test

March 4, 2019

In [1]:
```
import sys
```

In [35]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
%matplotlib inline
```

In [3]:
```
print("toto")
```

```
toto
```

In [4]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [5]:
```
# Load Data
df = pd.read_csv('bitcointweets.csv', header=None)
pd.set_option('display.max_colwidth', -1)
df = df[[1,7]]
df.columns = ['tweet','label']
df.head()
```

Out[5]:

	tweet	label
0	RT @ALXTOKEN: Paul Krugman, Nobel Luddite. I had to tweak the nose of this Bitcoin	['neutral']
1	@lopp @_Kevin_Pham @psycho_sage @naval But @ProfFaustus (dum b a ss) said you know	['neutral']
2	RT @tippereconomy: Another use case for #blockchain and #Tipper. The #TipperEconomy	['positive']
3	free coins https://t.co/DiuoePJdap	['positive']
4	RT @payvxofficial: WE are happy to announce that PayVX Presale Phase 1 is now LIVE!	['positive']

```
In [6]: df.tail()
```

Out[6]:

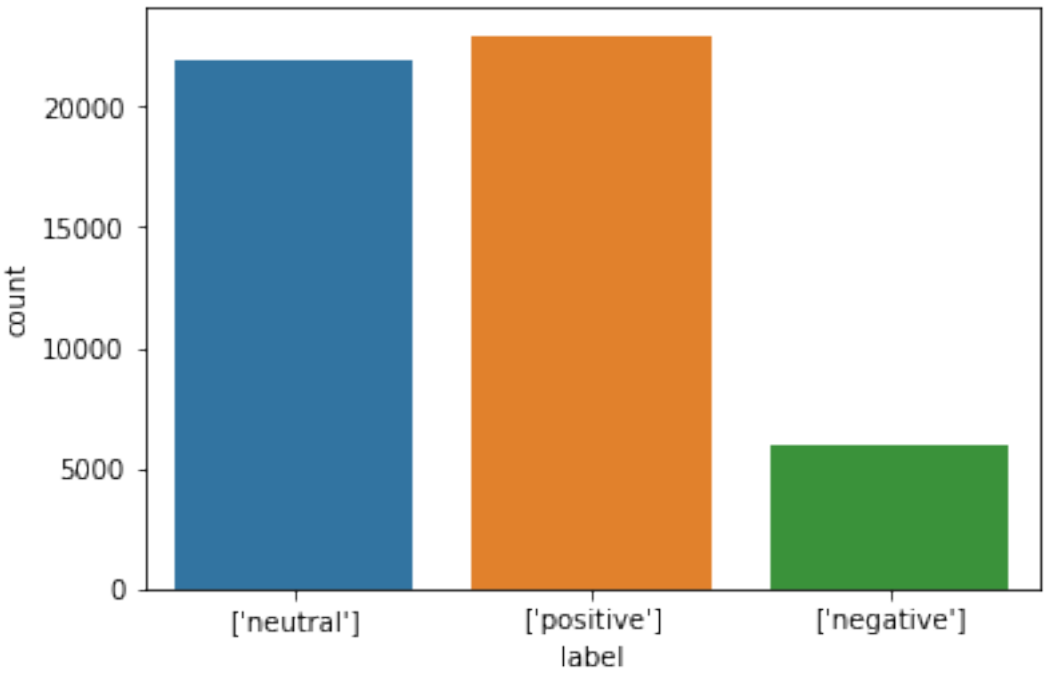
	tweet	label
50854	RT @fixy_app: Fixy Network brings popular cryptocurrencies and retailers as part	['positive']
50855	RT @bethereumteam: After a successful launch of our Bounty campaign, we've manag	['positive']
50856	RT @GymRewards: Buy #GYMRewards Tokens, Bonus Time is ending! https://t.co/HDvh	['neutral']
50857	I added a video to a @YouTube playlist https://t.co/ntFJrNvSvZ How To Bitcoin C	['positive']
50858	RT @Raybambs: Airdrop PhotoCoin Airdrop Round#2. 100 #PhotoCoin will be giving t	['positive']

```
In [7]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50859 entries, 0 to 50858
Data columns (total 2 columns):
tweet    50859 non-null object
label    50859 non-null object
dtypes: object(2)
memory usage: 794.8+ KB
```

```
In [8]: # inspect sentiment
        sns.countplot(df['label'])
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7ed8a0e278>
```

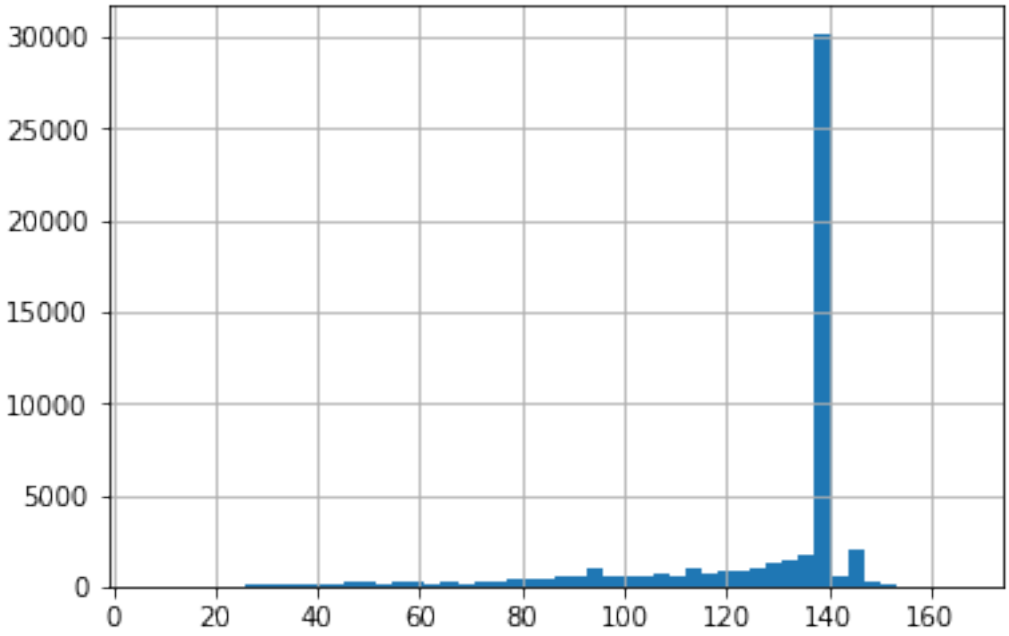



```
In [9]: # text length
        df['text_length'] = df['tweet'].apply(len)
        df[['label','text_length','tweet']].head()

Out[9]:          label  text_length  \
        0  ['neutral']   140
        1  ['neutral']   137
        2  ['positive']  140
        3  ['positive']  34
        4  ['positive']  146


        0  RT @ALXTOKEN: Paul Krugman, Nobel Luddite. I had to tweak the nose of this Bitcoin
        1  @lopp @_Kevin_Pham @psycho_sage @naval But @ProfFaustus (dum b a ss) said you know
        2  RT @tippereconomy: Another use case for #blockchain and #Tipper. The #TipperEconomy
        3  free coins https://t.co/DiuoePJdap
        4  RT @payvxofficial: WE are happy to announce that PayVX Presale Phase 1 is now LIVE!

In [10]: df['text_length'].describe()

Out[10]: count    50859.000000
         mean     127.650072
         std      23.595770
         min      7.000000
         25%      126.000000
```

```
50%      140.000000
75%      140.000000
max      166.000000
Name: text_length, dtype: float64
```

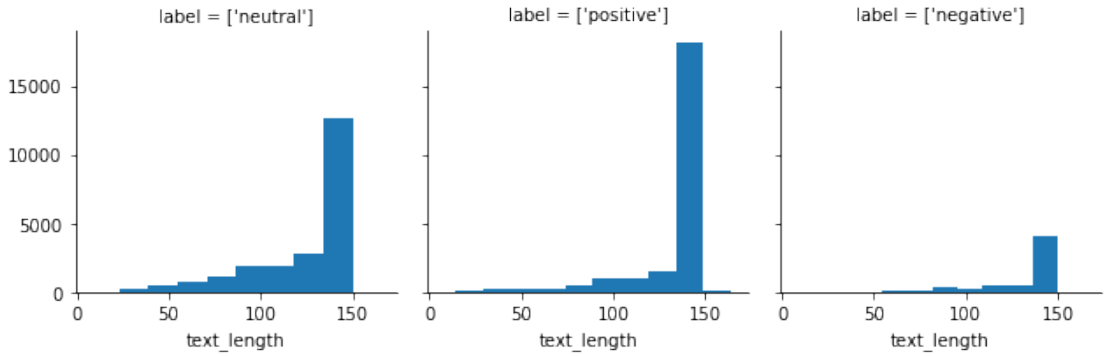
```
In [11]: df['text_length'].hist(bins=50)
```

```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7ed89e8ac8>
```



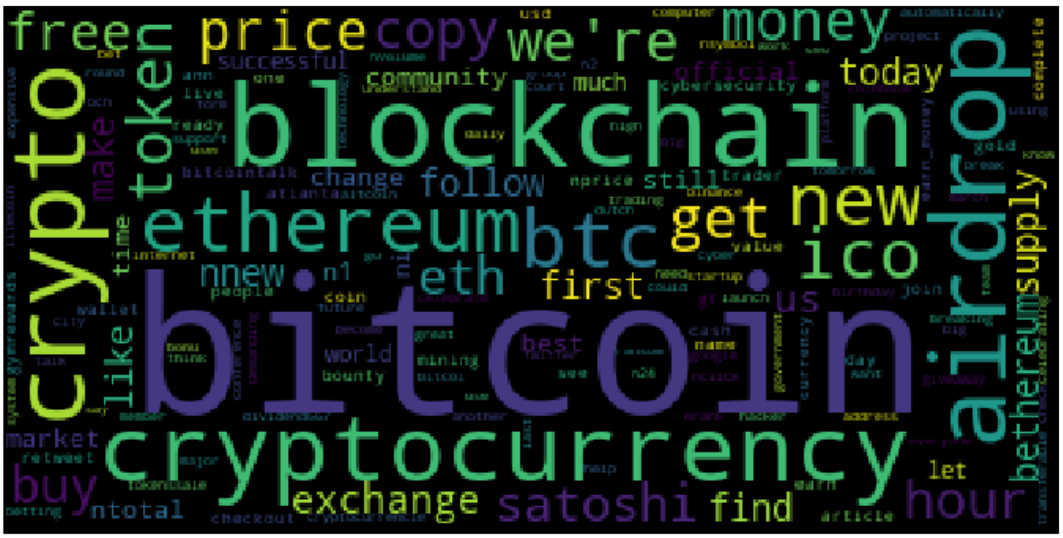

```
In [12]: g = sns.FacetGrid(df,col='label')
         g.map(plt.hist,'text_length')
```

```
Out[12]: <seaborn.axisgrid.FacetGrid at 0x7f7ed9fcc8d0>
```



In [13]:
```
# word cloud
import nltk
nltk.download("stopwords")
from nltk.corpus import stopwords
from wordcloud import WordCloud
import re

def clean_text(s):
    s = re.sub(r'http\S+', '', s)
    s = re.sub('(RT|via)((?:\\b\\W*@\\w+)+)', ' ', s)
    s = re.sub(r'@\S+', '', s)
    s = re.sub('&amp', ' ', s)
    return s
df['clean_tweet'] = df['tweet'].apply(clean_text)

text = df['clean_tweet'].to_string().lower()
wordcloud = WordCloud(
    collocations=False,
    relative_scaling=0.5,
    stopwords=set(stopwords.words('english'))).generate(text)

plt.figure(figsize=(12,12))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

```
[nltk_data] Downloading package stopwords to /home/jovyan/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```



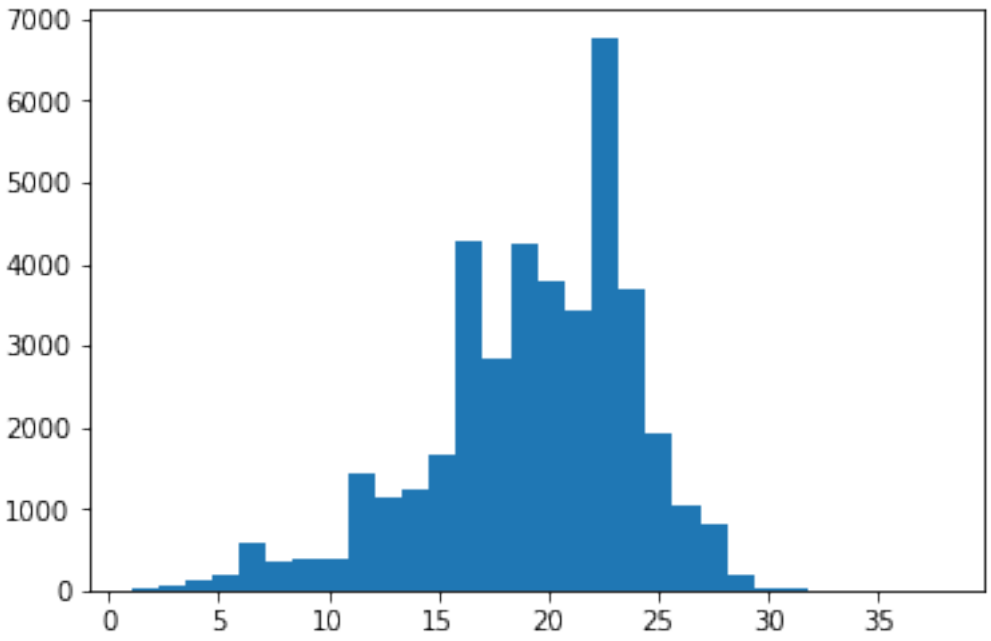
```
In [14]: seed = 101 # fix random seed for reproducibility
         np.random.seed(seed)

In [15]: print(type(df))
         df["tweet"].describe()

<class 'pandas.core.frame.DataFrame'>


Out[15]: count     50859
         unique    28136
         top       RT @GymRewards: https://t.co/Bm9sIxiiwU  Checkout our #bitcointalk #ANN http
         freq      672
         Name: tweet, dtype: object

In [16]: df.tweet.head()

Out[16]: 0    RT @ALXTOKEN: Paul Krugman, Nobel Luddite. I had to tweak the nose of this Bitco
         1    @lopp @_Kevin_Pham @psycho_sage @naval But @ProfFaustus (dum b a ss) said you kn
         2    RT @tippereconomy: Another use case for #blockchain and #Tipper. The #TipperEcon
         3    free coins https://t.co/DiuoePJdap
         4    RT @payvxofficial: WE are happy to announce that PayVX Presale Phase 1 is now LI
         Name: tweet, dtype: object

In [17]: df.label.head()

Out[17]: 0    ['neutral']
         1    ['neutral']
         2    ['positive']
         3    ['positive']
         4    ['positive']
         Name: label, dtype: object

In [38]: # Split Train Test sets
         import tensorflow
         from sklearn.model_selection import train_test_split
         X = df["tweet"]
         y = df["label"]

         # integer encode
         label_encoder = LabelEncoder()
         integer_encoded = label_encoder.fit_transform(y)
         onehot_encoder = OneHotEncoder(sparse=False)
         integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
         y = onehot_encoder.fit_transform(integer_encoded)

         X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                             test_size=0.2,
                                                             stratify=y,
```

```
                                                        random_state=seed)

        print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(40687,) (10172,) (40687, 3) (10172, 3)
```

In [40]:
```
# Tokenize Text
from keras.preprocessing.text import Tokenizer
max_features = 20000
tokenizer = Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(list(X_train))
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)
```

In [41]:
```
totalNumWords = [len(one_comment) for one_comment in X_train]
plt.hist(totalNumWords,bins = 30)
plt.show()
```



In [42]:
```
from keras.preprocessing import sequence
max_words = 30
X_train = sequence.pad_sequences(X_train, maxlen=max_words)
X_test = sequence.pad_sequences(X_test, maxlen=max_words)
print(X_train.shape,X_test.shape)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
print(X_train.shape[1])
```

```
(40687, 30) (10172, 30)
(40687, 30) (10172, 30) (40687, 3) (10172, 3)
30
```

In [43]:
```
import keras.backend as K
from keras.models import Sequential
from keras.layers import Dense,Embedding,Conv1D,MaxPooling1D,LSTM
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

batch_size = 128
epochs = 5
```

In [44]:
```
def get_model(max_features, embed_dim):
    np.random.seed(seed)
    K.clear_session()
    model = Sequential()
    model.add(Embedding(max_features, embed_dim, input_length=X_train.shape[1]))
    model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(num_classes, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accura
    print(model.summary())
    return model
```

In [45]:
```
def model_train(model):
    # train the model
    model_history = model.fit(X_train, y_train, validation_data=(X_test, y_test),
                              epochs=epochs, batch_size=batch_size, verbose=2)
    # plot train history
    plot_model_history(model_history)
```

In [46]:
```
def plot_model_history(model_history):
    fig, axs = plt.subplots(1,2,figsize=(15,5))
    # summarize history for accuracy
    axs[0].plot(range(1,len(model_history.history['acc'])+1),model_history.history['a
    axs[0].plot(range(1,len(model_history.history['val_acc'])+1),model_history.histor
    axs[0].set_title('Model Accuracy')
    axs[0].set_ylabel('Accuracy')
    axs[0].set_xlabel('Epoch')
    axs[0].set_xticks(np.arange(1,len(model_history.history['acc'])+1),len(model_hist
    axs[0].legend(['train', 'val'], loc='best')
    # summarize history for loss
    axs[1].plot(range(1,len(model_history.history['loss'])+1),model_history.history['
    axs[1].plot(range(1,len(model_history.history['val_loss'])+1),model_history.histo
    axs[1].set_title('Model Loss')
```

```
            axs[1].set_ylabel('Loss')
            axs[1].set_xlabel('Epoch')
            axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),len(model_hist
            axs[1].legend(['train', 'val'], loc='best')
            plt.show()
```

```
In [47]: def model_evaluate():
            # predict class with test set
            y_pred_test =  model.predict_classes(X_test, batch_size=batch_size, verbose=0)
            print('Accuracy:\t{:0.1f}%'.format(accuracy_score(np.argmax(y_test,axis=1),y_pred_
            #classification report
            print('\n')
            print(classification_report(np.argmax(y_test,axis=1), y_pred_test))

            #confusion matrix
            confmat = confusion_matrix(np.argmax(y_test,axis=1), y_pred_test)

            fig, ax = plt.subplots(figsize=(4, 4))
            ax.matshow(confmat, cmap=plt.cm.Blues, alpha=0.3)
            for i in range(confmat.shape[0]):
                for j in range(confmat.shape[1]):
                    ax.text(x=j, y=i, s=confmat[i, j], va='center', ha='center')
            plt.xlabel('Predicted label')
            plt.ylabel('True label')
            plt.tight_layout()
```

```
In [48]: # train the model
         max_features = 20000
         num_classes = 3
         embed_dim = 100
         model = get_model(max_features, embed_dim)
         model_train(model)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 30, 100)           2000000
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 30, 32)            9632
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 15, 32)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 15, 32)            3104
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 7, 32)             0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               53200
```

```
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 303
=================================================================
Total params: 2,066,239
Trainable params: 2,066,239
Non-trainable params: 0
_________________________________________________________________
None
Train on 40687 samples, validate on 10172 samples
Epoch 1/5
 - 41s - loss: 0.3237 - acc: 0.8690 - val_loss: 0.1030 - val_acc: 0.9696
Epoch 2/5
 - 36s - loss: 0.0553 - acc: 0.9841 - val_loss: 0.0877 - val_acc: 0.9735
Epoch 3/5
 - 36s - loss: 0.0232 - acc: 0.9934 - val_loss: 0.1049 - val_acc: 0.9733
Epoch 4/5
 - 37s - loss: 0.0127 - acc: 0.9965 - val_loss: 0.1060 - val_acc: 0.9761
Epoch 5/5
 - 36s - loss: 0.0082 - acc: 0.9978 - val_loss: 0.1203 - val_acc: 0.9769
```