CQF 2023 - FINAL PROJECT

DEEP LEARNING PROJECT - LSTM TO PREDICT BITCOIN UP/DOWN MOVEMENTS

AMEY ANANT PATIL

Contents

1 Introduction	3
2 Methods	4
2.1 Deep Learning	4
2.2 Sentiment Analysis	5
2.3 Technical Indicators	5
2.4 Python Libraries	7
3 Databases Processing, Feature Engineering, Data Analysis and Feature Selection	8
3.1 Datasets	8
3.2 Data Processing and Feature Engineering	8
3.3 Data Analysis	9
3.4 Feature Selection with Boruta	17
4 Model Training and Results	19
4.1 LSTM architecture	19
4.2 Hyperparameter Tuning and Compiling	19
4.3 Best Model	21
5 Results	24
5.1 Main Model - Classification	24
6 Conclusion	28

List of Figures

Figure 1	Simple Neural Network Structure. Source: Medium Article.	4
Figure 2	RNN. Source: IBM.	4
Figure 3	LSTM Structure. Source: Medium Article.	5
Figure 4	Query used.	8
Figure 5	BTC over time.	9
Figure 6	BTC Log Returns over time.	9
Figure 7	BTC Stationary over time.	10
Figure 8	Subreddits with the most BTC mentions.	10
Figure 9	Polarity distribution per Subreddit.	11
Figure 10	Words most frequently associated with BTC mentions on Reddit.	11
Figure 11	Volume of BTC mentions per day.	12
Figure 12	Aggregated Sum of Polarities of mentions per day.	12
Figure 13	Aggregated Mean of Polarities of mentions per day.	13
Figure 14	Aggregated Median of Polarities of mentions per day.	13
Figure 15	Bit coin Close x Volume of Reddit Mentions.	14
Figure 16	Bit coin Close x Mean of Polarity for Reddit Mentions.	14
Figure 17	Bit coin Close x Aggregated Sum of Polarity for Reddit Mentions.	15
Figure 18	Correlation Matrix - All variables	15
Figure 19	Correlation Matrix - Reddit Features.	16
Figure 20	Top 10 features most correlated to BTC returns.	16
Figure 21	Correlation matrix of Selected Features.	18
Figure 22	Histogram of maximum epochs reached per permutation.	21
Figure 23	Validation Cross Entropy Loss Distribution.	22
Figure 24	Validation Cross Entropy Loss Distribution.	22
Figure 25	Correlations: Metrics x Hyperparameters.	23
Figure 26	Metrics evolution over time.	24
Figure 27	Metrics evolution over time.	24
Figure 28	Confusion matrix.	25
Figure 29	Confusion matrix.	25
Figure 30	Metric evolutions over time.	27
Figure 31	Predicted x Observed for LSTM regression.	27

TABLE REFERENCE

Table 1	Labs used	7

ABSTRACT

Long Short Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) algorithm capable of learning both long and short dependencies over time. Therefore, it is known to perform well when time dependency is observed. This study aims to predict up/downward moves in Bit coin returns by using a multivariate LSTM classification with Reddit sentiment and technical indicators as features.

1 Introduction

The Efficient Markets Hypothesis (EMH), as expounded by Fama [1, 2], establishes that asset prices swiftly incorporate all accessible information, implying that price vacillations originate solely from unforeseeable new information. This perspective implies that asset prices adhere to a random walk model, making accurate predictions challenging—anticipating outcomes with over 50% accuracy seems implausible. However, Kahneman and Tversky's work [3] counters this notion, asserting that financial judgments are not solely based on rational fundamentals, but are equally influenced by emotions and perceived risks. In the contemporary era, the digital landscape has facilitated an overflow of both information and emotions via platforms like Twitter and Reddit.

In 2021, Twitter boasted a user base of 199 million, generating a staggering 500 million daily tweets, with one in every five American adults actively participating. The conciseness of tweets, limited to 280 characters, adds an additional layer of complexity. Simultaneously, Reddit hosted over 52 million daily active users, constituting another one in five American adults. Notably, Reddit comments can extend up to 40,000 characters, yielding an extensive wellspring of information. This surge in data is evidenced by the record of 2 billion comments in 2020, accounting for spam exclusions. This proliferation encompasses both news and sentiments, with hedge funds already leveraging social media as a barometer of market sentiment. Prominent instances, such as Elon Musk's tweets impacting cryptocurrency prices and the Gamestop Short Squeeze, underscore the persuasive influence of social media on asset dynamics. Furthermore, the accelerated dissemination of news online is exemplified by the 2009 US Airways plane crash, initially reported on Twitter rather than traditional television outlets, spotlighting the evolving landscape of information propagation.

Prior research by Nguyen and Shirai [10], Abraham et al [11], and Mohaptra et al [12] harnessed social media and news as proxies for market sentiment to anticipate asset shifts, yielding accuracy rates surpassing 50%. These findings propose that harnessing these contemporary data sources holds the potential for returns surpassing EMH's 50% benchmark.

This project derives inspiration from the aforementioned principles and insights from CQF lessons. Its core objective is to prognosticate daily upward/downward movements of Bitcoins. The predictive endeavor involves the application of technical indicators and Reddit sentiment as features within a Long Short-Term Memory (LSTM) Deep Learning architecture. Furthermore, a simplified LSTM regressor is constructed, employing sentiment as the singular feature.

The project framework is delineated across the following sections:

- Methods
- Database Processing, Feature Engineering, Data Analysis, and Feature Selection
- Model Training and Results
- Conclusion

All the code is available in the GitHub repository

https://github.com/ameypatil3737/Deep-Learning-LSTM-to-predict-up-down-of-scrip

2 Methods

In the section all the technical information used is explained, from Deep Learning to Python libraries used.

Deep Learning

Neural Networks stand as algorithms inspired by biological systems, mirroring the functionality of neurons. In this conceptual alignment, dendrites serve to receive information at synapses, while axons undertake the role of processing and transmitting this information to interconnected neurons. The fundamental architecture takes the form of an input-output mechanism, wherein input data is ingested, traverses through hidden layers for processing, and culminates in output layer revelations. Integral to this structure are connection weights, associated with each link, which undergo iterative adjustment. Information flow persists through these connections, steadily advancing until the output layer is reached. Here, activation functions exert their influence, orchestrating the transformation of received data into meaningful outputs.

Figura 1 – Redes Neurais

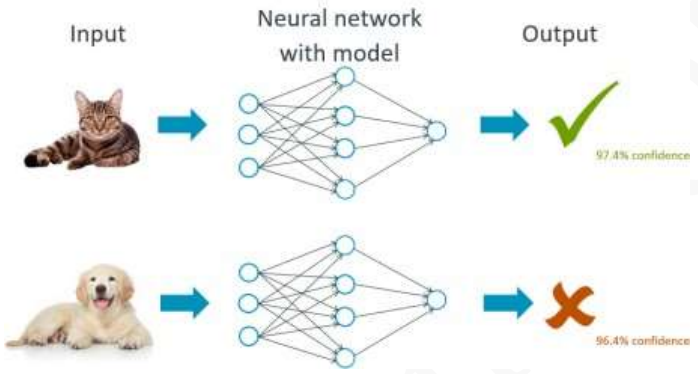

Figure 1: Simple Neural Network Structure. Source: Article.

In Neural Networks the neurons are independent and there is no memory (thus inputs and outputs are independent of one another), disallowing their usage for when the sequence matters. Rumelhart, Hinton and Williams (1996) [13] created the Recurring Neural Networks (RNN) model; RNN's name is self-explanatory and it circumvents vanilla Neural Networks limitations by adding *back propagation*: at each iteration the error is computed to adjust the connection weights retroactively, re-starting the process and minimizing the loss function (through gradient descent) until the input data can generate the outputs reasonably well.

Figura 2 – RNN x ANNs.

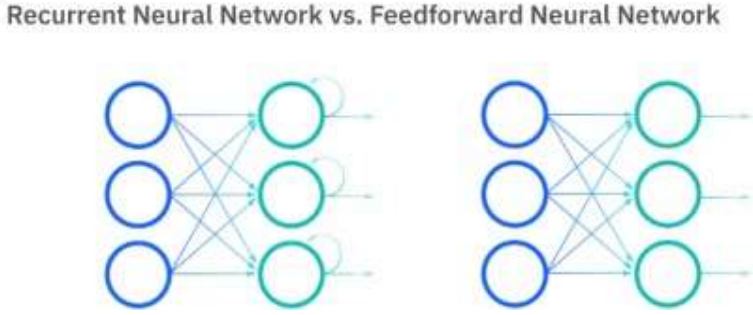

Figure 2: RNN. Source: IBM.

However, RNNs have a shortcoming: they have limited memory, failing when the problem involves carrying information over long series. Hochreiter and Shmidhuber (1997) [14] developed the Long Short Term Memory algorithm, a subclass of RNN capable of learning dependencies over long series. It comprises *forget gates* which are trained according to informational gain, filtering out which previous information should be discarded or propagated to the next layers. The figure below illustrates the idea.

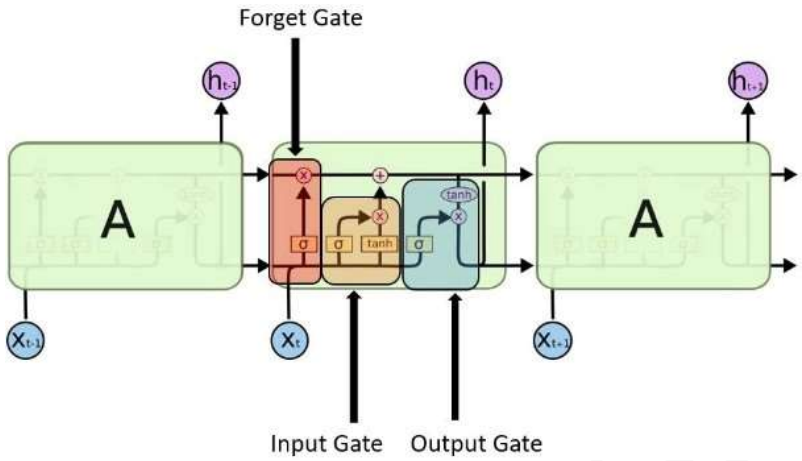

Figure 3: LSTM Structure. Source: Medium Article.

This study assumes Bit coin has a time dependent relationship; therefore the Long Short Term Memory algorithm is used.

Sentiment Analysis

Text data available in social media is non-structured. Meaning it is context-specific and made to be consumed by humans. The Natural Language Processing (NLP) field aims to yield information from such data, with many different methods to make unstructured data consumable by computers [11]. In NLP there are Senti- ment Analysis tools, aimed at extracting emotions from text with intricate data processing and machine learning algorithms. In a broad sense Sentiment Analysis can be divided into machine learning based approaches and *lexicon* based methods (dictionaries).

VADER (Valence Aware Dictionary and sentiment Reasoner) is one of the latter [15]: it is a lexicon specially calibrated to social media, boasting of good performance on Sentiment Analysis over Twitter, Facebook and Reddit data. It detects polarity (positive, neutral, negative) for each word and is also capable of interpreting internet-specific characters, such as emojis, symbols, contractions, slangs and CSS formatting. It alleviates the toil of processing text data too much before yielding sentiments. It was compared to other 11 sentiment analysis tools (2015) [15] and had the best performance. To analyze Reddit sentiment, this project uses VADER.

Technical Indicators

Besides sentiment, technical indicators were calculated in order to be used as features. The formulae are available in plenty of websites, but the used herein (all of which were implemented with TA-lib) are listed below. Look back periods of 5, 10, 21 and 50 were calculated for each, but the other parameters were set to their default values.

Exponential Moving Average:

$$EMA_t(p) = \alpha \cdot p_t + (1 - \alpha) \cdot EMA_{t-1},$$

Where α is the exponential smoothing, set to default value of 2 and the look back calculated for [5, 10, 21 50] D. It yields a trend, as the Moving Average, but placing greater height on more recent observations.

Average True Range:

$$ATR_t = \frac{ATR_{t-1} \times (n-1) + TR_t}{n}$$

$$TR = \max(high, close_{t-1}) - \min(low, close_{t-1})$$

which is basically a volatility measure.

Stochastic Oscillator:

$$\%K = 100 * \frac{(close_t - low_N)}{(high_N - low_N)}$$

is a momentum indicator based on resistance and support.

Moving Average Convergence Divergence:

$$MACD = EMA(M) - EMA(N))$$

is the difference between EMA over M days and EMA over N days. Here M and N are set to their usual values, 12 and 26, respectively.

Commodity Channel Index:

$$TP_t = p_t = \frac{high_t + low_t + close_t}{3},$$

$$CCI_t = \frac{1}{0.015} \cdot \frac{p_t - SMA_n(p_t)}{MAD_n(p_t)}$$

with SMA being the Simple moving average. It was designed to detect beginning and ending market trends.

Accumulation/Distribution Indicator:

$$A/D_t = A/D_{t-1} + MFM_t, \text{ where}$$

$$MFMt = \frac{Close - Low) - (High - Close)}{High - Low}$$

which gauges supply/demand to assess whether the stock is being accumulated or distributed.

Bollinger Bands:

$$\text{Upper Bollinger Band} = SMA(close, N) + \text{number of standard deviations} * st.dev(close, N)$$

$$\text{Lower Bollinger Band} = SMA(close, N) - \text{number of standard deviations} * st.dev(close, N)$$

yields price dynamics by gauging upper and lower bands using moving averages and standard deviations. N is changed for the previously mentioned look backs but the other parameters are kept at their default values (number of std deviations =2).

Momentum Indicator:

$$MOM = Price_t - Price_{t-N}$$

yields momentum, or simply the price difference between time t and N look backs.

Relative Strength Index:

$$\text{RSI} = 100 * \frac{\text{EMA}_N(\text{p}) \text{ of U}}{\text{EMA}_N(\text{p}) \text{ of U} + \text{EMA}_N(\text{p}) \text{ of D}}$$

U/D being, respectively, upward and downward changes over the period. If prices do not change, both are set to 0. It is a momentum indicator that captures both magnitude and velocity of price movements.

Drift-Independent Volatility Estimator:

Created by Yang and Zhang [16], it is considered one of the best measures for stock price volatility, with the added feature of considering overnight jumps. The formula is written below

```
def get_hvol_yz(data, look
    back=10): o = data.Open
    h = data.High
    l = data. Low
    c = data. Close

    k = 0.34 / (1.34 + (lookback+1)/(lookback-1))
    cc = np.log(c/c.shift(1))
    ho = np.log(h/o)
    lo = np.log(l/o)
    co = np.log(c/o)
    oc = np.log(o/c.shift(1))
    oc_sq = oc**2
    cc_sq = cc**2
    rs = ho*(ho-co)+lo*(lo-co)
    close_vol = cc_sq.rolling(look back).sum() * (1.0 / (look back -
    1.0)) open_vol = oc_sq.rolling(look back).sum() * (1.0 / (look back -
    1.0)) window_rs = rs.rolling(look back).sum() * (1.0 / (look back -
    1.0))
    result = (open_vol + k * close_vol + (1-k) * window_rs).apply(np.sqrt) * np.sqrt(252)
    result[:lookback-1] = np.nan

    return result * 100
```

Apart from the drift-independent volatility, all the indicators were readily available in the TA-lib library.

Python Libraries

Lib	Usage
TA-lib	Provides the main technical indicators easily
Fast Text	NLP to identify language
VADER	Sentiment Analysis toolkit
Tensorflow	Besides being a framework for building DL, with CUDAS, CUDNN and Keras allow DL algorithms to run on GPU
Talos	Simple and effective hyperparameter tuning tool
Boruta	Feature selection library that builds upon other FS methods (such as DTR), enhancing the selection by generating shadow copies of the data and testing for information gain

Table 1: Labs used

3 DATABASE PROCESSING,FEATURE ENGINEERING,DATA ANALYSIS AND FEATURE SELECTION

Datasets

Two datasets were used: a Reddit Comments BigQuery Dataset and Bit coin OHLC daily dataset. The first has 1.7 billion comments from 2015 to 2019 and the latter has daily OHLC Bit coin in USD from 2014 to 2021. No adjustments were required in the latter.

In order to retrieve only comments related to Bit coin, the following query was used on the BigQuery database:

```
SELECT
  subreddit,
  created_utc,
  body,
  score
FROM
  [reddit-btc-analysis:comments.reddit_btc_comments]
WHERE
  (LOWER(body) LIKE '% bitcoin%'
    OR LOWER(body) LIKE '% bitcoin %'
    OR LOWER(body) LIKE '% btc.%'
    OR LOWER(body) LIKE '% btc %')
```

Figure 4: Query used.

Which yielded a database with over 5 million comments mentioning Bit coin.

Data Processing and Feature Engineering

Within the confines of this notebook, an extensive sequence of operations was executed to extract and harness Reddit sentiments. This intricate process, which spanned over a laborious 12-hour period through parallel processing techniques, was orchestrated through the following steps:

Removal of Special Characters and Formatting: Preliminarily, the elimination of special characters and superfluous formatting components was carried out. This refinement streamlines the text data for subsequent analysis.

Language Identification with Fast Track: Fast Track language identification was employed as a discerning mechanism to retain solely English text. This was essential to adhere to the language constraints of the VADER sentiment analysis framework.

VADER Compound Polarity Calculation: VADER, a well-recognized sentiment analysis tool, was harnessed to gauge the compound polarity of each comment. This facilitated the quantification of sentiment intensity for individual comments.

Weighted Polarity Assessment: A nuanced approach was adopted by integrating the comment score (computed as up votes minus downvotes) to weigh the polarity score. This stratagem aimed to encapsulate the influence of the comment's sentiment within the context of its popularity.

The gamut of technical indicators, spanning various look back periods including 5, 10, 21, and 50 days, was computed utilizing TA-lib. In parallel, the target variable denoting upward/downward return movements was meticulously crafted. Notably, the computation of drift-independent volatility was meticulously executed, leveraging the bespoke function outlined in the Method section.

To synthesize both the Reddit sentiments and technical indicators, a pivotal consolidation phase was initiated. The final Reddit dataset underwent aggregation, coalescing on a daily basis with the Bit coin time series. During this critical aggregation juncture, pivotal statistical measures such as mean, median, sum, and the volume of comments were meticulously computed for the polarity scores.

Data Analysis

A thorough Data Analysis was performed to check for trends and relations between variables.

Bitcoin's price behaved as follows during the considered time span:

Figure 5: BTC over time.

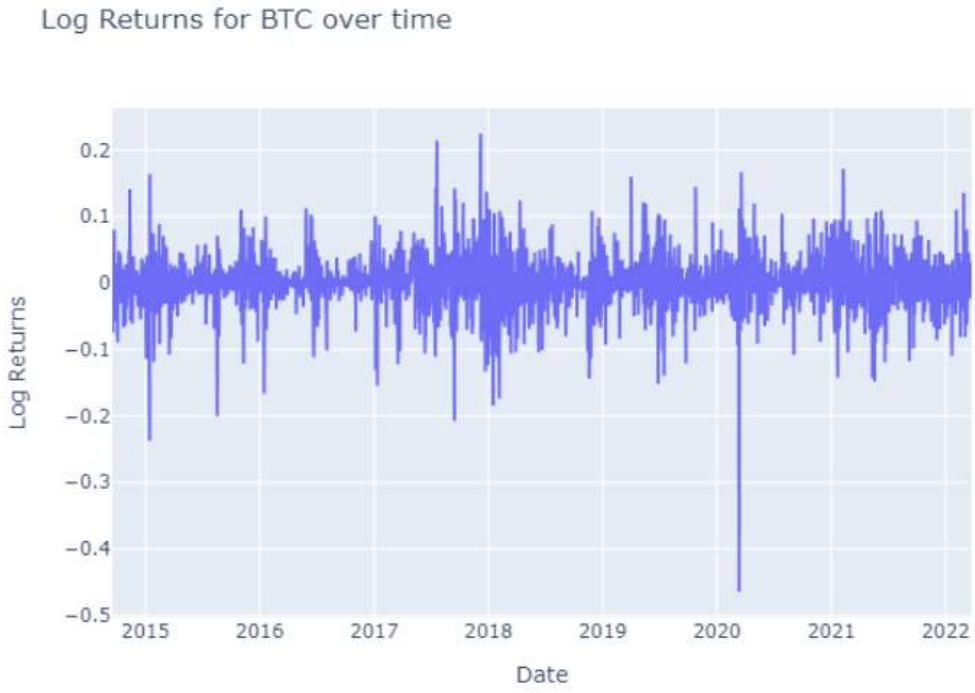

Figure 6: BTC Log Returns over time.

Figure 7: BTC Stationary over time.

It is clear that its series is far from stationary, with both moving average and standard deviation being far from constant, which instantly discards the usual ARIMA time series algorithms as viable alternatives. Then, the Reddit Sentiment data was analyzed:

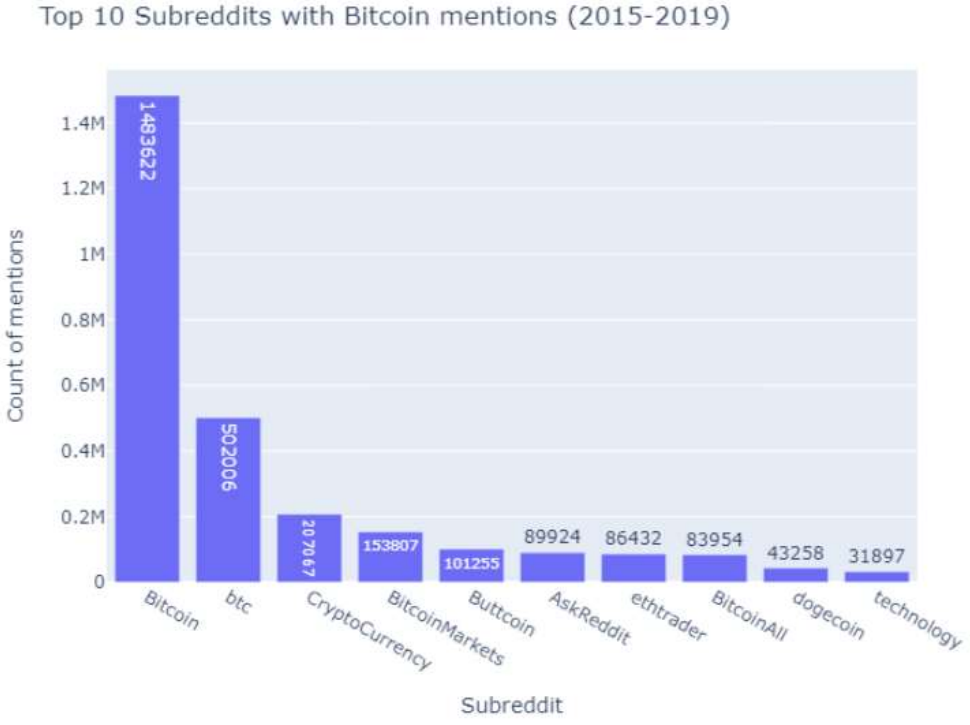

Figure 8: Subreddits with the most BTC mentions

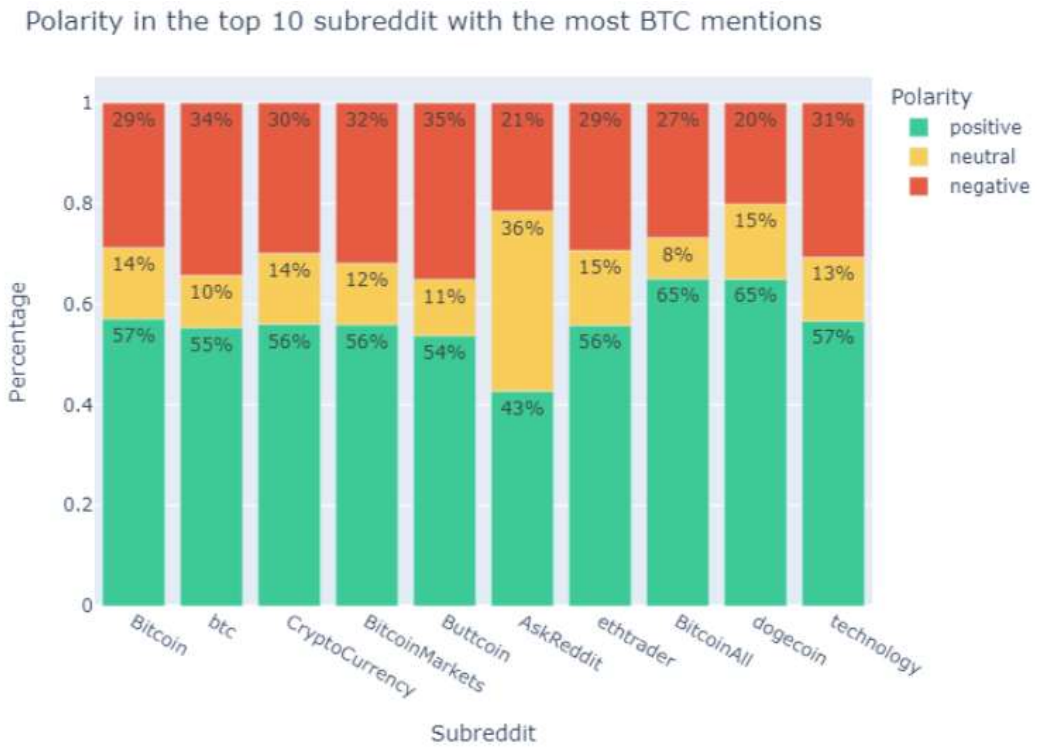

Figure 9: Polarity distribution per Subreddit.

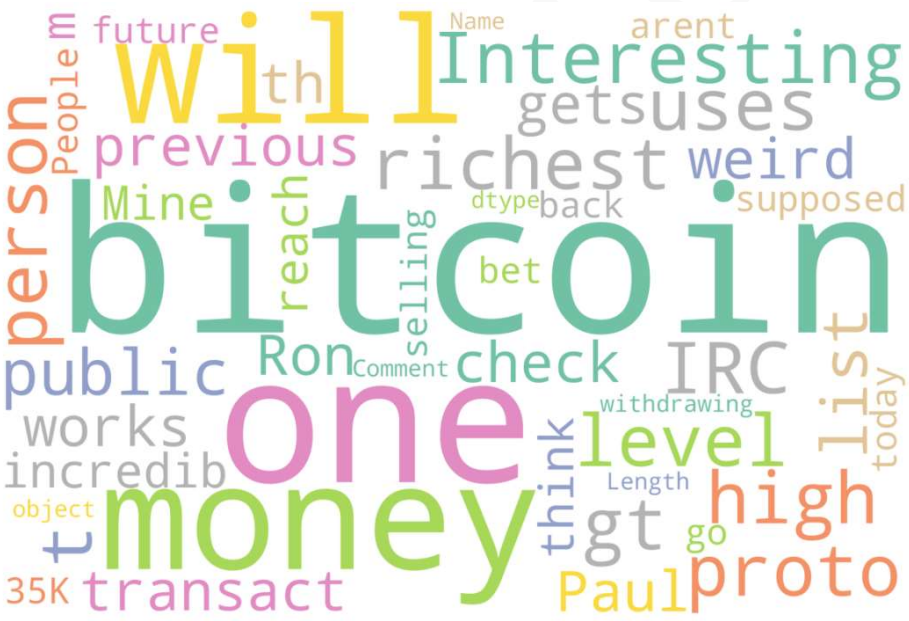

Figure 10: Words most frequently associated with BTC mentions on Reddit.

The word cloud shows words related to people seeking help ("money", "Interesting") and words related to regulations and opinions, proving it could be an effective proxy for market sentiment. Furthermore, some of the top 10 subreddits with the most mentions to Bit coin seem to be positively biased, which may be because some of those attract people that tend to see it as a value reserve instead of a speculative asset to profit from. Aggregating sentiment polarities by day, the following statistics are obtained:

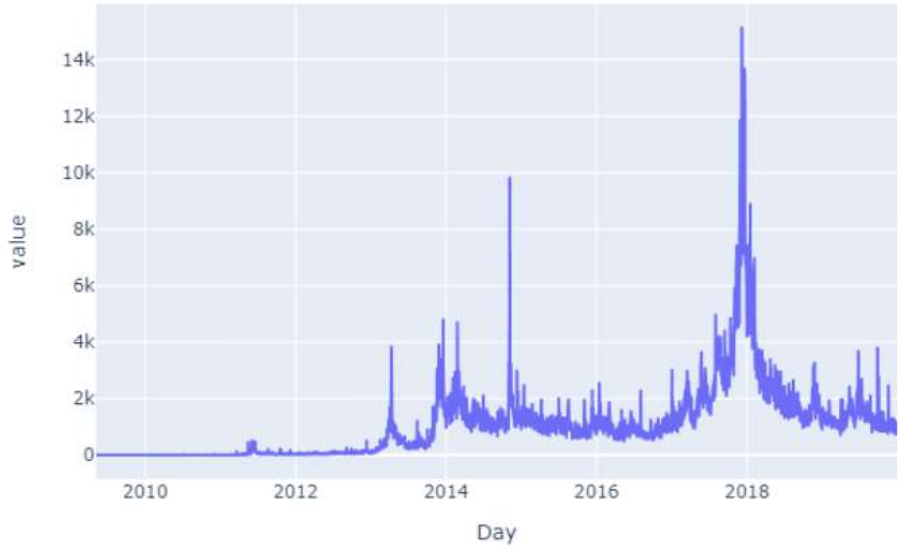

Figure 11: Volume of BTC mentions per day.

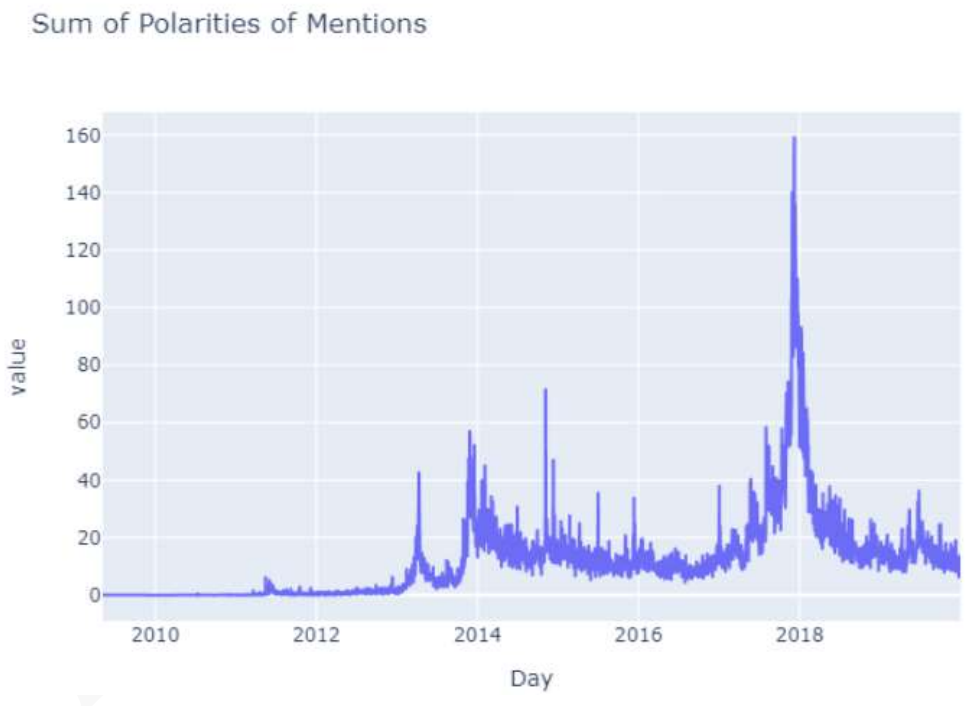

Figure 12: Aggregated Sum of Polarities of mentions per day.

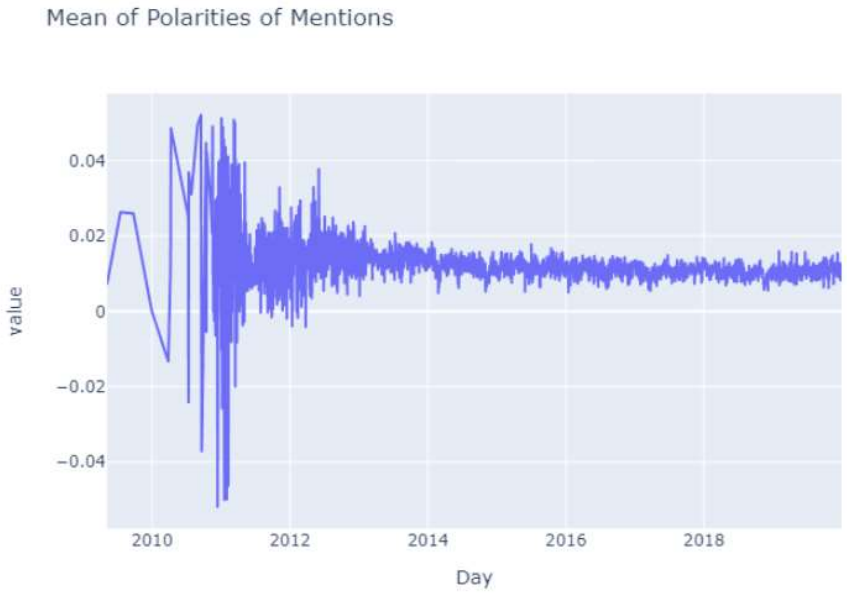

Figure 13: Aggregated Mean of Polarities of mentions per day.

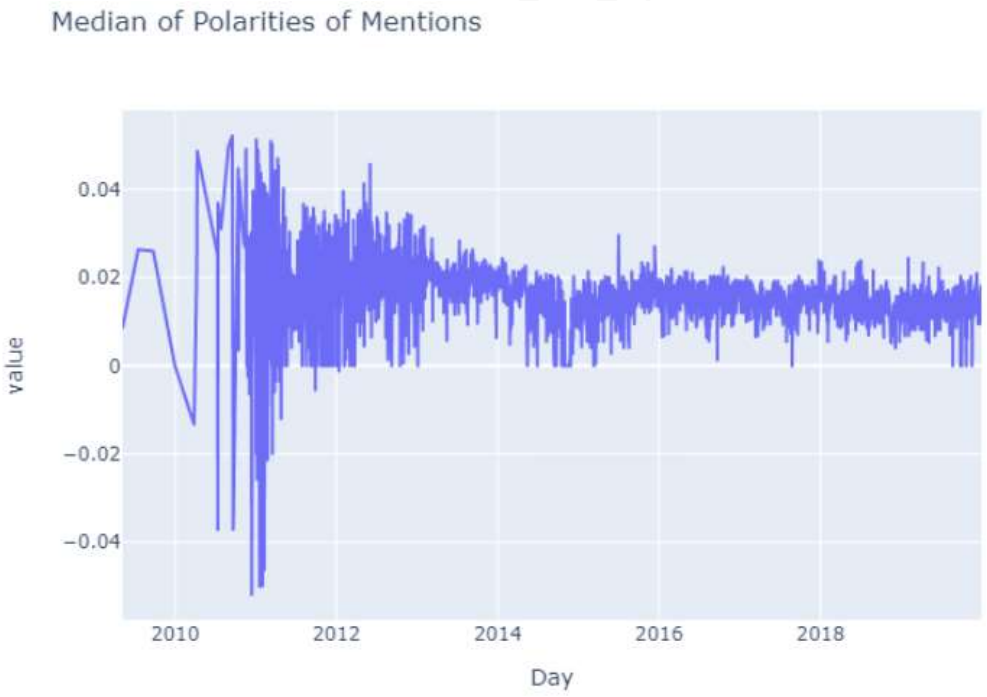

Figure 14: Aggregated Median of Polarities of mentions per day.

The mean daily polarity (already weighted by influence, check Feature Engineering section) shows a lot of volatility, but the aggregated sum o weighted sentiments seems to follow the volume of mentions evolution. Further analysis is done to check how both sentiment and Bit coin moved over time:

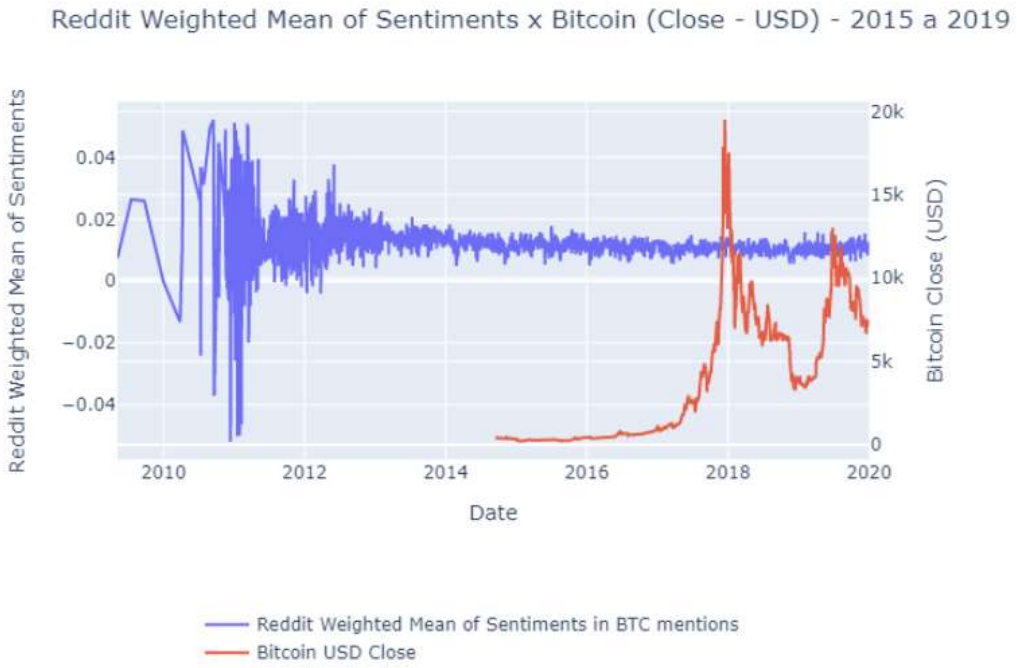

Figure 15: Bit coin Close x Volume of Reddit Mentions.

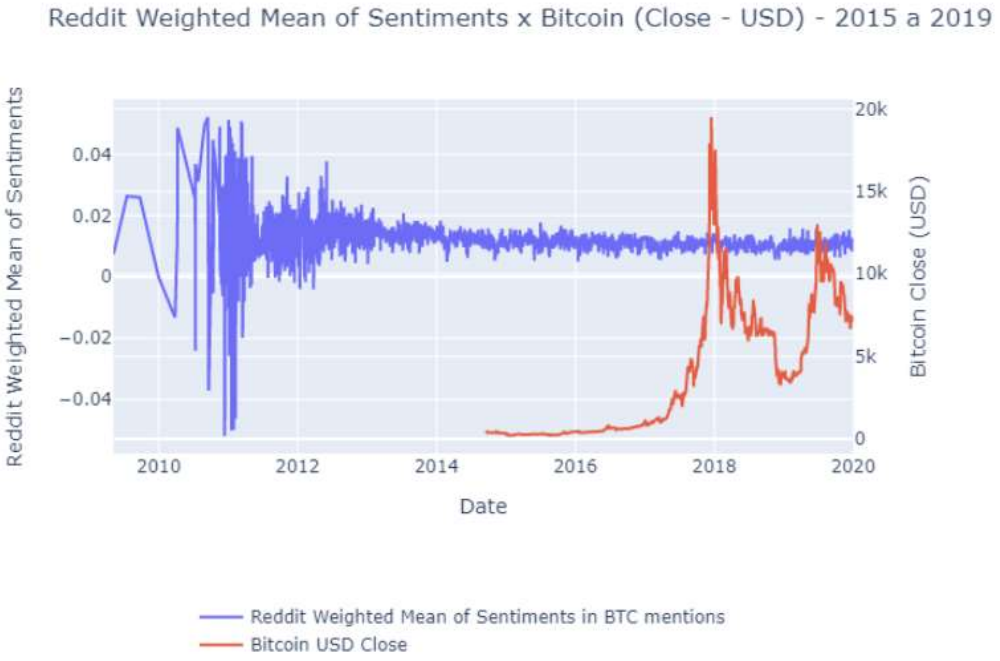

Figure 16: Bit coin Close x Mean of Polarity for Reddit Mentions.

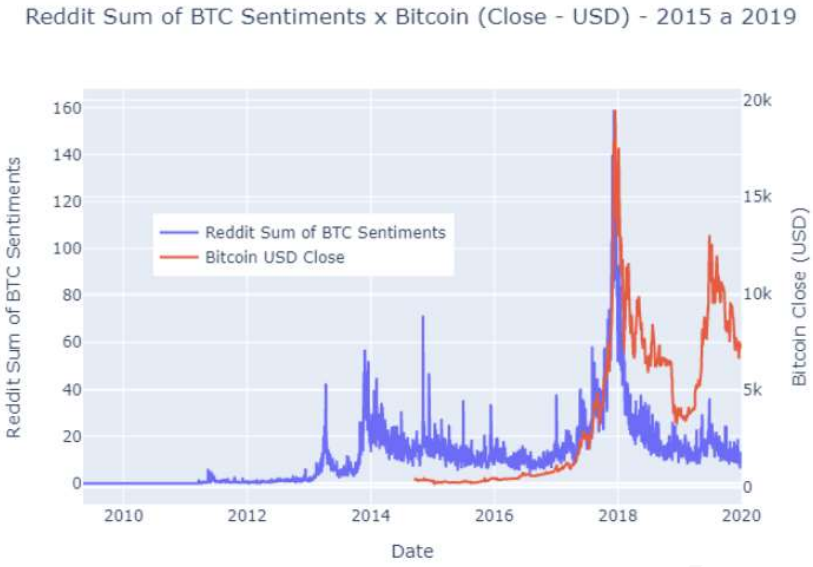

Figure 17: Bit coin Close x Aggregated Sum of Polarity for Reddit Mentions.

It is clear, especially considering Reddit Sentiment daily sums and volume that both time series seem to follow each other's trends. Regarding other variables, from all the technical indicators mentioned and all the different look backs (5, 10, 20 and 50 days) way too many features were created, making it difficult to spot individual correlations in the matrix:

Figure 18: Correlation Matrix - All variables.

As the target variable is categorical (up/down return movements), in order to grasp correlation, the percentage return change is considered. Checking Reddit Sentiment features' correlation to returns separately:

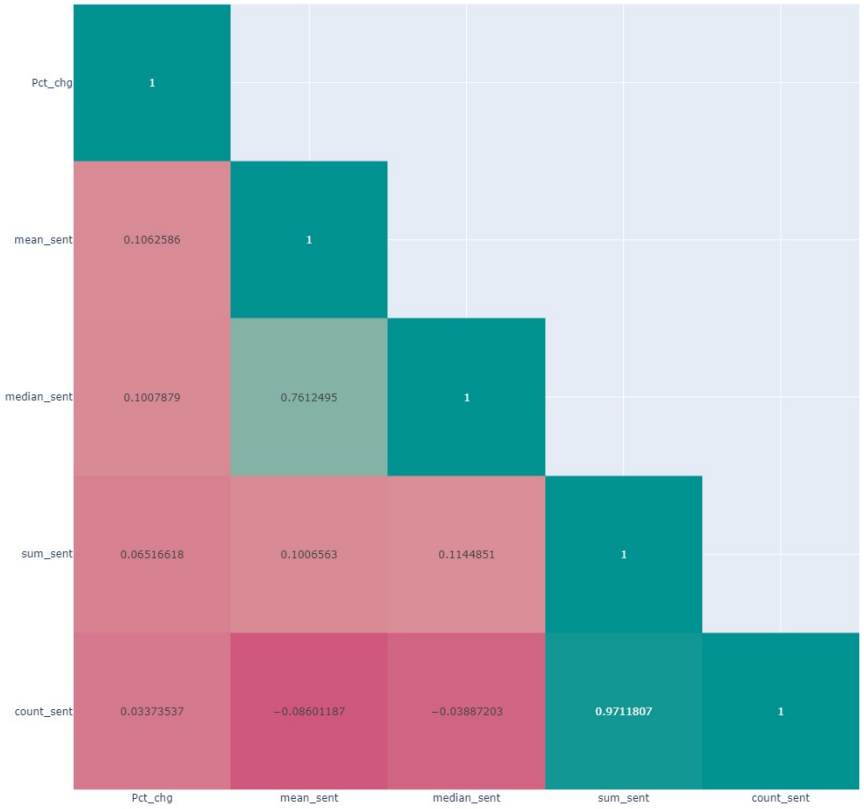

Figure 19: Correlation Matrix - Reddit Features.

Although daily volume and sum of mentions seemed to accompany BTC-USD oscillations over time in the previous visualizations, here both median and mean (redundant) of polarities boast the higher correlations to BTC. The top 10 most correlated to BTC returns are momentum indicators, such as MOM, RSI and CCI:

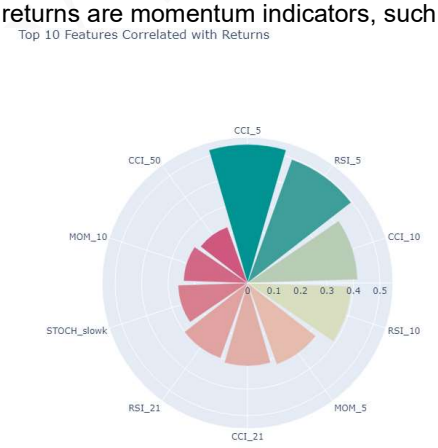

Figure 20: Top 10 features most correlated to BTC returns.

Feature Selection with Boruta

Within this project, an innovative approach called Boruta was harnessed to streamline the feature selection process. In essence, Boruta augmented the effectiveness of the Random Forest Classifier's feature importance methodology. This enhancement was achieved by introducing a shadow dataset comprised of features generated through the random shuffling of each feature. Among these shadow features, the genuine features that yielded the most informative content were singled out. Notably, Boruta follows an all-encompassing feature selection strategy, aiming to identify all features that contribute relevant information for prediction, as opposed to striving for a compact subset with minimal classifier error, as articulated on its GitHub repository.

While alternate techniques such as Self-Organizing Maps (SOMs) are available for feature selection and dimension reduction, Boruta was chosen due to its dual merits of efficiency and interpretability. The rationale behind this selection is underpinned by the fact that Boruta's implementation is relatively straightforward and user-friendly, aligning well with the project's objectives.

```
#Defining class weights (to deal with class imbalance):
def cwts(data):
    c0, c1 = np.bincount(data['target'])
    #making the weights inversely proportional to the amount of observations:
    w0, w1 = (1/c0)*(len(data))/2, (1/c1)*(len(data))/2
    return {0: w0, 1:w1}

# Define random forest classifier with the weighted classes:
forest = RandomForestClassifier(n_jobs=-1, class_weight=class_weights, max_depth=5)
forest.fit(X, y)

# define Boruta feature selection method
# percentage was set to 80 as the threshold for comparison between shadow and real features
# to allow for less drastic feature selection (way too many variables were being removed)
feat_selector = BorutaPy(forest, n_estimators='auto', verbose=2, random_state=1, perc=80)

# find all relevant features
feat_selector.fit(X, y)

feature_ranks = list(zip(df_.columns,
                         feat_selector.ranking_,
                         feat_selector.support_,
                         feat_selector.support_weak_))

# iterate through and print out the results
boruta = defaultdict(list)
for feat in feature_ranks:
    boruta['Feature'].append(feat[0])
    boruta['Rank'].append(feat[1])
    boruta['Keep'].append(feat[2])


# dataframe with ranks
boruta = pd.DataFrame(boruta).set_index('Feature')
```

The dataframe generated above returns information on rank of importance and whether Boruta's method would keep or exclude the feature. The following 17 features were kept:

```
['mean_sent', 'STOCH_slowk', 'STOCH_slowd', 'MACD','MACD_hist',
'CCI_5', 'CCI_10', 'CCI_21', 'CCI_50', 'MOM_5', 'MOM_10','MOM_21',
'RSI_5', 'RSI_10', 'RSI_21', 'RSI_50']
```

Finally, the correlation matrix between the selected features reiterates the selection, the features are considerably correlated to Bit coin's returns:

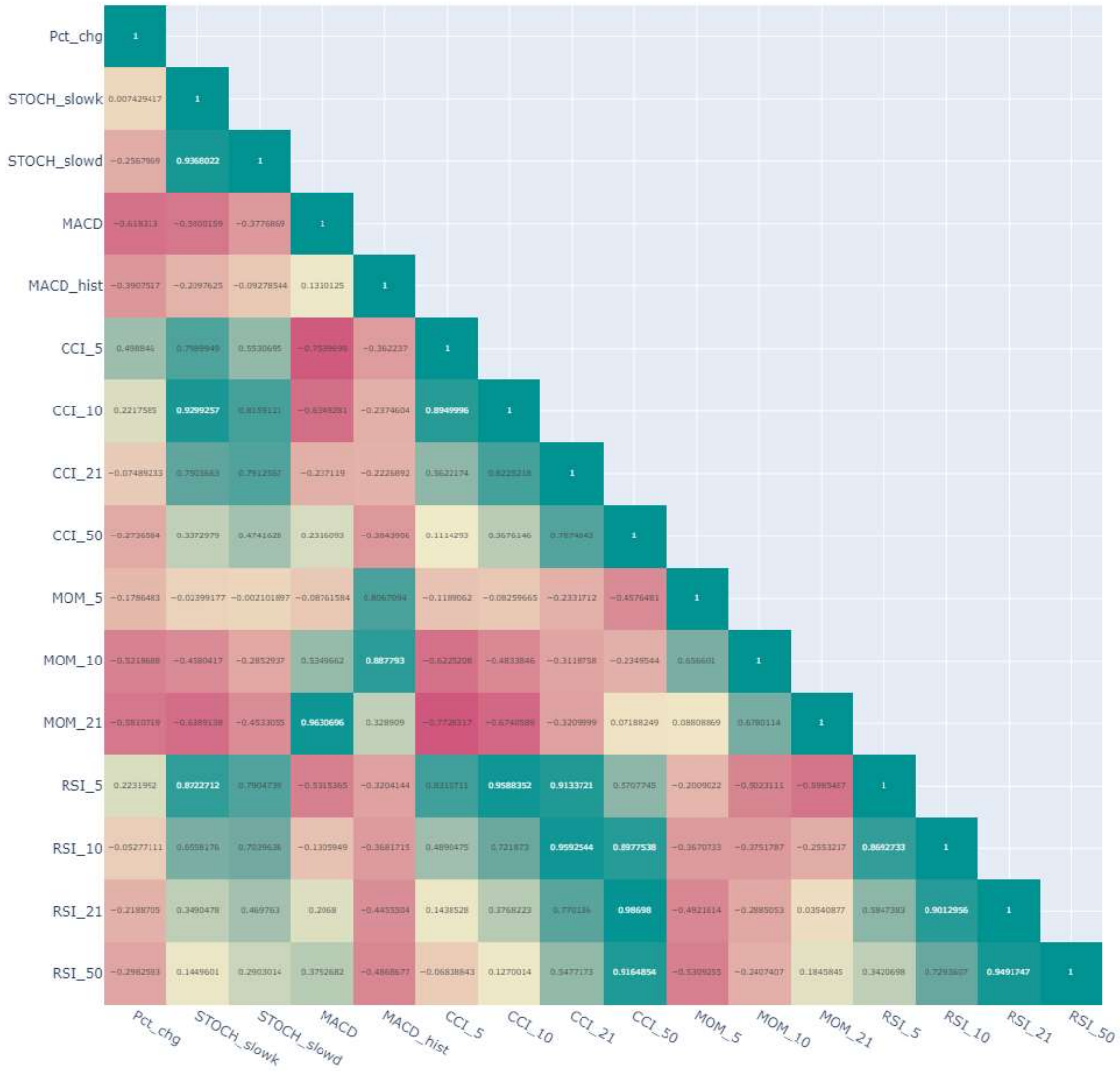

Figure 21: Correlation matrix of Selected Features.

4 MODEL TRAINING AND RESULTS

The aim was to predict the *target* (Bitcoin returns up/down movements) with the selected features. For further details, consult the full code in the notebook

LSTM architecture

```
model.add(LSTM(units=params['u1'], input_shape = (win_length, num_features), \
return_sequences=True))
# Dropout layer
model.add(Dropout(params['dropout']))
#Add second layer LSTM
model.add(LSTM(units=params['u2'], return_sequences=False))
#Dropout layer
model.add(Dropout(params['dropout']))
#Add a Dense layer
model.add(Dense(params['dense_neurons'], activation = params['activation']))

#Add the output layer - output layer
model.add(Dense(1, activation = 'sigmoid'))#output layer
```

1. A first LSTM and input layer with dynamic dimension defined by window lenght and number of features
2. A Dropout layer to prevent over fitting (it randomly shuts down a couple of neurons to avoid to do so)
3. Another LSTM layer, stacked 2 layer LSTM tend to provide enough complexity in most of the cases
4. Yet another dropout layer to prevent over fitting and regularize information
5. A Dense, output layer, fully connected to previous neurons to condense information before the output
6. Finally, a Dense output layer, wherein the activation function is a Sigmoid due to the problem at hand: 0 or 1 (up or down) dependent variable classification.

Hyperparameter Tuning and Compiling

The following parameters were tuned using Talos to avoid arbitrary selection.

```
params = {'lr': [1e-2,1e-3, 1e-4],
    'u1': [128,256],
    'u2': [128,256],
    'dropout': [0.3, 0.5],
    'batch_size': [32,64, 256],
    'epochs': [100],
    'optimizer': ['Adam'],
    'activation': ['relu','elu'],
    'dense_neurons': [32, 64],
    'window': [21,50,200],
    }
```

Each permutation of the values above for each hyperparameter was tested (with the time restriction, Talos was constrained to return only 20% of possible permutations):

1. lr: Learning Rate, the rate at which the model learns
2. u1 and u2: The number of nodes in the first and second LSTM layers
3. dropout: The number of nodes in the first and second LSTM layers
4. batch_size: Number of training samples to work through before the model's internal weights are updated
5. epochs: Number times that the learning algorithm will work through the entire training data (although this impacts the result, it was kept constant in 100 due to time restrictions)
6. optimizer: Many alternatives were tested, but in the end this project was restricted to Adam optimizer, an enhanced SGD, due to its efficiency, speed and good performance on large datasets
7. activation: Relu and Elu were both tested
8. dense_neurons: Number of nodes in the last, dense, hidden layer
9. window: Lookback or time window considered (how many days are being used to make the next day's prediction)

A couple of metric functions (Recall, Precision, Specificity and F1) were manually defined to be computed on each iteration, they were defined as follows:

```
    # Defining Recall, Precision, Specificity and F1 metrics:
#(https://medium.com/analytics-vidhya/custom-metrics-for-keras-tensorflow-ae7036654e05)
def recall(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall_keras = true_positives / (possible_positives + K.epsilon())
    return recall_keras


def precision(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision_keras = true_positives / (predicted_positives + K.epsilon())
    return precision_keras


def specificity(y_true, y_pred):
    tn = K.sum(K.round(K.clip((1 - y_true) * (1 - y_pred), 0, 1)))
    fp = K.sum(K.round(K.clip((1 - y_true) * y_pred, 0, 1)))
    return tn / (tn + fp + K.epsilon())


def f1(y_true, y_pred):
    p = precision(y_true, y_pred)
    r = recall(y_true, y_pred)
    return 2 * ((p * r) / (p + r + K.epsilon()))
```

Additionally, in order to avoid that the Binary Cross Entropy loss function's gradient descent taking sharp turns or diverging on later iterations, an exponential decay of -0.1 per epoch on the learning rate was set as follows (to be used as a callback when compiling the model):

```
# This function keeps the initial learning rate for the first ten epochs
# and decreases it exponentially after that.
def scheduler(epoch, lr):
    #rampup epochs
    if epoch<=10:
        return lr
    return lr * tf.math.exp(-0.1*epoch)
```

Finally, in order to avoid over fitting, an Early Stopping was set to stop after 10 consecutive epochs without reduction of the loss in the validation set.

```
early_stopping = EarlyStopping(monitor='val_loss',patience = 10, mode='min',\
restore_best_weights=True)
```

The Talos' tuning was set to return only 20% of all possible permutations due to time constraints.

```
talos.Scan(x = x_train, y = y_train, x_val = x_test, y_val = y_test,
params = params, model = lstm_model,experiment_name = 'lstm_model_val2',
fraction_limit = 0.2) # this yields only 20\% permutations
```

To check the whole code, please go to: DL-modified

Best Model

The hyperparameter tuning for 20% of the possible permutations returned 172 different models.

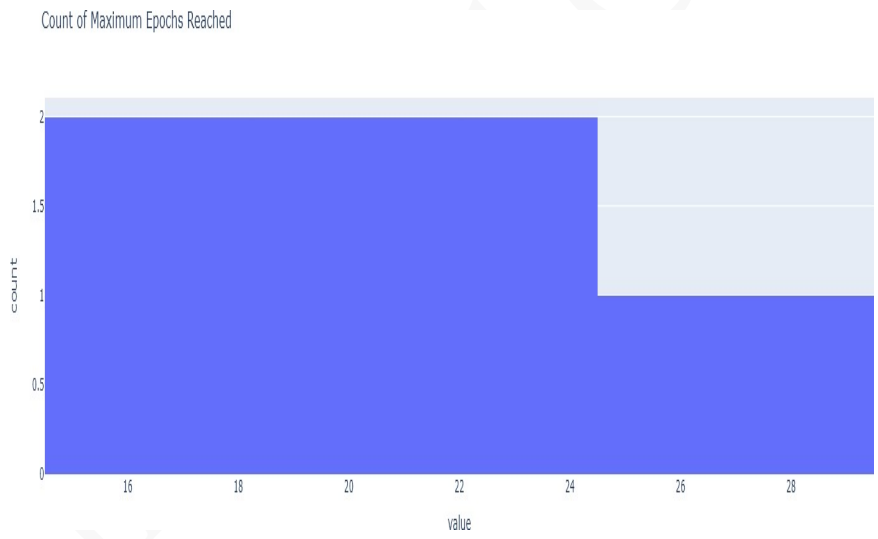

Figure 22: Histogram of maximum epochs reached per permutation.

No model went over 30 epochs, even though the epochs were set to 100. Clearly, either the early stop was too stringent or the exponential decay was too intense, making the gradient descent stop too early. For all the computed models, the Validation Binary Cross Entropy loss stayed in the 0.68-0.69 range. Accuracy is almost normally distributed but also highly concentrated.

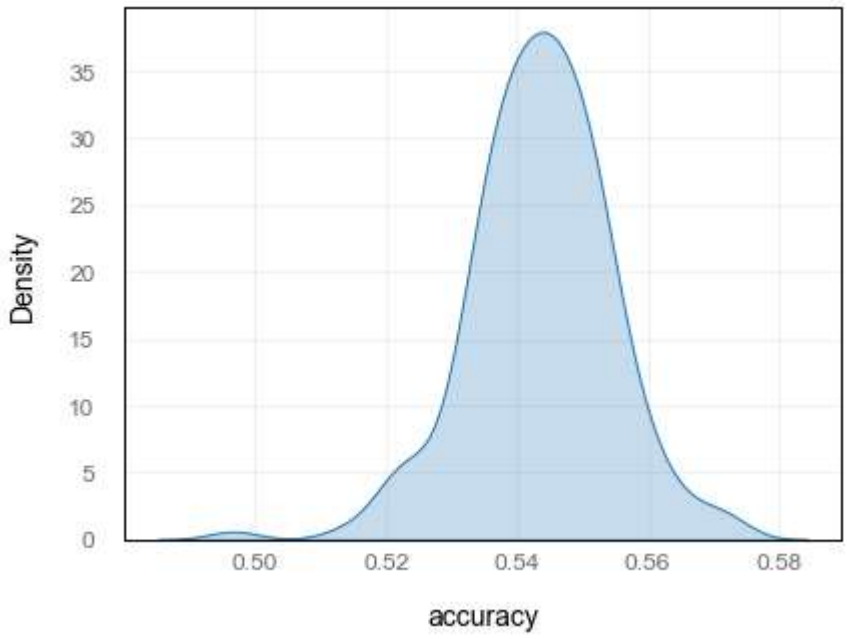

Figure 23: Validation Cross Entropy Loss Distribution.

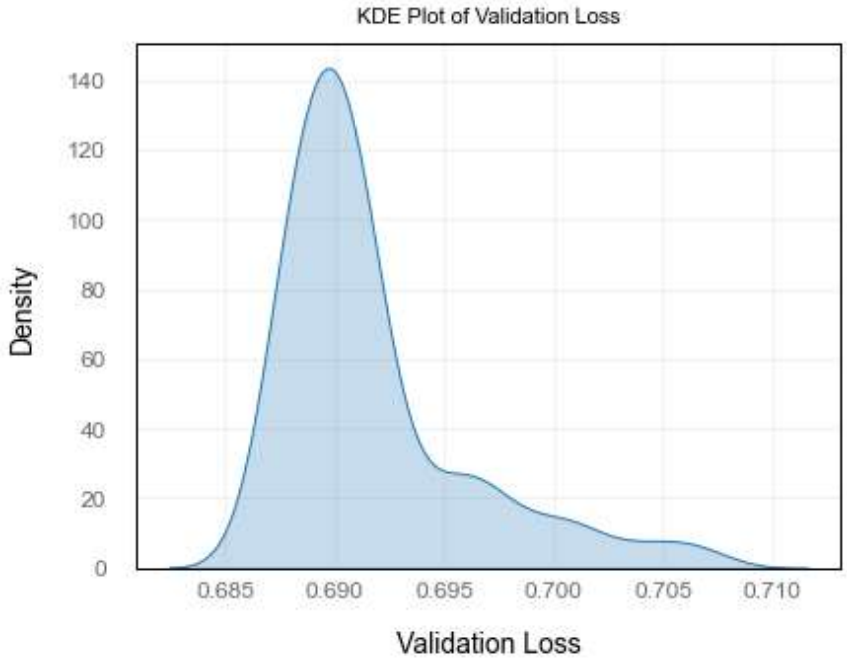

Figure 24: Validation Cross Entropy Loss Distribution.

The heat map on the next page shows that there is, weirdly, no clear relation between the tuned hyper parameters and the metrics considered. Bearing in mind the shortfall engendered by many restrictions and the considerably high minimum value for the validation loss, these were the best possible hyper parameters produced after running 20% of the total possible permutations:

```
{'lr': 0.0001,'u1': 128,'u2': 256,'dropout': 0.5,'batch_size': 32,
'epochs': 100,'optimizer': 'Adam','activation': 'relu','dense_neurons': 64,'window': 200}
```

The best model produced has 128 neurons in the first layer, 256 in the second and 64 in the last, dense layer with ReLU activation. Besides, the dropout rates are of 0.5, the batch size is 32 and 200 previous days are used to classify the next day price movement. The learning rate is the smallest between the alternatives

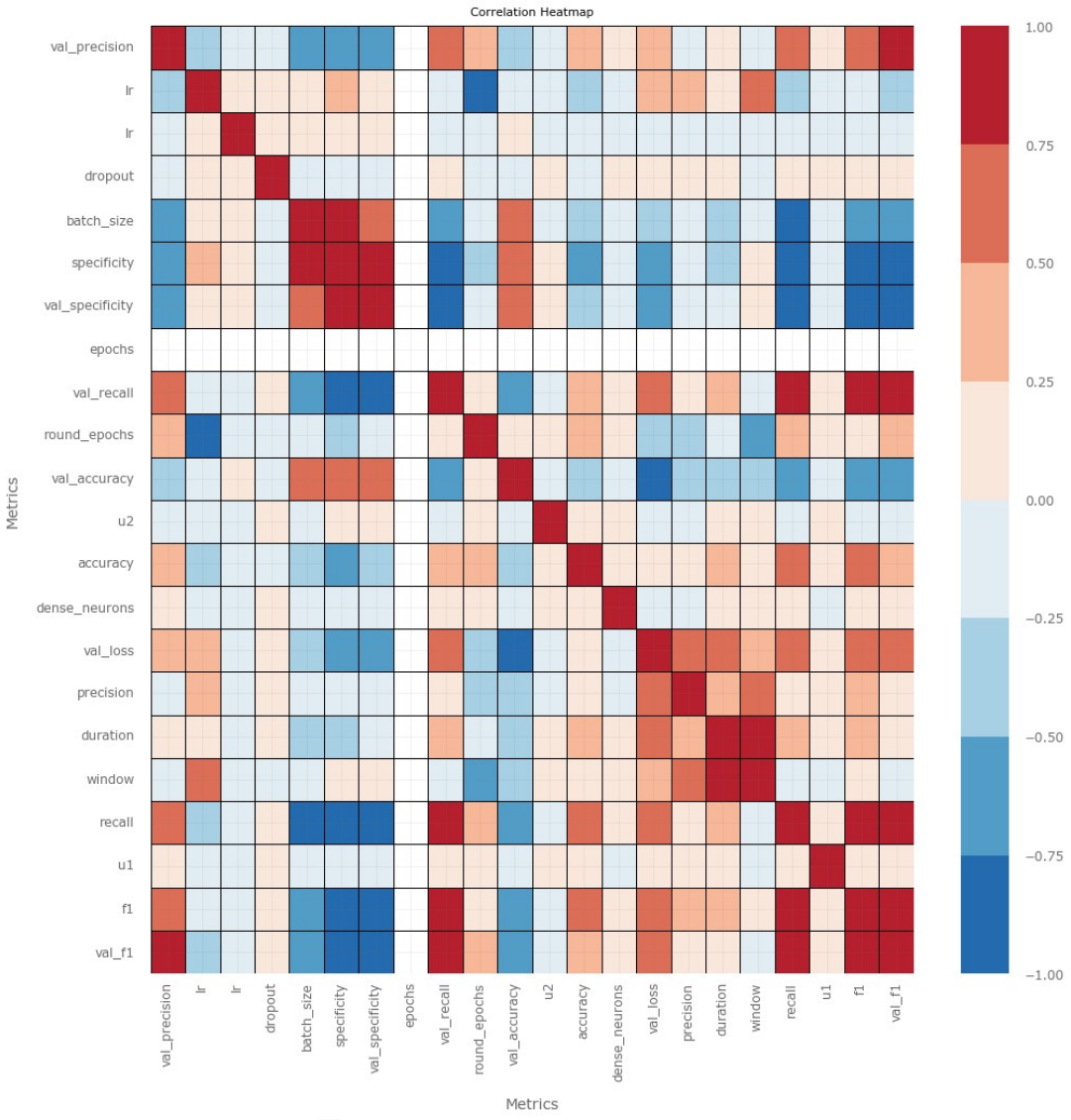

Figure 25: Correlations: Metrics x Hyper parameters.

5 RESULTS

Main Model - Classification

Using the optimal parameters, a slightly different model is trained: for longer epochs (200) and with increased early stopping (stops only after 30 stale periods) tolerance. Over the training, the metrics evolved as follows:

Figure 26: Metrics evolution over time.

Figure 27: Metrics evolution over time.

Unfortunately, increasing the number of epochs caused over fitting: the validation loss starts to increase whilst the loss decreases, with a subsequent increasing accuracy metric. Furthermore, even with the improvement of Precision, Recall and F1-score over the epochs, it came at the cost of Specificity, the true negative rate, worsening over time.

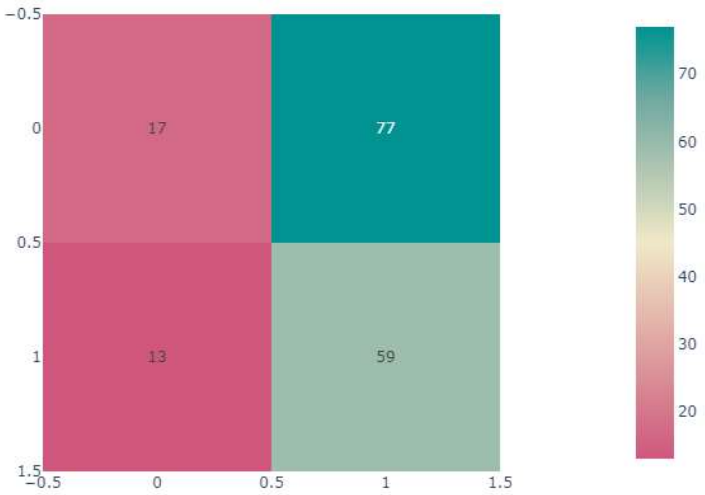

Figure 28: Confusion matrix.

The confusion matrix shows that the model only predicted downward moves.

Figure 29: Confusion matrix.

Apart from not converging, the trained model also failed to predict every single upward movement on the test set. The accuracy is over 50% only due to it always predicting movements as downwards in a test set where up and down movements are somewhat distributed. Likely to be either an architecture or data processing issue, although thus far its cause could not be pinpointed.

6 CONCLUSION

In order to manage the constraints of a time-bound approach and significant processing requirements, several limitations were implemented. This encompassed constraining hyperparameter tuning to a maximum of 20% of all possible permutations, with a focus on testing a fixed and limited set of values for each hyperparameter. Additionally, due to the considerable effort invested in sourcing and processing Reddit BTC sentiment data from scratch, the scope was narrowed to analyze a single asset.

Although this report does not explicitly outline baseline models, it's worth noting that various architectures were conceived and explored throughout the project. However, all of these efforts failed to converge below the threshold of 0.68 Binary Cross-Entropy log loss. Potential explanations for this outcome include the possibility that the chosen features were insufficient to adequately explain the up/down movements of Bitcoin, or that the parameter configurations were suboptimal. Numerous attempts were made with different architectures, yielding no favorable outcomes. Notably, it could be argued that allowing the gradient descent to run for a more extensive number of epochs might lead to improved fitting. Furthermore, considering the logic of the Adam optimizer, the exponential decay specified might be eliminated, as it might not be necessary.

An intriguing observation emerged when an alternative model was employed. Instead of classifying Bitcoin's direction, this model aimed to predict its closing price. Impressively, this approach yielded significantly more promising outcomes. However, it's important to note that only the aggregated daily sums of weighted sentiments were utilized as a single feature in this scenario. While this singular variable or the incorporation of LeakyReLU activation layers could explain this positive performance, similar architectural adjustments applied to the classification task (changing the final dense layer's activation to sigmoid) resulted in subpar results.

Despite the lackluster performance of the classification models, the alternative regression model exhibited remarkable achievements. It effectively tracked trends with a slight, discernible lag. This success may indicate the capacity of Reddit, serving as a niche community for crypto enthusiasts during the considered period, to reasonably reflect overall market sentiment and conditions. However, it's important to acknowledge that this efficacy might not persist in contemporary times, given the altered dynamics of the cryptocurrency landscape with the participation of prominent players.

REFERENCES

[1] Eugene F. FAMA. Efficient capital markets: Ii. *The Journal of Finance*, 46(5):1575– 1617, 1991.

[2] Eugene F. Fama, Lawrence Fisher, Michael C. Jensen, and Richard Roll. The adjustment of stock prices to new information. *International Economic Review*, 10(1):1–21, 1969.

[3] Daniel Kahneman and Amos Tversky. Choices, values, and frames. In *Handbook of the fundamentals of financial decision making: Part I*, pages 269–278. World Scientific, 2013.

[4] Twitter. Twitter investor fact sheet q12021. Available in: https://s22.q4cdn.com/826641620/files/doc_financials/2021/q1/Q1'21_InvestorFactSheet.pdf. Accessed on: 12 de Jul. 2021, 2021.

[5] Brian Dean. Reddit user and growth stats. Available in: https://backlinko.com/reddit-users. Accessed on: 12 de Jul. 2021, 2021.

[6] CNN. Wall street reddit hedge funds. Available in: https://edition.cnn.com/2021/02/03/investing/wall-street-reddit-hedge-funds/index.html. Accessed on: 12 de Jul. 2021, 2021.

[7] Ron Shevlin. How elon musk moves the price of bitcoin with his twitter activity. Available in: https://www.forbes.com/sites/ronshevlin/2021/02/21/ how-elon-musk-moves-the-price-of-bitcoin-with-his-twitter-activity/?sh=750ef005d27b. Accessed on: 12 de Jul. 2021, 2021.

[8] Emil Sayegh. Losing touch with reality – a gamestop lesson. Available in: https://www.forbes.com/sites/emilsayegh/2021/03/09/losing-touch-with-reality--a-gamestop-lesson/?sh=4c74871735a5. Accessed on: 12 de Jul. 2021, 2021.

[9] Kit Smith. 60 incredible and interesting twitter stats and statistics. Available in: https://www.brandwatch.com/blog/twitter-stats-and-statistics/. Accessed on: 12 de Jul. 2021, 2020.

[10] Thien Hai Nguyen and Kiyoaki Shirai. Topic modeling based sentiment analysis on social media for stock market prediction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Inter- national Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1354–1364, 2015.

[11] Jethin Abraham, Daniel Higdon, John Nelson, and Juan Ibarra. Cryptocurrency price prediction using tweet volumes and sentiment analysis. *SMU Data Science Review*, 1(3):1, 2018.

[12] Shubhankar Mohapatra, Nauman Ahmed, and Paulo Alencar. Kryptooracle: A real-time cryptocurrency price prediction platform using twitter sentiments. pages 5544–5551, 12 2019.

[13] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] C.J. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. 01 2015.

[16] Dennis Yang and Qiang Zhang. Drift-independent volatility estimation based on high, low, open, and close prices. *The Journal of Business*, 73(3):477–492, 2000.