Predictive Model Testing

Securities and Digital Assets

Andrei, Jon, Lareeb, Michelle, Tara

Agenda

- ➢ Motivation/Summary
- ➢ Model Summary
- ➢ Data Cleanup & Model Training
- ➢ Model Evaluation
- ➢ Discussion
- ➢ Postmortem
- ➢ Questions

Motivation & Summary

- ➢ Project one peaked interest in delving more into predictability of 4 assets
  - SPY, Gold, BTC and ETH
- ➢ Can we use machine learning to predict future prices of assets
- ➢ What model is best at predicting future prices
- ➢ Best tool to evaluate model

Model Summary

- Model used and why
  - LSTM was used for prediction of 4 assets
    - LSTM was used as it is more sophisticated than a standard RNN model
      - LSTM more suited to classifying, processing and making predictions on time series data
      - LSTM deals with vanishing gradient problem that can be encountered when training traditional RNNs
      - LSTM architecture allows model to learn longer term dependencies

Data Cleanup and Model Training

We used the DataFrames we created for each asset in the prior project.

We created models for each asset, and then trained and tested each model.

We ran each model for 100 Epochs.

```
split = int(0.7 * len(X))
X_train = X[: split]
X_test = X[split:]
y_train = y[: split]
y_test = y[split:]
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
scaler.fit(y_train)
y_train = scaler.transform(y_train)
y_test = scaler.transform(y_test)
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))
print (f"X_train sample values:\n{X_train[:3]} \n")
print (f"X_test sample values:\n{X_test[:3]}")
```

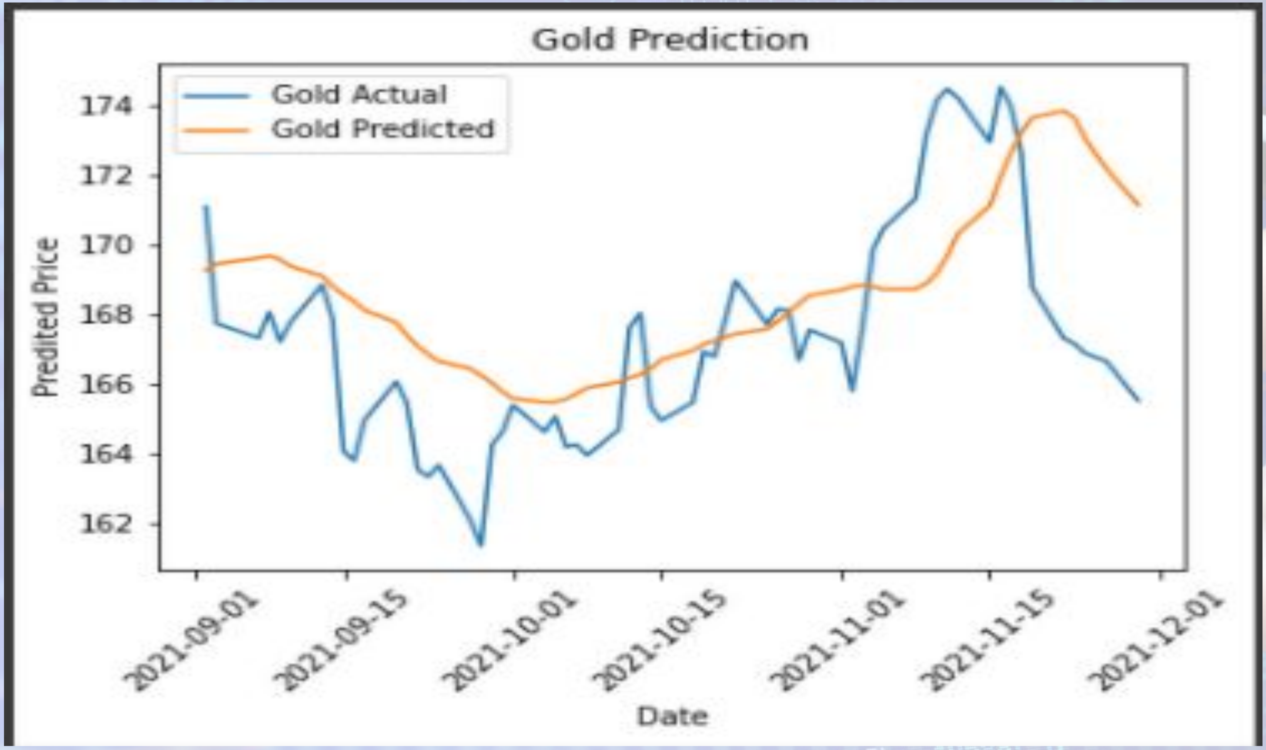
Model Evaluation - Gold

Mean Absolute Percentage Error (MAPE) is commonly used because it's easy to interpret and explain. For example, a MAPE value of 11.5% (.115) means the average difference between the forecasted and actual value is 11.5%. The lower the value for MAPE, the better a model is able to forecast values. Feb 27, 2020 – statology.org

```
from sklearn.metrics import mean_absolute_percentage_error
y_true = stocks["SPY Actual"]
y_pred = stocks["SPY Predicted"]
mean_absolute_percentage_error(y_true, y_pred)
```

Asset	**Mean Absolute % Error**
SPY	0.021370928133390049
GOLD	0.014784009823776893
BTC	0.06374552301709513
ETH	0.22953520557197069

Model Evaluation - Gold



Creating a LSTM

```
[28] # Create the LSTM RNN Model Structure
     # Define the LSTM RNN 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))
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 50, 30)            3840

 dropout (Dropout)           (None, 50, 30)            0

 lstm_1 (LSTM)               (None, 50, 30)            7320

 dropout_1 (Dropout)         (None, 50, 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
_________________________________________________________________
```

Making Sense of the Data

```
[17] #  Create the Features X and Target y Data
     def window_data(df, window, feature_col_number, target_col_number):
         """
         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 - window to predict Xt.
         It returns two numpy arrays of X and y.
         """
         X = []
         y = []
         for i in range(len(df) - window):
             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)
```

```
[31] #  Train the Model
     # Train the model
     model.fit(X_train, y_train, epochs=100, shuffle=False, batch_size=90, verbose=1)
```

```
[32] # Model Performance
     # Evaluate the model

     model.evaluate(X_test, y_test, verbose=0)
```

```
0.0222518064081669
```

```
[33] # Make Predictions
     # Make predictions using the testing data X_test
     predicted = model.predict(X_test)
```

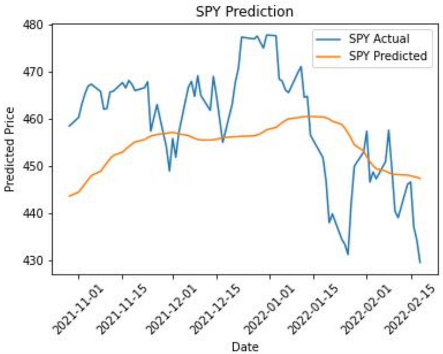
```
[34] # Recover the original prices instead of the scaled version
     predicted_prices = scaler.inverse_transform(predicted)
     real_prices = scaler.inverse_transform(y_test.reshape(-1, 1))
```

```
#  Plotting Predicted Vs. Real Prices
 # Create a DataFrame of Real and Predicted values
stocks = pd.DataFrame({
    "SPY Actual": real_prices.ravel(),
    "SPY Predicted": predicted_prices.ravel()
}, index = spy_historical.index[-len(real_prices): ])

# Show the DataFrame's head
stocks.head()
```

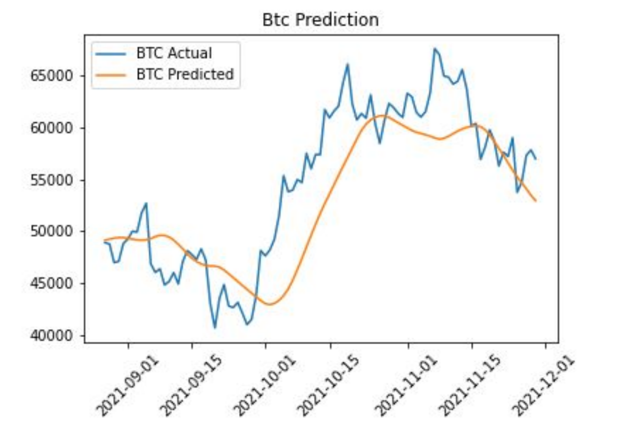
	SPY Actual	SPY Predicted
2021-10-29	458.429443	443.604523
2021-11-01	460.282898	444.471008
2021-11-02	463.093048	445.329987
2021-11-03	465.275391	446.190186
2021-11-04	466.889709	447.058197

```
[36]  # Plot the real vs predicted prices as a line chart
     stocks.plot(title="SPY Prediction", xlabel="Date", ylabel="Predicted Price",rot=45);
```



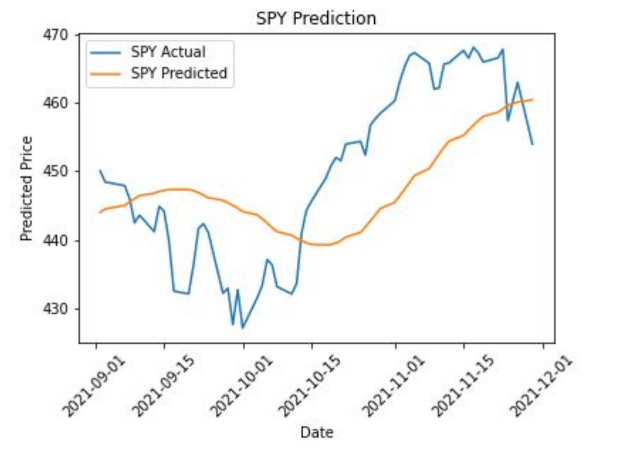
Model Evaluation - Original TimeFrame

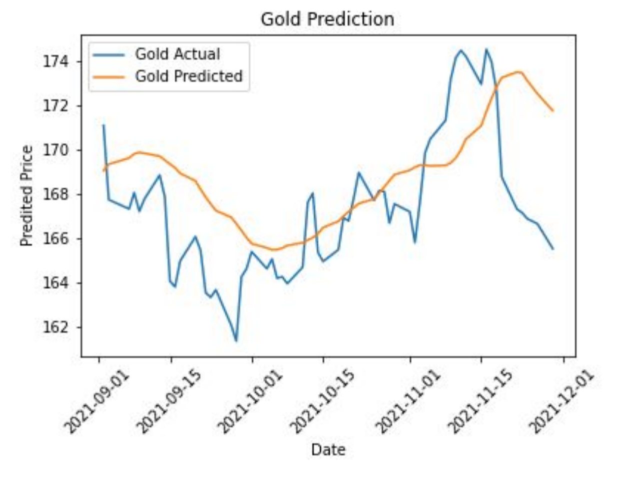
```
model.evaluate(X_test, y_test, verbose=0)
```




```
0.01722021773457527
```

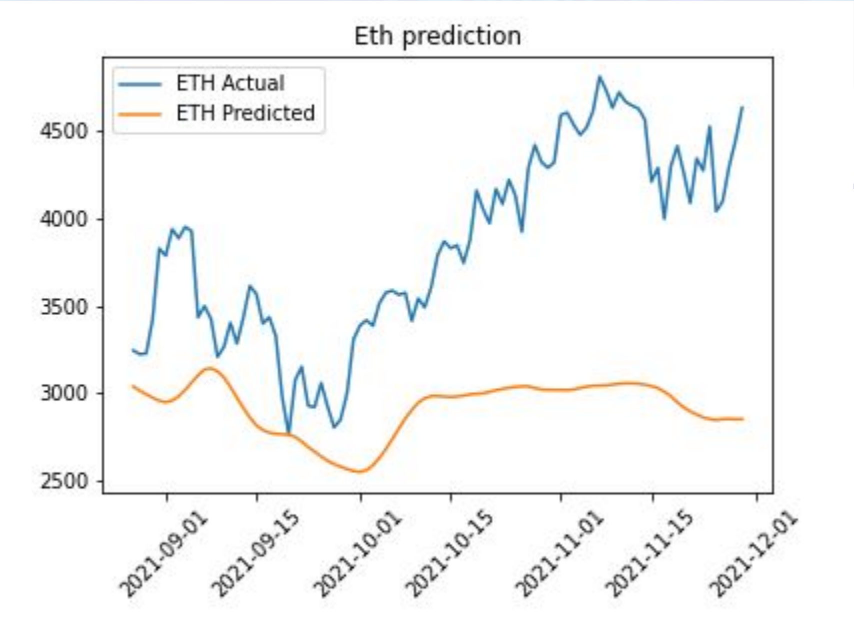
```
0.018557678908109665
```






```
0.020605500787496567
```

Model Evaluation - Original TimeFrame ETH

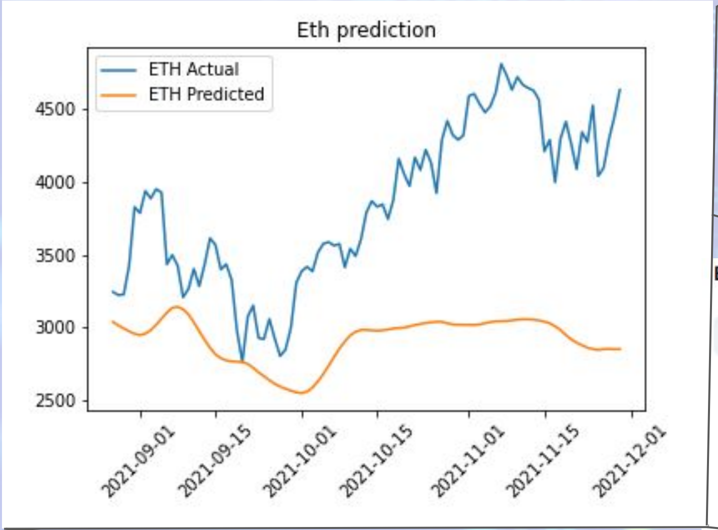



```
model.evaluate(X_test, y_test, verbose=0)
```

```
0.11493466794490814
```

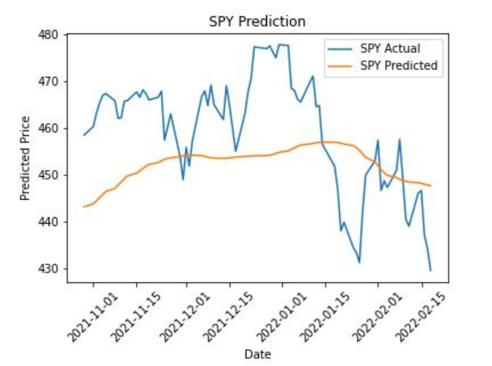
Jon →

Model Evaluation - ETH

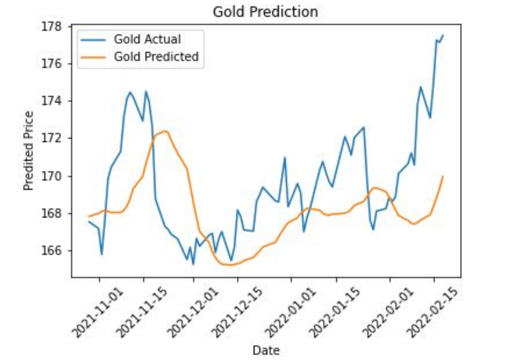


Model Evaluation - NEW TimeFrame

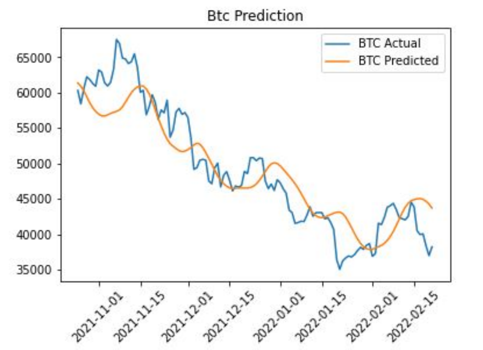
```
model.evaluate(X_test, y_test, verbose=0)
```



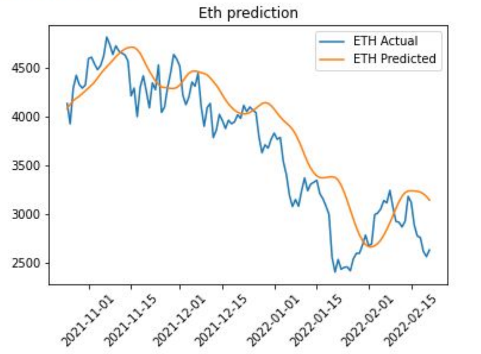
0.02613447047770023



0.025733275339007378

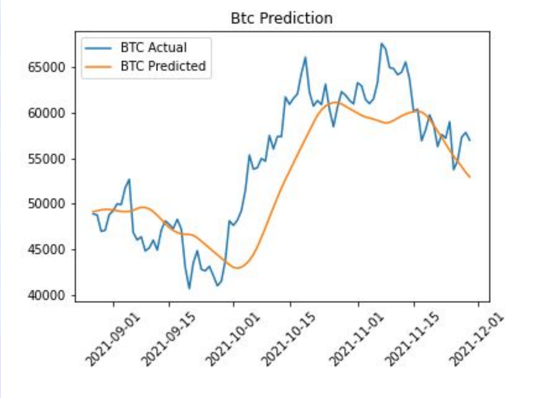


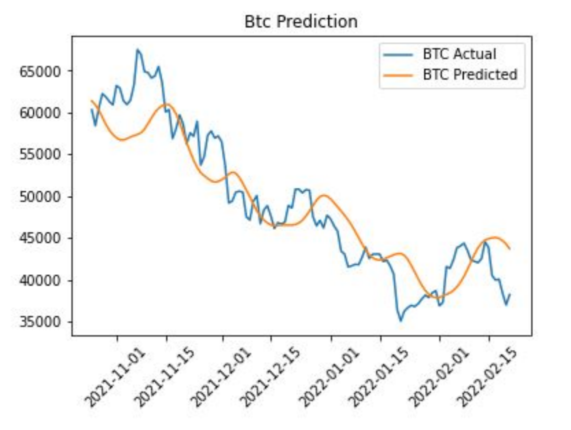
0.010466416366398335

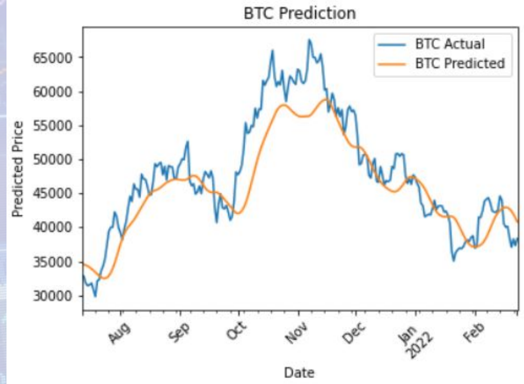


0.01303309015929699

Model Evaluation - BTC

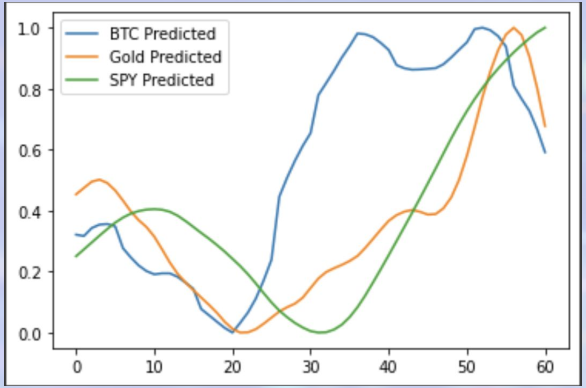






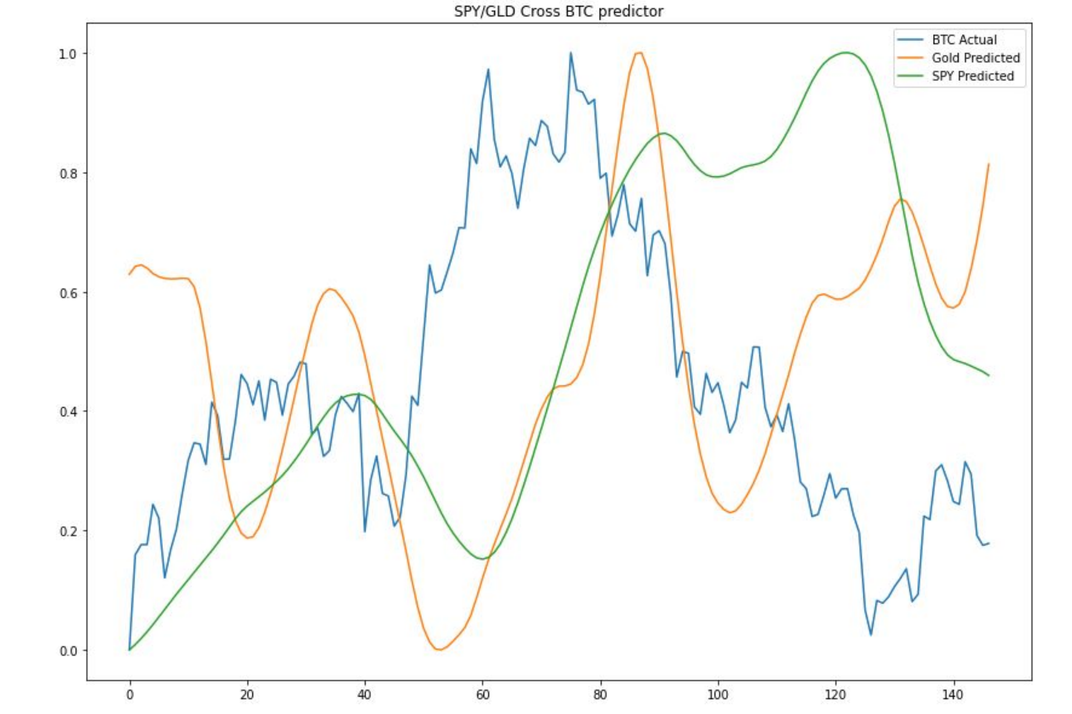
Trying to Predict BTC via Gold and SPY

```
scaled_needed_predictions.plot()
```



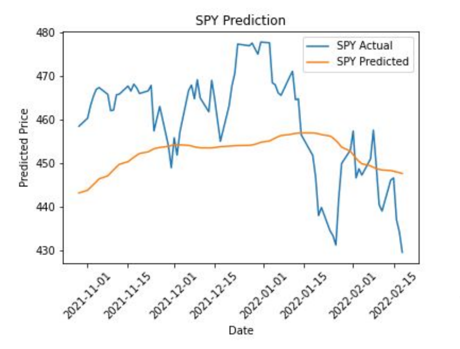

```
[138] andrei_spygld_btc_pred.plot(figsize=(14, 10), title='SPY/GLD Cross BTC predictor')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7ff1f2c00d50>

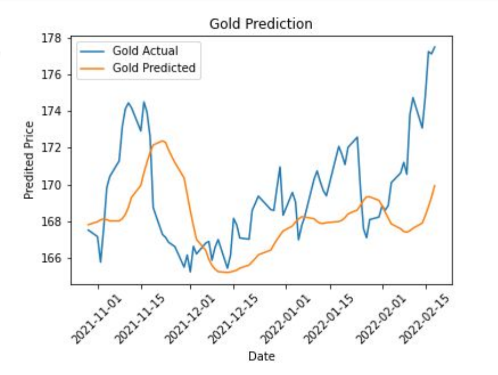


So you’ve got a model..Now What??

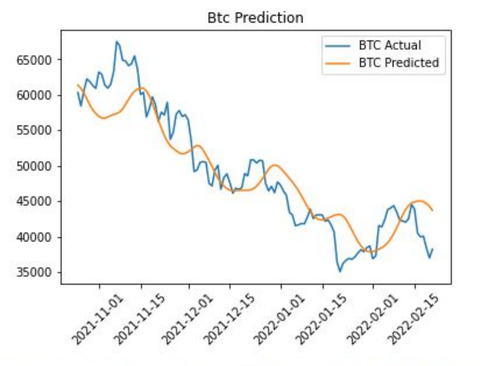
```
model.evaluate(X_test, y_test, verbose=0)
```



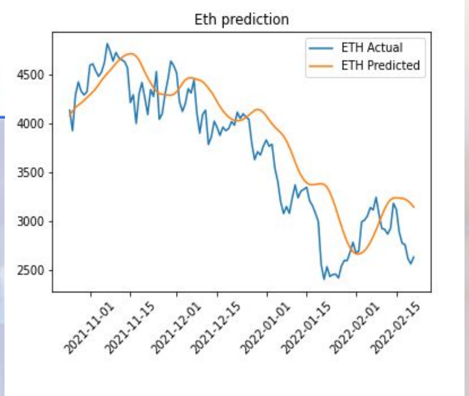
0.026134470477700233



0.025733275339007378

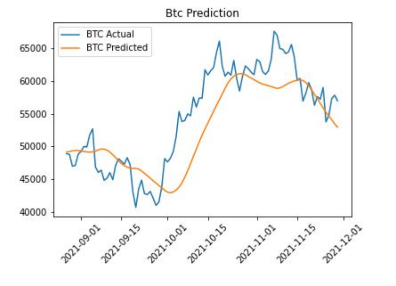


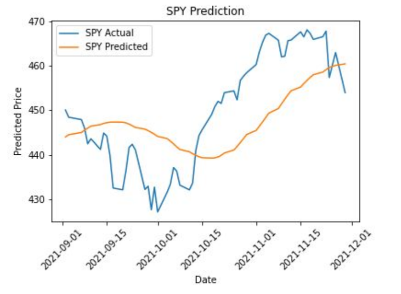
0.010466416366398335

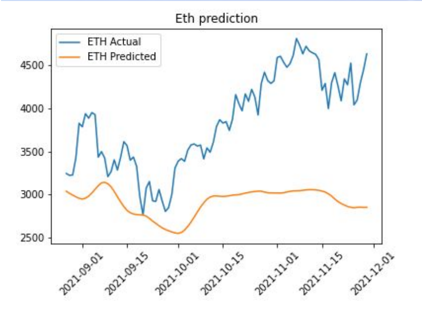


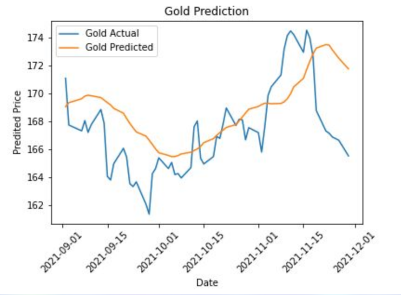
0.01303309015929699

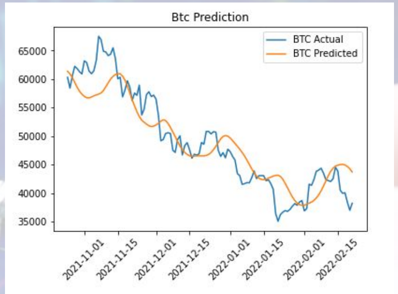
Bull vs. Bear Market

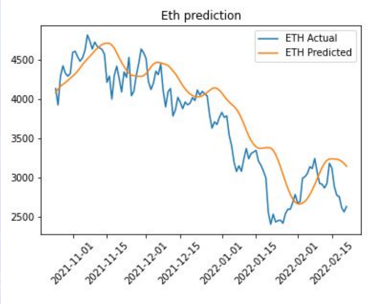


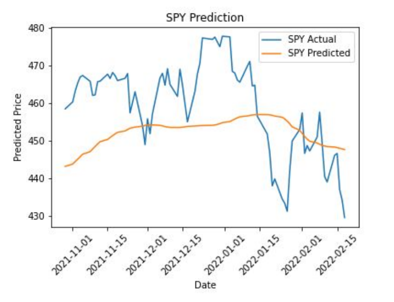


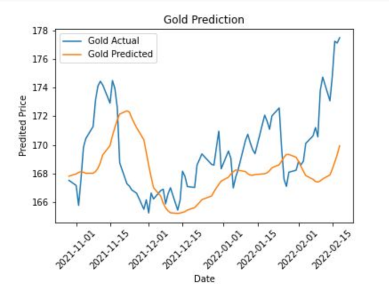




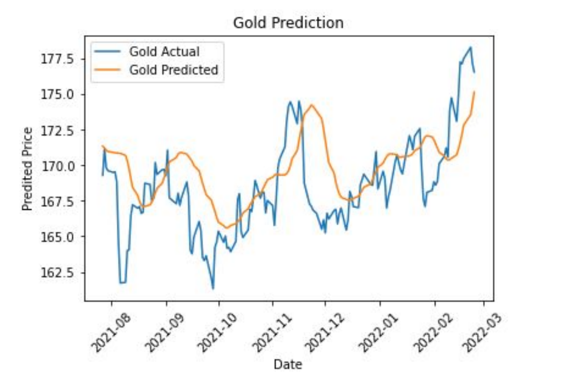


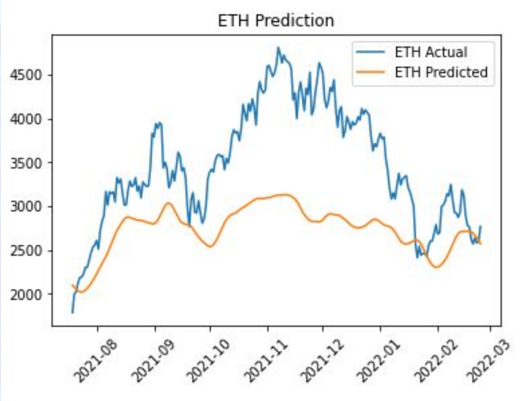


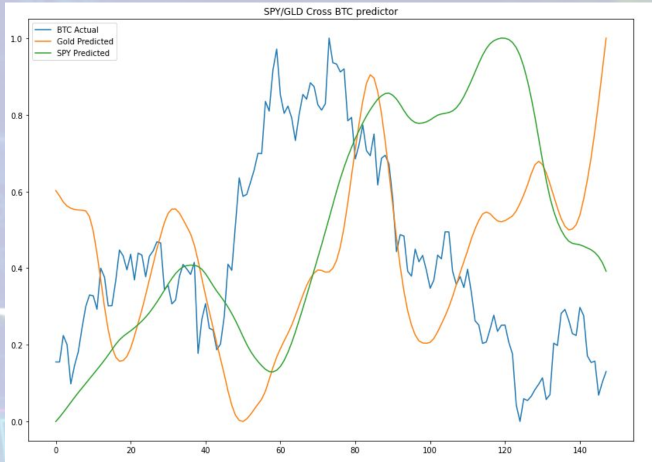




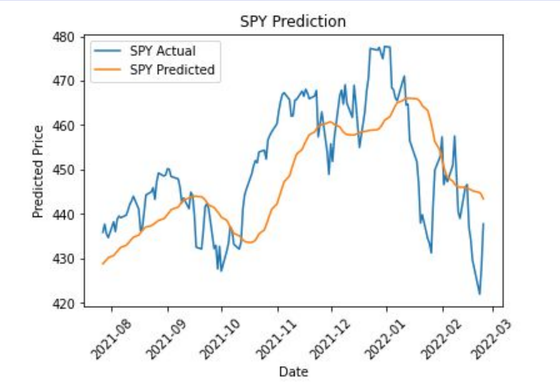
QUESTIONS????

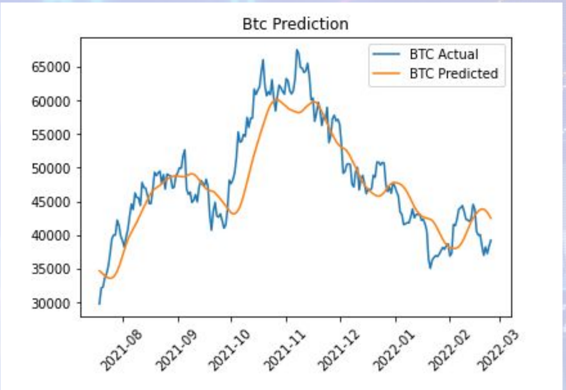






Most Recent Data






```
[133] from sklearn.metrics import mean_absolute_percentage_error
      y_true = stocks["SPY Actual"]
      y_pred = stocks["SPY Predicted"]
      mean_absolute_percentage_error(y_true, y_pred)

      0.020434298316266856

[134] from sklearn.metrics import mean_absolute_percentage_error
      y_true = gold["Gold Actual"]
      y_pred = gold["Gold Predicted"]
      mean_absolute_percentage_error(y_true, y_pred)

      0.014603110796334268

[135] from sklearn.metrics import mean_absolute_percentage_error
      y_true = bitcoin["BTC Actual"]
      y_pred = bitcoin["BTC Predicted"]
      mean_absolute_percentage_error(y_true, y_pred)

      0.06407523552875073

[136] from sklearn.metrics import mean_absolute_percentage_error
      y_true = ethereum["ETH Actual"]
      y_pred = ethereum["ETH Predicted"]
      mean_absolute_percentage_error(y_true, y_pred)

      0.17821599566673724
```

```
[69] # Create the LSTM RNN Model Structure
     # Define the LSTM RNN 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))
```