Sentiment Analysis with Twitter API

Feedback and Reviews matter!

- Companies need reviews and feedback on products during development stage
- Governments need to know popular opinion over different subjects
- Brands need to know general preferences before launching new products
- Services providers need to track clients satisfaction

But where to find all this information?

On the Internet of course!

Social Media platforms provide tons of thoughts, reviews, opinions and feedback on the most diverse topics and fields. On Twitter, more than 340 million people engage on the platform providing an infinite source of data.

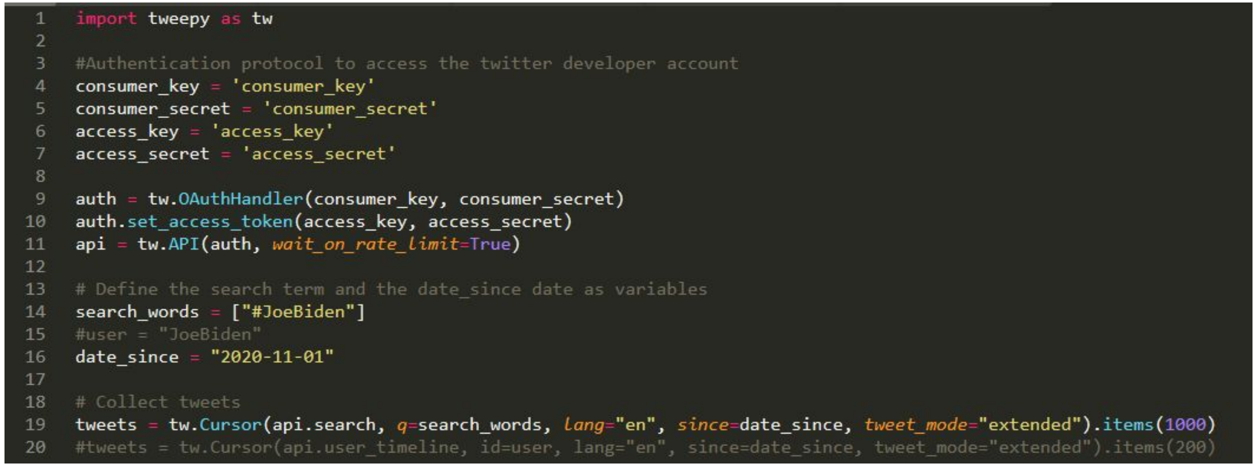
Analysing the US elections of 2020

The project:

- Twitter API
- TextBLob
- SKlearn
- MatplotLib
- Decision Tree Algorithm
- Tags on twitter: #DonaldTrump, #JoeBiden and #USElections2020

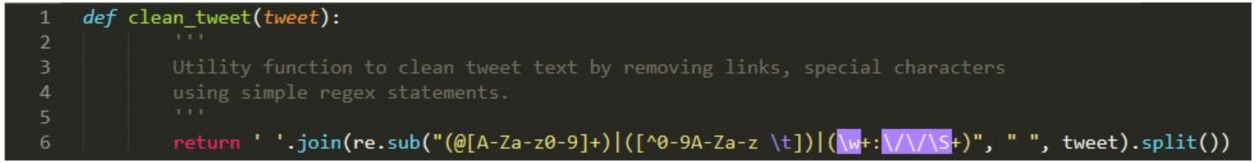
Data Gathering

Twitter API



```python
import tweepy as tw

#Authentication protocol to access the twitter developer account
consumer_key = 'consumer_key'
consumer_secret = 'consumer_secret'
access_key = 'access_key'
access_secret = 'access_secret'

auth = tw.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)
api = tw.API(auth, wait_on_rate_limit=True)

# Define the search term and the date_since date as variables
search_words = ["#JoeBiden"]
#user = "JoeBiden"
date_since = "2020-11-01"

# Collect tweets
tweets = tw.Cursor(api.search, q=search_words, lang="en", since=date_since, tweet_mode="extended").items(1000)
#tweets = tw.Cursor(api.user_timeline, id=user, lang="en", since=date_since, tweet_mode="extended").items(200)
```

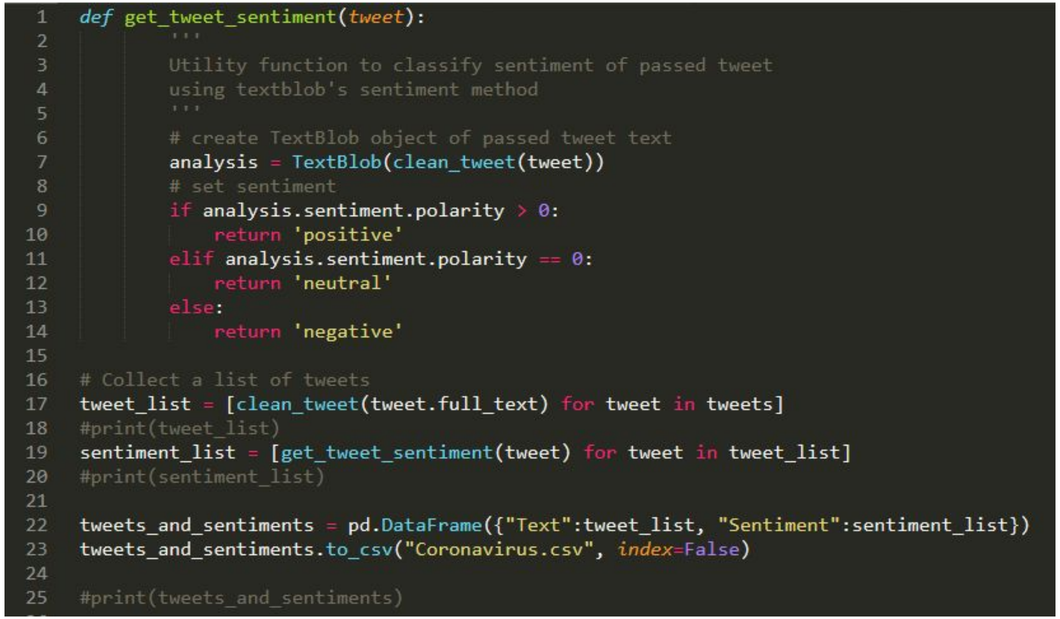
Data Cleaning



```
def clean_tweet(tweet):
        '''
        Utility function to clean tweet text by removing links, special characters
        using simple regex statements.
        '''
        return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", tweet).split())
```

Getting the Sentiment

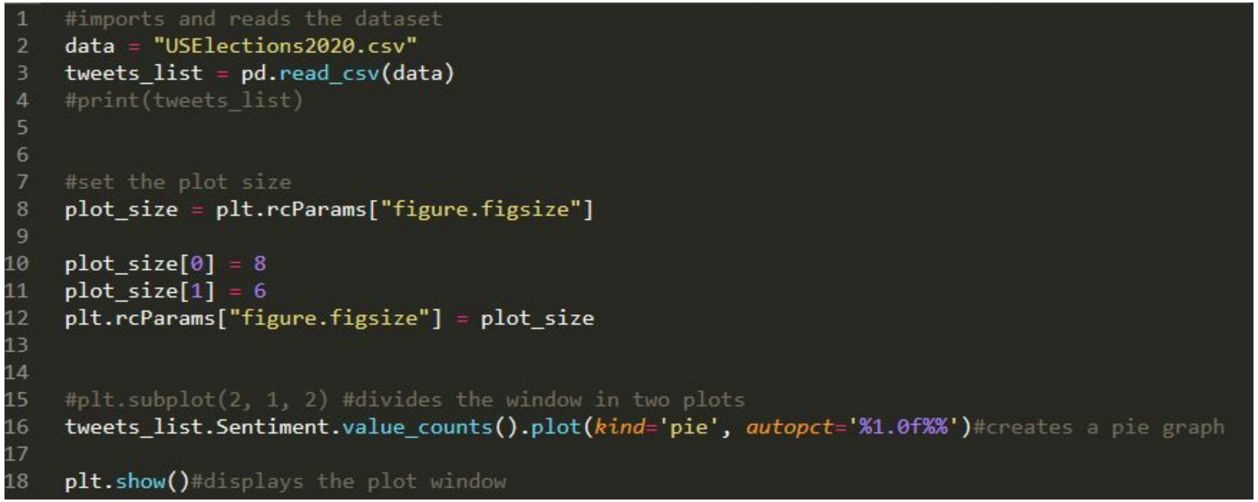
TextBlob and Natural Language Processing (NLP)



```python
def get_tweet_sentiment(tweet):
        '''
        Utility function to classify sentiment of passed tweet
        using textblob's sentiment method
        '''
        # create TextBlob object of passed tweet text
        analysis = TextBlob(clean_tweet(tweet))
        # set sentiment
        if analysis.sentiment.polarity > 0:
            return 'positive'
        elif analysis.sentiment.polarity == 0:
            return 'neutral'
        else:
            return 'negative'

# Collect a list of tweets
tweet_list = [clean_tweet(tweet.full_text) for tweet in tweets]
#print(tweet_list)
sentiment_list = [get_tweet_sentiment(tweet) for tweet in tweet_list]
#print(sentiment_list)

tweets_and_sentiments = pd.DataFrame({"Text":tweet_list, "Sentiment":sentiment_list})
tweets_and_sentiments.to_csv("Coronavirus.csv", index=False)

#print(tweets_and_sentiments)
```

Visualization Tools

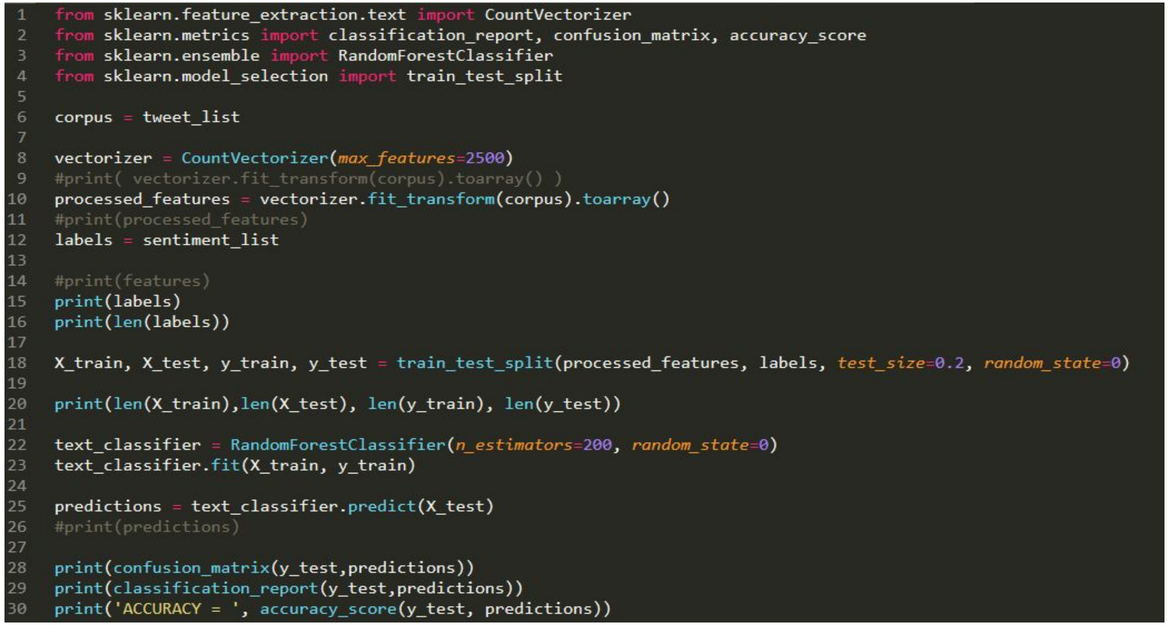
MatplotLib



```
#imports and reads the dataset
data = "USElections2020.csv"
tweets_list = pd.read_csv(data)
#print(tweets_list)


#set the plot size
plot_size = plt.rcParams["figure.figsize"]

plot_size[0] = 8
plot_size[1] = 6
plt.rcParams["figure.figsize"] = plot_size


#plt.subplot(2, 1, 2) #divides the window in two plots
tweets_list.Sentiment.value_counts().plot(kind='pie', autopct='%1.0f%%')#creates a pie graph

plt.show()#displays the plot window
```

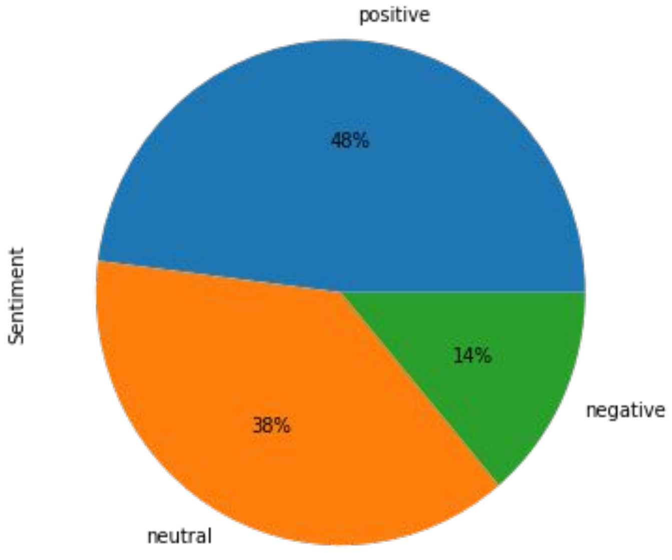
Prediction Model

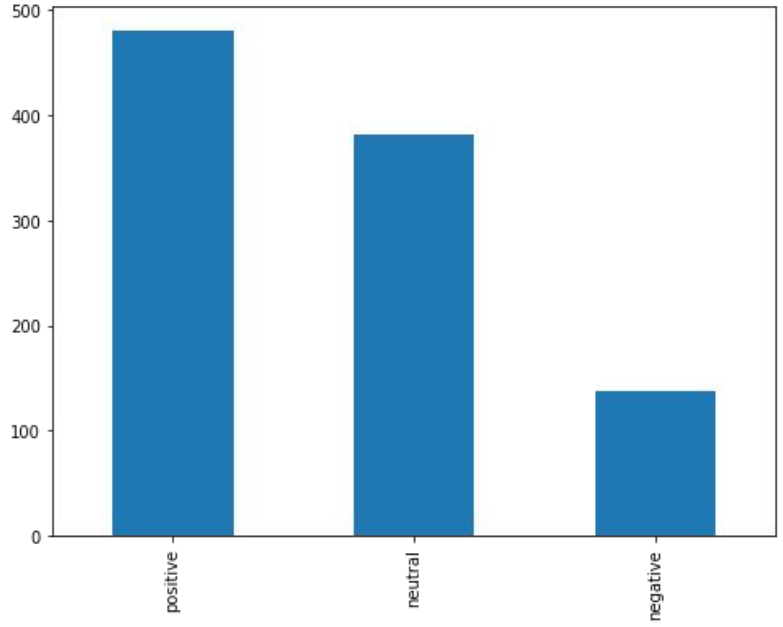
SKlean and Decision Trees



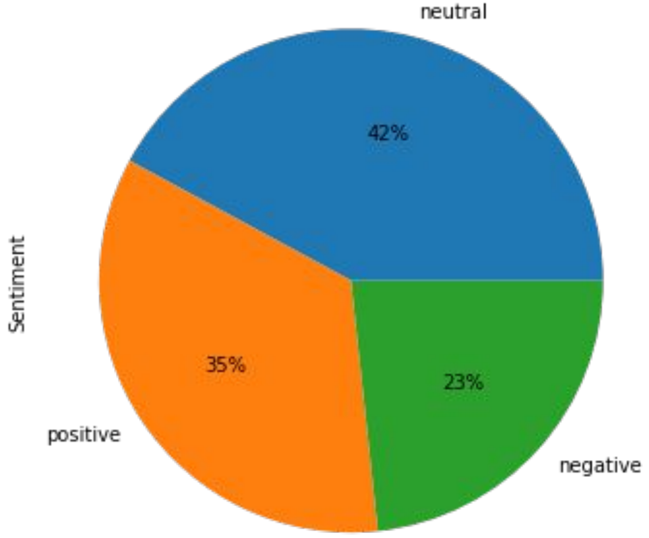
```python
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

corpus = tweet_list

vectorizer = CountVectorizer(max_features=2500)
#print( vectorizer.fit_transform(corpus).toarray() )
processed_features = vectorizer.fit_transform(corpus).toarray()
#print(processed_features)
labels = sentiment_list

#print(features)
print(labels)
print(len(labels))

X_train, X_test, y_train, y_test = train_test_split(processed_features, labels, test_size=0.2, random_state=0)

print(len(X_train),len(X_test), len(y_train), len(y_test))

text_classifier = RandomForestClassifier(n_estimators=200, random_state=0)
text_classifier.fit(X_train, y_train)

predictions = text_classifier.predict(X_test)
#print(predictions)

print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print('ACCURACY = ', accuracy_score(y_test, predictions))
```

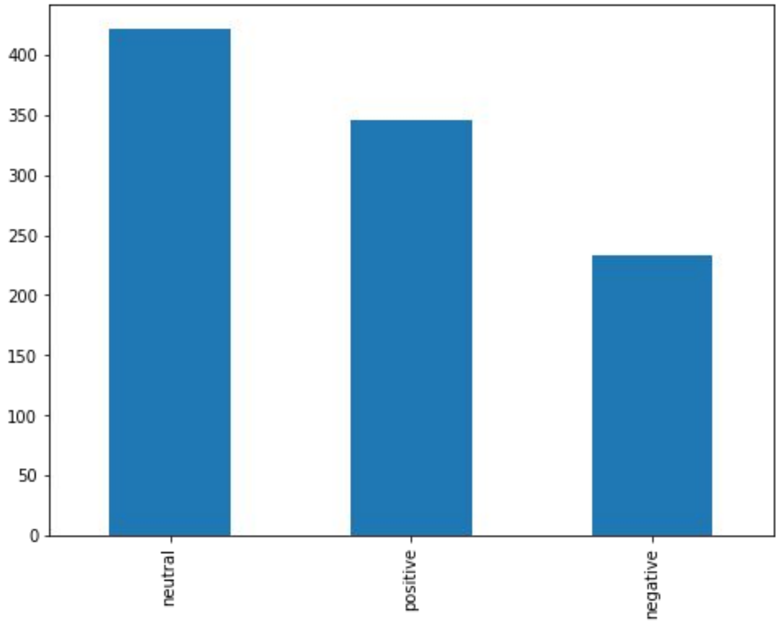
#USElections2020 Results



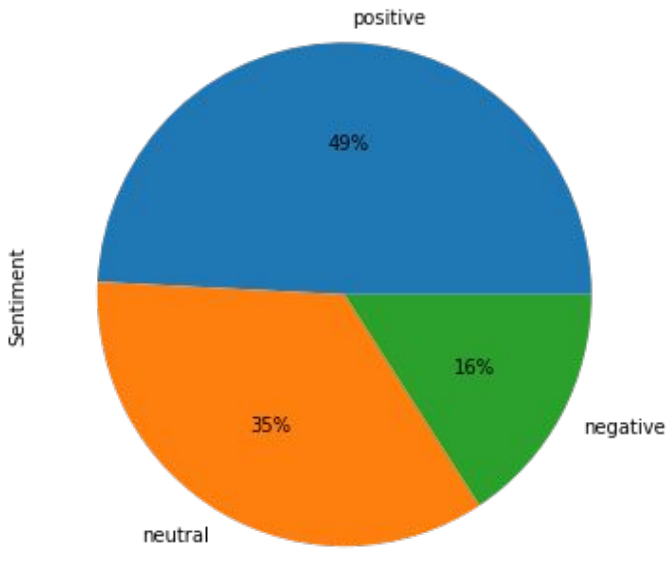


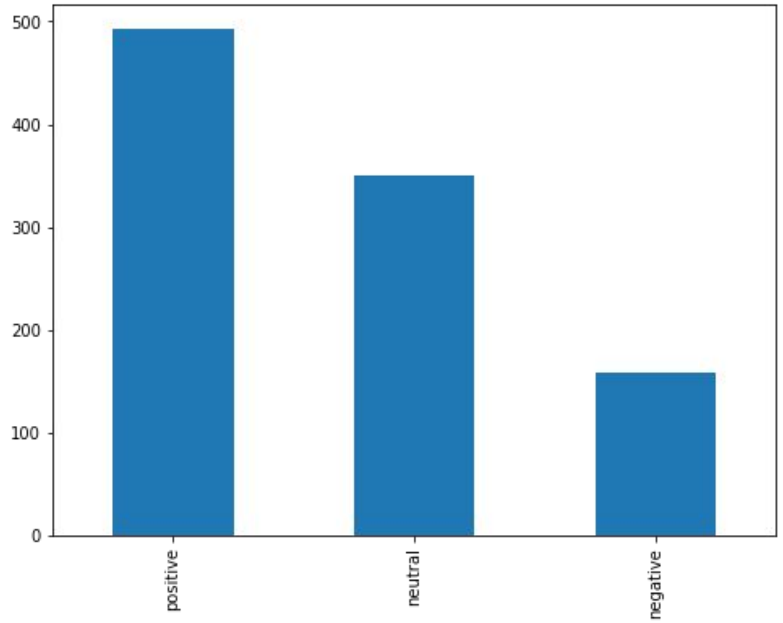
#DonaldTrump Results





#JoeBiden Results





Prediction Model Metrics

ACCURACY = 0.82

	precision	recall	f1-score	support
negative	0.91	0.36	0.51	28
neutral	0.80	0.93	0.86	80
positive	0.83	0.87	0.85	92
accuracy			0.82	200
macro avg	0.85	0.72	0.74	200
weighted avg	0.83	0.82	0.81	200

Conclusion and Further Improvements

- Good Results with simple and cheap implementation
- More Data = More Accuracy
- Fields Flexibility
- NLP is still a developing technology
- Improvements on the specific vocabulary
- Possibilities are limitless!!

Thank you!!