```
%tensorflow_version 2.x
import json
import requests
from keras.models import Sequential
from keras.layers import Activation, Dense, Dropout, LSTM
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.metrics import mean_absolute_error
%matplotlib inline
```

```
     Colab only includes TensorFlow 2.x; %tensorflow_version has no effect.
```

```
endpoint = 'https://min-api.cryptocompare.com/data/histoday'
res = requests.get(endpoint + '?fsym=BTC&tsym=CAD&limit=500')
hist = pd.DataFrame(json.loads(res.content)['Data'])
hist = hist.set_index('time')
hist.index = pd.to_datetime(hist.index, unit='s')
target_col = 'close'
```

```
hist.drop(["conversionType", "conversionSymbol"], axis = 'columns', inplace = True)
```

```
hist.head(500)
```

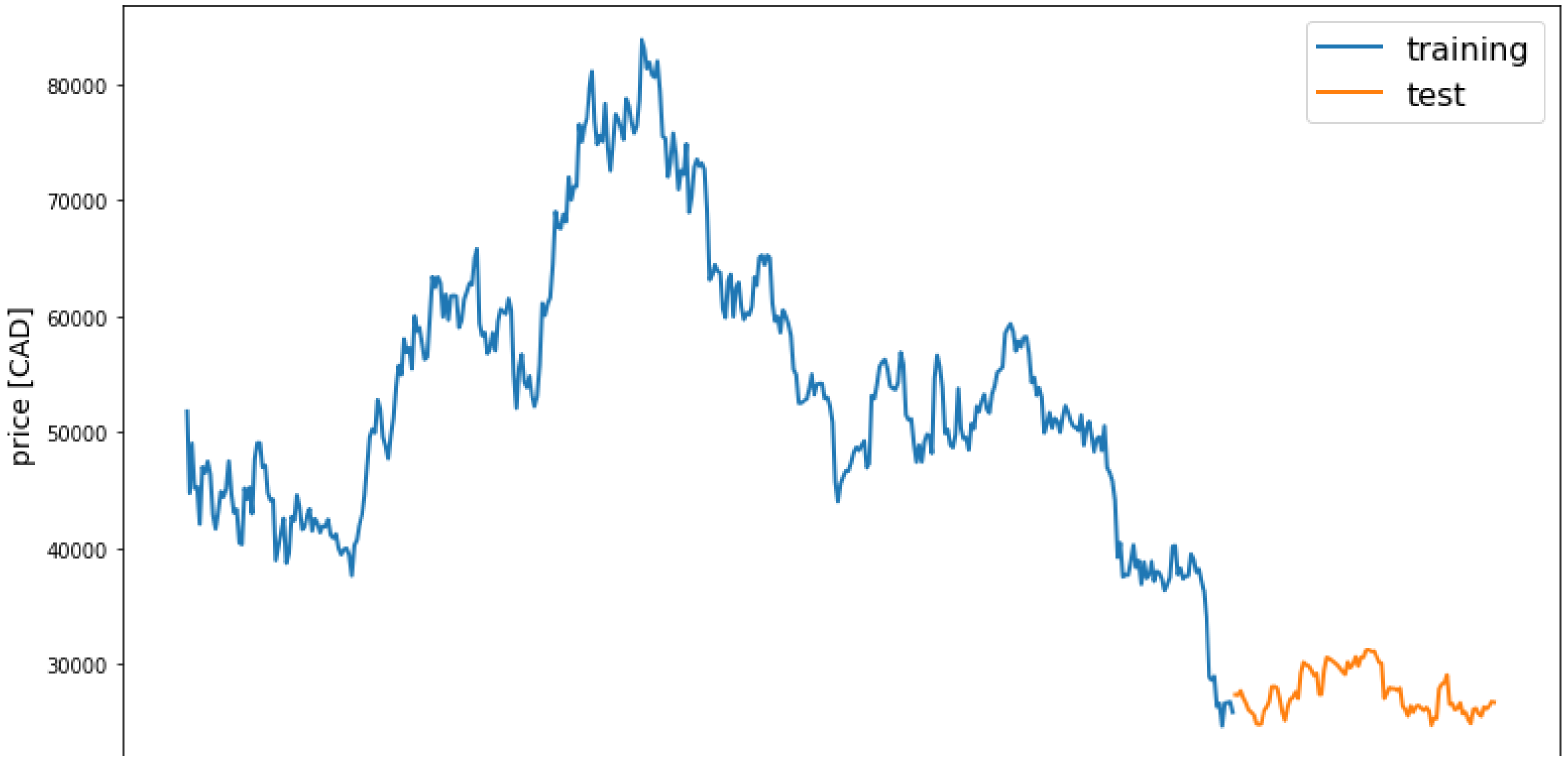
	high	low	open	volumefrom	volumeto	close
time						
2021-05-18	55233.78	51188.02	52491.57	103.92	5505788.51	51756.01
2021-05-19	52581.04	37161.97	51756.01	388.18	17992653.18	44720.39
2021-05-20	51549.23	42572.44	44720.39	226.30	10939478.97	49029.01
2021-05-21	51385.89	40627.45	49029.01	235.88	10836710.18	45155.71
2021-05-22	47102.04	42790.32	45155.71	95.25	4330980.22	45273.75
...	...	...	...	...	...	...
2022-09-25	26015.37	25394.85	25727.84	51.11	1318310.10	25514.32
2022-09-29	26859.84	25938.05	26379.93	118.57	3137920.87	26772.05

```
def train_test_split(df, test_size=0.2):
    split_row = len(df) - int(test_size * len(df))
    train_data = df.iloc[:split_row]
    test_data = df.iloc[split_row:]
    return train_data, test_data
```

```
train, test = train_test_split(hist, test_size=0.2)
```

```
def line_plot(line1, line2, label1=None, label2=None, title='', lw=2):
    fig, ax = plt.subplots(1, figsize=(13, 7))
    ax.plot(line1, label=label1, linewidth=lw)
    ax.plot(line2, label=label2, linewidth=lw)
    ax.set_ylabel('price [CAD]', fontsize=14)
    ax.set_title(title, fontsize=16)
    ax.legend(loc='best', fontsize=16);
```

```
line_plot(train[target_col], test[target_col], 'training', 'test', title='')
```



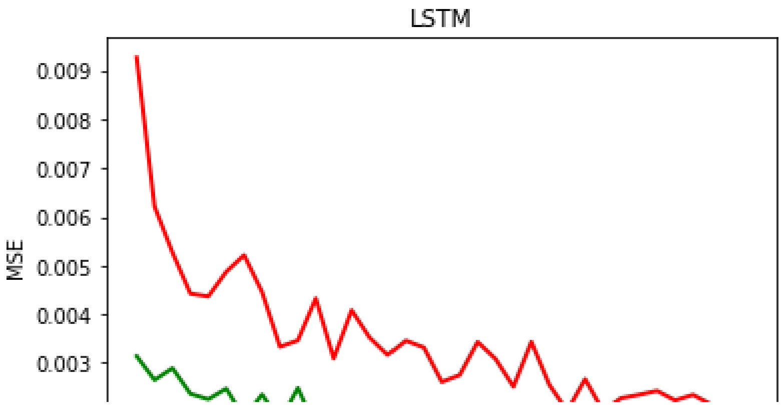
```
def normalise_zero_base(df):
    return df / df.iloc[0] - 1

def normalise_min_max(df):
    return (df - df.min()) / (data.max() - df.min())


def extract_window_data(df, window_len=5, zero_base=True):
    window_data = []
    for idx in range(len(df) - window_len):
        tmp = df[idx: (idx + window_len)].copy()
        if zero_base:
            tmp = normalise_zero_base(tmp)
        window_data.append(tmp.values)
    return np.array(window_data)
```

```
def prepare_data(df, target_col, window_len=10, zero_base=True, test_size=0.2):
    train_data, test_data = train_test_split(df, test_size=test_size)
    X_train = extract_window_data(train_data, window_len, zero_base)
    X_test = extract_window_data(test_data, window_len, zero_base)
    y_train = train_data[target_col][window_len:].values
    y_test = test_data[target_col][window_len:].values
    if zero_base:
        y_train = y_train / train_data[target_col][:-window_len].values - 1
        y_test = y_test / test_data[target_col][:-window_len].values - 1

    return train_data, test_data, X_train, X_test, y_train, y_test


def build_lstm_model(input_data, output_size, neurons=100, activ_func='linear',
                     dropout=0.2, loss='mse', optimizer='adam'):
    model = Sequential()
    model.add(LSTM(neurons, input_shape=(input_data.shape[1], input_data.shape[2])))
    model.add(Dropout(dropout))
    model.add(Dense(units=output_size))
    model.add(Activation(activ_func))

    model.compile(loss=loss, optimizer=optimizer)
    return model


np.random.seed(42)
window_len = 5
test_size = 0.2
zero_base = True
lstm_neurons = 100
epochs = 35
batch_size = 40
loss = 'mse'
dropout = 0.2
optimizer = 'adam'


train, test, X_train, X_test, y_train, y_test = prepare_data(
    hist, target_col, window_len=window_len, zero_base=zero_base, test_size=test_size)


model = build_lstm_model(
    X_train, output_size=1, neurons=lstm_neurons, dropout=dropout, loss=loss,
    optimizer=optimizer)
history = model.fit(
    X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=batch_size, verbose=1, shuffle=True)
```

```
Epoch 7/35
10/10 [==============================] - 0s 13ms/step - loss: 0.0052 - val_loss: 0.0019
Epoch 8/35
10/10 [==============================] - 0s 10ms/step - loss: 0.0045 - val_loss: 0.0024
Epoch 9/35
10/10 [==============================] - 0s 11ms/step - loss: 0.0033 - val_loss: 0.0018
Epoch 10/35
10/10 [==============================] - 0s 10ms/step - loss: 0.0035 - val_loss: 0.0025
Epoch 11/35
10/10 [==============================] - 0s 11ms/step - loss: 0.0043 - val_loss: 0.0017
Epoch 12/35
10/10 [==============================] - 0s 10ms/step - loss: 0.0031 - val_loss: 0.0019
Epoch 13/35
10/10 [==============================] - 0s 11ms/step - loss: 0.0041 - val_loss: 0.0021
Epoch 14/35
10/10 [==============================] - 0s 11ms/step - loss: 0.0035 - val_loss: 0.0016
Epoch 15/35
10/10 [==============================] - 0s 12ms/step - loss: 0.0032 - val_loss: 0.0015
Epoch 16/35
10/10 [==============================] - 0s 12ms/step - loss: 0.0035 - val_loss: 0.0017
Epoch 17/35
10/10 [==============================] - 0s 13ms/step - loss: 0.0033 - val_loss: 0.0016
Epoch 18/35
10/10 [==============================] - 0s 10ms/step - loss: 0.0026 - val_loss: 0.0016
Epoch 19/35
10/10 [==============================] - 0s 11ms/step - loss: 0.0028 - val_loss: 0.0015
Epoch 20/35
10/10 [==============================] - 0s 13ms/step - loss: 0.0034 - val_loss: 0.0014
Epoch 21/35
10/10 [==============================] - 0s 12ms/step - loss: 0.0031 - val_loss: 0.0013
Epoch 22/35
10/10 [==============================] - 0s 12ms/step - loss: 0.0025 - val_loss: 0.0013
Epoch 23/35
```

```
    Epoch 23/35
    10/10 [==============================] - 0s 12ms/step - loss: 0.0034 - val_loss: 0.0015
    Epoch 24/35
    10/10 [==============================] - 0s 12ms/step - loss: 0.0026 - val_loss: 0.0013
    Epoch 25/35
    10/10 [==============================] - 0s 10ms/step - loss: 0.0020 - val_loss: 0.0014
    Epoch 26/35
    10/10 [==============================] - 0s 12ms/step - loss: 0.0027 - val_loss: 0.0012
    Epoch 27/35
    10/10 [==============================] - 0s 11ms/step - loss: 0.0020 - val_loss: 0.0013
    Epoch 28/35
    10/10 [==============================] - 0s 11ms/step - loss: 0.0023 - val_loss: 0.0012
    Epoch 29/35
    10/10 [==============================] - 0s 12ms/step - loss: 0.0023 - val_loss: 0.0014
    Epoch 30/35
    10/10 [==============================] - 0s 11ms/step - loss: 0.0024 - val_loss: 0.0012
    Epoch 31/35
    10/10 [==============================] - 0s 12ms/step - loss: 0.0022 - val_loss: 0.0012
    Epoch 32/35
    10/10 [==============================] - 0s 10ms/step - loss: 0.0023 - val_loss: 0.0012
    Epoch 33/35
    10/10 [==============================] - 0s 12ms/step - loss: 0.0022 - val_loss: 0.0013
    Epoch 34/35
    10/10 [==============================] - 0s 12ms/step - loss: 0.0018 - val_loss: 0.0011
    Epoch 35/35
    10/10 [==============================] - 0s 10ms/step - loss: 0.0021 - val_loss: 0.0011
```

```
import matplotlib.pyplot as plt
plt.plot(history.history['loss'],'r',linewidth=2, label='Train loss')
plt.plot(history.history['val_loss'], 'g',linewidth=2, label='Validation loss')
plt.title('LSTM')
plt.xlabel('Epochs')
plt.ylabel('MSE')
plt.show()
```



```
targets = test[target_col][window_len:]
preds = model.predict(X_test).squeeze()
mean_absolute_error(preds, y_test)
```

```
     0.02554982940364087
```

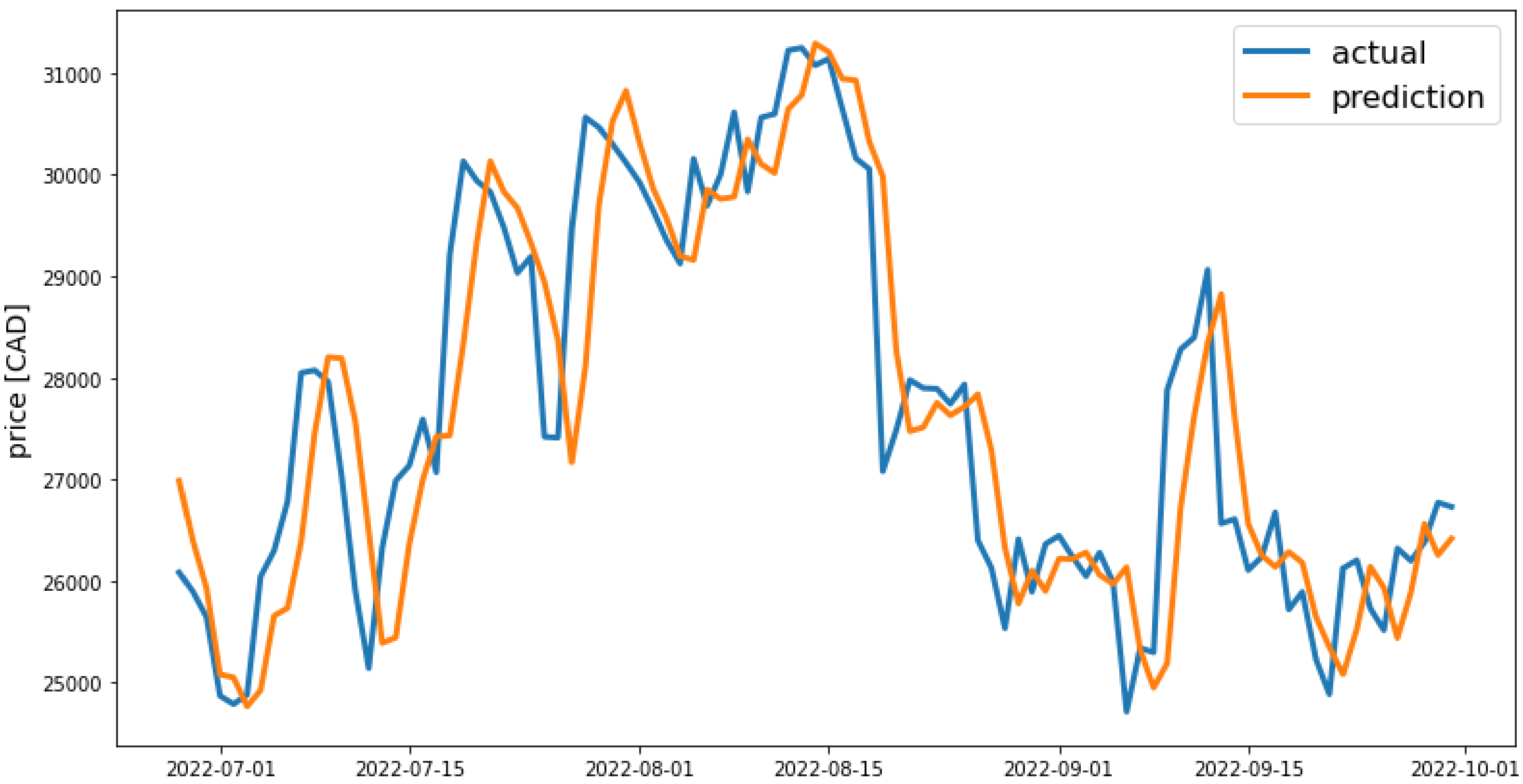
```
from sklearn.metrics import mean_squared_error
MAE=mean_squared_error(preds, y_test)
MAE
```

```
     0.0011294702449779806
```

```
from sklearn.metrics import r2_score
R2=r2_score(y_test, preds)
R2
```

```
     0.7135908938100886
```

```
preds = test[target_col].values[:-window_len] * (preds + 1)
preds = pd.Series(index=targets.index, data=preds)
line_plot(targets, preds, 'actual', 'prediction', lw=3)
line_plot(targets, preds, 'actual', 'prediction', lw=3)
```



✓ 0s completed at 2:18 PM ● ✕