Appendix A: Predicting Bitcoin Prices with LSTM

This notebook uses the Keras wrapper around TensorFlow to demonstrate how a Long-Short Term Memory neural network can be used to effectively predict cryptocurrency prices. Predictions are based on daily (close) historic Bitcoin prices; a positive/negative semantic score computed from tweets by cryptocurrency experts; daily historic Economic Policy Uncertainty Index data; and Bitcoin market data including daily volume, bid-ask spread, and total transactions and transactions per minute. We explored numerous references for coding Long-Short Term Memory networks. One of the most useful guides is Jason Brownlee's [*"Machine Learning Mastery"* (https://machinelearningmastery.com/use-features-lstm-networks-time-series-forecasting/)](https://machinelearningmastery.com/use-features-lstm-networks-time-series-forecasting/) series, and we have adapted some of his code to produce our models.

In [1]:

```
import pandas as pd
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from math import sqrt
import matplotlib.pyplot as plt
import numpy
from numpy import concatenate
import seaborn as sns
import matplotlib.pyplot as plt
```

```
/Users/tjd/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36: F
utureWarning: Conversion of the second argument of issubdtype from `flo
at` to `np.floating` is deprecated. In future, it will be treated as `n
p.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

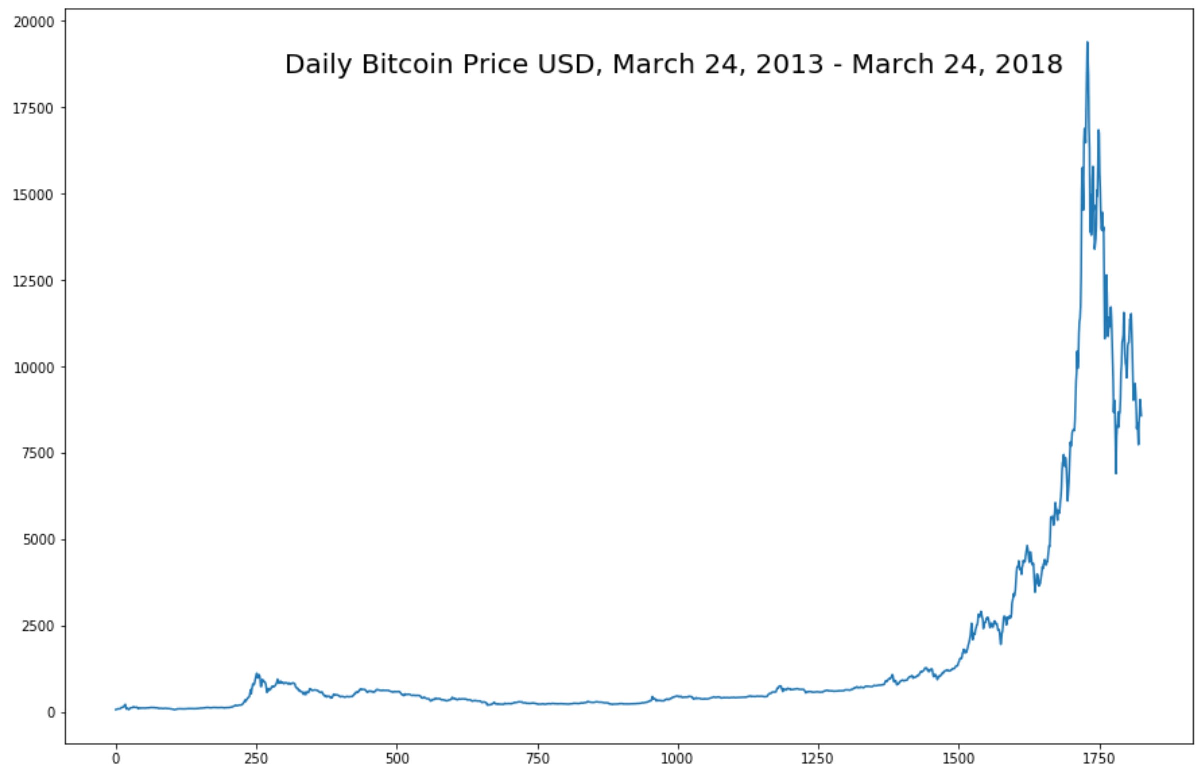
In [2]:

```
# plot size
plt.rcParams["figure.figsize"] = (15,10)
```

Part 1: Data preparation

Bitcoin price, volume, total transactions, trades per minute, bidask, the Economic Policy Uncertainty Index, and semantic (tweet) scores were collected separately (see other appendices) and are loaded from a csv file here. The data cover 1,826 days of trading from March 24, 2013, through March 24, 2018.

In [3]:
```
# get the data and view price timeline
fname='btc_new_3_24.csv'
df = pd.read_csv(fname)
df['price'].plot()
plt.text(300, 18500, "Daily Bitcoin Price USD, March 24, 2013 - March 2
4, 2018", fontsize=20)
# plt.title("Daily Bitcoin Price USD, March 24, 2013 - March 24, 2018",
 fontsize=18)
plt.show()
```



Function deck

These are some of the functions [Brownlee (https://machinelearningmastery.com/use-features-lstm-networks-time-series-forecasting/)](https://machinelearningmastery.com/use-features-lstm-networks-time-series-forecasting/) devised to transform data and wrap around the LSTM modeling code. We are using adapted versions for our models. Similar functions are available in pandas and Keras. Note wrappers for 3 models.

In [4]:
```
# create a differenced series
def difference(dataset, interval=1):
        diff = list()
        for i in range(interval, len(dataset)):
                value = dataset[i] - dataset[i - interval]
                diff.append(value)
        return pd.Series(diff)

# invert differenced value
def inverse_difference(history, yhat, interval=1):
        return yhat + history[-interval]

# frame a sequence as a supervised learning problem; NOT USED
def timeseries_to_supervised(data, lag=1):
        df = pd.DataFrame(data)
        columns = [df.shift(i) for i in range(1, lag+1)]
        columns.append(df)
        df = pd.concat(columns, axis=1)
        return df

# scale train and test data to [-1, 1]
def scale(train, test):
        # fit scaler
        scaler = MinMaxScaler(feature_range=(-1, 1))
        scaler = scaler.fit(train)
        # transform train
        train = train.reshape(train.shape[0], train.shape[1])
        train_scaled = scaler.transform(train)
        # transform test
        test = test.reshape(test.shape[0], test.shape[1])
        test_scaled = scaler.transform(test)
        return scaler, train_scaled, test_scaled

# inverse scaling for a forecasted value
def invert_scale(scaler, X, yhat):
        new_row = [x for x in X] + [yhat]
        array = numpy.array(new_row)
        array = array.reshape(1, len(array))
        inverted = scaler.inverse_transform(array)
        return inverted[0, -1]

# fit an LSTM network to training data
# this base model uses hidden layers of 100, 20, 10 and 5 neurons
# the function can be easily modified to change layers
# or use differently shaped input data
def fit_lstm(train, batch_size, nb_epoch, neurons):
    X, y = train[:, 0:-1], train[:, -1]
    X = X.reshape(X.shape[0], 1, X.shape[1])
    model = Sequential()
    model.add(LSTM(neurons, return_sequences=True, batch_input_shape=(ba
tch_size, X.shape[1], X.shape[2])))
    model.add(LSTM(20, return_sequences=True))
    model.add(LSTM(10, return_sequences=True))
    model.add(LSTM(5))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
```

```
    for i in range(nb_epoch):
        model.fit(X, y, epochs=1, batch_size=batch_size, verbose=2, shuf
fle=False)
        model.reset_states()
    return model

# model 2 uses fewer neurons in hidden layers
def fit2_lstm(train, batch_size, nb_epoch, neurons):
    X, y = train[:, 0:-1], train[:, -1]
    X = X.reshape(X.shape[0], 1, X.shape[1])
    model = Sequential()
    model.add(LSTM(neurons, return_sequences=True, batch_input_shape=(ba
tch_size, X.shape[1], X.shape[2])))
    model.add(LSTM(10, return_sequences=True))
    model.add(LSTM(5, return_sequences=True))
    model.add(LSTM(3))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    for i in range(nb_epoch):
        model.fit(X, y, epochs=1, batch_size=batch_size, verbose=2, shuf
fle=False)
        model.reset_states()
    return model

# model 3, takes 200 neurons and then 100
def fit3_lstm(train, batch_size, nb_epoch, neurons):
    X, y = train[:, 0:-1], train[:, -1]
    X = X.reshape(X.shape[0], 1, X.shape[1])
    model = Sequential()
    model.add(LSTM(neurons, return_sequences=True, batch_input_shape=(ba
tch_size, X.shape[1], X.shape[2])))
    model.add(LSTM(100))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    for i in range(nb_epoch):
        model.fit(X, y, epochs=1, batch_size=batch_size, verbose=2, shuf
fle=False)
        model.reset_states()
    return model

# model 4, takes a single neuron over 500 epochs
# fit an LSTM network to training data
def fit4_lstm(train, batch_size, nb_epoch, neurons):
        X, y = train[:, 0:-1], train[:, -1]
        X = X.reshape(X.shape[0], 1, X.shape[1])
        model = Sequential()
        model.add(LSTM(neurons, batch_input_shape=(batch_size, X.shape[1
], X.shape[2]), stateful=True))
        model.add(Dense(1))
        model.compile(loss='mean_squared_error', optimizer='adam')
        for i in range(nb_epoch):
                model.fit(X, y, epochs=1, batch_size=batch_size, verbose
=0, shuffle=False)
                model.reset_states()
        return model

# model 5, deeper network, more epochs
```

```
def fit5_lstm(train, batch_size, nb_epoch, neurons):
    X, y = train[:, 0:-1], train[:, -1]
    X = X.reshape(X.shape[0], 1, X.shape[1])
    model = Sequential()
    model.add(LSTM(neurons, return_sequences=True, batch_input_shape=(ba
tch_size, X.shape[1], X.shape[2])))
    model.add(LSTM(50, return_sequences=True))
    model.add(LSTM(30, return_sequences=True))
    model.add(LSTM(20, return_sequences=True))
    model.add(LSTM(10, return_sequences=True))
    model.add(LSTM(5))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    for i in range(nb_epoch):
        model.fit(X, y, epochs=1, batch_size=batch_size, verbose=2, shuf
fle=False)
        model.reset_states()
    return model

# make a one-step forecast
def forecast_lstm(model, batch_size, X):
        X = X.reshape(1, 1, len(X))
        yhat = model.predict(X, batch_size=batch_size)
        return yhat[0,0]

# this function runs the lstm model for the repeat experiment function b
elow
# I've commented out the data prep part, since we've done that already
# run a repeated experiment
def experiment(n_repeats, batch_size=1, n_epochs=1, n_neurons=1):
        error_scores = list()
        for r in range(n_repeats):
                # fit the base model
                lstm_model = fit_lstm(train_scaled, batch_size, n_epochs
, n_neurons)
                # forecast test dataset
                predictions = list()
                for i in range(len(test_scaled)):
                        # predict
                        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
                        yhat = forecast_lstm(lstm_model, 1, X)
                        # invert scaling
                        yhat = invert_scale(scaler, X, yhat)
                        # invert differencing
                        yhat = inverse_difference(raw_values, yhat, len(
test_scaled)+1-i)
                        # store forecast
                        predictions.append(yhat)
                # report performance
                rmse = sqrt(mean_squared_error(raw_values[-600:], predic
tions))
                print('')
                print('%d) Test RMSE: %.3f' % (r+1, rmse))
                print('')
                error_scores.append(rmse)
        return error_scores
```

```
# this function can be used to configure and run multiple trials of a mo
del
# default is set to 2 repeats for a batch size of one with 10 epochs and
 100 neurons
def run():
        n_repeats = 2
        batch_size = 1
        n_epochs = 10
        n_neurons = 100
        # run the experiment
        results = pd.DataFrame()
        results['results'] = experiment(n_repeats, batch_size, n_epochs,
 n_neurons)
        # summarize results
        print(results.describe())
        # save results
        results.to_csv('experiment.csv', index=False)
    # save boxplot
        results.boxplot()
        plt.show()

# to run it
# run()
```

In [5]:
```
df.head()
```

Out[5]:

	day	epu_idx	price	volume	bidask	tpm	trans	exp_sem
0	3/24/13	126.08	68.287939	3.361108e+06	0.761886	2.203646	51335	0.24065
1	3/25/13	178.57	73.648607	6.917269e+06	0.978241	2.824132	48993	0.20230
2	3/26/13	163.29	77.090928	5.322638e+06	0.931310	2.560417	49061	0.00000
3	3/27/13	177.06	85.256510	7.356470e+06	1.033937	3.198785	53207	0.21230
4	3/28/13	72.92	90.658741	1.477020e+07	1.555560	5.049826	60989	0.20240

Difference the data

Several variables are not stationary (please refer to stationarity tests in Appendix C). For LSTM to work, they have to be differenced. We do that for transactions, bid-ask spread, volume and trades per minute. We also will difference price in a subsequent step.

In [6]:
```
# temporarily split off the variables to be differenced
df_temp = df[['trans', 'bidask', 'volume', 'tpm']].diff(1,0)
```

In [7]:
```
# concatenate the data, rename columns
new_df = pd.DataFrame(pd.concat([df_temp, df[['price','epu_idx', 'exp_se
m']]], axis=1))
```

In [8]:
```
# note that differenced data is NaN in row 0, as it should be
new_df.head()
```

Out[8]:

	trans	bidask	volume	tpm	price	epu_idx	exp_sem
0	NaN	NaN	NaN	NaN	68.287939	126.08	0.24065
1	-2342.0	0.216354	3556161.550	0.620486	73.648607	178.57	0.20230
2	68.0	-0.046930	-1594630.877	-0.263715	77.090928	163.29	0.00000
3	4146.0	0.102627	2033831.837	0.638368	85.256510	177.06	0.21230
4	7782.0	0.521623	7413732.351	1.851042	90.658741	72.92	0.20240

Create the 1-day lagged price as target variable 'y'

We are trying to predict the next day's price based on the data we know today. The next series of steps adds our features together with y = t+1. So now we have the next day's true price (y) and today's knowledge (X) in each observation.

Step 1: Difference the price

In [9]:
```
# first we have to difference the price
# we're using Brownlee's function to do this so we can undifference it l
ater
raw_values = new_df['price'].values
diff_values = difference(raw_values, 1)
diff_values.head()
```

Out[9]:
```
0    5.360668
1    3.442321
2    8.165582
3    5.402231
4   -0.559414
dtype: float64
```

In [10]:
```
len(diff_values)
```

Out[10]:
```
1825
```

In [11]:
```
# check it
new_df['price'].diff(1).head()
```

Out[11]:
```
0         NaN
1    5.360668
2    3.442321
3    8.165582
4    5.402231
Name: price, dtype: float64
```

Step 2: Create our lagged ahead target price 'y'

In [12]:
```
# transform data to be supervised learning
supervised = timeseries_to_supervised(diff_values, 1)
```

In [13]:
```
supervised[0:6]
```

Out[13]:

	0	0
0	NaN	5.360668
1	5.360668	3.442321
2	3.442321	8.165582
3	8.165582	5.402231
4	5.402231	-0.559414
5	-0.559414	1.400072

Step 3: Combine with our other features

In [14]:
```
# note that data differenced data has NaN at row 0 as expected
new_df.head(5)
```

Out[14]:

	trans	bidask	volume	tpm	price	epu_idx	exp_sem
0	NaN	NaN	NaN	NaN	68.287939	126.08	0.24065
1	-2342.0	0.216354	3556161.550	0.620486	73.648607	178.57	0.20230
2	68.0	-0.046930	-1594630.877	-0.263715	77.090928	163.29	0.00000
3	4146.0	0.102627	2033831.837	0.638368	85.256510	177.06	0.21230
4	7782.0	0.521623	7413732.351	1.851042	90.658741	72.92	0.20240

In [15]:
```
# concatenate the data, rename columns
df2 = pd.DataFrame(pd.concat([new_df, supervised], axis=1))
```

In [16]:
```
df2.columns.values[7] = "d-price"
df2.columns.values[8] = "y"
```

```
In [17]: df2.head()
```

Out[17]:

	trans	bidask	volume	tpm	price	epu_idx	exp_sem	d-price	
0	NaN	NaN	NaN	NaN	68.287939	126.08	0.24065	NaN	5.
1	-2342.0	0.216354	3556161.550	0.620486	73.648607	178.57	0.20230	5.360668	3.
2	68.0	-0.046930	-1594630.877	-0.263715	77.090928	163.29	0.00000	3.442321	8.
3	4146.0	0.102627	2033831.837	0.638368	85.256510	177.06	0.21230	8.165582	5.
4	7782.0	0.521623	7413732.351	1.851042	90.658741	72.92	0.20240	5.402231	-0

```
In [18]: df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1826 entries, 0 to 1825
Data columns (total 9 columns):
trans      1825 non-null float64
bidask     1825 non-null float64
volume     1825 non-null float64
tpm        1825 non-null float64
price      1826 non-null float64
epu_idx    1826 non-null float64
exp_sem    1826 non-null float64
d-price    1824 non-null float64
y          1825 non-null float64
dtypes: float64(9)
memory usage: 128.5 KB
```

```
In [19]: # delete old price column
         df2.drop(['price'], axis=1, inplace=True)
```

Step 4: Inspect

```
In [20]: # inspect; we should have seven features (X) and our target(y)
         df2.head()
```

Out[20]:

	trans	bidask	volume	tpm	epu_idx	exp_sem	d-price	y
0	NaN	NaN	NaN	NaN	126.08	0.24065	NaN	5.360668
1	-2342.0	0.216354	3556161.550	0.620486	178.57	0.20230	5.360668	3.442321
2	68.0	-0.046930	-1594630.877	-0.263715	163.29	0.00000	3.442321	8.165582
3	4146.0	0.102627	2033831.837	0.638368	177.06	0.21230	8.165582	5.402231
4	7782.0	0.521623	7413732.351	1.851042	72.92	0.20240	5.402231	-0.559414

In [21]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1826 entries, 0 to 1825
Data columns (total 8 columns):
trans       1825 non-null float64
bidask      1825 non-null float64
volume      1825 non-null float64
tpm         1825 non-null float64
epu_idx     1826 non-null float64
exp_sem     1826 non-null float64
d-price     1824 non-null float64
y           1825 non-null float64
dtypes: float64(8)
memory usage: 114.2 KB
```

In [22]:
```
# we must drop any NaN rows for LSTM to run
df2.dropna(inplace=True)
```

In [23]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1824 entries, 1 to 1824
Data columns (total 8 columns):
trans       1824 non-null float64
bidask      1824 non-null float64
volume      1824 non-null float64
tpm         1824 non-null float64
epu_idx     1824 non-null float64
exp_sem     1824 non-null float64
d-price     1824 non-null float64
y           1824 non-null float64
dtypes: float64(8)
memory usage: 128.2 KB
```

In [24]:
```
df2.head()
```

Out[24]:

	trans	bidask	volume	tpm	epu_idx	exp_sem	d-price	y
1	-2342.0	0.216354	3556161.550	0.620486	178.57	0.202300	5.360668	3.442321
2	68.0	-0.046930	-1594630.877	-0.263715	163.29	0.000000	3.442321	8.165582
3	4146.0	0.102627	2033831.837	0.638368	177.06	0.212300	8.165582	5.402231
4	7782.0	0.521623	7413732.351	1.851042	72.92	0.202400	5.402231	-0.559414
5	1270.0	-0.430127	-5853453.366	-1.613368	90.71	0.240067	-0.559414	1.400072

In [25]:
```
len(df2)
```

Out[25]: 1824

Convert to an array

We now have a completed dataset with a lagged y to be predicted from other variables in each row. The next step is to convert the pandas dataframe to an array. This is required by LSTM.

In [26]:
```
supervised_values = df2.values
supervised_values[0:2]
```

Out[26]:
```
array([[-2.34200000e+03,  2.16354374e-01,  3.55616155e+06,
         6.20486111e-01,  1.78570000e+02,  2.02300000e-01,
         5.36066752e+00,  3.44232118e+00],
       [ 6.80000000e+01, -4.69303670e-02, -1.59463088e+06,
        -2.63715277e-01,  1.63290000e+02,  0.00000000e+00,
         3.44232118e+00,  8.16558203e+00]])
```

In [27]:
```
len(supervised_values)
```

Out[27]:
```
1824
```

Make training, test sets; scale the data

We have 1,824 observations. We split the data into a training set of 1,224 rows and test set of 600 and scale each set between (-1,1) to improve model performance.

In [28]:
```
# split data into train and test-sets
train, test = supervised_values[0:-600, :], supervised_values[-600:, :]

# transform the scale of the data
scaler, train_scaled, test_scaled = scale(train, test)
```

In [29]:
```
len(train), len(test)
```

Out[29]:
```
(1224, 600)
```

Part 2: Modeling

At this stage, we've attempted five LSTM models of different configurations. All reported similar results.

Model 1

This is a single-batch model run over 10 epochs with hidden layers of 100, 20, 10 and 5 hidden layers. Generally, deeper layers provide better results.

In [30]:
```
numpy.random.seed(1234)
lstm_model = fit_lstm(train_scaled, 1, 10, 100)
```

```
Epoch 1/1
 - 13s - loss: 0.0133
Epoch 1/1
 - 13s - loss: 0.0117
Epoch 1/1
 - 13s - loss: 0.0116
Epoch 1/1
 - 13s - loss: 0.0115
Epoch 1/1
 - 13s - loss: 0.0114
Epoch 1/1
 - 13s - loss: 0.0113
Epoch 1/1
 - 14s - loss: 0.0112
Epoch 1/1
 - 13s - loss: 0.0111
Epoch 1/1
 - 13s - loss: 0.0111
Epoch 1/1
 - 14s - loss: 0.0110
```

Predictions on test data

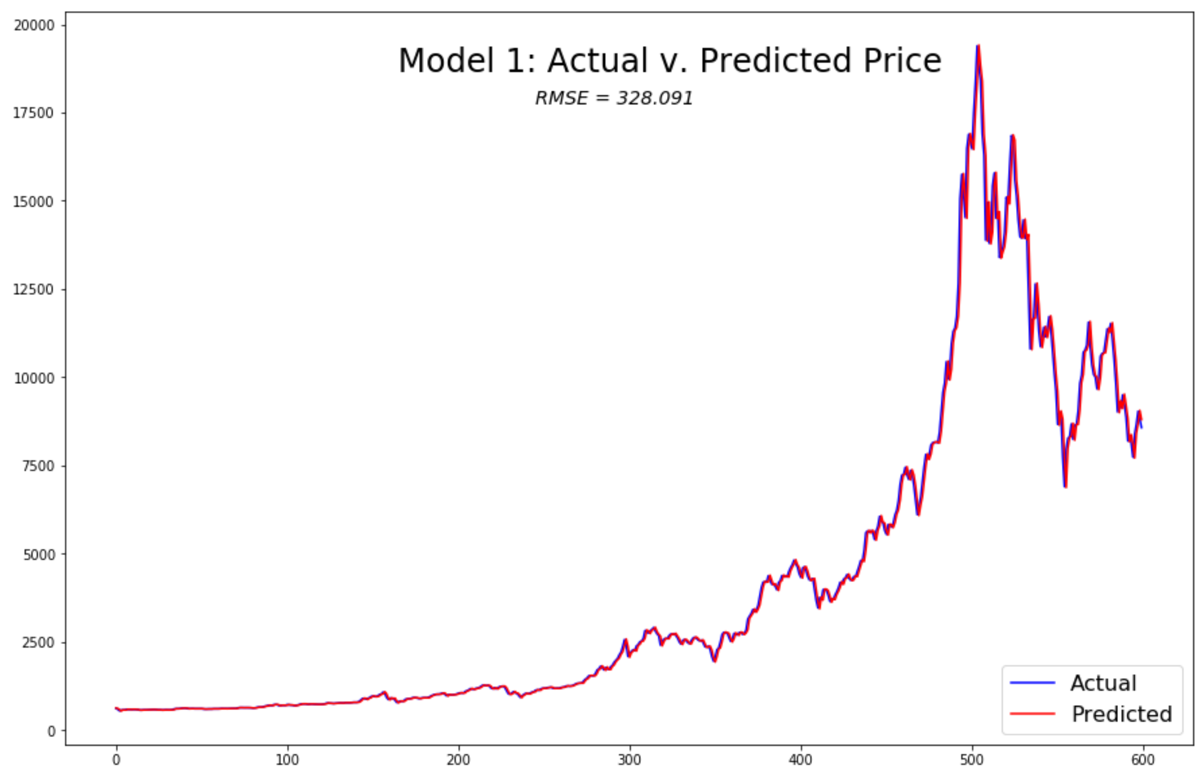
In [31]:
```
predictions = list()
for i in range(len(test_scaled)):
        # predict
        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
        yhat = forecast_lstm(lstm_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)
        # invert differencing
        yhat = inverse_difference(raw_values, yhat, len(test_scaled)+1-i
)
        # store forecast
        predictions.append(yhat)
# report performance
rmse = sqrt(mean_squared_error(raw_values[-600:], predictions))
print('%d) Test RMSE: %.3f' % (1 ,rmse))
```

```
1) Test RMSE: 328.091
```

In [32]:
```
lstm_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_1 (LSTM)                (1, 1, 100)               43200     
_________________________________________________________________
lstm_2 (LSTM)                (1, 1, 20)                9680      
_________________________________________________________________
lstm_3 (LSTM)                (1, 1, 10)                1240      
_________________________________________________________________
lstm_4 (LSTM)                (1, 5)                    320       
_________________________________________________________________
dense_1 (Dense)              (1, 1)                    6         
=================================================================
Total params: 54,446
Trainable params: 54,446
Non-trainable params: 0
_________________________________________________________________
```

In [40]:
```
#Plot actual versus predicted:
orig = plt.plot(raw_values[-600:], color='blue',label='Actual')
pred = plt.plot(predictions, color='red', label='Predicted')
plt.legend(loc=4, fontsize=16)
plt.text(165, 18650, 'Model 1: Actual v. Predicted Price', fontsize=24)
plt.text(245, 17750, 'RMSE = 328.091', fontsize=14, style='italic' )
plt.show()
```



In [37]:

```
# save model
# serialize model to JSON
from keras.models import model_from_json
lstm_model_1 = lstm_model.to_json()
with open("lstm_model_1", "w") as json_file:
    json_file.write(lstm_model_1)
# serialize weights to HDF5
lstm_model.save_weights("model_1_weights.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

Model 2

This is a single-batch model run over 200 epochs with 50, 10, 5 and 3 hidden layers. This model takes appox. 40 minutes to complete.

In [38]:
```
numpy.random.seed(1234)
lstm_model = fit2_lstm(train_scaled, 1, 200, 50)
```

```
Epoch 1/1
 - 12s - loss: 0.0136
Epoch 1/1
 - 12s - loss: 0.0116
Epoch 1/1
 - 12s - loss: 0.0115
Epoch 1/1
 - 12s - loss: 0.0114
Epoch 1/1
 - 12s - loss: 0.0113
Epoch 1/1
 - 12s - loss: 0.0112
Epoch 1/1
 - 12s - loss: 0.0111
Epoch 1/1
 - 13s - loss: 0.0111
Epoch 1/1
 - 12s - loss: 0.0110
Epoch 1/1
 - 13s - loss: 0.0110
Epoch 1/1
 - 13s - loss: 0.0109
Epoch 1/1
 - 13s - loss: 0.0109
Epoch 1/1
 - 13s - loss: 0.0109
Epoch 1/1
 - 229s - loss: 0.0109
Epoch 1/1
 - 12s - loss: 0.0109
Epoch 1/1
 - 12s - loss: 0.0108
Epoch 1/1
 - 12s - loss: 0.0108
Epoch 1/1
 - 12s - loss: 0.0108
Epoch 1/1
 - 12s - loss: 0.0108
Epoch 1/1
 - 13s - loss: 0.0108
Epoch 1/1
 - 13s - loss: 0.0108
Epoch 1/1
 - 13s - loss: 0.0108
Epoch 1/1
 - 13s - loss: 0.0108
Epoch 1/1
 - 13s - loss: 0.0108
Epoch 1/1
 - 13s - loss: 0.0108
Epoch 1/1
 - 13s - loss: 0.0108
Epoch 1/1
 - 13s - loss: 0.0108
Epoch 1/1
 - 13s - loss: 0.0107
Epoch 1/1
```

```
 - 1055s - loss: 0.0107
Epoch 1/1
 - 12s - loss: 0.0107
Epoch 1/1
 - 12s - loss: 0.0107
Epoch 1/1
 - 12s - loss: 0.0106
Epoch 1/1
 - 12s - loss: 0.0106
Epoch 1/1
 - 12s - loss: 0.0106
Epoch 1/1
 - 12s - loss: 0.0105
Epoch 1/1
 - 12s - loss: 0.0105
Epoch 1/1
 - 12s - loss: 0.0104
Epoch 1/1
 - 12s - loss: 0.0104
Epoch 1/1
 - 12s - loss: 0.0103
Epoch 1/1
 - 12s - loss: 0.0102
Epoch 1/1
 - 12s - loss: 0.0101
Epoch 1/1
 - 12s - loss: 0.0100
Epoch 1/1
 - 12s - loss: 0.0100
Epoch 1/1
 - 901s - loss: 0.0099
Epoch 1/1
 - 12s - loss: 0.0098
Epoch 1/1
 - 12s - loss: 0.0097
Epoch 1/1
 - 12s - loss: 0.0096
Epoch 1/1
 - 12s - loss: 0.0095
Epoch 1/1
 - 12s - loss: 0.0094
Epoch 1/1
 - 12s - loss: 0.0093
Epoch 1/1
 - 12s - loss: 0.0092
Epoch 1/1
 - 13s - loss: 0.0091
Epoch 1/1
 - 12s - loss: 0.0091
Epoch 1/1
 - 12s - loss: 0.0089
Epoch 1/1
 - 13s - loss: 0.0088
Epoch 1/1
 - 12s - loss: 0.0088
Epoch 1/1
 - 12s - loss: 0.0087
```

```
Epoch 1/1
 - 13s - loss: 0.0085
Epoch 1/1
 - 13s - loss: 0.0085
Epoch 1/1
 - 12s - loss: 0.0084
Epoch 1/1
 - 12s - loss: 0.0083
Epoch 1/1
 - 12s - loss: 0.0082
Epoch 1/1
 - 12s - loss: 0.0082
Epoch 1/1
 - 12s - loss: 0.0082
Epoch 1/1
 - 12s - loss: 0.0081
Epoch 1/1
 - 12s - loss: 0.0080
Epoch 1/1
 - 12s - loss: 0.0079
Epoch 1/1
 - 12s - loss: 0.0080
Epoch 1/1
 - 12s - loss: 0.0078
Epoch 1/1
 - 12s - loss: 0.0078
Epoch 1/1
 - 12s - loss: 0.0078
Epoch 1/1
 - 12s - loss: 0.0076
Epoch 1/1
 - 12s - loss: 0.0076
Epoch 1/1
 - 12s - loss: 0.0077
Epoch 1/1
 - 12s - loss: 0.0073
Epoch 1/1
 - 13s - loss: 0.0074
Epoch 1/1
 - 13s - loss: 0.0075
Epoch 1/1
 - 12s - loss: 0.0071
Epoch 1/1
 - 13s - loss: 0.0071
Epoch 1/1
 - 12s - loss: 0.0071
Epoch 1/1
 - 840s - loss: 0.0070
Epoch 1/1
 - 12s - loss: 0.0069
Epoch 1/1
 - 12s - loss: 0.0068
Epoch 1/1
 - 14s - loss: 0.0068
Epoch 1/1
 - 13s - loss: 0.0068
Epoch 1/1
```

```
 - 12s - loss: 0.0067
Epoch 1/1
 - 12s - loss: 0.0070
Epoch 1/1
 - 12s - loss: 0.0066
Epoch 1/1
 - 13s - loss: 0.0066
Epoch 1/1
 - 13s - loss: 0.0068
Epoch 1/1
 - 13s - loss: 0.0064
Epoch 1/1
 - 13s - loss: 0.0065
Epoch 1/1
 - 13s - loss: 0.0065
Epoch 1/1
 - 13s - loss: 0.0064
Epoch 1/1
 - 13s - loss: 0.0065
Epoch 1/1
 - 13s - loss: 0.0063
Epoch 1/1
 - 13s - loss: 0.0064
Epoch 1/1
 - 13s - loss: 0.0063
Epoch 1/1
 - 13s - loss: 0.0063
Epoch 1/1
 - 13s - loss: 0.0063
Epoch 1/1
 - 13s - loss: 0.0062
Epoch 1/1
 - 13s - loss: 0.0062
Epoch 1/1
 - 13s - loss: 0.0061
Epoch 1/1
 - 13s - loss: 0.0062
Epoch 1/1
 - 13s - loss: 0.0062
Epoch 1/1
 - 13s - loss: 0.0061
Epoch 1/1
 - 13s - loss: 0.0061
Epoch 1/1
 - 13s - loss: 0.0060
Epoch 1/1
 - 13s - loss: 0.0060
Epoch 1/1
 - 13s - loss: 0.0060
Epoch 1/1
 - 13s - loss: 0.0059
Epoch 1/1
 - 13s - loss: 0.0059
Epoch 1/1
 - 13s - loss: 0.0059
Epoch 1/1
 - 13s - loss: 0.0058
```

```
Epoch 1/1
 - 13s - loss: 0.0058
Epoch 1/1
 - 13s - loss: 0.0059
Epoch 1/1
 - 13s - loss: 0.0057
Epoch 1/1
 - 13s - loss: 0.0057
Epoch 1/1
 - 13s - loss: 0.0057
Epoch 1/1
 - 13s - loss: 0.0057
Epoch 1/1
 - 13s - loss: 0.0056
Epoch 1/1
 - 13s - loss: 0.0056
Epoch 1/1
 - 13s - loss: 0.0056
Epoch 1/1
 - 13s - loss: 0.0054
Epoch 1/1
 - 13s - loss: 0.0056
Epoch 1/1
 - 13s - loss: 0.0054
Epoch 1/1
 - 13s - loss: 0.0054
Epoch 1/1
 - 13s - loss: 0.0056
Epoch 1/1
 - 12s - loss: 0.0055
Epoch 1/1
 - 13s - loss: 0.0053
Epoch 1/1
 - 13s - loss: 0.0055
Epoch 1/1
 - 13s - loss: 0.0053
Epoch 1/1
 - 13s - loss: 0.0054
Epoch 1/1
 - 13s - loss: 0.0052
Epoch 1/1
 - 13s - loss: 0.0054
Epoch 1/1
 - 13s - loss: 0.0052
Epoch 1/1
 - 12s - loss: 0.0052
Epoch 1/1
 - 13s - loss: 0.0052
Epoch 1/1
 - 13s - loss: 0.0052
Epoch 1/1
 - 13s - loss: 0.0055
Epoch 1/1
 - 13s - loss: 0.0053
Epoch 1/1
 - 13s - loss: 0.0052
Epoch 1/1
```

```
 - 13s - loss: 0.0051
Epoch 1/1
 - 13s - loss: 0.0051
Epoch 1/1
 - 13s - loss: 0.0052
Epoch 1/1
 - 13s - loss: 0.0051
Epoch 1/1
 - 13s - loss: 0.0050
Epoch 1/1
 - 13s - loss: 0.0050
Epoch 1/1
 - 13s - loss: 0.0050
Epoch 1/1
 - 12s - loss: 0.0051
Epoch 1/1
 - 12s - loss: 0.0051
Epoch 1/1
 - 13s - loss: 0.0050
Epoch 1/1
 - 13s - loss: 0.0050
Epoch 1/1
 - 13s - loss: 0.0049
Epoch 1/1
 - 12s - loss: 0.0048
Epoch 1/1
 - 13s - loss: 0.0048
Epoch 1/1
 - 13s - loss: 0.0050
Epoch 1/1
 - 12s - loss: 0.0050
Epoch 1/1
 - 13s - loss: 0.0047
Epoch 1/1
 - 13s - loss: 0.0046
Epoch 1/1
 - 13s - loss: 0.0048
Epoch 1/1
 - 12s - loss: 0.0046
Epoch 1/1
 - 12s - loss: 0.0050
Epoch 1/1
 - 12s - loss: 0.0049
Epoch 1/1
 - 12s - loss: 0.0045
Epoch 1/1
 - 12s - loss: 0.0045
Epoch 1/1
 - 13s - loss: 0.0047
Epoch 1/1
 - 12s - loss: 0.0045
Epoch 1/1
 - 12s - loss: 0.0046
Epoch 1/1
 - 12s - loss: 0.0045
Epoch 1/1
 - 12s - loss: 0.0046
```

```
Epoch 1/1
 - 12s - loss: 0.0045
Epoch 1/1
 - 12s - loss: 0.0045
Epoch 1/1
 - 12s - loss: 0.0045
Epoch 1/1
 - 12s - loss: 0.0044
Epoch 1/1
 - 13s - loss: 0.0044
Epoch 1/1
 - 13s - loss: 0.0045
Epoch 1/1
 - 13s - loss: 0.0044
Epoch 1/1
 - 12s - loss: 0.0044
Epoch 1/1
 - 13s - loss: 0.0042
Epoch 1/1
 - 12s - loss: 0.0044
Epoch 1/1
 - 12s - loss: 0.0042
Epoch 1/1
 - 13s - loss: 0.0042
Epoch 1/1
 - 13s - loss: 0.0041
Epoch 1/1
 - 13s - loss: 0.0044
Epoch 1/1
 - 13s - loss: 0.0042
Epoch 1/1
 - 13s - loss: 0.0043
Epoch 1/1
 - 12s - loss: 0.0041
Epoch 1/1
 - 12s - loss: 0.0042
Epoch 1/1
 - 12s - loss: 0.0042
Epoch 1/1
 - 12s - loss: 0.0040
Epoch 1/1
 - 12s - loss: 0.0041
Epoch 1/1
 - 12s - loss: 0.0041
Epoch 1/1
 - 12s - loss: 0.0040
Epoch 1/1
 - 13s - loss: 0.0042
Epoch 1/1
 - 12s - loss: 0.0040
Epoch 1/1
 - 12s - loss: 0.0040
Epoch 1/1
 - 12s - loss: 0.0041
Epoch 1/1
 - 12s - loss: 0.0040
```

```
Epoch 1/1
 - 12s - loss: 0.0039
```

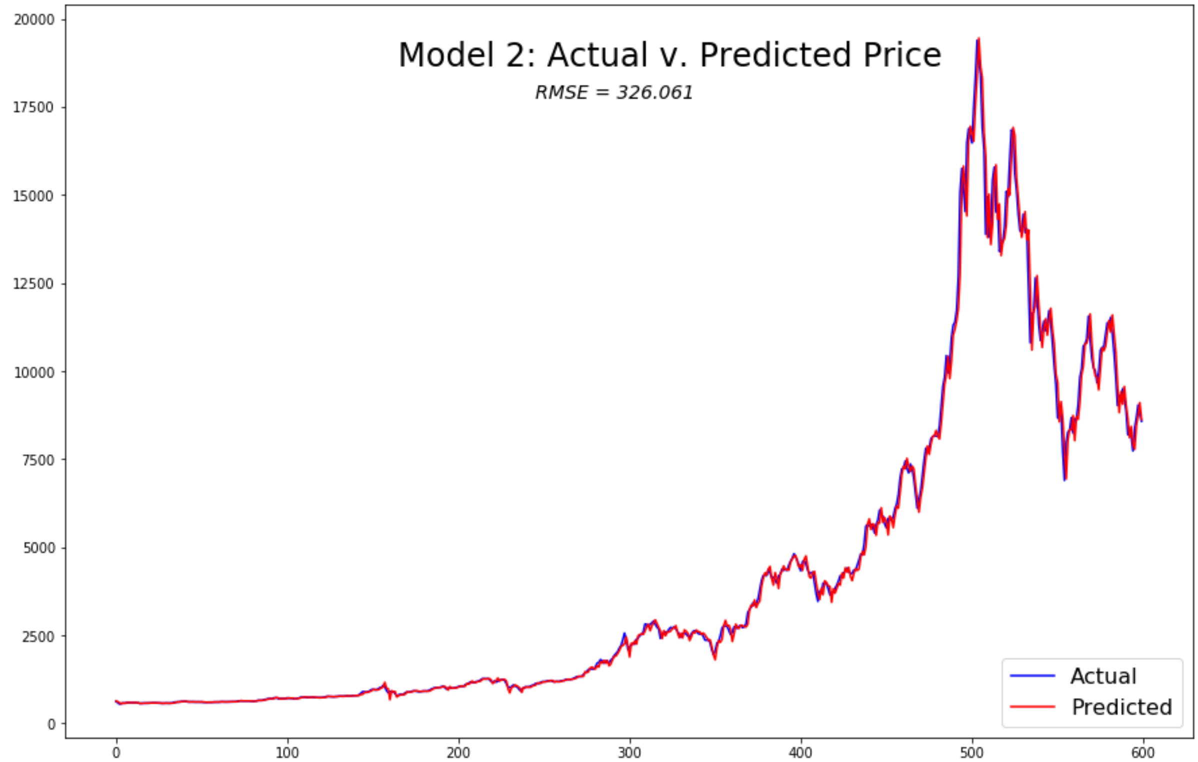
In [39]:
```
predictions_2 = list()
for i in range(len(test_scaled)):
        # predict
        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
        yhat = forecast_lstm(lstm_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)
        # invert differencing
        yhat = inverse_difference(raw_values, yhat, len(test_scaled)+1-i
)
        # store forecast
        predictions_2.append(yhat)
# report performance
rmse_2 = sqrt(mean_squared_error(raw_values[-600:], predictions_2))
print('%d) Test RMSE: %.3f' % (1 ,rmse_2))
```

```
1) Test RMSE: 326.061
```

In [41]:
```
lstm_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_5 (LSTM)                (1, 1, 50)                11600     
_________________________________________________________________
lstm_6 (LSTM)                (1, 1, 10)                2440      
_________________________________________________________________
lstm_7 (LSTM)                (1, 1, 5)                 320       
_________________________________________________________________
lstm_8 (LSTM)                (1, 3)                    108       
_________________________________________________________________
dense_2 (Dense)              (1, 1)                    4         
=================================================================
Total params: 14,472
Trainable params: 14,472
Non-trainable params: 0
_________________________________________________________________
```

In [42]:
```
#Plot actual versus predicted:
orig = plt.plot(raw_values[-600:], color='blue',label='Actual')
pred = plt.plot(predictions_2, color='red', label='Predicted')
plt.legend(loc=4, fontsize=16)
plt.text(165, 18650, 'Model 2: Actual v. Predicted Price', fontsize=24)
plt.text(245, 17750, 'RMSE = 326.061', fontsize=14, style='italic' )
plt.show()
```



In [43]:
```
# save model
# serialize model to JSON
from keras.models import model_from_json
lstm_model_2 = lstm_model.to_json()
with open("lstm_model_2", "w") as json_file:
    json_file.write(lstm_model_2)
# serialize weights to HDF5
lstm_model.save_weights("model_2_weights.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

Model 3

Model with 100 epochs and five hidden layers of 200, 10, 5, 3, and 1 neurons.

In [44]:
```
numpy.random.seed(1234)
lstm_model = fit3_lstm(train_scaled, 1, 100, 200)
```

```
Epoch 1/1
 - 9s - loss: 0.0125
Epoch 1/1
 - 8s - loss: 0.0119
Epoch 1/1
 - 8s - loss: 0.0117
Epoch 1/1
 - 9s - loss: 0.0117
Epoch 1/1
 - 8s - loss: 0.0116
Epoch 1/1
 - 8s - loss: 0.0116
Epoch 1/1
 - 9s - loss: 0.0115
Epoch 1/1
 - 9s - loss: 0.0114
Epoch 1/1
 - 9s - loss: 0.0114
Epoch 1/1
 - 9s - loss: 0.0113
Epoch 1/1
 - 9s - loss: 0.0113
Epoch 1/1
 - 9s - loss: 0.0113
Epoch 1/1
 - 9s - loss: 0.0112
Epoch 1/1
 - 9s - loss: 0.0112
Epoch 1/1
 - 9s - loss: 0.0112
Epoch 1/1
 - 9s - loss: 0.0111
Epoch 1/1
 - 9s - loss: 0.0111
Epoch 1/1
 - 9s - loss: 0.0111
Epoch 1/1
 - 9s - loss: 0.0111
Epoch 1/1
 - 9s - loss: 0.0110
Epoch 1/1
 - 8s - loss: 0.0110
Epoch 1/1
 - 8s - loss: 0.0110
Epoch 1/1
 - 9s - loss: 0.0110
Epoch 1/1
 - 9s - loss: 0.0110
Epoch 1/1
 - 9s - loss: 0.0110
Epoch 1/1
 - 9s - loss: 0.0109
Epoch 1/1
 - 9s - loss: 0.0109
Epoch 1/1
 - 9s - loss: 0.0109
Epoch 1/1
```

```
 - 9s - loss: 0.0109
Epoch 1/1
 - 9s - loss: 0.0109
Epoch 1/1
 - 9s - loss: 0.0109
Epoch 1/1
 - 9s - loss: 0.0108
Epoch 1/1
 - 9s - loss: 0.0108
Epoch 1/1
 - 9s - loss: 0.0108
Epoch 1/1
 - 9s - loss: 0.0108
Epoch 1/1
 - 9s - loss: 0.0107
Epoch 1/1
 - 9s - loss: 0.0107
Epoch 1/1
 - 9s - loss: 0.0106
Epoch 1/1
 - 9s - loss: 0.0106
Epoch 1/1
 - 9s - loss: 0.0105
Epoch 1/1
 - 9s - loss: 0.0104
Epoch 1/1
 - 9s - loss: 0.0103
Epoch 1/1
 - 9s - loss: 0.0102
Epoch 1/1
 - 9s - loss: 0.0101
Epoch 1/1
 - 9s - loss: 0.0098
Epoch 1/1
 - 9s - loss: 0.0096
Epoch 1/1
 - 9s - loss: 0.0093
Epoch 1/1
 - 9s - loss: 0.0090
Epoch 1/1
 - 8s - loss: 0.0088
Epoch 1/1
 - 9s - loss: 0.0086
Epoch 1/1
 - 9s - loss: 0.0085
Epoch 1/1
 - 9s - loss: 0.0086
Epoch 1/1
 - 9s - loss: 0.0086
Epoch 1/1
 - 9s - loss: 0.0085
Epoch 1/1
 - 9s - loss: 0.0083
Epoch 1/1
 - 9s - loss: 0.0084
Epoch 1/1
 - 9s - loss: 0.0084
```

```
Epoch 1/1
 - 9s - loss: 0.0082
Epoch 1/1
 - 9s - loss: 0.0080
Epoch 1/1
 - 9s - loss: 0.0083
Epoch 1/1
 - 9s - loss: 0.0082
Epoch 1/1
 - 9s - loss: 0.0081
Epoch 1/1
 - 9s - loss: 0.0080
Epoch 1/1
 - 9s - loss: 0.0082
Epoch 1/1
 - 9s - loss: 0.0081
Epoch 1/1
 - 9s - loss: 0.0080
Epoch 1/1
 - 9s - loss: 0.0078
Epoch 1/1
 - 9s - loss: 0.0081
Epoch 1/1
 - 8s - loss: 0.0079
Epoch 1/1
 - 9s - loss: 0.0081
Epoch 1/1
 - 9s - loss: 0.0082
Epoch 1/1
 - 9s - loss: 0.0078
Epoch 1/1
 - 9s - loss: 0.0078
Epoch 1/1
 - 9s - loss: 0.0079
Epoch 1/1
 - 9s - loss: 0.0080
Epoch 1/1
 - 8s - loss: 0.0078
Epoch 1/1
 - 9s - loss: 0.0077
Epoch 1/1
 - 9s - loss: 0.0079
Epoch 1/1
 - 9s - loss: 0.0077
Epoch 1/1
 - 8s - loss: 0.0078
Epoch 1/1
 - 9s - loss: 0.0078
Epoch 1/1
 - 9s - loss: 0.0079
Epoch 1/1
 - 9s - loss: 0.0078
Epoch 1/1
 - 8s - loss: 0.0077
Epoch 1/1
 - 9s - loss: 0.0077
Epoch 1/1
```

```
 - 9s - loss: 0.0078
Epoch 1/1
 - 8s - loss: 0.0078
Epoch 1/1
 - 9s - loss: 0.0076
Epoch 1/1
 - 955s - loss: 0.0078
Epoch 1/1
 - 16s - loss: 0.0076
Epoch 1/1
 - 9s - loss: 0.0077
Epoch 1/1
 - 9s - loss: 0.0076
Epoch 1/1
 - 8s - loss: 0.0075
Epoch 1/1
 - 9s - loss: 0.0077
Epoch 1/1
 - 8s - loss: 0.0076
Epoch 1/1
 - 8s - loss: 0.0077
Epoch 1/1
 - 8s - loss: 0.0075
Epoch 1/1
 - 9s - loss: 0.0076
Epoch 1/1
 - 9s - loss: 0.0075
Epoch 1/1
 - 9s - loss: 0.0078
```

In [45]:

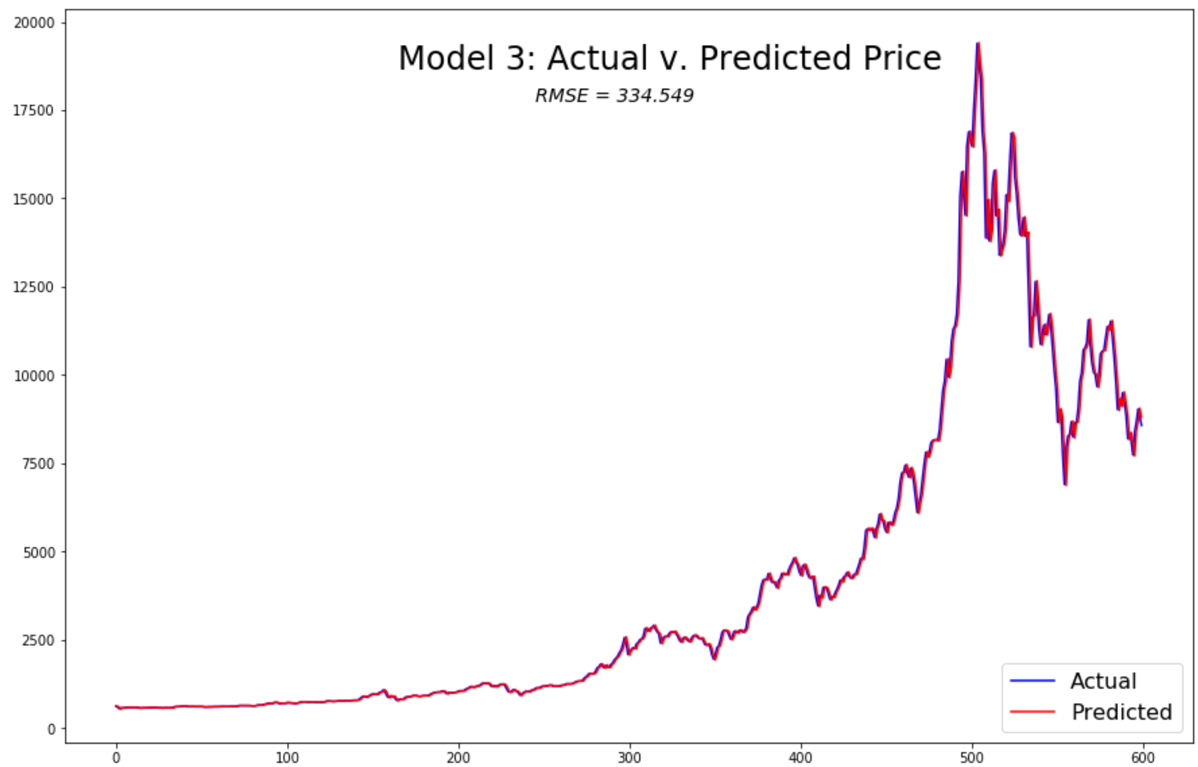
```
predictions_3 = list()
for i in range(len(test_scaled)):
        # predict
        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
        yhat = forecast_lstm(lstm_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)
        # invert differencing
        yhat = inverse_difference(raw_values, yhat, len(test_scaled)+1-i
)
        # store forecast
        predictions_3.append(yhat)
# report performance
rmse_3 = sqrt(mean_squared_error(raw_values[-600:], predictions_3))
print('%d) Test RMSE: %.3f' % (1 , rmse_3))
```

```
1) Test RMSE: 334.549
```

In [87]:
```
lstm_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_15 (LSTM)               (1, 1, 200)               166400
_________________________________________________________________
lstm_16 (LSTM)               (1, 100)                  120400
_________________________________________________________________
dense_5 (Dense)              (1, 1)                    101
=================================================================
Total params: 286,901
Trainable params: 286,901
Non-trainable params: 0
_________________________________________________________________
```

In [46]:
```
#Plot actual versus predicted:
orig = plt.plot(raw_values[-600:], color='blue',label='Actual')
pred = plt.plot(predictions, color='red', label='Predicted')
plt.legend(loc=4, fontsize=16)
plt.text(165, 18650, 'Model 3: Actual v. Predicted Price', fontsize=24)
plt.text(245, 17750, 'RMSE = 334.549', fontsize=14, style='italic')
plt.show()
```



In [47]:
```
# save model
# serialize model to JSON
from keras.models import model_from_json
lstm_model_3 = lstm_model.to_json()
with open("lstm_model_3", "w") as json_file:
    json_file.write(lstm_model_3)
# serialize weights to HDF5
lstm_model.save_weights("model_3_weights.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

Model 4

Model with 500 epochs and one layer of 10 neurons.

In [ ]:
```
numpy.random.seed(1234)
lstm_model = fit4_lstm(train_scaled, 1, 500, 100)
```

In [94]:
```
predictions_4 = list()
for i in range(len(test_scaled)):
        # predict
        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
        yhat = forecast_lstm(lstm_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)
        # invert differencing
        yhat = inverse_difference(raw_values, yhat, len(test_scaled)+1-i
)
        # store forecast
        predictions_4.append(yhat)
# report performance
rmse_4 = sqrt(mean_squared_error(raw_values[-600:], predictions_4))
print('%d) Test RMSE: %.3f' % (1 , rmse_4))
```

```
1) Test RMSE: 328.127
```

In [ ]:
```
lstm_model.summary()
```

In [ ]:
```
#Plot actual versus predicted:
orig = plt.plot(raw_values[-600:], color='blue',label='Actual')
pred = plt.plot(predictions, color='red', label='Predicted')
plt.legend(loc=4, fontsize=16)
plt.text(165, 18650, 'Model 4: Actual v. Predicted Price', fontsize=24)
plt.text(245, 17750, 'RMSE = 328.127', fontsize=14)
plt.show()
```

In [ ]:
```
# save model
# serialize model to JSON
from keras.models import model_from_json
lstm_model_4 = lstm_model.to_json()
with open("lstm_model_4", "w") as json_file:
    json_file.write(lstm_model_4)
# serialize weights to HDF5
lstm_model.save_weights("model_4_weights.h5")
print("Saved model to disk")
```

Model 5

Model with 200 epochs and hidden layers of 100, 50, 30, 20, 10 and 5 neurons.

In [100]:

```
numpy.random.seed(1234)
lstm_model = fit5_lstm(train_scaled, 1, 200, 100)
```

```
Epoch 1/1
 - 23s - loss: 0.0135
Epoch 1/1
 - 20s - loss: 0.0119
Epoch 1/1
 - 20s - loss: 0.0118
Epoch 1/1
 - 20s - loss: 0.0117
Epoch 1/1
 - 20s - loss: 0.0116
Epoch 1/1
 - 20s - loss: 0.0113
Epoch 1/1
 - 20s - loss: 0.0111
Epoch 1/1
 - 20s - loss: 0.0110
Epoch 1/1
 - 20s - loss: 0.0110
Epoch 1/1
 - 20s - loss: 0.0110
Epoch 1/1
 - 20s - loss: 0.0110
Epoch 1/1
 - 20s - loss: 0.0110
Epoch 1/1
 - 20s - loss: 0.0110
Epoch 1/1
 - 20s - loss: 0.0110
Epoch 1/1
 - 20s - loss: 0.0110
Epoch 1/1
 - 20s - loss: 0.0110
Epoch 1/1
 - 21s - loss: 0.0110
Epoch 1/1
 - 21s - loss: 0.0110
Epoch 1/1
 - 20s - loss: 0.0110
Epoch 1/1
 - 21s - loss: 0.0110
Epoch 1/1
 - 21s - loss: 0.0110
Epoch 1/1
 - 21s - loss: 0.0110
Epoch 1/1
 - 22s - loss: 0.0110
Epoch 1/1
 - 21s - loss: 0.0109
Epoch 1/1
 - 21s - loss: 0.0109
Epoch 1/1
 - 20s - loss: 0.0109
Epoch 1/1
 - 20s - loss: 0.0108
Epoch 1/1
 - 21s - loss: 0.0108
Epoch 1/1
```

```
 - 20s - loss: 0.0107
Epoch 1/1
 - 21s - loss: 0.0106
Epoch 1/1
 - 22s - loss: 0.0106
Epoch 1/1
 - 21s - loss: 0.0105
Epoch 1/1
 - 20s - loss: 0.0103
Epoch 1/1
 - 22s - loss: 0.0102
Epoch 1/1
 - 20s - loss: 0.0102
Epoch 1/1
 - 20s - loss: 0.0099
Epoch 1/1
 - 20s - loss: 0.0097
Epoch 1/1
 - 20s - loss: 0.0096
Epoch 1/1
 - 20s - loss: 0.0095
Epoch 1/1
 - 20s - loss: 0.0093
Epoch 1/1
 - 20s - loss: 0.0093
Epoch 1/1
 - 20s - loss: 0.0095
Epoch 1/1
 - 20s - loss: 0.0094
Epoch 1/1
 - 20s - loss: 0.0092
Epoch 1/1
 - 20s - loss: 0.0091
Epoch 1/1
 - 20s - loss: 0.0089
Epoch 1/1
 - 20s - loss: 0.0088
Epoch 1/1
 - 20s - loss: 0.0090
Epoch 1/1
 - 20s - loss: 0.0091
Epoch 1/1
 - 20s - loss: 0.0085
Epoch 1/1
 - 21s - loss: 0.0084
Epoch 1/1
 - 20s - loss: 0.0085
Epoch 1/1
 - 20s - loss: 0.0087
Epoch 1/1
 - 20s - loss: 0.0083
Epoch 1/1
 - 20s - loss: 0.0084
Epoch 1/1
 - 20s - loss: 0.0093
Epoch 1/1
 - 20s - loss: 0.0088
```

```
Epoch 1/1
 - 20s - loss: 0.0083
Epoch 1/1
 - 20s - loss: 0.0081
Epoch 1/1
 - 20s - loss: 0.0082
Epoch 1/1
 - 19s - loss: 0.0082
Epoch 1/1
 - 20s - loss: 0.0081
Epoch 1/1
 - 20s - loss: 0.0081
Epoch 1/1
 - 20s - loss: 0.0079
Epoch 1/1
 - 20s - loss: 0.0077
Epoch 1/1
 - 20s - loss: 0.0079
Epoch 1/1
 - 20s - loss: 0.0088
Epoch 1/1
 - 20s - loss: 0.0079
Epoch 1/1
 - 20s - loss: 0.0077
Epoch 1/1
 - 20s - loss: 0.0085
Epoch 1/1
 - 20s - loss: 0.0075
Epoch 1/1
 - 20s - loss: 0.0075
Epoch 1/1
 - 21s - loss: 0.0075
Epoch 1/1
 - 20s - loss: 0.0073
Epoch 1/1
 - 20s - loss: 0.0073
Epoch 1/1
 - 20s - loss: 0.0075
Epoch 1/1
 - 20s - loss: 0.0072
Epoch 1/1
 - 20s - loss: 0.0072
Epoch 1/1
 - 20s - loss: 0.0079
Epoch 1/1
 - 20s - loss: 0.0072
Epoch 1/1
 - 20s - loss: 0.0071
Epoch 1/1
 - 20s - loss: 0.0073
Epoch 1/1
 - 20s - loss: 0.0072
Epoch 1/1
 - 20s - loss: 0.0071
Epoch 1/1
 - 20s - loss: 0.0072
Epoch 1/1
```

```
 - 21s - loss: 0.0077
Epoch 1/1
 - 20s - loss: 0.0071
Epoch 1/1
 - 20s - loss: 0.0068
Epoch 1/1
 - 20s - loss: 0.0068
Epoch 1/1
 - 20s - loss: 0.0068
Epoch 1/1
 - 21s - loss: 0.0071
Epoch 1/1
 - 22s - loss: 0.0073
Epoch 1/1
 - 20s - loss: 0.0071
Epoch 1/1
 - 21s - loss: 0.0074
Epoch 1/1
 - 21s - loss: 0.0070
Epoch 1/1
 - 23s - loss: 0.0069
Epoch 1/1
 - 21s - loss: 0.0067
Epoch 1/1
 - 22s - loss: 0.0065
Epoch 1/1
 - 21s - loss: 0.0064
Epoch 1/1
 - 20s - loss: 0.0063
Epoch 1/1
 - 20s - loss: 0.0063
Epoch 1/1
 - 22s - loss: 0.0064
Epoch 1/1
 - 22s - loss: 0.0063
Epoch 1/1
 - 21s - loss: 0.0067
Epoch 1/1
 - 20s - loss: 0.0072
Epoch 1/1
 - 22s - loss: 0.0069
Epoch 1/1
 - 20s - loss: 0.0064
Epoch 1/1
 - 20s - loss: 0.0067
Epoch 1/1
 - 20s - loss: 0.0059
Epoch 1/1
 - 20s - loss: 0.0058
Epoch 1/1
 - 20s - loss: 0.0059
Epoch 1/1
 - 20s - loss: 0.0057
Epoch 1/1
 - 21s - loss: 0.0058
Epoch 1/1
 - 21s - loss: 0.0061
```

```
Epoch 1/1
 - 20s - loss: 0.0062
Epoch 1/1
 - 20s - loss: 0.0057
Epoch 1/1
 - 20s - loss: 0.0057
Epoch 1/1
 - 20s - loss: 0.0058
Epoch 1/1
 - 21s - loss: 0.0059
Epoch 1/1
 - 21s - loss: 0.0059
Epoch 1/1
 - 21s - loss: 0.0056
Epoch 1/1
 - 20s - loss: 0.0056
Epoch 1/1
 - 20s - loss: 0.0061
Epoch 1/1
 - 21s - loss: 0.0056
Epoch 1/1
 - 22s - loss: 0.0055
Epoch 1/1
 - 21s - loss: 0.0058
Epoch 1/1
 - 20s - loss: 0.0056
Epoch 1/1
 - 20s - loss: 0.0054
Epoch 1/1
 - 20s - loss: 0.0058
Epoch 1/1
 - 20s - loss: 0.0055
Epoch 1/1
 - 21s - loss: 0.0055
Epoch 1/1
 - 20s - loss: 0.0055
Epoch 1/1
 - 20s - loss: 0.0054
Epoch 1/1
 - 20s - loss: 0.0054
Epoch 1/1
 - 20s - loss: 0.0053
Epoch 1/1
 - 20s - loss: 0.0054
Epoch 1/1
 - 20s - loss: 0.0058
Epoch 1/1
 - 20s - loss: 0.0060
Epoch 1/1
 - 20s - loss: 0.0053
Epoch 1/1
 - 20s - loss: 0.0051
Epoch 1/1
 - 20s - loss: 0.0050
Epoch 1/1
 - 20s - loss: 0.0052
Epoch 1/1
```

```
 - 20s - loss: 0.0049
Epoch 1/1
 - 20s - loss: 0.0049
Epoch 1/1
 - 20s - loss: 0.0048
Epoch 1/1
 - 20s - loss: 0.0052
Epoch 1/1
 - 20s - loss: 0.0048
Epoch 1/1
 - 20s - loss: 0.0049
Epoch 1/1
 - 20s - loss: 0.0046
Epoch 1/1
 - 20s - loss: 0.0051
Epoch 1/1
 - 20s - loss: 0.0049
Epoch 1/1
 - 20s - loss: 0.0049
Epoch 1/1
 - 21s - loss: 0.0046
Epoch 1/1
 - 20s - loss: 0.0045
Epoch 1/1
 - 21s - loss: 0.0046
Epoch 1/1
 - 21s - loss: 0.0044
Epoch 1/1
 - 22s - loss: 0.0044
Epoch 1/1
 - 21s - loss: 0.0046
Epoch 1/1
 - 21s - loss: 0.0045
Epoch 1/1
 - 21s - loss: 0.0045
Epoch 1/1
 - 20s - loss: 0.0044
Epoch 1/1
 - 21s - loss: 0.0043
Epoch 1/1
 - 20s - loss: 0.0042
Epoch 1/1
 - 20s - loss: 0.0043
Epoch 1/1
 - 20s - loss: 0.0043
Epoch 1/1
 - 20s - loss: 0.0044
Epoch 1/1
 - 20s - loss: 0.0049
Epoch 1/1
 - 20s - loss: 0.0040
Epoch 1/1
 - 20s - loss: 0.0041
Epoch 1/1
 - 21s - loss: 0.0039
Epoch 1/1
 - 21s - loss: 0.0042
```

```
Epoch 1/1
 - 21s - loss: 0.0039
Epoch 1/1
 - 22s - loss: 0.0038
Epoch 1/1
 - 21s - loss: 0.0038
Epoch 1/1
 - 21s - loss: 0.0037
Epoch 1/1
 - 21s - loss: 0.0037
Epoch 1/1
 - 20s - loss: 0.0037
Epoch 1/1
 - 20s - loss: 0.0037
Epoch 1/1
 - 20s - loss: 0.0046
Epoch 1/1
 - 22s - loss: 0.0046
Epoch 1/1
 - 22s - loss: 0.0036
Epoch 1/1
 - 20s - loss: 0.0033
Epoch 1/1
 - 20s - loss: 0.0033
Epoch 1/1
 - 20s - loss: 0.0032
Epoch 1/1
 - 20s - loss: 0.0035
Epoch 1/1
 - 20s - loss: 0.0037
Epoch 1/1
 - 20s - loss: 0.0039
Epoch 1/1
 - 20s - loss: 0.0034
Epoch 1/1
 - 22s - loss: 0.0032
Epoch 1/1
 - 22s - loss: 0.0030
Epoch 1/1
 - 22s - loss: 0.0033
Epoch 1/1
 - 22s - loss: 0.0034
Epoch 1/1
 - 23s - loss: 0.0029
Epoch 1/1
 - 22s - loss: 0.0028
Epoch 1/1
 - 22s - loss: 0.0028
Epoch 1/1
 - 22s - loss: 0.0029
Epoch 1/1
 - 23s - loss: 0.0028
Epoch 1/1
 - 22s - loss: 0.0027
Epoch 1/1
 - 20s - loss: 0.0028
```

```
Epoch 1/1
 - 20s - loss: 0.0026
```

In [102]:

```python
predictions_5 = list()
for i in range(len(test_scaled)):
        # predict
        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
        yhat = forecast_lstm(lstm_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)
        # invert differencing
        yhat = inverse_difference(raw_values, yhat, len(test_scaled)+1-i
)
        # store forecast
        predictions_5.append(yhat)
# report performance
rmse_5 = sqrt(mean_squared_error(raw_values[-600:], predictions_5))
print('%d) Test RMSE: %.3f' % (1 , rmse_5))
```

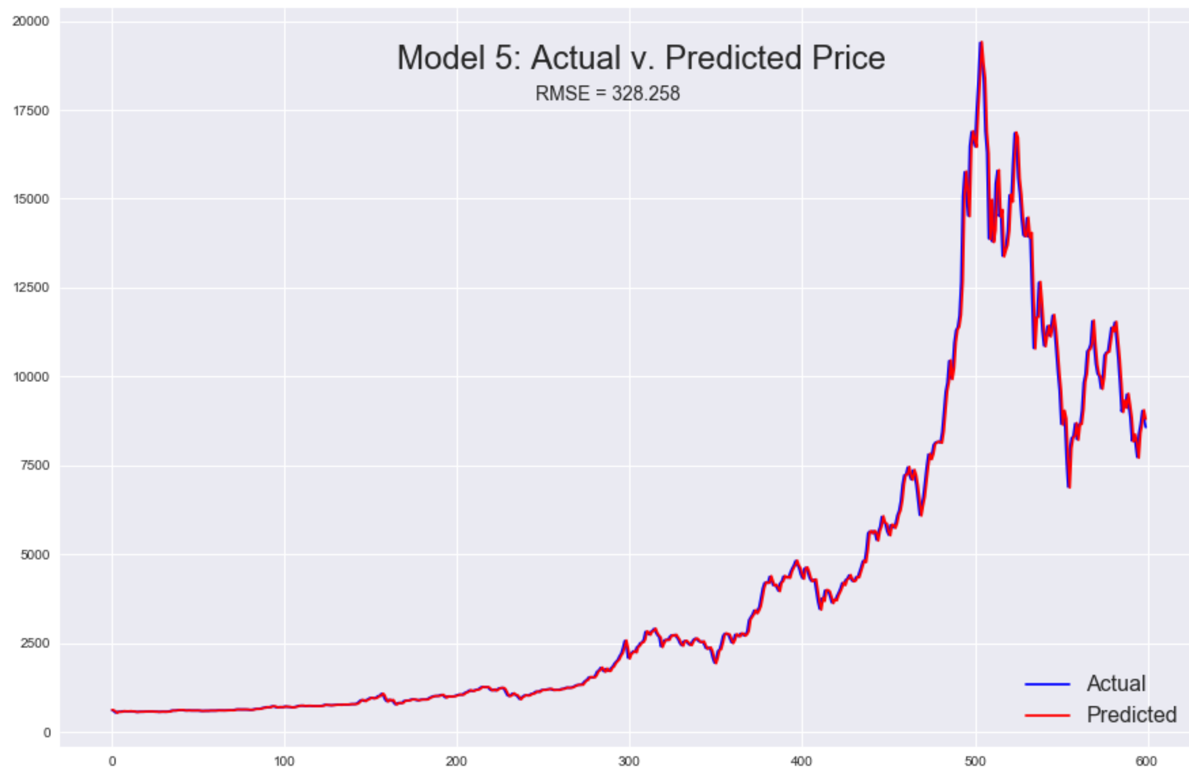
```
1) Test RMSE: 328.258
```

In [103]:

```python
lstm_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_19 (LSTM)               (1, 1, 100)               43200
_________________________________________________________________
lstm_20 (LSTM)               (1, 1, 50)                30200
_________________________________________________________________
lstm_21 (LSTM)               (1, 1, 30)                9720
_________________________________________________________________
lstm_22 (LSTM)               (1, 1, 20)                4080
_________________________________________________________________
lstm_23 (LSTM)               (1, 1, 10)                1240
_________________________________________________________________
lstm_24 (LSTM)               (1, 5)                    320
_________________________________________________________________
dense_8 (Dense)              (1, 1)                    6
=================================================================
Total params: 88,766
Trainable params: 88,766
Non-trainable params: 0
_________________________________________________________________
```

In [104]:
```
#Plot actual versus predicted:
orig = plt.plot(raw_values[-600:], color='blue',label='Actual')
pred = plt.plot(predictions, color='red', label='Predicted')
plt.legend(loc=4, fontsize=16)
plt.text(165, 18650, 'Model 5: Actual v. Predicted Price', fontsize=24)
plt.text(245, 17750, 'RMSE = 328.258', fontsize=14)
plt.show()
```



In [ ]:
```
# save model
# serialize model to JSON
from keras.models import model_from_json
lstm_model_5 = lstm_model.to_json()
with open("lstm_model_5", "w") as json_file:
    json_file.write(lstm_model_5)
# serialize weights to HDF5
lstm_model.save_weights("model_5_weights.h5")
print("Saved model to disk")
```

In [ ]:
```
# pd.DataFrame(predictions).to_csv('predict_3_24.csv'); new_df.to_csv('n
ew_df_3_24.csv')
```

In [ ]:
```
# pd.DataFrame(raw_values[-600:]).to_csv('raw_values_3_24.csv')
```

Part 3: Backtesting

We use results from our best-performing model to simulate gross profits/losses (i.e., excluding trading fees and taxes) from a $10,000 investment over the duration our five-year period. Long and long-short strategies show considerably larger gross profits than a buy-and-hold.

In [105]:
```
finaldf = pd.DataFrame()
```

In [106]:
```
finaldf['yhat'] = predictions_5
finaldf['actual'] = raw_values[-600:]
```

Long only strategy

In [109]:

```python
# Strategy is starting with $100,000. If we are predicting higher price
 tomorrow, we are going long.
# If we are long and predicting lower, we are closing the position.

long = False
ballance = 1000000
position = 0
tbalance = 0

for i in range(1,len(finaldf)):
    if finaldf['yhat'][i] > finaldf['actual'][i-1] and not long:
        position = ballance/finaldf['actual'][i-1]
        ballance = ballance - finaldf['actual'][i-1]*position
        long = True
        print("Action: Buy", "\t" "price", finaldf['actual'][i-1], "\t"
"Balance: ", ballance )
    if finaldf['yhat'][i] < finaldf['actual'][i-1] and long:
        ballance = ballance + position*finaldf['actual'][i-1]
        #tbalance = tbalance + (position*finaldf['actual'][i-1] -10000)
        position = 0
        long = False
        print("Action: sell","\t" "price", finaldf['actual'][i-1], "\t"
"Balance: ", ballance)
    if i == len(finaldf) and long:
        ballance = ballance + position*finaldf['actual'][i]
        position = 0
        long = False
        print("Closed long")


#Final PNL
ballance
```

Action: Buy	price 550.2323023	Balance:	0.0
Action: sell	price 587.0532616	Balance:	1066918.93432
Action: Buy	price 593.4479862	Balance:	0.0
Action: sell	price 587.3153874	Balance:	1055893.5607
Action: Buy	price 587.2248494	Balance:	0.0
Action: sell	price 576.646713	Balance:	1036872.93151
Action: Buy	price 573.1198682	Balance:	0.0
Action: sell	price 579.3445188	Balance:	1048134.40066
Action: Buy	price 575.1770015	Balance:	0.0
Action: sell	price 574.2403722	Balance:	1046427.59843
Action: Buy	price 604.8187711	Balance:	0.0
Action: sell	price 612.2567719	Balance:	1059296.46045
Action: Buy	price 622.2924153	Balance:	0.0
Action: sell	price 622.9217192	Balance:	1060367.6922
Action: Buy	price 610.5118938	Balance:	0.0
Action: sell	price 612.321211	Balance:	1063510.20511
Action: Buy	price 614.0080525	Balance:	0.0
Action: sell	price 611.002838	Balance:	1058304.9374
Action: Buy	price 610.5484313	Balance:	0.0
Action: sell	price 610.1854495	Balance:	1057675.75646
Action: Buy	price 610.5267214	Balance:	0.0
Action: sell	price 601.4110143	Balance:	1041883.71647
Action: Buy	price 600.8648835	Balance:	0.0
Action: sell	price 604.2527135	Balance:	1047758.12352
Action: Buy	price 606.1522112	Balance:	0.0
Action: sell	price 608.0561961	Balance:	1051049.23689
Action: Buy	price 607.8343391	Balance:	0.0
Action: sell	price 612.9725346	Balance:	1059934.05322
Action: Buy	price 611.004079	Balance:	0.0
Action: sell	price 615.8840712	Balance:	1068399.57757
Action: Buy	price 618.3920061	Balance:	0.0
Action: sell	price 637.8551278	Balance:	1102026.12966
Action: Buy	price 640.2696824	Balance:	0.0
Action: sell	price 639.0261185	Balance:	1099885.71922
Action: Buy	price 637.6674787	Balance:	0.0
Action: sell	price 632.5352201	Balance:	1091033.30299
Action: Buy	price 629.6080977	Balance:	0.0
Action: sell	price 651.3984951	Balance:	1128793.3784
Action: Buy	price 655.3074792	Balance:	0.0
Action: sell	price 681.6864268	Balance:	1174232.17213
Action: Buy	price 686.0609328	Balance:	0.0
Action: sell	price 704.0784751	Balance:	1205070.217
Action: Buy	price 721.7589711	Balance:	0.0
Action: sell	price 724.5645218	Balance:	1209754.44779
Action: Buy	price 703.141842	Balance:	0.0
Action: sell	price 705.2771552	Balance:	1213428.25083
Action: Buy	price 725.4650081	Balance:	0.0
Action: sell	price 715.6629188	Balance:	1197033.06714
Action: Buy	price 716.1461885	Balance:	0.0
Action: sell	price 709.9249062	Balance:	1186634.2397
Action: Buy	price 698.3639265	Balance:	0.0
Action: sell	price 703.32663	Balance:	1195066.68255
Action: Buy	price 711.9112755	Balance:	0.0
Action: sell	price 743.8171194	Balance:	1248626.18123
Action: Buy	price 749.7805789	Balance:	0.0
Action: sell	price 741.9046409	Balance:	1235510.20748
Action: Buy	price 744.3066091	Balance:	0.0

```
Action: sell    price 736.6682418       Balance:  1222830.91557
Action: Buy     price 735.7482894       Balance:  0.0
Action: sell    price 736.2990081       Balance:  1223746.22297
Action: Buy     price 765.0813086       Balance:  0.0
Action: sell    price 756.7646454       Balance:  1210443.73464
Action: Buy     price 757.9309461       Balance:  0.0
Action: sell    price 762.5863612       Balance:  1217878.60462
Action: Buy     price 774.9232831       Balance:  0.0
Action: sell    price 769.1368634       Balance:  1208784.59893
Action: Buy     price 776.2038665       Balance:  0.0
Action: sell    price 778.1382984       Balance:  1211797.0955
Action: Buy     price 780.3870174       Balance:  0.0
Action: sell    price 792.499409        Balance:  1230605.40552
Action: Buy     price 814.4280861       Balance:  0.0
Action: sell    price 903.3235778       Balance:  1364926.99202
Action: Buy     price 918.7565343       Balance:  0.0
Action: sell    price 952.933137        Balance:  1415700.58195
Action: Buy     price 975.3794824       Balance:  0.0
Action: sell    price 1015.567064       Balance:  1474030.27177
Action: Buy     price 1021.924406       Balance:  0.0
Action: sell    price 1046.983904       Balance:  1510176.25128
Action: Buy     price 932.5794354       Balance:  0.0
Action: sell    price 871.3523672       Balance:  1411027.95268
Action: Buy     price 913.4413105       Balance:  0.0
Action: sell    price 844.7248234       Balance:  1304878.95001
Action: Buy     price 785.7758794       Balance:  0.0
Action: sell    price 808.3866016       Balance:  1342426.87712
Action: Buy     price 827.850596        Balance:  0.0
Action: sell    price 818.776514        Balance:  1327712.51728
Action: Buy     price 828.9852566       Balance:  0.0
Action: sell    price 886.0109378       Balance:  1419045.51764
Action: Buy     price 894.0286368       Balance:  0.0
Action: sell    price 922.8638161       Balance:  1464814.10966
Action: Buy     price 921.1066432       Balance:  0.0
Action: sell    price 908.8113035       Balance:  1445261.12173
Action: Buy     price 897.7349632       Balance:  0.0
Action: sell    price 919.615029        Balance:  1480485.78128
Action: Buy     price 972.2134194       Balance:  0.0
Action: sell    price 1012.727232       Balance:  1542180.18119
Action: Buy     price 1017.895875       Balance:  0.0
Action: sell    price 1042.072152       Balance:  1578808.85428
Action: Buy     price 1054.0024         Balance:  0.0
Action: sell    price 1021.67818        Balance:  1530389.8329
Action: Buy     price 1007.699577       Balance:  0.0
Action: sell    price 1053.551128       Balance:  1600024.42348
Action: Buy     price 1098.990636       Balance:  0.0
Action: sell    price 1164.845901       Balance:  1695903.34088
Action: Buy     price 1160.792967       Balance:  -2.32830643654e-10
Action: sell    price 1252.329126       Balance:  1829636.47183
Action: Buy     price 1175.556969       Balance:  0.0
Action: sell    price 1221.803154       Balance:  1901614.01864
Action: Buy     price 1019.41057        Balance:  0.0
Action: sell    price 1062.290732       Balance:  1981602.90592
Action: Buy     price 1003.931965       Balance:  0.0
Action: sell    price 1039.352979       Balance:  2051518.38498
Action: Buy     price 1053.977146       Balance:  0.0
Action: sell    price 1081.516214       Balance:  2105121.92327
```

```
Action: Buy     price 1143.5689       Balance:  0.0
Action: sell    price 1192.112059     Balance:  2194481.88072
Action: Buy     price 1186.628193     Balance:  0.0
Action: sell    price 1187.944108     Balance:  2196915.45818
Action: Buy     price 1185.925684     Balance:  0.0
Action: sell    price 1213.75341      Balance:  2248466.04203
Action: Buy     price 1214.565918     Balance:  0.0
Action: sell    price 1253.905279     Balance:  2321293.06279
Action: Buy     price 1250.085926     Balance:  0.0
Action: sell    price 1272.448042     Balance:  2362817.42816
Action: Buy     price 1323.925936     Balance:  0.0
Action: sell    price 1334.408423     Balance:  2381525.57663
Action: Buy     price 1388.339179     Balance:  0.0
Action: sell    price 1441.308628     Balance:  2472388.16949
Action: Buy     price 1468.571456     Balance:  0.0
Action: sell    price 1535.046833     Balance:  2584301.64499
Action: Buy     price 1542.320304     Balance:  0.0
Action: sell    price 1602.892181     Balance:  2685795.47929
Action: Buy     price 1771.554671     Balance:  0.0
Action: sell    price 1781.41046      Balance:  2700737.51521
Action: Buy     price 1724.079595     Balance:  0.0
Action: sell    price 1927.798999     Balance:  3019860.04213
Action: Buy     price 2037.688691     Balance:  0.0
Action: sell    price 2212.994415     Balance:  3279663.58985
Action: Buy     price 2370.584191     Balance:  0.0
Action: sell    price 2090.43164      Balance:  2892077.21995
Action: Buy     price 2180.564241     Balance:  0.0
Action: sell    price 2249.844868     Balance:  2983963.95246
Action: Buy     price 2387.948555     Balance:  0.0
Action: sell    price 2506.637528     Balance:  3132276.86994
Action: Buy     price 2522.659848     Balance:  0.0
Action: sell    price 2823.107831     Balance:  3505330.04575
Action: Buy     price 2808.620249     Balance:  0.0
Action: sell    price 2748.831043     Balance:  3430709.4557
Action: Buy     price 2818.873258     Balance:  4.65661287308e-10
Action: sell    price 2814.415619     Balance:  3425284.28654
Action: Buy     price 2731.557526     Balance:  0.0
Action: sell    price 2679.848374     Balance:  3360442.69191
Action: Buy     price 2490.600899     Balance:  0.0
Action: sell    price 2585.80772      Balance:  3488900.47331
Action: Buy     price 2591.504975     Balance:  0.0
Action: sell    price 2724.993128     Balance:  3668613.37553
Action: Buy     price 2708.803666     Balance:  0.0
Action: sell    price 2610.463412     Balance:  3535428.24451
Action: Buy     price 2500.995273     Balance:  0.0
Action: sell    price 2544.72019      Balance:  3597238.16003
Action: Buy     price 2570.197731     Balance:  4.65661287308e-10
Action: sell    price 2479.267623     Balance:  3469972.75533
Action: Buy     price 2538.844273     Balance:  0.0
Action: sell    price 2608.125695     Balance:  3564663.33929
Action: Buy     price 2635.637271     Balance:  0.0
Action: sell    price 2606.260828     Balance:  3524932.10216
Action: Buy     price 2555.505174     Balance:  4.65661287308e-10
Action: sell    price 2092.468244     Balance:  2886242.83021
Action: Buy     price 2069.55207      Balance:  0.0
Action: sell    price 2324.181149     Balance:  3241354.14357
Action: Buy     price 2495.685191     Balance:  0.0
```

```
Action: sell    price 2764.690577      Balance:  3590733.83524
Action: Buy     price 2751.010088      Balance:  0.0
Action: sell    price 2633.923639      Balance:  3437907.68753
Action: Buy     price 2583.746771      Balance:  0.0
Action: sell    price 2724.361001      Balance:  3625007.57971
Action: Buy     price 2775.183747      Balance:  0.0
Action: sell    price 2828.760316      Balance:  3694990.50208
Action: Buy     price 3147.13161       Balance:  0.0
Action: sell    price 3354.059282      Balance:  3937940.55229
Action: Buy     price 3413.142136      Balance:  4.65661287308e-10
Action: sell    price 4267.39414       Balance:  4923540.763
Action: Buy     price 4148.029651      Balance:  0.0
Action: sell    price 3981.514068      Balance:  4725893.60771
Action: Buy     price 4183.225291      Balance:  0.0
Action: sell    price 4343.959435      Balance:  4907478.96609
Action: Buy     price 4494.437119      Balance:  0.0
Action: sell    price 4716.671551      Balance:  5150136.89449
Action: Buy     price 4441.606165      Balance:  0.0
Action: sell    price 4528.176639      Balance:  5250517.20187
Action: Buy     price 4271.211468      Balance:  0.0
Action: sell    price 3997.29913       Balance:  4913802.09112
Action: Buy     price 3465.158917      Balance:  0.0
Action: sell    price 3697.029978      Balance:  5242609.09008
Action: Buy     price 3980.720882      Balance:  0.0
Action: sell    price 3960.290209      Balance:  5215701.89534
Action: Buy     price 3820.775985      Balance:  0.0
Action: sell    price 3705.50656       Balance:  5058348.79199
Action: Buy     price 3831.259125      Balance:  0.0
Action: sell    price 3926.398515      Balance:  5183959.77333
Action: Buy     price 4041.845466      Balance:  0.0
Action: sell    price 4146.85416       Balance:  5318641.03468
Action: Buy     price 4278.981012      Balance:  0.0
Action: sell    price 4319.341364      Balance:  5368807.70374
Action: Buy     price 4258.735188      Balance:  0.0
Action: sell    price 4357.281562      Balance:  5493040.95811
Action: Buy     price 4642.577228      Balance:  0.0
Action: sell    price 4784.869249      Balance:  5661399.14818
Action: Buy     price 5103.569255      Balance:  0.0
Action: sell    price 5602.540466      Balance:  6214908.8681
Action: Buy     price 5616.88043       Balance:  -9.31322574615e-10
Action: sell    price 5407.695032      Balance:  5983451.53136
Action: Buy     price 5639.431298      Balance:  0.0
Action: sell    price 5914.172506      Balance:  6274952.68012
Action: Buy     price 5661.714545      Balance:  0.0
Action: sell    price 5550.539122      Balance:  6151735.49689
Action: Buy     price 5820.928683      Balance:  -9.31322574615e-10
Action: sell    price 5753.306695      Balance:  6080270.51138
Action: Buy     price 6099.049665      Balance:  0.0
Action: sell    price 6947.575925      Balance:  6926184.13403
Action: Buy     price 6544.375848      Balance:  0.0
Action: sell    price 6107.615699      Balance:  6463942.77066
Action: Buy     price 6352.721124      Balance:  0.0
Action: sell    price 8161.023041      Balance:  8303903.92674
Action: Buy     price 8439.600966      Balance:  0.0
Action: sell    price 9946.465749      Balance:  9786540.42093
Action: Buy     price 10209.38848      Balance:  0.0
Action: sell    price 15147.33044      Balance:  14519964.8256
```

```
Action: Buy     price 16476.6898        Balance:  0.0
Action: sell    price 16687.43696       Balance:  14705684.2503
Action: Buy     price 16473.9132        Balance:  0.0
Action: sell    price 18848.27709       Balance:  16825195.5793
Action: Buy     price 14951.28644       Balance:  0.0
Action: sell    price 13798.72283       Balance:  15528176.2069
Action: Buy     price 14094.40131       Balance:  -1.86264514923e-09
Action: sell    price 14512.3731        Balance:  15988666.8274
Action: Buy     price 14672.56474       Balance:  1.86264514923e-09
Action: sell    price 13395.63709       Balance:  14597201.1143
Action: Buy     price 13530.34288       Balance:  0.0
Action: sell    price 15591.355         Balance:  16820722.623
Action: Buy     price 15141.87142       Balance:  0.0
Action: sell    price 14454.05864       Balance:  16056648.7733
Action: Buy     price 13943.20689       Balance:  1.86264514923e-09
Action: sell    price 13931.89035       Balance:  16043616.9285
Action: Buy     price 11636.14171       Balance:  0.0
Action: sell    price 12086.58129       Balance:  16664671.5917
Action: Buy     price 11149.89693       Balance:  -1.86264514923e-09
Action: sell    price 11146.83673       Balance:  16660097.807
Action: Buy     price 11354.79056       Balance:  0.0
Action: sell    price 11382.30768       Balance:  16700471.7714
Action: Buy     price 10147.93641       Balance:  0.0
Action: sell    price 9620.999648       Balance:  15833291.2765
Action: Buy     price 9026.190483       Balance:  0.0
Action: sell    price 8844.390109       Balance:  15514386.1657
Action: Buy     price 7963.871181       Balance:  0.0
Action: sell    price 10886.11897       Balance:  21207205.6049
Action: Buy     price 10358.41938       Balance:  0.0
Action: sell    price 10011.92585       Balance:  20497815.5656
Action: Buy     price 9671.487812       Balance:  0.0
Action: sell    price 11286.79258       Balance:  23921303.2295
Action: Buy     price 11525.99812       Balance:  0.0
Action: sell    price 11058.83504       Balance:  22951742.973
Action: Buy     price 9312.040508       Balance:  0.0
Action: sell    price 9137.994765       Balance:  22522765.7627
Action: Buy     price 9501.060708       Balance:  0.0
Action: sell    price 9224.963591       Balance:  21868262.9777
Action: Buy     price 8864.711601       Balance:  0.0
Action: sell    price 8195.082981       Balance:  20216363.241
Action: Buy     price 8356.61907        Balance:  0.0
Action: sell    price 7734.753936       Balance:  18711944.8476
Action: Buy     price 8390.472654       Balance:  0.0
Action: sell    price 8828.079342       Balance:  19687869.8697
```

```
Out[109]: 19687869.869729411
```

Long / Short strategy

In [110]:
```
#Strategy is starting with $10,000. Long/short.
long = False
short = False
ballance = 1000000
longposition = 0
shortposition = 0

for i in range(1,len(finaldf)):
    if finaldf['yhat'][i] > finaldf['actual'][i-1] and not long and not 
short:
        longposition = ballance/finaldf['actual'][i-1]
        ballance = 0
        long = True
        print("Action: none, going long", "\t""price: ", finaldf['actua
l'][i-1], "\t""amount long: ", longposition, "\t""Ballance: ",ballance)

    if finaldf['yhat'][i] < finaldf['actual'][i-1] and not long and not 
short:
        shortposition = ballance/finaldf['actual'][i-1]
        ballance = ballance + finaldf['actual'][i-1]*(shortposition)
        short = True
        print("none, going short", "\t" "price: ", finaldf['actual'][i-1
], "\t""amount short: ", shortposition, "\t""Ballance: ", ballance )

    if finaldf['yhat'][i] < finaldf['actual'][i-1] and long:
        ballance = ballance + longposition*finaldf['actual'][i-1] # sell
        longposition=0
        shortposition = ballance/finaldf['actual'][i-1] #short
        ballance = ballance + finaldf['actual'][i-1]*(ballance/finaldf[
'actual'][i-1])
        long = False
        short = True
        print("long, going short", "\t""price: ", finaldf['actual'][i-1
], "\t""amount short: ", shortposition,"\t""Ballance: ", ballance)

    if finaldf['yhat'][i] > finaldf['actual'][i-1] and short:
        ballance = ballance - shortposition*finaldf['actual'][i-1] #cove
r
        shorptosition = 0
        short = False
        longposition = ballance/finaldf['actual'][i-1] # go long
        ballance = 0
        long = True
        print("short, going long", "\t""price: ", finaldf['actual'][i-1
], "\t""amount long: ", longposition, "\t""Ballance: ",ballance)

    if i == (len(finaldf)-1):
        if long:
            ballance = ballance + position*finaldf['actual'][i]
            position = 0
            long = False
            print("Closing long")
        if short:
            ballance = ballance - shortposition*finaldf['actual'][i]
            position = 0
            short = False
```

```
            print("Covered short")

        #print("Cashing out")


#Final PNL
ballance
```

```
none, going short       price:  620.2563179     amount short:  1612.236
70141   Ballance:  2000000.0
short, going long       price:  550.2323023     amount long:  2022.5917
0041    Ballance:  0
long, going short       price:  587.0532616     amount short:  2022.591
70041   Ballance:  2374738.10922
short, going long       price:  593.4479862     amount long:  1979.0026
5064    Ballance:  0
long, going short       price:  587.3153874     amount short:  1979.002
65064   Ballance:  2324597.41686
short, going long       price:  587.2248494     amount long:  1979.6128
937     Ballance:  0
long, going short       price:  576.646713      amount short:  1979.612
8937    Ballance:  2283074.53632
short, going long       price:  573.1198682     amount long:  2003.9770
3749    Ballance:  0
long, going short       price:  579.3445188     amount short:  2003.977
03749   Ballance:  2321986.22494
short, going long       price:  575.1770015     amount long:  2033.0171
7279    Ballance:  0
long, going short       price:  574.2403722     amount short:  2033.017
17279   Ballance:  2334881.07598
short, going long       price:  604.8187711     amount long:  1827.4468
0014    Ballance:  0
long, going short       price:  612.2567719     amount short:  1827.446
80014   Ballance:  2237733.35735
short, going long       price:  622.2924153     amount long:  1768.5047
2093    Ballance:  0
long, going short       price:  622.9217192     amount short:  1768.504
72093   Ballance:  2203280.00235
short, going long       price:  610.5118938     amount long:  1840.4012
2296    Ballance:  0
long, going short       price:  612.321211      amount short:  1840.401
22296   Ballance:  2253833.41113
short, going long       price:  614.0080525     amount long:  1830.2890
9121    Ballance:  0
long, going short       price:  611.002838      amount short:  1830.289
09121   Ballance:  2236623.65818
short, going long       price:  610.5484313     amount long:  1833.0135
1268    Ballance:  0
long, going short       price:  610.1854495     amount short:  1833.013
51268   Ballance:  2236956.34834
short, going long       price:  610.5267214     amount long:  1830.9642
7885    Ballance:  0
long, going short       price:  601.4110143     amount short:  1830.964
27885   Ballance:  2202324.16818
short, going long       price:  600.8648835     amount long:  1834.2926
3441    Ballance:  0
long, going short       price:  604.2527135     amount short:  1834.292
63441   Ballance:  2216752.60339
short, going long       price:  606.1522112     amount long:  1822.7963
9773    Ballance:  0
long, going short       price:  608.0561961     amount short:  1822.796
39773   Ballance:  2216725.28774
short, going long       price:  607.8343391     amount long:  1824.1270
2391    Ballance:  0
long, going short       price:  612.9725346     amount short:  1824.127
```

```
02391   Ballance:  2236279.53055
short, going long       price:  611.004079      amount long:  1835.8805
07      Ballance:  0
long, going short       price:  615.8840712     amount short:  1835.880
507     Ballance:  2261379.12177
short, going long       price:  618.3920061     amount long:  1820.9894
0637    Ballance:  0
long, going short       price:  637.8551278     amount short:  1820.989
40637   Ballance:  2323054.86104
short, going long       price:  640.2696824     amount long:  1807.2549
4894    Ballance:  0
long, going short       price:  639.0261185     amount short:  1807.254
94894   Ballance:  2309766.23032
short, going long       price:  637.6674787     amount long:  1814.9561
681     Ballance:  0
long, going short       price:  632.5352201     amount short:  1814.956
1681    Ballance:  2296047.39852
short, going long       price:  629.6080977     amount long:  1831.8320
5923    Ballance:  0
long, going short       price:  651.3984951     amount short:  1831.832
05923   Ballance:  2386505.29331
short, going long       price:  655.3074792     amount long:  1809.9778
8353    Ballance:  0
long, going short       price:  681.6864268     amount short:  1809.977
88353   Ballance:  2467674.71202
short, going long       price:  686.0609328     amount long:  1786.8960
8792    Ballance:  0
long, going short       price:  704.0784751     amount short:  1786.896
08792   Ballance:  2516230.1455
short, going long       price:  721.7589711     amount long:  1699.3510
4754    Ballance:  0
long, going short       price:  724.5645218     amount short:  1699.351
04754   Ballance:  2462578.95827
short, going long       price:  703.141842      amount long:  1802.8995
8124    Ballance:  0
long, going short       price:  705.2771552     amount short:  1802.899
58124   Ballance:  2543087.77553
short, going long       price:  725.4650081     amount long:  1702.5593
2738    Ballance:  0
long, going short       price:  715.6629188     amount short:  1702.559
32738   Ballance:  2436917.15532
short, going long       price:  716.1461885     amount long:  1700.2614
8555    Ballance:  0
long, going short       price:  709.9249062     amount short:  1700.261
48555   Ballance:  2414115.95128
short, going long       price:  698.3639265     amount long:  1756.5550
247     Ballance:  0
long, going short       price:  703.32663       amount short:  1756.555
0247    Ballance:  2470863.85186
short, going long       price:  711.9112755     amount long:  1714.1918
7438    Ballance:  0
long, going short       price:  743.8171194     amount short:  1714.191
87438   Ballance:  2550090.5242
short, going long       price:  749.7805789     amount long:  1686.9238
5996    Ballance:  0
long, going short       price:  741.9046409     amount short:  1686.923
85996   Ballance:  2503073.2811
```

```
short, going long      price:  744.3066091     amount long:  1676.0360
4191    Ballance:  0
long, going short      price:  736.6682418     amount short:  1676.036
04191   Ballance:  2469365.04837
short, going long      price:  735.7482894     amount long:  1680.2273
4864    Ballance:  0
long, going short      price:  736.2990081     amount short:  1680.227
34864   Ballance:  2474299.46037
short, going long      price:  765.0813086     amount long:  1553.8072
9913    Ballance:  0
long, going short      price:  756.7646454     amount short:  1553.807
29913   Ballance:  2351732.85949
short, going long      price:  757.9309461     amount long:  1549.0253
1589    Ballance:  0
long, going short      price:  762.5863612     amount short:  1549.025
31589   Ballance:  2362531.1581
short, going long      price:  774.9232831     amount long:  1499.7037
772     Ballance:  0
long, going short      price:  769.1368634     amount short:  1499.703
7772    Ballance:  2306954.91845
short, going long      price:  776.2038665     amount long:  1472.3954
586     Ballance:  0
long, going short      price:  778.1382984     amount short:  1472.395
4586    Ballance:  2291454.59346
short, going long      price:  780.3870174     amount long:  1463.9099
1599    Ballance:  0
long, going short      price:  792.499409      amount short:  1463.909
91599   Ballance:  2320295.48651
short, going long      price:  814.4280861     amount long:  1385.0776
4486    Ballance:  0
long, going short      price:  903.3235778     amount short:  1385.077
64486   Ballance:  2502346.58737
short, going long      price:  918.7565343     amount long:  1338.5455
2837    Ballance:  0
long, going short      price:  952.933137      amount short:  1338.545
52837   Ballance:  2551088.77873
short, going long      price:  975.3794824     amount long:  1276.9378
0377    Ballance:  0
long, going short      price:  1015.567064     amount short:  1276.937
80377   Ballance:  2593631.95257
short, going long      price:  1021.924406     amount long:  1261.0502
679     Ballance:  0
long, going short      price:  1046.983904     amount short:  1261.050
2679    Ballance:  2640598.66524
short, going long      price:  932.5794354     amount long:  1570.4497
2557    Ballance:  0
long, going short      price:  871.3523672     amount short:  1570.449
72557   Ballance:  2736830.17189
short, going long      price:  913.4413105     amount long:  1425.7254
4237    Ballance:  0
long, going short      price:  844.7248234     amount short:  1425.725
44237   Ballance:  2408691.34504
short, going long      price:  785.7758794     amount long:  1639.6414
2392    Ballance:  0
long, going short      price:  808.3866016     amount short:  1639.641
42392   Ballance:  2650928.31706
short, going long      price:  827.850596      amount long:  1562.5406
```

```
2422    Ballance:  0
long, going short       price:  818.776514      amount short:  1562.540
62422   Ballance:  2558743.13056
short, going long       price:  828.9852566     amount long:  1524.0560
4344    Ballance:  0
long, going short       price:  886.0109378     amount short:  1524.056
04344   Ballance:  2700660.64862
short, going long       price:  894.0286368     amount long:  1496.7204
0315    Ballance:  0
long, going short       price:  922.8638161     amount short:  1496.720
40315   Ballance:  2762538.20577
short, going long       price:  921.1066432     amount long:  1502.4309
1789    Ballance:  0
long, going short       price:  908.8113035     amount short:  1502.430
91789   Ballance:  2730852.40181
short, going long       price:  897.7349632     amount long:  1539.5051
9216    Ballance:  0
long, going short       price:  919.615029      amount short:  1539.505
19216   Ballance:  2831504.22386
short, going long       price:  972.2134194     amount long:  1372.9255
225     Ballance:  0
long, going short       price:  1012.727232     amount short:  1372.925
5225    Ballance:  2780798.12829
short, going long       price:  1017.895875     amount long:  1358.9827
1741    Ballance:  0
long, going short       price:  1042.072152     amount short:  1358.982
71741   Ballance:  2832316.08973
short, going long       price:  1054.0024       amount long:  1328.2180
7998    Ballance:  0
long, going short       price:  1021.67818      amount short:  1328.218
07998   Ballance:  2714022.86119
short, going long       price:  1007.699577     amount long:  1365.0676
2058    Ballance:  0
long, going short       price:  1053.551128     amount short:  1365.067
62058   Ballance:  2876337.06291
short, going long       price:  1098.990636     amount long:  1252.1858
561     Ballance:  0
long, going short       price:  1164.845901     amount short:  1252.185
8561    Ballance:  2917207.12354
short, going long       price:  1160.792967     amount long:  1260.9299
2464    Ballance:  0
long, going short       price:  1252.329126     amount short:  1260.929
92464   Ballance:  3158198.54094
short, going long       price:  1175.556969     amount long:  1425.6251
503     Ballance:  0
long, going short       price:  1221.803154     amount short:  1425.625
1503    Ballance:  3483666.61011
short, going long       price:  1019.41057      amount long:  1991.7090
5501    Ballance:  0
long, going short       price:  1062.290732     amount short:  1991.709
05501   Ballance:  4231548.13995
short, going long       price:  1003.931965     amount long:  2223.2659
5074    Ballance:  0
long, going short       price:  1039.352979     amount short:  2223.265
95074   Ballance:  4621516.17803
short, going long       price:  1053.977146     amount long:  2161.5693
3299    Ballance:  0
```

```
long, going short	price:  1081.516214	amount short:  2161.569
33299   Ballance:  4675544.56262
short, going long	price:  1143.5689	amount long:  1926.9858
5824    Ballance:  0
long, going short	price:  1192.112059	amount short:  1926.985
85824   Ballance:  4594366.15827
short, going long	price:  1186.628193	amount long:  1944.7965
4619    Ballance:  0
long, going short	price:  1187.944108	amount short:  1944.796
54619   Ballance:  4620619.19662
short, going long	price:  1185.925684	amount long:  1951.4165
6308    Ballance:  0
long, going short	price:  1213.75341	amount short:  1951.416
56308   Ballance:  4737077.01553
short, going long	price:  1214.565918	amount long:  1948.8056
8532    Ballance:  0
long, going short	price:  1253.905279	amount short:  1948.805
68532   Ballance:  4887235.47314
short, going long	price:  1250.085926	amount long:  1960.7139
4968    Ballance:  0
long, going short	price:  1272.448042	amount short:  1960.713
94968   Ballance:  4989813.25239
short, going long	price:  1323.925936	amount long:  1808.2380
1108    Ballance:  0
long, going short	price:  1334.408423	amount short:  1808.238
01108   Ballance:  4825856.06554
short, going long	price:  1388.339179	amount long:  1667.7541
2293    Ballance:  0
long, going short	price:  1441.308628	amount short:  1667.754
12293   Ballance:  4807496.81353
short, going long	price:  1468.571456	amount long:  1605.8331
4032    Ballance:  0
long, going short	price:  1535.046833	amount short:  1605.833
14032   Ballance:  4930058.15274
short, going long	price:  1542.320304	amount long:  1590.6871
5443    Ballance:  0
long, going short	price:  1602.892181	amount short:  1590.687
15443   Ballance:  5099400.00451
short, going long	price:  1771.554671	amount long:  1287.8014
9059    Ballance:  0
long, going short	price:  1781.41046	amount short:  1287.801
49059   Ballance:  4588206.09149
short, going long	price:  1724.079595	amount long:  1373.4480
8558    Ballance:  0
long, going short	price:  1927.798999	amount short:  1373.448
08558   Ballance:  5295463.68913
short, going long	price:  2037.688691	amount long:  1225.3118
2928    Ballance:  0
long, going short	price:  2212.994415	amount short:  1225.311
82928   Ballance:  5423216.46964
short, going long	price:  2370.584191	amount long:  1062.4012
5437    Ballance:  0
long, going short	price:  2090.43164	amount short:  1062.401
25437   Ballance:  4441754.39303
short, going long	price:  2180.564241	amount long:  974.57353
8443    Ballance:  0
long, going short	price:  2249.844868	amount short:  974.5735
```

```
38443   Ballance:  4385278.54791
short, going long       price:  2387.948555     amount long:  861.84732
5285    Ballance:  0
long, going short       price:  2506.637528     amount short:  861.8473
25285   Ballance:  4320677.69793
short, going long       price:  2522.659848     amount long:  850.89952
0611    Ballance:  0
long, going short       price:  2823.107831     amount short:  850.8995
20611   Ballance:  4804362.20006
short, going long       price:  2808.620249     amount long:  859.67783
5574    Ballance:  0
long, going short       price:  2748.831043     amount short:  859.6778
35574   Ballance:  4726218.24281
short, going long       price:  2818.873258     amount long:  816.95598
5899    Ballance:  0
long, going short       price:  2814.415619     amount short:  816.9559
85899   Ballance:  4598507.3735
short, going long       price:  2731.557526     amount long:  866.51848
9645    Ballance:  0
long, going short       price:  2679.848374     amount short:  866.5184
89645   Ballance:  4644276.33103
short, going long       price:  2490.600899     amount long:  998.20272
3977    Ballance:  0
long, going short       price:  2585.80772      amount short:  998.2027
23977   Ballance:  5162320.61957
short, going long       price:  2591.504975     amount long:  993.81375
6551    Ballance:  0
long, going short       price:  2724.993128     amount short:  993.8137
56551   Ballance:  5416271.31423
short, going long       price:  2708.803666     amount long:  1005.6930
3023    Ballance:  0
long, going short       price:  2610.463412     amount short:  1005.693
03023   Ballance:  5250649.71824
short, going long       price:  2500.995273     amount long:  1093.7310
5702    Ballance:  0
long, going short       price:  2544.72019      amount short:  1093.731
05702   Ballance:  5566479.00647
short, going long       price:  2570.197731     amount long:  1072.0474
5073    Ballance:  0
long, going short       price:  2479.267623     amount short:  1072.047
45073   Ballance:  5315785.06982
short, going long       price:  2538.844273     amount long:  1021.7340
0974    Ballance:  0
long, going short       price:  2608.125695     amount short:  1021.734
00974   Ballance:  5329621.44851
short, going long       price:  2635.637271     amount long:  1000.4036
7482    Ballance:  0
long, going short       price:  2606.260828     amount short:  1000.403
67482   Ballance:  5214625.81973
short, going long       price:  2555.505174     amount long:  1040.1423
0912    Ballance:  0
long, going short       price:  2092.468244     amount short:  1040.142
30912   Ballance:  4352929.50213
short, going long       price:  2069.55207      amount long:  1063.1773
2475    Ballance:  0
long, going short       price:  2324.181149     amount short:  1063.177
32475   Ballance:  4942033.39244
```

```
short, going long       price:  2495.685191     amount long:  917.05375
9792    Ballance:  0
long, going short       price:  2764.690577     amount short:  917.0537
59792   Ballance:  5070739.7766
short, going long       price:  2751.010088     amount long:  926.17458
7032    Ballance:  0
long, going short       price:  2633.923639     amount short:  926.1745
87032   Ballance:  4878946.27725
short, going long       price:  2583.746771     amount long:  962.14757
0545    Ballance:  0
long, going short       price:  2724.361001     amount short:  962.1475
70545   Ballance:  5242474.6368
short, going long       price:  2775.183747     amount long:  926.90739
4723    Ballance:  0
long, going short       price:  2828.760316     amount short:  926.9073
94723   Ballance:  5243997.7096
short, going long       price:  3147.13161      amount long:  739.37109
6121    Ballance:  0
long, going short       price:  3354.059282     amount short:  739.3710
96121   Ballance:  4959788.97557
short, going long       price:  3413.142136     amount long:  713.77347
7983    Ballance:  0
long, going short       price:  4267.39414      amount short:  713.7734
77983   Ballance:  6091905.51446
short, going long       price:  4148.029651     amount long:  754.85284
0297    Ballance:  0
long, going short       price:  3981.514068     amount short:  754.8528
40297   Ballance:  6010914.40582
short, going long       price:  4183.225291     amount long:  682.05624
0062    Ballance:  0
long, going short       price:  4343.959435     amount short:  682.0562
40062   Ballance:  5925649.27843
short, going long       price:  4494.437119     amount long:  636.38456
1654    Ballance:  0
long, going short       price:  4716.671551     amount short:  636.3845
61654   Ballance:  6003233.91489
short, going long       price:  4441.606165     amount long:  715.20621
2468    Ballance:  0
long, going short       price:  4528.176639     amount short:  715.2062
12468   Ballance:  6477160.12673
short, going long       price:  4271.211468     amount long:  801.26286
7852    Ballance:  0
long, going short       price:  3997.29913      amount short:  801.2628
67852   Ballance:  6405774.72913
short, going long       price:  3465.158917     amount long:  1047.3607
8335    Ballance:  0
long, going short       price:  3697.029978     amount short:  1047.360
78335   Ballance:  7744248.42763
short, going long       price:  3980.720882     amount long:  898.07790
9089    Ballance:  0
long, going short       price:  3960.290209     amount short:  898.0779
09089   Ballance:  7113298.30057
short, going long       price:  3820.775985     amount long:  963.66387
5432    Ballance:  0
long, going short       price:  3705.50656      amount short:  963.6638
75432   Ballance:  7141725.6241
short, going long       price:  3831.259125     amount long:  900.40362
```

```
5901    Ballance:  0
long, going short       price:  3926.398515     amount short:  900.4036
25901   Ballance:  7070686.91928
short, going long       price:  4041.845466     amount long:  848.96729
3585    Ballance:  0
long, going short       price:  4146.85416      amount short:  848.9672
93585   Ballance:  7041087.10621
short, going long       price:  4278.981012     amount long:  796.53828
0398    Ballance:  0
long, going short       price:  4319.341364     amount short:  796.5382
80398   Ballance:  6881041.48507
short, going long       price:  4258.735188     amount long:  819.20939
6155    Ballance:  0
long, going short       price:  4357.281562     amount short:  819.2093
96155   Ballance:  7139051.99457
short, going long       price:  4642.577228     amount long:  718.52528
0936    Ballance:  0
long, going short       price:  4784.869249     amount short:  718.5252
80936   Ballance:  6876099.04276
short, going long       price:  5103.569255     amount long:  628.78651
1888    Ballance:  0
long, going short       price:  5602.540466     amount short:  628.7865
11888   Ballance:  7045603.75466
short, going long       price:  5616.88043      amount long:  625.57591
2675    Ballance:  0
long, going short       price:  5407.695032     amount short:  625.5759
12675   Ballance:  6765847.51023
short, going long       price:  5639.431298     amount long:  574.16341
4343    Ballance:  0
long, going short       price:  5914.172506     amount short:  574.1634
14343   Ballance:  6791402.95811
short, going long       price:  5661.714545     amount long:  625.36773
5477    Ballance:  0
long, going short       price:  5550.539122     amount short:  625.3677
35477   Ballance:  6942256.1628
short, going long       price:  5820.928683     amount long:  567.26947
7735    Ballance:  0
long, going short       price:  5753.306695     amount short:  567.2694
77735   Ballance:  6527350.56825
short, going long       price:  6099.049665     amount long:  502.95472
5505    Ballance:  0
long, going short       price:  6947.575925     amount short:  502.9547
25505   Ballance:  6988632.28457
short, going long       price:  6544.375848     amount long:  564.92897
2939    Ballance:  0
long, going short       price:  6107.615699     amount short:  564.9289
72939   Ballance:  6900738.12788
short, going long       price:  6352.721124     amount long:  521.33595
0892    Ballance:  0
long, going short       price:  8161.023041     amount short:  521.3359
50892   Ballance:  8509269.41467
short, going long       price:  8439.600966     amount long:  486.91899
4922    Ballance:  0
long, going short       price:  9946.465749     amount short:  486.9189
94922   Ballance:  9686246.21105
short, going long       price:  10209.38848     amount long:  461.83971
1834    Ballance:  0
```

long, going short price: 15147.33044 amount short: 461.8397
11834 Ballance: 13991277.4509
short, going long price: 16476.6898 amount long: 387.31613
3227 Ballance: 0
long, going short price: 16687.43696 amount short: 387.3161
33227 Ballance: 12926627.1136
short, going long price: 16473.9132 amount long: 397.35639
4587 Ballance: 0
long, going short price: 18848.27709 amount short: 397.3563
94587 Ballance: 14978966.8573
short, going long price: 14951.28644 amount long: 604.49497
9035 Ballance: 0
long, going short price: 13798.72283 amount short: 604.4949
79035 Ballance: 16682517.3356
short, going long price: 14094.40131 amount long: 579.13226
1933 Ballance: 0
long, going short price: 14512.3731 amount short: 579.1322
61933 Ballance: 16809166.9188
short, going long price: 14672.56474 amount long: 566.48659
9984 Ballance: 0
long, going short price: 13395.63709 amount short: 566.4865
99984 Ballance: 15176897.8195
short, going long price: 13530.34288 amount long: 555.20691
1708 Ballance: 0
long, going short price: 15591.355 amount short: 555.2069
11708 Ballance: 17312856.1178
short, going long price: 15141.87142 amount long: 588.16933
536 Ballance: 0
long, going short price: 14454.05864 amount short: 588.1693
3536 Ballance: 17002868.1271
short, going long price: 13943.20689 amount long: 631.26807
6795 Ballance: 0
long, going short price: 13931.89035 amount short: 631.2680
76795 Ballance: 17589515.2547
short, going long price: 11636.14171 amount long: 880.35971
9867 Ballance: 0
long, going short price: 12086.58129 amount short: 880.3597
19867 Ballance: 21281078.6372
short, going long price: 11149.89693 amount long: 1028.2748
4159 Ballance: 0
long, going short price: 11146.83673 amount short: 1028.274
84159 Ballance: 22924023.5454
short, going long price: 11354.79056 amount long: 990.61079
3011 Ballance: 0
long, going short price: 11382.30768 amount short: 990.6107
93011 Ballance: 22550873.6744
short, going long price: 10147.93641 amount long: 1231.6019
5678 Ballance: 0
long, going short price: 9620.999648 amount short: 1231.601
95678 Ballance: 23698483.9853
short, going long price: 9026.190483 amount long: 1393.9225
1336 Ballance: 0
long, going short price: 8844.390109 amount short: 1393.922
51336 Ballance: 24656788.9798
short, going long price: 7963.871181 amount long: 1702.1583
2715 Ballance: 0
long, going short price: 10886.11897 amount short: 1702.158

```
32715   Ballance:  37059796.1103
short, going long       price:  10358.41938     amount long:  1875.5879
2454    Ballance:  0
long, going short       price:  10011.92585     amount short:  1875.587
92454   Ballance:  37556494.4514
short, going long       price:  9671.487812     amount long:  2007.6299
61      Ballance:  0
long, going short       price:  11286.79258     amount short:  2007.629
961     Ballance:  45319405.8944
short, going long       price:  11525.99812     amount long:  1924.2990
0711    Ballance:  0
long, going short       price:  11058.83504     amount short:  1924.299
00711   Ballance:  42561010.5745
short, going long       price:  9312.040508     amount long:  2646.2363
6996    Ballance:  0
long, going short       price:  9137.994765     amount short:  2646.236
36996   Ballance:  48362588.1913
short, going long       price:  9501.060708     amount long:  2443.9940
4511    Ballance:  0
long, going short       price:  9224.963591     amount short:  2443.994
04511   Ballance:  45091512.1655
short, going long       price:  8864.711601     amount long:  2642.6364
2806    Ballance:  0
long, going short       price:  8195.082981     amount short:  2642.636
42806   Ballance:  43313249.6332
short, going long       price:  8356.61907      amount long:  2540.4704
3254    Ballance:  0
long, going short       price:  7734.753936     amount short:  2540.470
43254   Ballance:  39299827.3548
short, going long       price:  8390.472654     amount long:  2143.3929
1764    Ballance:  0
long, going short       price:  8828.079342     amount short:  2143.392
91764   Ballance:  37844085.476
Covered short
```

Out[110]: 19465853.69784224

Buy and Hold strategy

In [111]:
```
# Buying in with 10000 holding and selling at the end of test window.
(1000000/finaldf['actual'][0])*finaldf['actual'][599]
```

Out[111]: 13823904.864089414