A Hybrid Approach of LSTM and NLP to Crypto Price Prediction

By:

Anita, Rakeshwer, Awais

Project’s Goal

- To accurately predict the trend if the price goes up or down and the actual price using different architecture.

- We have used 4 different models with different features and compared their effectiveness for our parameters and size of the data.

The Future of Money: Crypto Currencies are...

Distributed

Safe

Fast & Cheap

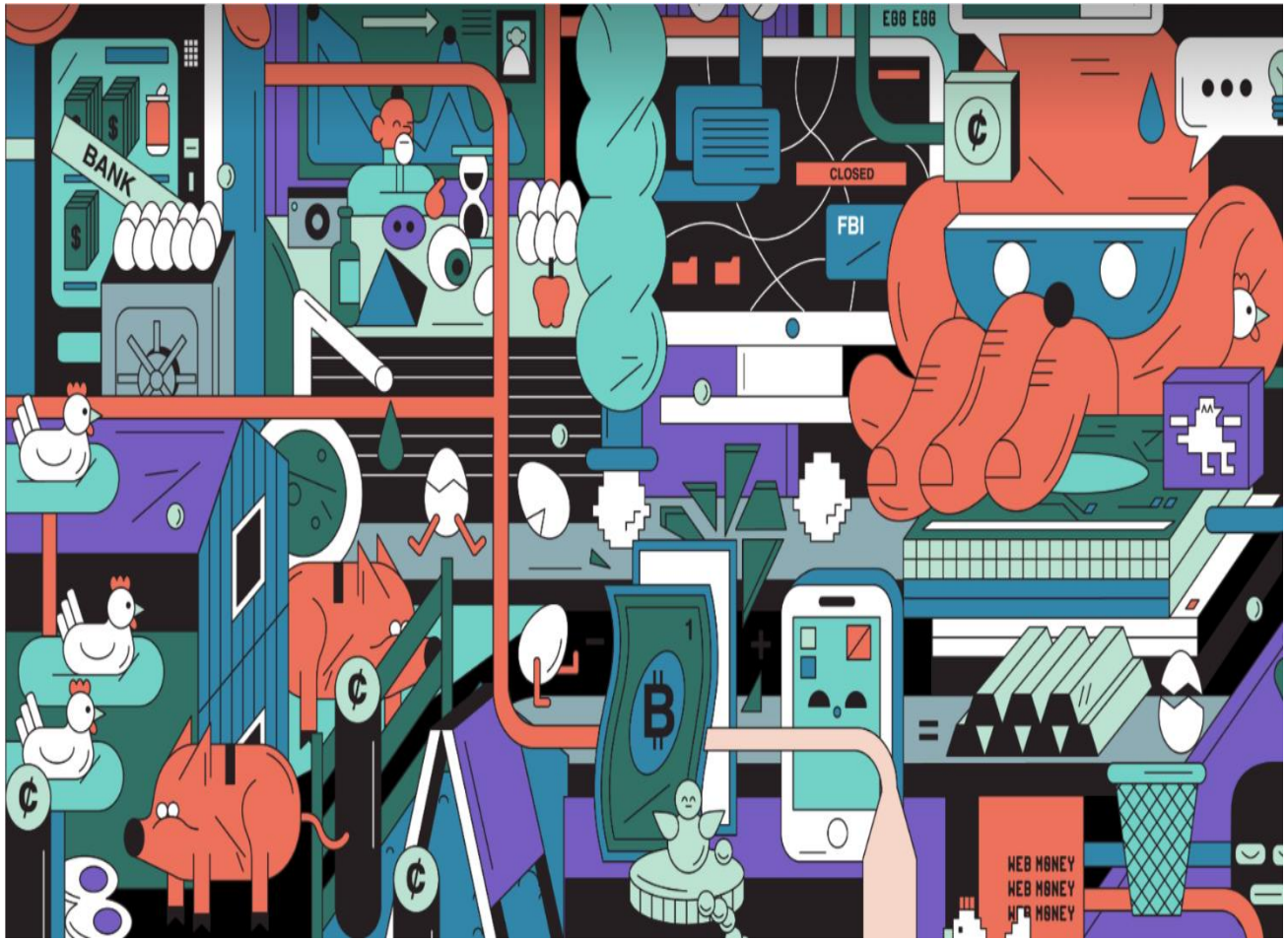
Blockgeeks

What makes the prediction a difficult task?

A volatile market with swinging prices

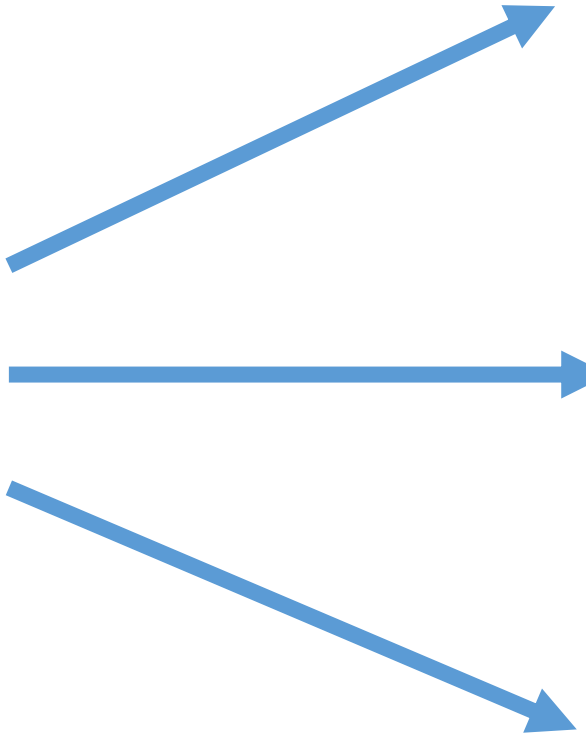
Key Factors

- Technical analysis
- Fundamental analysis
- Sentimental analysis



What we have tried?

Combining technical and sentimental features we designed three models



- A LSTM model with historical prices
- A NLP + FFNN & CNN/LSTM model with historical prices and sentiment scores
- A LSTM model with historical prices and Fear&Greed Index

Data Collection

- Historical Prices from Yahoo Finance API

  (contains High, Low, Open, Close, Volume, and Adj Close Price)

- Twitter Bitcoin dataset from Kaggle. (latest one)

  (contains more than 50000 tweets)

- Fear and greed indexes from bitcoinstools website

  (Index for year 2021)

Data Preprocessing

- The texts are used only for this study: Column header ‘tweets’
- Rows containing non-English tweets removed
- All currency signs, URLS, emoticons, #hashtags and @mentions remained

Tweets Sentiments

- Vader
- A commonly used tool for sentiment analysis.

TWEETS	NEG	NEU	POS	COMP
'RT @GracieBrowning Show Support for Autism Awareness https//tco/Y6kTWae5WI'	0	0.722	0.278	0.4019
'RT @Autistic_Mind who wants to be "cured" of my Autism I embrace my Autism and it is me Autism Speaks completely disregard thaΓÇª'	0.09	0.824	0.086	-0.024
'RT @xtinamurillo10 Supporting my older brother especially on this day so #LightItUpBlue and @autismspeaks today https//tco/MYΓÇª'	0	0.854	0.146	0.4404
'#memes #funny #cringe #offensive #fnaf #kek #autism #banter #dankmemes #edgy #filth #papafranku #furrypride #ΓÇª https//tco/3h2qS3alJI'	0	1	0	0
'Remember to not support Autism Speaks They're not for autism they're against it'	0.158	0.842	0	-0.3089

- Labeled data corpus for Sentiment Analysis
- *SentimentIntensityAnalyzer*: library for classifying into groups of sentiments
- Outputs:
  - Probability of positive, neutral and negative
  - Compound values in a range of -1 to 1, where -1 represents negative for each tweet
- The compound value is comparable to a single measure of polarity

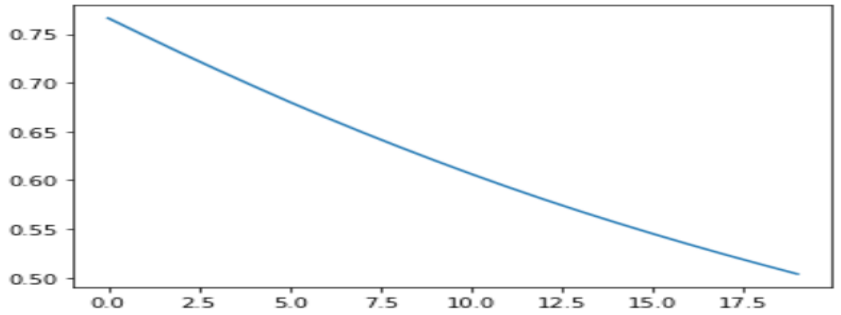
How ? We did?

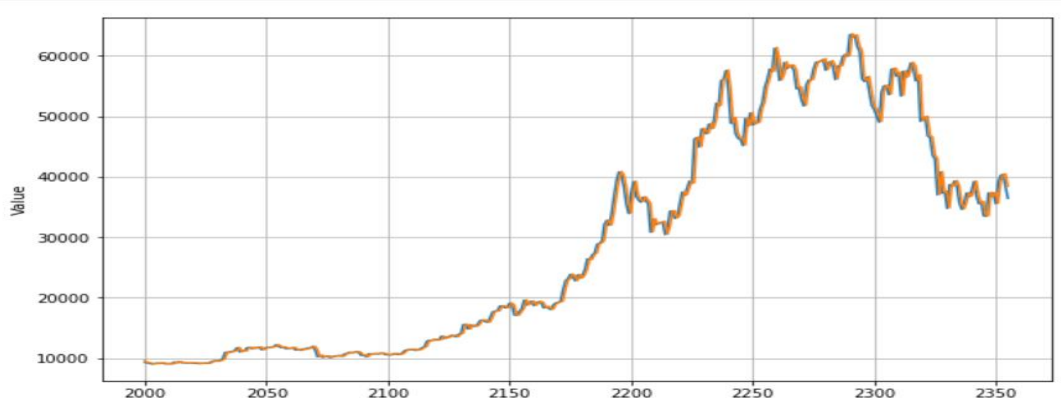
1. Data Collection from mentioned APIs and sites.
2. Preprocessed (took a lot of time)
3. Received the sentiments as pos (1) or neg(0)
4. Used sentiments as labels
5. Merged the two datasets crpto and sentiments.
6. Trained and predicted.

Feed Forward NN

- With this simple model of we have great great results.
- 2 layers
- Some of the models also use drop out to avoid over fitting in sequential model.
- But In our case does not improve the performance.

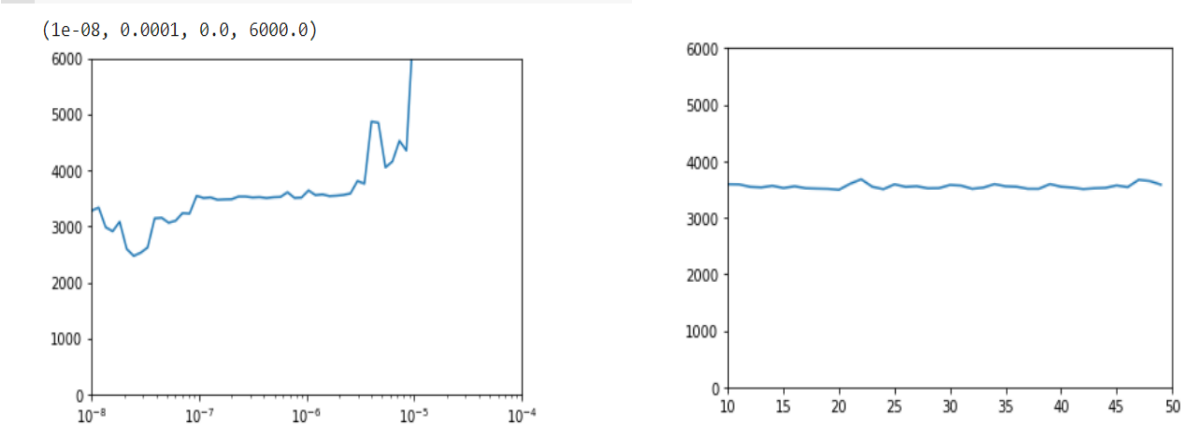
```
from tensorflow import keras
from tensorflow.keras import layers



model = keras.Sequential()
x_train, x_test, y_train, y_test = train_test_split(training_x, Y, test_size=0.25)

model.add(layers.Dense(64,input_shape=(1,)))
model.add(layers.Activation('relu'))

opt = keras.optimizers.Adam(learning_rate=0.00001)
model.compile(loss='mean_squared_error', optimizer=opt)


history = model.fit(x_train, y_train, epochs=20)
```





CNN + LSTM + Normal Layer

- Learning rate of 1e-6
- Conv1D : with 15/50/30 filters and 3/5 kernels , stides : 1 and padding is casual
- 2 layers LSTM with 30 layers
- 2 layers of Dense with 15 and 10 neurons
- Adjusting hyper parameters did not improve the model.
- The shown figure is of loss. Which is in fluctuation mode.
- Changing epochs also does not make the model more generous.



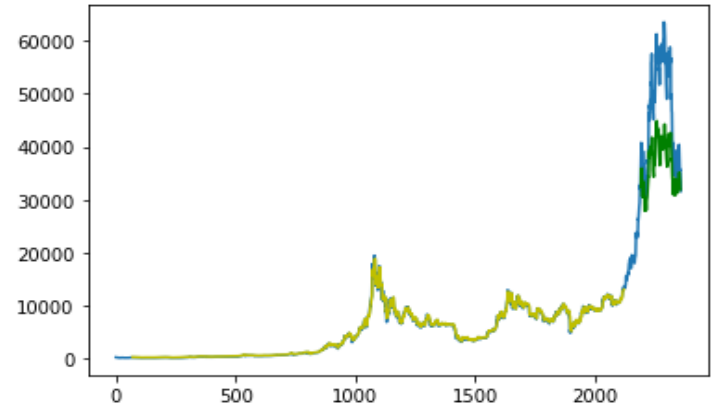

```
train_set = windowed_dataset(x_train, window_size=40, batch_size=64, shuffle_buffer=shuffle_buffer_size)
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv1D(filters=50, kernel_size=5,
                         strides=1, padding="causal",
                         activation="relu",
                         input_shape=[None, 1]),
  tf.keras.layers.LSTM(30, return_sequences=True),
  tf.keras.layers.LSTM(30, return_sequences=True),
  tf.keras.layers.Dense(15, activation="relu"),
  tf.keras.layers.Dense(10, activation="relu"),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 40000)
])
```

Prediction Model LSTM

- No significant correlation between general Tweets and price action.
- Volatility Clustering is exhibited by the data.
- Very short memory retention.
  - Significant differences in predicted vs actual values for longer windows.
  - Much better results for shorter window size.
- Results reinforce the autoregressive nature of the price action.

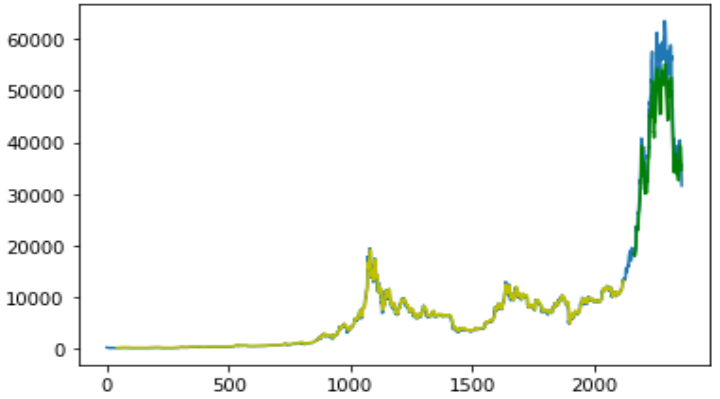
- Lookback: 70 days

```
model.summary()

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 25)                9600
_________________________________________________________________
dropout_1 (Dropout)          (None, 25)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 26
=================================================================
Total params: 9,626
Trainable params: 9,626
Non-trainable params: 0
_________________________________________________________________
```



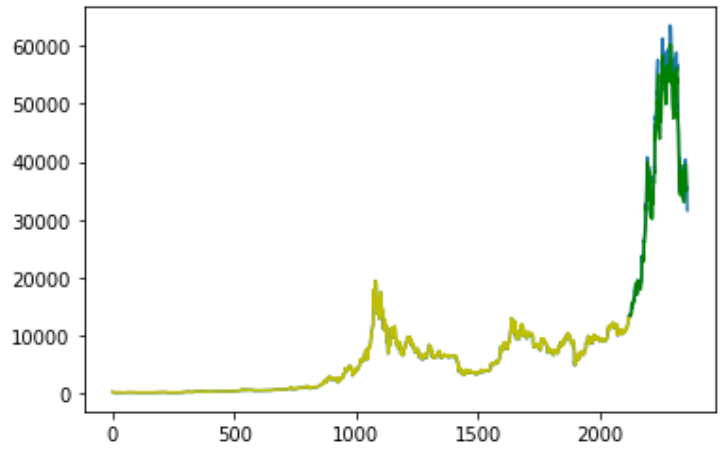
- Lookback: 42 days

```
model.summary()

Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 25)                6800
_________________________________________________________________
dropout_2 (Dropout)          (None, 25)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 26
=================================================================
Total params: 6,826
Trainable params: 6,826
Non-trainable params: 0
_________________________________________________________________
```



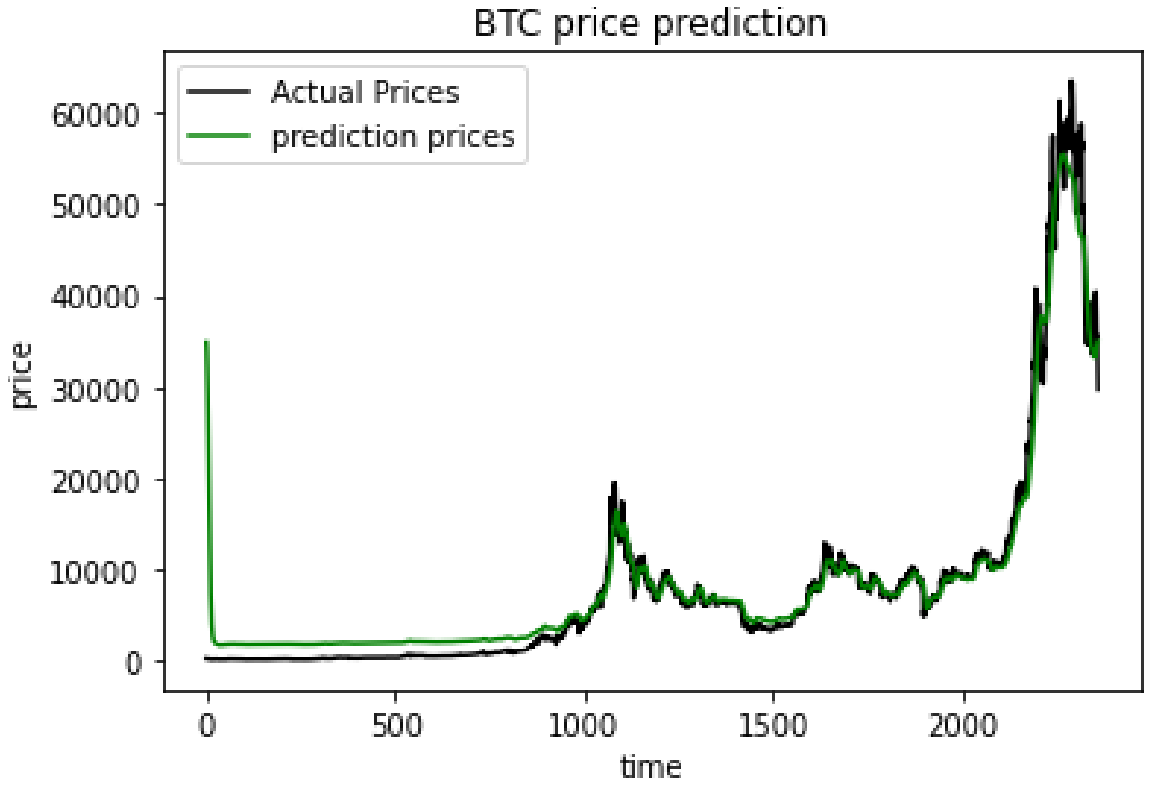
- Lookback: 1 day

```
model.summary()

Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_5 (LSTM)                (None, 25)                2700
_________________________________________________________________
dropout_5 (Dropout)          (None, 25)                0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 26
=================================================================
Total params: 2,726
Trainable params: 2,726
Non-trainable params: 0
_________________________________________________________________
```



LSTM (Best in our case)

- A model with the window size of 14-> lower performs outclass some how.
- 2 lstm layers, optimizer: “Adam”. Activation relu.
- It says : it will go down even in future.



Prediction Model with Fear&Greed Index

- **Fear&Greed Index:** volume, open interest, social media and search data
- **What success look like:** price goes up or down?
- **Inputs:** one year prices + Fear and Greed Index
- **Model architecture:** One layer LSTM and one dense layer, with 0.2 dropout and a dense output layer

```
[ ]  model = Sequential()
     model.add(LSTM(100, return_sequences=True,
                         input_shape=(None, normalized_df1.shape[-1]),
                         kernel_initializer='random_uniform'))
     model.add(Dropout(0.2))
     model.add(LSTM(100, dropout=0.0, activation='relu'))
     model.add(Dropout(0.2))
     model.add(Dense(50,activation='relu'))
     model.add(layers.Dense(2, activation='sigmoid'))
     model.compile(loss='mean_squared_error', optimizer='Adam',metrics=['accuracy'])

     es = EarlyStopping(monitor='val_loss', mode='min', verbose=1,
                              patience=50, min_delta=0.0001, restore_best_weights = True)

     history = model.fit_generator(train_gen,
                                   steps_per_epoch=2,
                                   epochs=100,
                                   validation_data=val_gen,
                                   validation_steps=val_steps,
                                   callbacks=[es])
```

Prediction Model with Fear&Greed Index

- **Result:** was not satisfying

```
test_loss, test_acc = model.evaluate_generator(test_gen, steps=3)
print('test acc:', test_acc)
print("test_loss:", test_loss)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/eng
  warnings.warn('`Model.evaluate_generator` is deprecated and '
test acc: 0.40625
test_loss: 0.2509402632713318
```

Conclusion

- In our case, LSTM performs well.
- Undoubtedly Other methods can works fine, but not as LSTM due to its nature of remembering values via gates architecture.
- CNN+LSTM should have worked well but the results were not satisfying. Adjusting hypermeters does not improve even the performance.
- Trying with different classification like fear and greed index makes us believe the LSTM model performs more rational in terms of financial data.