project_two_meek

July 5, 2022

[1]:
```
!pip install hvplot
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: hvplot in /usr/local/lib/python3.7/dist-packages
(0.8.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages
(from hvplot) (1.3.5)
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.7/dist-
packages (from hvplot) (1.21.6)
Requirement already satisfied: bokeh>=1.0.0 in /usr/local/lib/python3.7/dist-
packages (from hvplot) (2.3.3)
Requirement already satisfied: colorcet>=2 in /usr/local/lib/python3.7/dist-
packages (from hvplot) (3.0.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-
packages (from hvplot) (21.3)
Requirement already satisfied: holoviews>=1.11.0 in
/usr/local/lib/python3.7/dist-packages (from hvplot) (1.14.9)
Requirement already satisfied: tornado>=5.1 in /usr/local/lib/python3.7/dist-
packages (from bokeh>=1.0.0->hvplot) (5.1.1)
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.7/dist-
packages (from bokeh>=1.0.0->hvplot) (3.13)
Requirement already satisfied: python-dateutil>=2.1 in
/usr/local/lib/python3.7/dist-packages (from bokeh>=1.0.0->hvplot) (2.8.2)
Requirement already satisfied: Jinja2>=2.9 in /usr/local/lib/python3.7/dist-
packages (from bokeh>=1.0.0->hvplot) (2.11.3)
Requirement already satisfied: typing-extensions>=3.7.4 in
/usr/local/lib/python3.7/dist-packages (from bokeh>=1.0.0->hvplot) (4.1.1)
Requirement already satisfied: pillow>=7.1.0 in /usr/local/lib/python3.7/dist-
packages (from bokeh>=1.0.0->hvplot) (7.1.2)
Requirement already satisfied: pyct>=0.4.4 in /usr/local/lib/python3.7/dist-
packages (from colorcet>=2->hvplot) (0.4.8)
Requirement already satisfied: param>=1.7.0 in /usr/local/lib/python3.7/dist-
packages (from colorcet>=2->hvplot) (1.12.1)
Requirement already satisfied: panel>=0.8.0 in /usr/local/lib/python3.7/dist-
packages (from holoviews>=1.11.0->hvplot) (0.12.1)
Requirement already satisfied: pyviz-comms>=0.7.4 in
/usr/local/lib/python3.7/dist-packages (from holoviews>=1.11.0->hvplot) (2.2.0)
```

```
Requirement already satisfied: MarkupSafe>=0.23 in
/usr/local/lib/python3.7/dist-packages (from Jinja2>=2.9->bokeh>=1.0.0->hvplot)
(2.0.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from packaging->hvplot) (3.0.9)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-
packages (from pandas->hvplot) (2022.1)
Requirement already satisfied: markdown in /usr/local/lib/python3.7/dist-
packages (from panel>=0.8.0->holoviews>=1.11.0->hvplot) (3.3.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.7/dist-packages
(from panel>=0.8.0->holoviews>=1.11.0->hvplot) (5.0.0)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-
packages (from panel>=0.8.0->holoviews>=1.11.0->hvplot) (2.23.0)
Requirement already satisfied: tqdm>=4.48.0 in /usr/local/lib/python3.7/dist-
packages (from panel>=0.8.0->holoviews>=1.11.0->hvplot) (4.64.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-
packages (from python-dateutil>=2.1->bokeh>=1.0.0->hvplot) (1.15.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.7/dist-
packages (from bleach->panel>=0.8.0->holoviews>=1.11.0->hvplot) (0.5.1)
Requirement already satisfied: importlib-metadata>=4.4 in
/usr/local/lib/python3.7/dist-packages (from
markdown->panel>=0.8.0->holoviews>=1.11.0->hvplot) (4.11.4)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-
metadata>=4.4->markdown->panel>=0.8.0->holoviews>=1.11.0->hvplot) (3.8.0)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from
requests->panel>=0.8.0->holoviews>=1.11.0->hvplot) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from
requests->panel>=0.8.0->holoviews>=1.11.0->hvplot) (2022.6.15)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests->panel>=0.8.0->holoviews>=1.11.0->hvplot) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from
requests->panel>=0.8.0->holoviews>=1.11.0->hvplot) (1.24.3)
```

[2]:

```python
# Initial imports
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import pandas as pd

# Other dependancies
import os
import requests
import pandas as pd
import json
```

```python
import numpy as np
import hvplot.pandas

# Set the random seed for reproducibility
# Note: This is for the homework solution, but it is good practice to comment␣
↪this out and run multiple experiments to evaluate your model
from numpy.random import seed
seed(1)
from tensorflow import random
random.set_seed(2)
```

[3]:
```python
# Upload eth_gas_prices.csv data to Colab
from google.colab import files

# Upload th_gas_prices.csv
csv_file = files.upload()
```

<IPython.core.display.HTML object>

Saving eth_gas_prices.csv to eth_gas_prices (2).csv

[4]:
```python
# Upload eth-usd.csv data to Colab
from google.colab import files

# Upload eth-usd.csv
csv_file = files.upload()
```

<IPython.core.display.HTML object>

Saving ETH-USD.csv to ETH-USD (1).csv

[5]:
```python
# Read in data
gas_df = pd.read_csv("eth_gas_prices.csv", delimiter=",", index_col="date")
gas_df = gas_df.drop(columns=["open", "close", "low", "high"])
gas_df.head()
```

[5]:

	avgGas
date	
2022-07-04	90313.448417
2022-07-03	98196.350563
2022-07-02	91501.283451
2022-07-01	94811.137054
2022-06-30	89419.183382

[6]:
```python
# Load the historical closing prices for Bitcoin
eth_df = pd.read_csv('ETH-USD.csv', index_col="Date",␣
↪infer_datetime_format=True, parse_dates=True)["Close"]
eth_df = eth_df.sort_index(ascending=False)
```

```
eth_df = pd.DataFrame(eth_df, index = eth_df.index)
eth_df.index = eth_df.index.strftime("%Y-%m-%d")
eth_df.tail()
```

[6]:

```
                 Close
Date
2017-11-13  316.716003
2017-11-12  307.907990
2017-11-11  314.681000
2017-11-10  299.252991
2017-11-09  320.884003
```

[7]:

```
# Join historical data with highest gas fees
df = gas_df.join(eth_df, how="inner")
df
```

[7]:

```
                   avgGas        Close
2022-03-25   91193.974019  3122.535889
2022-03-24  105808.226105  3108.062012
2022-03-23   94992.370915  3031.067139
2022-03-22   91776.874180  2973.131104
2022-03-21   89850.877301  2897.976563
…                     …            …
2019-10-26   86257.287893   179.835480
2019-10-25   68881.416953   181.523209
2019-10-24   80838.205435   162.168549
2019-10-23   82505.347888   162.402786
2019-10-22   76896.891914   172.300858

[883 rows x 2 columns]
```

[8]:

```
# This function accepts the column number for the features (X) and the target␣
 ↪(y)
# It chunks the data up with a rolling window of Xt-n to predict Xt
# It returns a numpy array of X any y
def window_data(df, window, feature_col_number, target_col_number):
    X = []
    y = []
    for i in range(len(df) - window - 1):
        features = df.iloc[i:(i + window), feature_col_number]
        target = df.iloc[(i + window), target_col_number]
        X.append(features)
        y.append(target)
    return np.array(X), np.array(y).reshape(-1, 1)
```

[9]:

```
# Predict Closing Prices using a 10 day window of previous closing prices
```

```
# Then, experiment with window sizes anywhere from 1 to 10 and see how the
 ↪model performance changes
window_size = 10

# Column index 0 is the 'open' column
# Column index 1 is the `close` column
feature_column = 1
target_column = 1
X, y = window_data(df, window_size, feature_column, target_column)
```

[10]:
```
# Use 70% of the data for training and the remainder for testing
split = int(0.7 * len(X))
X_train = X[: split]
y_train = y[: split]

X_test = X[split:]
y_test = y[split:]
```

[11]:
```
from sklearn.preprocessing import MinMaxScaler
# Use the MinMaxScaler to scale data between 0 and 1.
scaler = MinMaxScaler()
scaler.fit(X_train)

# Scale thr X_train and X_test sets
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

# fit the MinMaxScaler object with the target daya
scaler.fit(y_train)

# Scale the y_train and y_test sets
y_train = scaler.transform(y_train)
y_test = scaler.transform(y_test)
```

[12]:
```
# Reshape the features for the model
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
```

[13]:
```
# Load the saved model to make predictions
from tensorflow.keras.models import model_from_json

# load json and create model
file_path = "model.json"
with open(file_path, "r") as json_file:
    model_json = json_file.read()
loaded_model = model_from_json(model_json)
```

```
# load weights into new model
file_path = "model.h5"
loaded_model.load_weights(file_path)
```

[14]:
```
# Build and Train the LSTM RNN
# Import dependancies
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
```

[15]:
```
# Build the LSTM model.
# The return sequences need to be set to True if you are adding additional LSTM␣
↪layers, but
# You don't have to do this for the final layer.
# Note: The dropouts help prevent overfitting
# Note: The input shape is the number of time steps and the number of indicators
# Note: Batching inputs has a different input shape of Samples/TimeSteps/
↪Features

# Define model
model = Sequential()

# Initial model setup
number_units = 30
dropout_fraction = 0.2

# Layer 1
model.add(LSTM(
    units=number_units,
    return_sequences=True,
    input_shape=(X_train.shape[1], 1))
    )
model.add(Dropout(dropout_fraction))

# Layer 2
model.add(LSTM(units=number_units, return_sequences=True))
model.add(Dropout(dropout_fraction))

# Layer 3
model.add(LSTM(units=number_units))
model.add(Dropout(dropout_fraction))

# Output layer
model.add(Dense(1))
```

[16]:
```
# Compile the model
model.compile(optimizer="adam", loss="mean_squared_error")
```

[17]:
```python
# Summarize the model
model.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 10, 30)            3840

 dropout (Dropout)           (None, 10, 30)            0

 lstm_1 (LSTM)               (None, 10, 30)            7320

 dropout_1 (Dropout)         (None, 10, 30)            0

 lstm_2 (LSTM)               (None, 30)                7320

 dropout_2 (Dropout)         (None, 30)                0

 dense (Dense)               (None, 1)                 31

=================================================================
Total params: 18,511
Trainable params: 18,511
Non-trainable params: 0
_________________________________________________________________
```

[18]:
```python
# Train the model
# Experiement with the batch size, but a smaller batch size is recommended
model.fit(X_train, y_train, epochs=30, shuffle=False, batch_size=5, verbose=1)
```

```
Epoch 1/30
122/122 [==============================] - 4s 7ms/step - loss: 0.0295
Epoch 2/30
122/122 [==============================] - 1s 7ms/step - loss: 0.0213
Epoch 3/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0240
Epoch 4/30
122/122 [==============================] - 1s 7ms/step - loss: 0.0242
Epoch 5/30
122/122 [==============================] - 1s 7ms/step - loss: 0.0299
Epoch 6/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0294
Epoch 7/30
122/122 [==============================] - 1s 7ms/step - loss: 0.0264
Epoch 8/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0204
Epoch 9/30
```

```
122/122 [==============================] - 1s 7ms/step - loss: 0.0202
Epoch 10/30
122/122 [==============================] - 1s 7ms/step - loss: 0.0181
Epoch 11/30
122/122 [==============================] - 1s 7ms/step - loss: 0.0157
Epoch 12/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0155
Epoch 13/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0140
Epoch 14/30
122/122 [==============================] - 1s 7ms/step - loss: 0.0146
Epoch 15/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0138
Epoch 16/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0124
Epoch 17/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0133
Epoch 18/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0141
Epoch 19/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0128
Epoch 20/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0120
Epoch 21/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0112
Epoch 22/30
122/122 [==============================] - 1s 7ms/step - loss: 0.0117
Epoch 23/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0105
Epoch 24/30
122/122 [==============================] - 1s 7ms/step - loss: 0.0106
Epoch 25/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0093
Epoch 26/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0095
Epoch 27/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0087
Epoch 28/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0091
Epoch 29/30
122/122 [==============================] - 1s 7ms/step - loss: 0.0085
Epoch 30/30
122/122 [==============================] - 1s 6ms/step - loss: 0.0096
```

```
[18]: <keras.callbacks.History at 0x7f56d325cad0>
```

[19]:
```python
# Evaluate the model
model.evaluate(X_test, y_test, verbose = 0)
```

[19]: 5.2680632506962866e-05

[20]:
```python
# Make some predictions
predicted = model.predict(X_test)
```

[21]:
```python
# Recover the original prices instead of the scaled version
predicted_prices = scaler.inverse_transform(predicted)
real_prices = scaler.inverse_transform(y_test.reshape(-1, 1))
```

[31]:
```python
# Create a DataFrame of Real and Predicted values
crypto = pd.DataFrame({
    "real": real_prices.ravel(),
    "predicted": predicted_prices.ravel()
}, index = df.index[-len(real_prices): ])
crypto.head()
```

[31]:

	real	predicted
2020-07-12	239.604584	263.990540
2020-07-11	242.131699	263.934509
2020-07-10	239.458176	264.581879
2020-07-09	240.984985	265.308899
2020-07-08	243.015961	265.967224

[63]:
```python
# Plot the real vs predicted eth values as a line chart
hvplot.extension('bokeh')

crypto.hvplot.line(title = "actual eth price vs. predicted eth prices").
↪opts(width = 800)
```

[63]:
```
:NdOverlay   [Variable]
   :Curve   [index]   (value)
```

[ ]:
```python
```