STAT 495R - Final

Cam Pitcher - Winter 2022

The Project and the Objective

How? - 2 APIs

Twitter API

- Get tweets every hour
- Sentiment Analysis
- 1 Week of Free Data

CoinGecko

- Get hourly price data
- Also gives total volume traded

Sentiment Analysis - What is it?

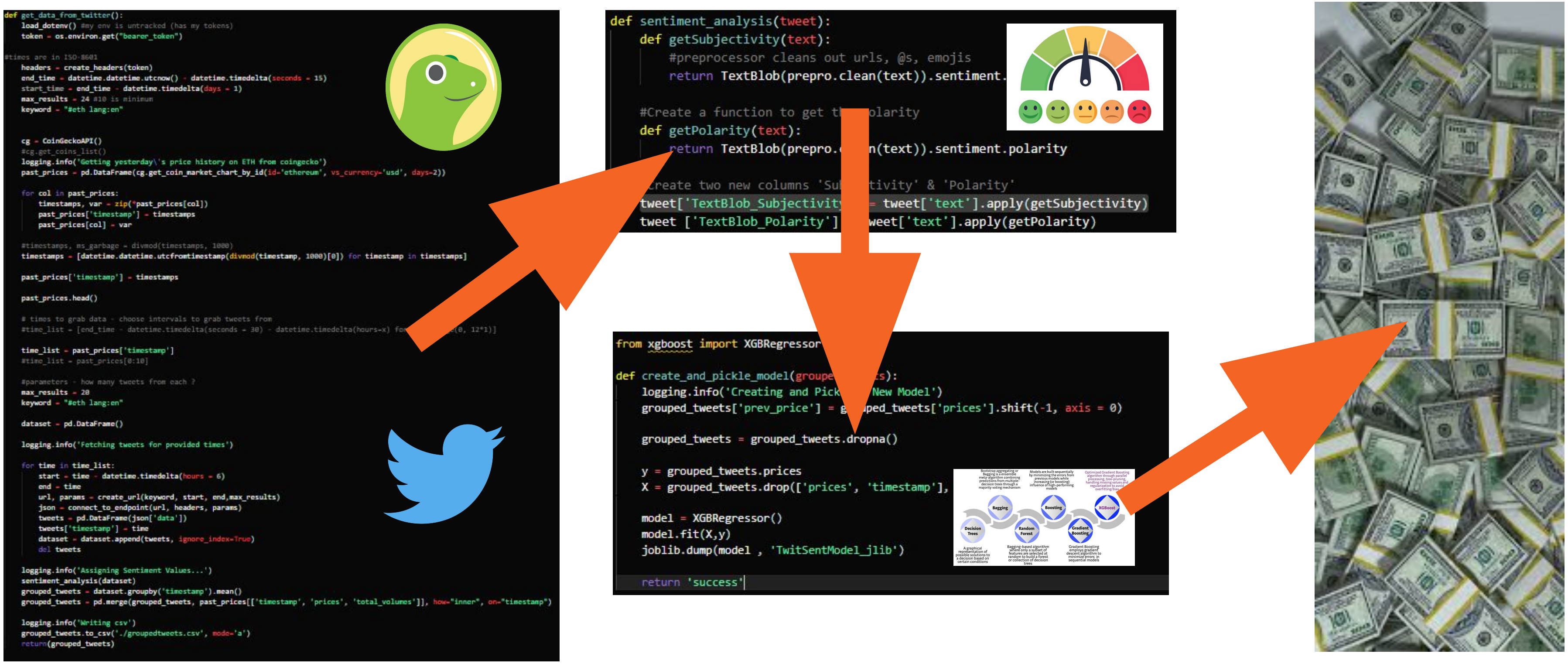
```
def sentiment_analysis(tweet):
    def getSubjectivity(text):
        #preprocessor cleans out urls, @s, emojis
        return TextBlob(prepro.clean(text)).sentiment.subjectivity

    #Create a function to get the polarity
    def getPolarity(text):
        return TextBlob(prepro.clean(text)).sentiment.polarity

    #Create two new columns 'Subjectivity' & 'Polarity'
    tweet['TextBlob_Subjectivity'] = tweet['text'].apply(getSubjectivity)
    tweet ['TextBlob_Polarity'] = tweet['text'].apply(getPolarity)
```

Shoutout libraries

Flow

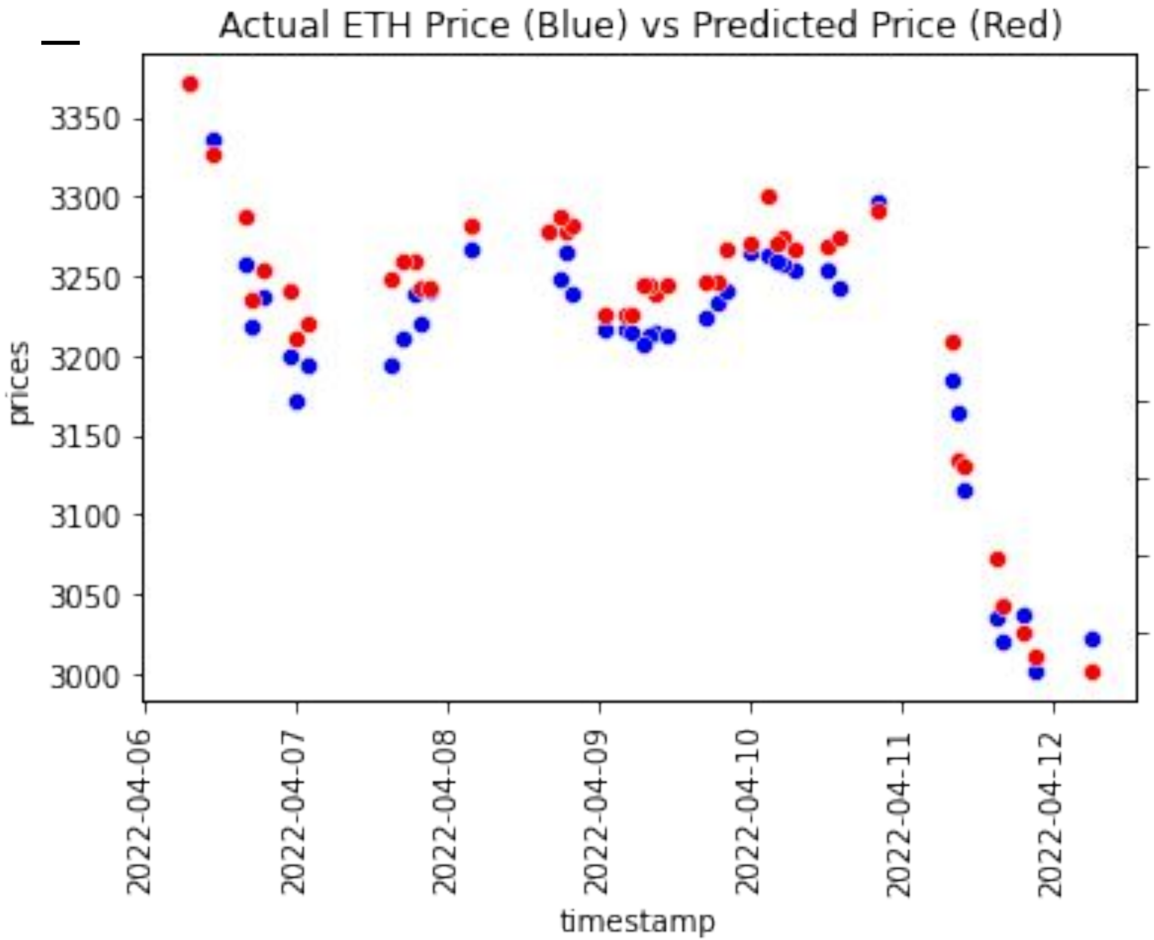

def get_data_from_twitter():
load_dotenv() #my env is untracked (has my tokens)
token = os.environ.get("bearer_token")
#times are in ISO-8601
headers = create_headers(token)
end_time = datetime.datetime.utcnow() - datetime.timedelta(seconds = 15)
start_time = end_time - datetime.timedelta(days = 1)
max_results = 24 #10 is minimum
keyword = "#eth lang:en"
cg = CoinGeckoAPI()
#cg.get_coins_list()
logging.info('Getting yesterday\'s price history on ETH from coingecko')
past_prices = pd.DataFrame(cg.get_coin_market_chart_by_id(id='ethereum', vs_currency='usd', days=2))
for col in past_prices:
timestamps, var = zip(*past_prices[col])
past_prices['timestamp'] = timestamps
past_prices[col] = var
#timestamps, ms_garbage = divmod(timestamps, 1000)
timestamps = [datetime.datetime.utcfromtimestamp(divmod(timestamp, 1000)[0]) for timestamp in timestamps]
past_prices['timestamp'] = timestamps
past_prices.head()
# times to grab data - choose intervals to grab tweets from
#time_list = [end_time - datetime.timedelta(seconds = 30) - datetime.timedelta(hours=x) fo
(0, 12*1)]
time_list = past_prices['timestamp']
#time_list = past_prices[0:10]
#parameters - how many tweets from each ?
max_results = 20
keyword = "#eth lang:en"
dataset = pd.DataFrame()
logging.info('Fetching tweets for provided times')
for time in time_list:
start = time - datetime.timedelta(hours = 6)
end = time
url, params = create_url(keyword, start, end,max_results)
json = connect_to_endpoint(url, headers, params)
tweets = pd.DataFrame(json['data'])
tweets['timestamp'] = time
dataset = dataset.append(tweets, ignore_index=True)
del tweets
logging.info('Assigning Sentiment Values...')
sentiment_analysis(dataset)
grouped_tweets = dataset.groupby('timestamp').mean()
grouped_tweets = pd.merge(grouped_tweets, past_prices[['timestamp', 'prices', 'total_volumes']], how="inner", on="timestamp")
logging.info('Writing csv')
grouped_tweets.to_csv('./groupedtweets.csv', mode='a')
return(grouped_tweets)
def sentiment_analysis(tweet):
def getSubjectivity(text):
#preprocessor cleans out urls, @s, emojis
return TextBlob(prepro.clean(text)).sentiment.
#Create a function to get t
olarity
def getPolarity(text):
return TextBlob(prepro.
n(text)).sentiment.polarity
reate two new columns 'Su
tivity' & 'Polarity'
tweet['TextBlob_Subjectivity
= tweet['text'].apply(getSubjectivity)
tweet ['TextBlob_Polarity']
weet['text'].apply(getPolarity)
from xgboost import XGBRegressor
def create_and_pickle_model(groupe
s):
logging.info('Creating and Pick
New Model')
grouped_tweets['prev_price'] = g
ped_tweets['prices'].shift(-1, axis = 0)
grouped_tweets = grouped_tweets.dropna()
y = grouped_tweets.prices
X = grouped_tweets.drop(['prices', 'timestamp'],
model = XGBRegressor()
model.fit(X,y)
joblib.dump(model , 'TwitSentModel_jlib')
return 'success'
Bagging
Boosting
XGBoost
Decision Trees
Random Forest
Gradient Boosting

The Model - XGBoost

- Extreme Gradient Boost
- Corrects when mistakes are made (boost)
- Ensemble method (multiple tiny models are fitted and work together to produce results)

Demo

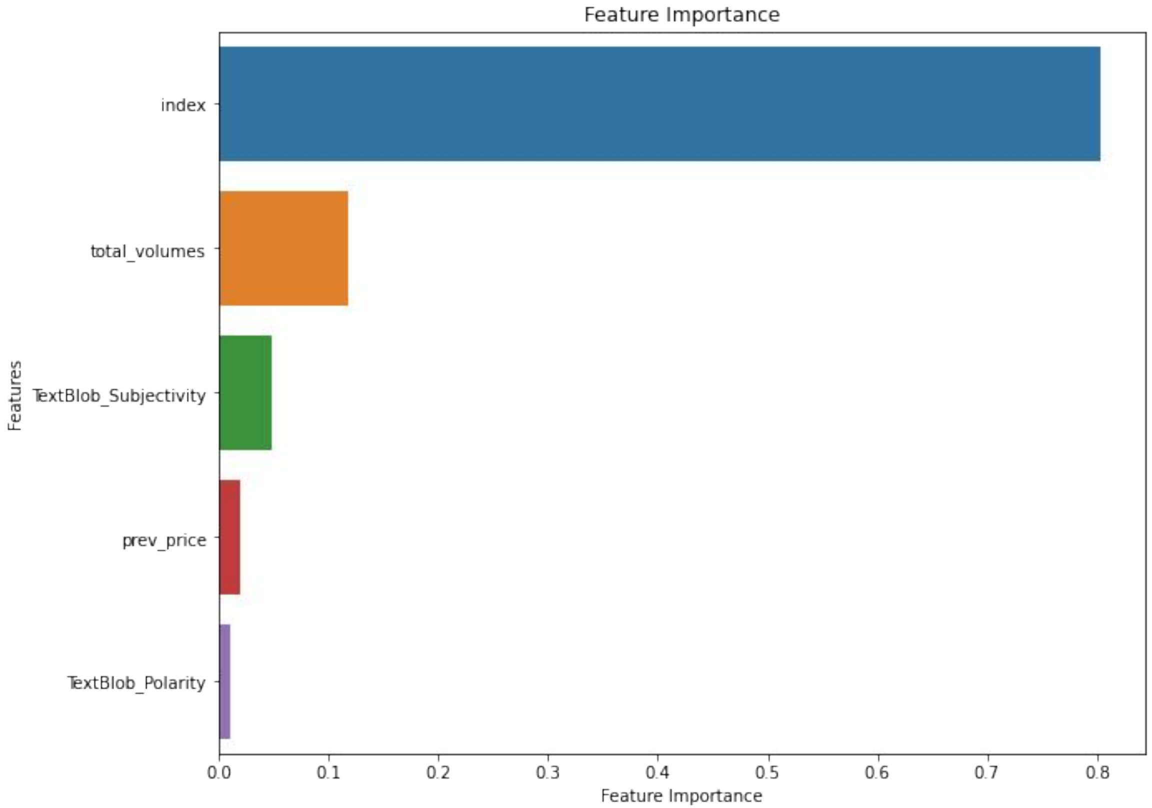
Predictions



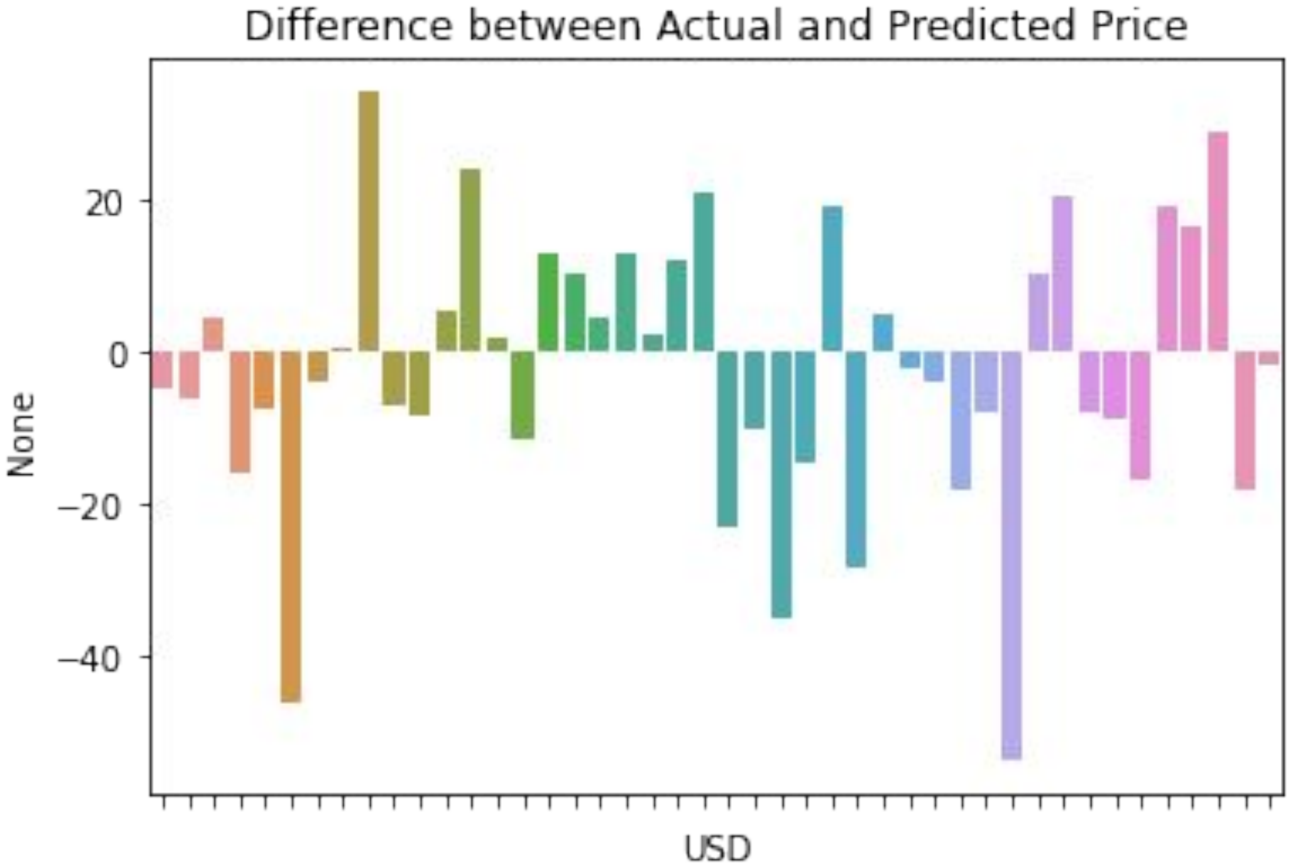

```
% of Predicted Increase (Hour to Hour
False     0.727273
True      0.272727
Name: pred_increase, dtype: float64

% of Actual Increase (Hour to Hour)
True      0.613636
False     0.386364
Name: actual_increase, dtype: float64

% of Correctly Predicted Movement Directions (Hour to Hour)
True      0.613636
False     0.386364
Name: correct_pred_direction, dtype: float64
```



The Ugly (What is Index?)



On this set of test data...

The mean difference between the model and the actual price when the correct direction was predicted is 9.9169698

The mean difference between the model and the actual price when the incorrect direction was predicted is 20.5810859

Conclusion

Could and would this make money?

Probably not

1. Hindsight is 2020
2. Crypto is volatile
3. The model is mediocre at best

Potential Improvements

1. More data
    a. XGBoost compensates heavily when mistakes are made as it makes predictions
    b. Twitter Free API restrictions
2. Better Preprocessing
    a. Remove certain alpha-numeric chars
    b. Retweets?
    c. Utilize Emoji instead of removing them
3. Experiment with other model types
4. Try different feature combos and add more data features
5. Analyze Sentiment of Tweets individually (don’t group)
6. Try different coins and tickers