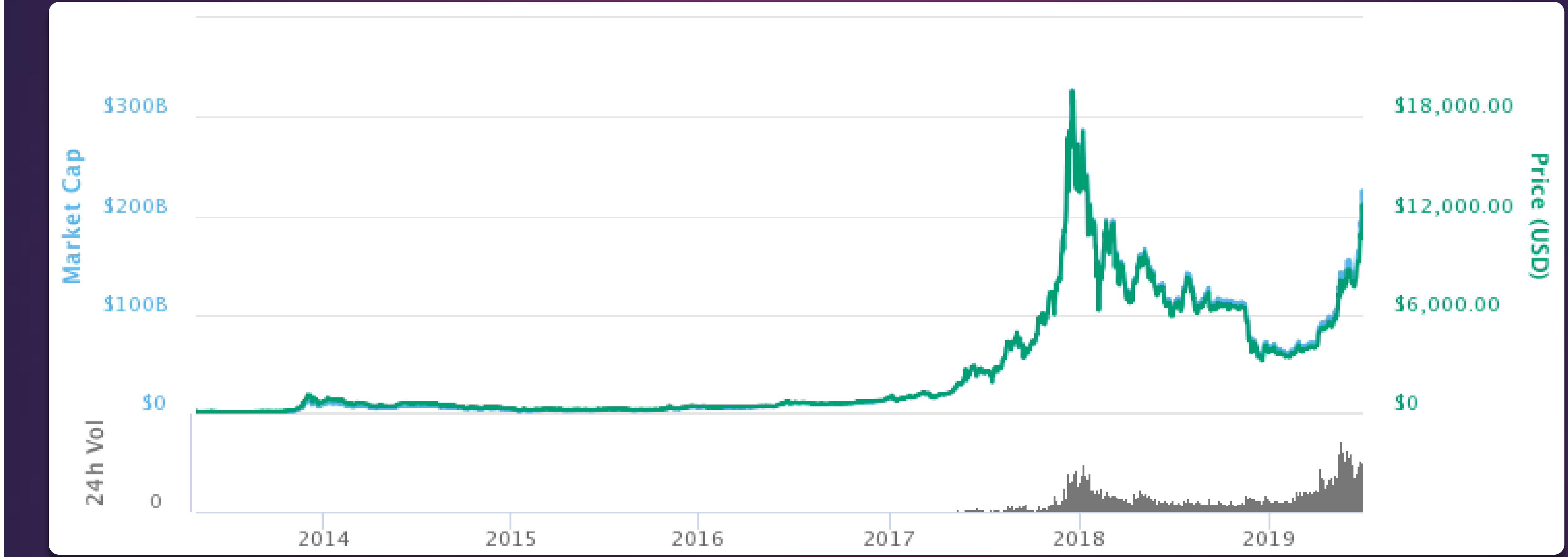
What does the world feel about Bitcoin?

TEAM COVFEEFEE

- FARZAD KHATOONABADI
- DAN FOX
- BRIAN REYES
- ROBERT SAYLER
- PRAVIN VENKATRAMAN

Volatility of Bitcoin



Bitcoin Value peaked to $20k in 2018
Current value- $12,600/-

Source- https://en.bitcoinwiki.org/wiki/Bitcoin_history

Purpose:
Sentiment Analysis

- Opinion mining
- How people feel about a subject in general
- Recommend based on the analysis
- Extract polarity and Subjectivity
- Turn Unstructured data into structured data

Considerations:

Facts or Opinions Automatic Method or Manual Method

- Machine Learning
- Positive or Negative
- Features
- Machine learning Algorithm
- Classifier
- New Data

Ways to find the answer

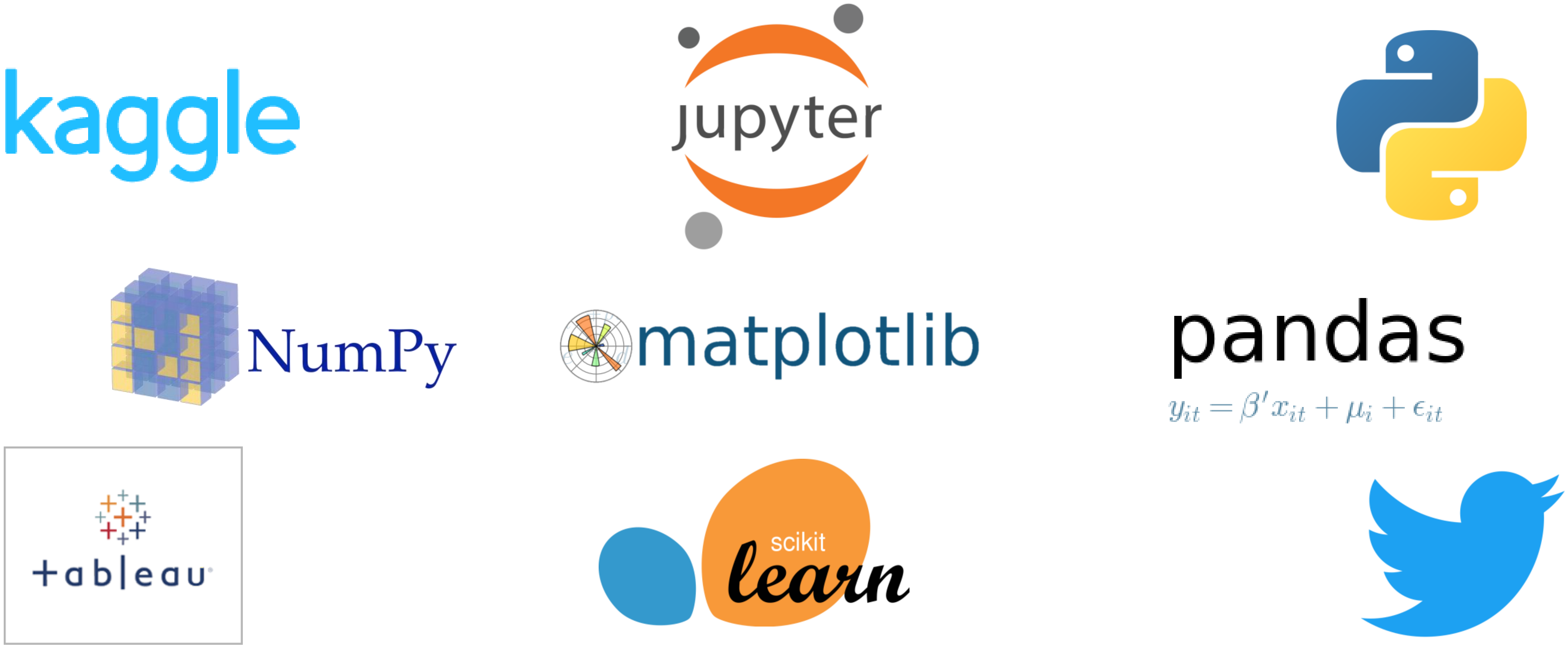
Rules Based Approach

- Pros
  - No training required
  - Easy to debug
- Cons
  - Not accurate

Automated Approach ✓

- Pros
  - Scalable
  - Better accuracy
- Cons
  - Length of time it takes to train data

Tools used to evaluate sentiment



Specific libraries used within python

- Tweepy
  - OAuthHandler
  - Stream
  - StreamListener
- SKlearn
- Ntlk
- Gensim-word2vec
- Re
- Wordcloud
- JSON
- Pandaspyplot
- NumPy
- Time
- Beautiful Soup
- Word2vec
  - Stopwords

Training Process

Read in the CSV

Replace Sentiment Column list into strings of categorical variables

#Using sklearn to encode categorical variables

```
from sklearn import preprocessing
encoder= preprocessing.LabelEncoder()
categories=["positive","negative","neutral"]
encoder.fit(categories)
```

#Use train_test_split to create training and testing data

```
from sklearn.model_selection
Import train_test_split
X= df['Tweet’]
y= df['Sentiment’]
X_train, X_test, y_train, y_test =
train_test_split(X, y)#
```

#This function converts a text to a sequence of words

```
def review_wordlist(review, remove_stopwords=False):
    review_text = BeautifulSoup(review).get_text()
    review_text = re.sub("[^a-zA-Z]"," ",review_text)
    words = review_text.lower().split()
    if remove_stopwords:
    stops = set(stopwords.words("english"))
    words = [w for w in words if not w in stops]
    return(words)
```

#Using punkt tokenizer better sentence splitting

```
import nltk.data nltk.download('popular’)
tokenizer =
nltk.data.load('tokenizers/punkt/english.pickle')
```

#This function splits a review into sentences

```
def review_sentences(review,
tokenizer,remove_stopwords=False):
    raw_sentences = tokenizer.tokenize(review.strip())
    sentences = for raw_sentence in raw_sentences:
    if len(raw_sentence)>0:
    sentences.append(review_wordlist
    (raw_sentence, remove_stopwords))
```

#This function converts a text to a sequence of words

```
def review_wordlist(review, remove_stopwords=False):
    review_text = BeautifulSoup(review).get_text()
    review_text = re.sub("[^a-zA-Z]"," ",review_text) words =
    review_text.lower().split()
    if remove_stopwords:
    stops = set(stopwords.words("english"))
    words = [w for w in words if not w in stops] return(words)
```

Training Process

# Creating the model and setting values

#for the various parameters

```
    num_features = 300 # Word vector dimensionality
    min_word_count = 40 # Minimum word count
    num_workers = 4 # Number of parallel threads
    context = 10 # Context window size
    downsampling = 1e-3 # (0.001) Downsample setting
    for frequent words
```

# Initializing the train model from gensim.models

```
    import word2vec print("Training model....")
    model = word2vec.Word2Vec(sentences,\
                        workers=num_workers,\
                        size=num_features,\
                        min_count=min_word_count,\
                        window=context, \
                        sample=downsampling)
```

# To make the model memory efficient

```
    model.init_sims(replace=True)
```

# Saving the model for later use.

#Can be loaded using Word2Vec.load()

```
    model_name = "300features_40minwords_10context.h5"
    model.save(model_name)
```

# Function to average all word vectors in a paragraph

```
def featureVecMethod(words, model, num_features):
    # Pre-initialising empty numpy array for speed
    featureVec = np.zeros(num_features,dtype="float32")
    nwords = 0
```

# Dividing the result by number of words to get average

```
    featureVec=np.divide(featureVec, nwords) reture featu
```

#Calculating average feature vector for training set

```
    clean_train_reviews = [] for review in X_train:
    clean_train_reviews.append(review_wordlist(review, \
                                remove_stopwords=True))
```

#Fitting a random forest classifier to the training data

```
    from sklearn.ensemble import RandomForestClassifier
    forest =RandomForestClassifier(n_estimators = 100)
    print("Fitting random forest to training data....")
    forest = forest.fit(trainDataVecs, y_train)
```

Training Process

# Determining our score on training data

```
forest.score(trainDataVecs, y_train)
```

0.9970637583892618

# Determining our score on testing data

```
forest.score(testDataVecs, y_test)
```

0.8993314982304365

Read Tweet Process

1 Creating the authentication object

AUTH OAUTHHANDLER
(CONSUMER_KEY,
CONSUMER_SECRET)

AUTH.SET_ACCESS_TOKEN
(ACCESS_TOKEN,
ACCESS_SECRET)

API
TWEEPY.API(AUTH)

2 Collecting tweets

serach words

date_since

tweets = tw.Cursor
(api.search,
q=search_words,
lang="en",
since=date_since).items(1000)

PRINT(TWEETS)

3 Collecting the tweet metadata and create a dataframe

Date Time

Handle

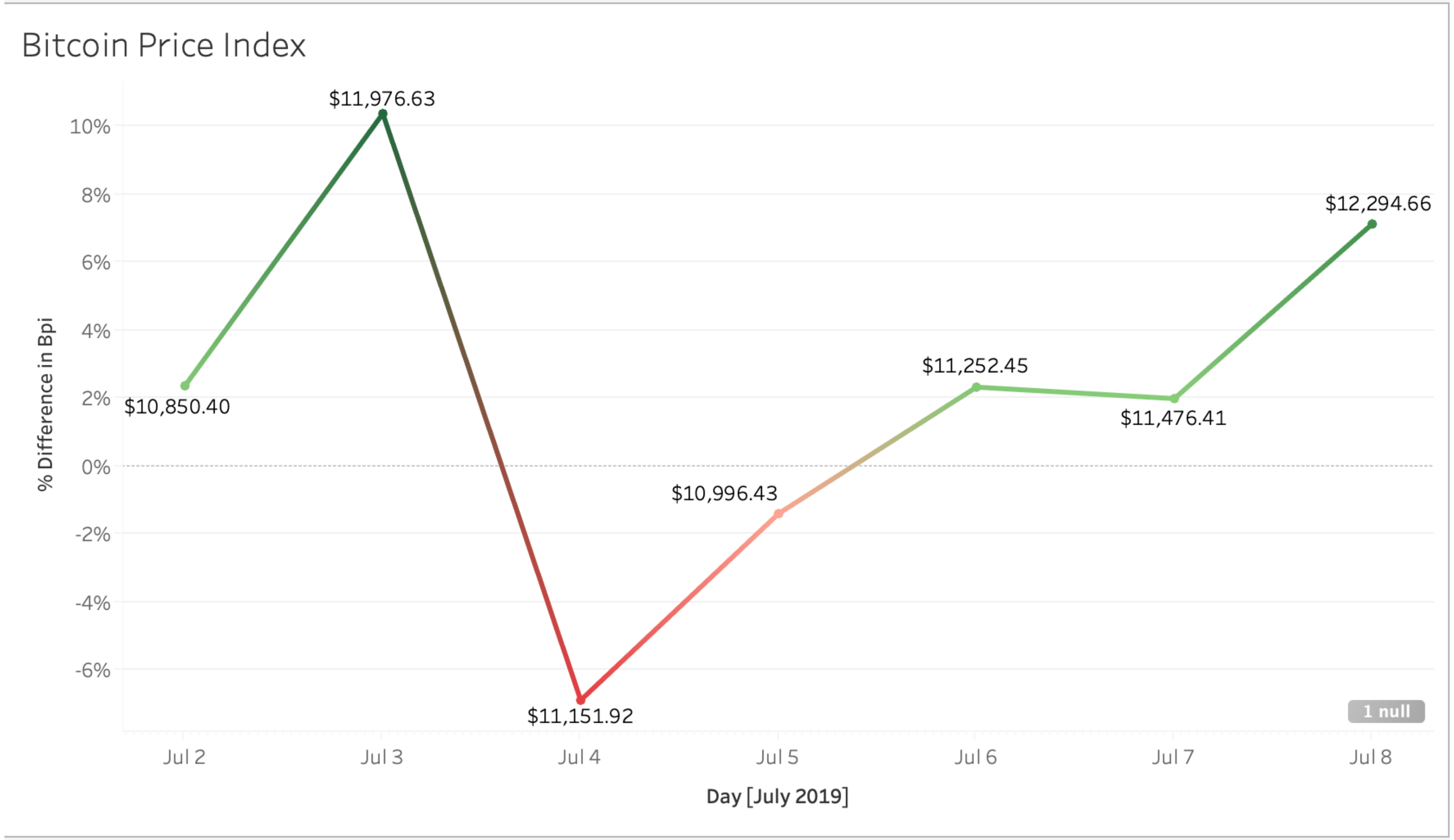
Tweet

Retweet Count

Followers

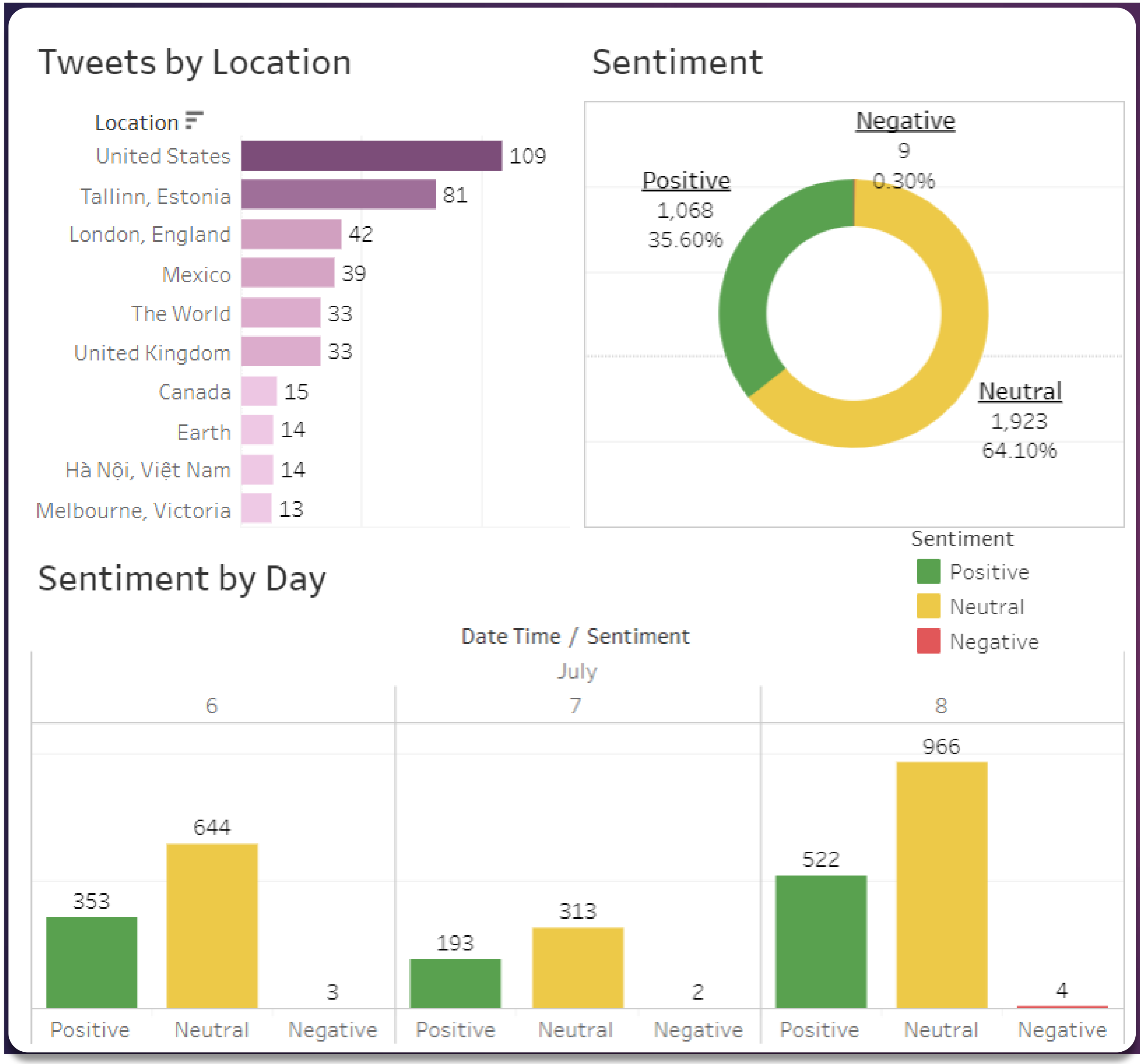
Location

Bitcoin Price Index (Percent Change)



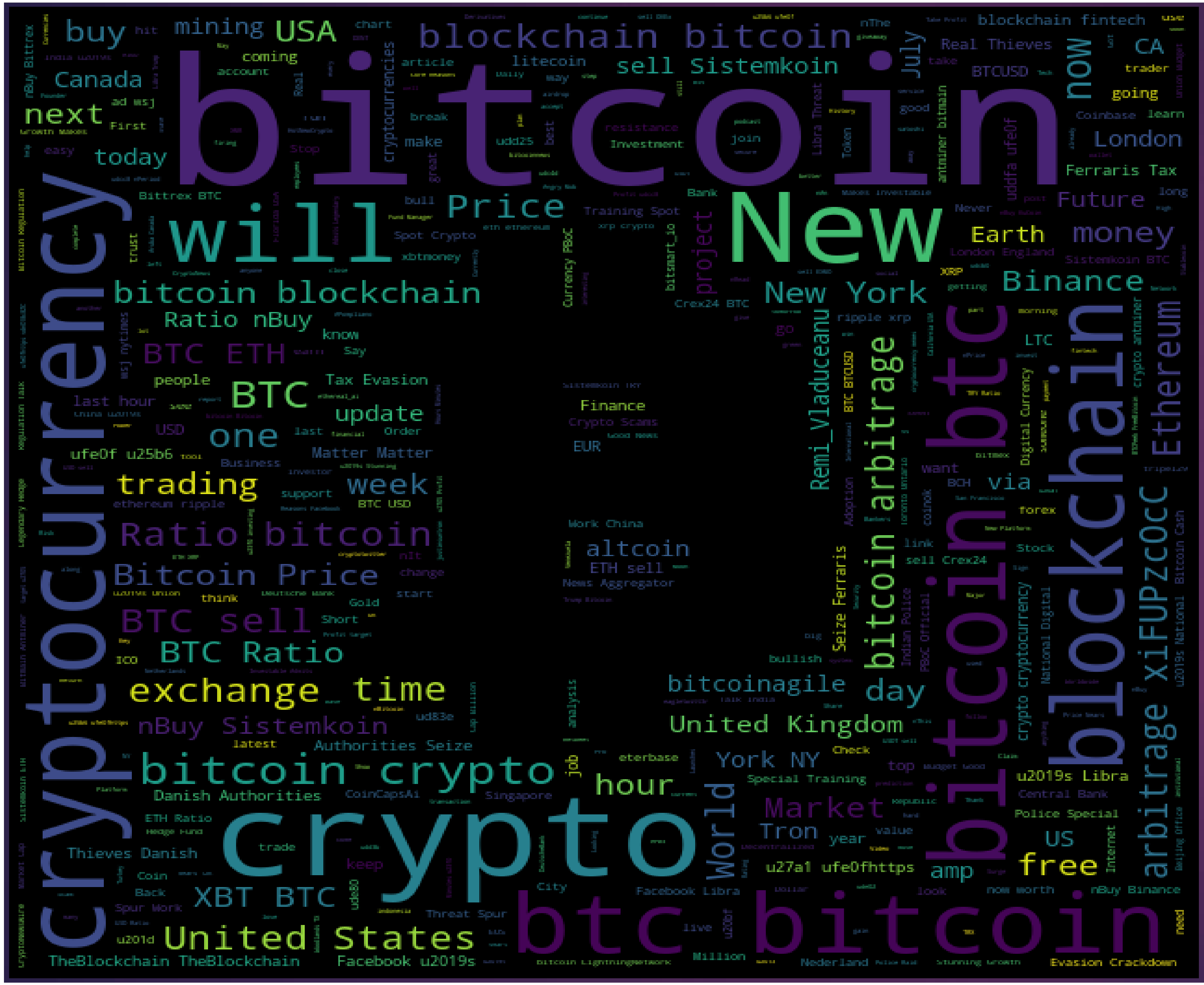
Since the beginning of July the value has increased 13%; however the day to day change is volatile.

source: Bitcoin Price Index https://www.coindesk.com



Majority of twitter users feel positive about bitcoin

Tableau Dash Board



Most common words used in tweets are-

- bitcoin
- blockchain
- BTC
- BTC ratio
- Binance
- cryptocurrency
- crypto
- trading
- digital currency
- bitcoinagile

Questions?

Considerations & Resources

- 3000 tweets were pulled (July 6th, July 7th, & July 8th)
- Kaggle data source- https://www.kaggle.com/varun08/imdb-dataset/downloads/labeledTrainData.tsv/data
- Bitcoin Price Index- https://www.coindesk.com
- Sentimental Analysis Code- Kaggle website