = lstm_01(inputs)
output = dense_01(x)

simple_lstm_model = tf.keras.Model(inputs=inputs, outputs=output, name = 'simp
le_lstm_model')
simple_lstm_model.compile(optimizer='adam', loss='mae')


EPOCHS = 10
# steps_per_epoch * batch_size = number_of_rows_in_train_data
STEPS_PER_EPOCH = TRAIN_SPLIT/BATCH_SIZE
simple_lstm_model.fit(train, epochs=EPOCHS,
                      steps_per_epoch=STEPS_PER_EPOCH,
                      validation_data=val, validation_steps=50)
```

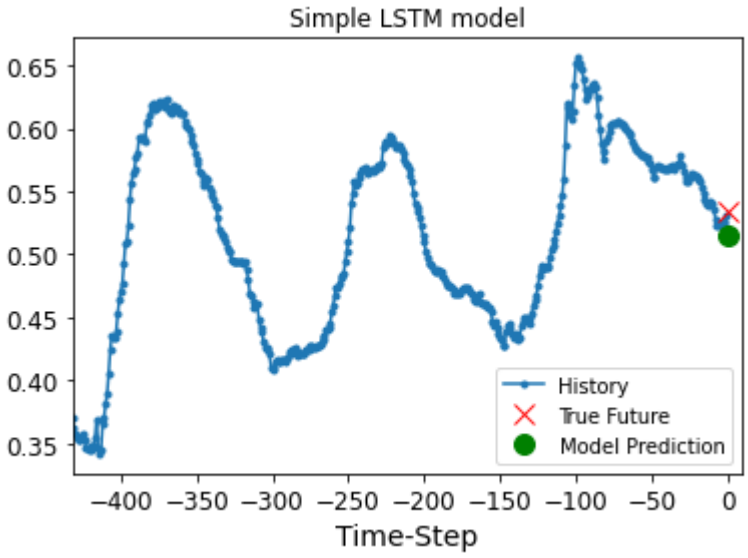
```
Epoch 1/10
157/156 [==============================] - 3s 21ms/step - loss: 0.0770 - val_
loss: 0.0183
Epoch 2/10
157/156 [==============================] - 3s 18ms/step - loss: 0.0213 - val_
loss: 0.0114
Epoch 3/10
157/156 [==============================] - 3s 19ms/step - loss: 0.0165 - val_
loss: 0.0100
Epoch 4/10
157/156 [==============================] - 3s 18ms/step - loss: 0.0145 - val_
loss: 0.0087
Epoch 5/10
157/156 [==============================] - 3s 19ms/step - loss: 0.0131 - val_
loss: 0.0095
Epoch 6/10
157/156 [==============================] - 3s 19ms/step - loss: 0.0122 - val_
loss: 0.0083
Epoch 7/10
157/156 [==============================] - 3s 19ms/step - loss: 0.0117 - val_
loss: 0.0080
Epoch 8/10
157/156 [==============================] - 3s 18ms/step - loss: 0.0112 - val_
loss: 0.0074
Epoch 9/10
157/156 [==============================] - 3s 18ms/step - loss: 0.0108 - val_
loss: 0.0081
Epoch 10/10
157/156 [==============================] - 3s 18ms/step - loss: 0.0106 - val_
loss: 0.0068
```

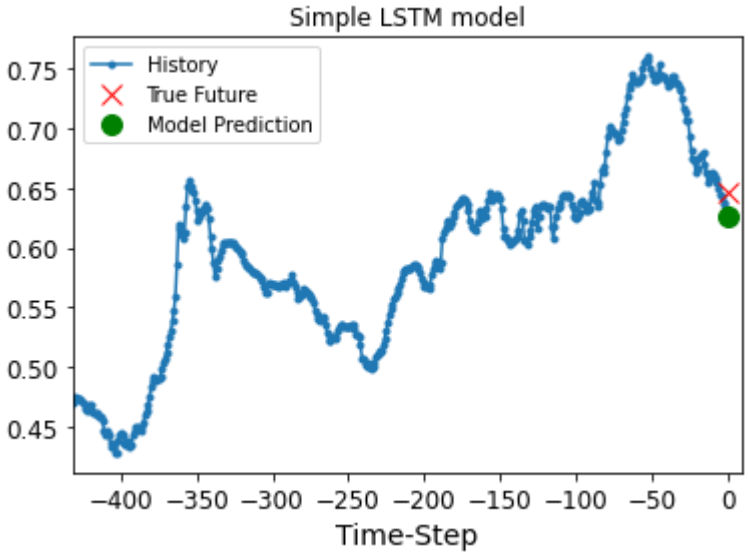
Out[96]: <tensorflow.python.keras.callbacks.History at 0x7fadf33fe0f0>

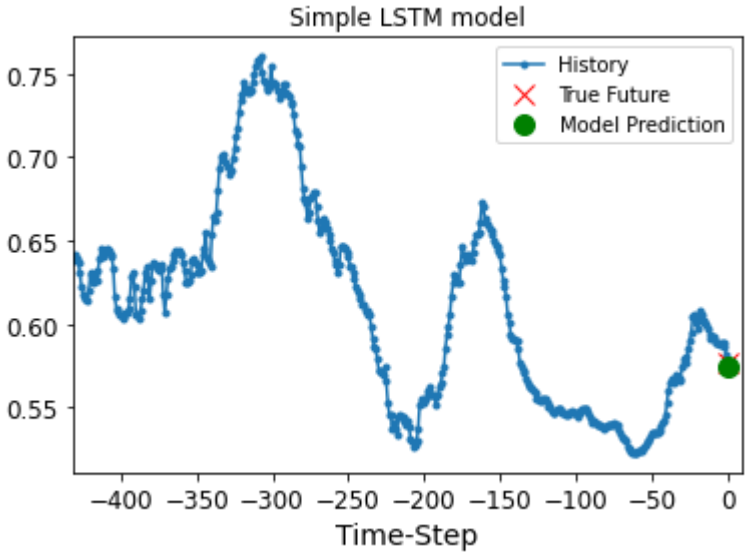
In [97]:
```
simple_lstm_model.summary()
```

```
Model: "simple_lstm_model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
model_input (InputLayer)     [(None, 432, 1)]          0
_________________________________________________________________
lstm_1 (LSTM)                (None, 8)                 320
_________________________________________________________________
dense_01 (Dense)             (None, 1)                 9
=================================================================
Total params: 329
Trainable params: 329
Non-trainable params: 0
_________________________________________________________________
```

In [105]:
```
for x, y in val.take(3):
    plot = show_plot([x[0].numpy(), y[0].numpy(),
                      simple_lstm_model.predict(x)[0]], 0, 'Simple LSTM model')
    plot.show()
```







In [143]:
```
# Load test data for target week 3/30-4/3
jena_test = pd.read_csv(r'jena_climate_2009_2016_test_data1.csv')
print("Number of data points 12/31/14 7am-11am: ", jena_test.size)
```

```
Number of data points 12/31/14 7am-11am:  50
```

In [0]:
```
# trimming down data to previous 3 days before the prediction interval
test_inputs = jena_weather[len(jena_weather) - len(jena_test) - 432:].values
```

In [145]:
```
test_inputs.shape
```

Out[145]:
```
(457, 2)
```

In [146]:
```
# reshape and scale test_inputs so that the model will have the correct tensor
shape
test_inputs = test_inputs[:,1]
test_inputs = test_inputs[np.newaxis,:,np.newaxis]
test_inputs.shape
```

Out[146]:
```
(1, 457, 1)
```

In [0]:
```
# break test inputs into separate 3 day long features for each of the 4 hours
 to be predicted
#test_features = []
#for i in range(432, 456):
    test_features.append(test_inputs[i-432:i, 0])
```

In [136]:
```
#test_features.shape
```

Out[136]:
```
(24,)
```

In [0]:
```
# turn test feature into an array and reshape it so that the model will have t
he correct tensor shape input
#test_features = np.array(test_features)
#test_features = test_features[np.newaxis,:]
```

In [147]:
```
# generate predictions on each test feature (4 hours in total)
predictions = simple_lstm_model.predict(test_inputs)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-147-9b853b560bb7> in <module>()
----> 1 predictions = simple_lstm_model.predict(test_inputs)

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/trainin
g.py in _method_wrapper(self, *args, **kwargs)
     86       raise ValueError('{} is not supported in multi-worker mode.'.fo
rmat(
     87           method.__name__))
---> 88     return method(self, *args, **kwargs)
     89
     90   return tf_decorator.make_decorator(

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/trainin
g.py in predict(self, x, batch_size, verbose, steps, callbacks, max_queue_siz
e, workers, use_multiprocessing)
   1184           workers=workers,
   1185           use_multiprocessing=use_multiprocessing,
-> 1186           model=self)
   1187
   1188       # Container that configures and calls `tf.keras.Callback`s.

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/data_ad
apter.py in __init__(self, x, y, sample_weight, batch_size, steps_per_epoch,
 initial_epoch, epochs, shuffle, class_weight, max_queue_size, workers, use_m
ultiprocessing, model)
   1110         use_multiprocessing=use_multiprocessing,
   1111         distribution_strategy=ds_context.get_strategy(),
-> 1112         model=model)
   1113
   1114     strategy = ds_context.get_strategy()

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/data_ad
apter.py in __init__(self, x, y, sample_weights, sample_weight_modes, batch_s
ize, epochs, steps, shuffle, **kwargs)
    263                **kwargs):
    264     super(TensorLikeDataAdapter, self).__init__(x, y, **kwargs)
--> 265     x, y, sample_weights = _process_tensorlike((x, y, sample_weights)
)
    266     sample_weight_modes = broadcast_sample_weight_modes(
    267         sample_weights, sample_weight_modes)

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/data_ad
apter.py in _process_tensorlike(inputs)
   1011     return x
   1012
-> 1013   inputs = nest.map_structure(_convert_numpy_and_scipy, inputs)
   1014   return nest._list_to_tuple(inputs)  # pylint: disable=protected-acc
ess
   1015

/usr/local/lib/python3.6/dist-packages/tensorflow/python/util/nest.py in map_
structure(func, *structure, **kwargs)
    615
    616   return pack_sequence_as(
--> 617       structure[0], [func(*x) for x in entries],
```

```
    618       expand_composites=expand_composites)
    619
```

/usr/local/lib/python3.6/dist-packages/tensorflow/python/util/nest.py in <listcomp>(.0)

```
    615
    616   return pack_sequence_as(
--> 617       structure[0], [func(*x) for x in entries],
    618       expand_composites=expand_composites)
    619
```

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/data_adapter.py in _convert_numpy_and_scipy(x)

```
   1006       if issubclass(x.dtype.type, np.floating):
   1007         dtype = backend.floatx()
-> 1008       return ops.convert_to_tensor(x, dtype=dtype)
   1009     elif scipy_sparse and scipy_sparse.issparse(x):
   1010       return _scipy_sparse_to_sparse_tensor(x)
```

/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/ops.py in convert_to_tensor(value, dtype, name, as_ref, preferred_dtype, dtype_hint, ctx, accepted_result_types)

```
   1339
   1340     if ret is None:
-> 1341       ret = conversion_func(value, dtype=dtype, name=name, as_ref=as_ref)
   1342
   1343     if ret is NotImplemented:
```

/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/tensor_conversion_registry.py in _default_conversion_function(***failed resolving arguments***)

```
     50 def _default_conversion_function(value, dtype, name, as_ref):
     51   del as_ref  # Unused.
---> 52   return constant_op.constant(value, dtype, name=name)
     53
     54
```

/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/constant_op.py in constant(value, dtype, shape, name)

```
    260   """
    261   return _constant_impl(value, dtype, shape, name, verify_shape=False,
--> 262                         allow_broadcast=True)
    263
    264
```

/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/constant_op.py in _constant_impl(value, dtype, shape, name, verify_shape, allow_broadcast)

```
    268   ctx = context.context()
    269   if ctx.executing_eagerly():
--> 270     t = convert_to_eager_tensor(value, ctx, dtype)
    271     if shape is None:
    272       return t
```

/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/constant_o

```
p.py in convert_to_eager_tensor(value, ctx, dtype)
     94       dtype = dtypes.as_dtype(dtype).as_datatype_enum
     95   ctx.ensure_initialized()
---> 96   return ops.EagerTensor(value, ctx.device_name, dtype)
     97
     98

ValueError: Failed to convert a NumPy array to a Tensor (Unsupported object t
ype float).
```

In [0]:
```
# untransform the predictions back to actual scale
predictions = scaler.inverse_transform(predictions)
```

In [0]:
```
# plot prediction versus actual
plt.figure(figsize=(10,6))
plt.plot(apple_testing_processed, color='blue', label='Actual Apple Stock Pric
e')
plt.plot(predictions , color='red', label='Predicted Apple Stock Price')
plt.title('Apple Stock Price Prediction')
plt.xlabel('Date')
plt.ylabel('Apple Stock Price')
plt.legend()
plt.show()
```

Problem 4

In [0]:
```

```