**User Guide and Start Up Guide –**

**How to build a Sentiment Analysis Program Using Twitter**

By Grant Martin

Banner Number: B00340010

University of the West of Scotland

Software Development: Assessment 1

Purpose of Document: To help people build and carry out their very own sentiment analysis from Twitter data, to aid in gathering research for their respected business or purpose.

Who Is the Document For: This document is for people who have very little computing and software development experience and this user-friendly guide will help them on their journey into sentiment analysis.

Purpose of a Sentiment Analysis Program: Sentiment Analysis is the measure of positive and negative language or text. It is a method of gauging written or spoken language to determine if the statement is favourable, unfavourable or neutral and to what extent. Sentiment analysis is used to monitor social media platforms such as Twitter. It allows businesses, companies, groups and even political parties to oversee what the public thinks about their products and ideas.

Note to users: I would recommend you don’t jump straight into this project if you have no experience of using any programming language, I would recommend learning the basics first as this will make it easier to build a sentiment analysis program. Also, if you come across any terms you don’t understand please research them before carrying on in the user and start-up guide, it will increase your understanding of the code and program. Thank You and enjoy performing sentiment analysis.

**Contents of User Guide and Start Up Guide**

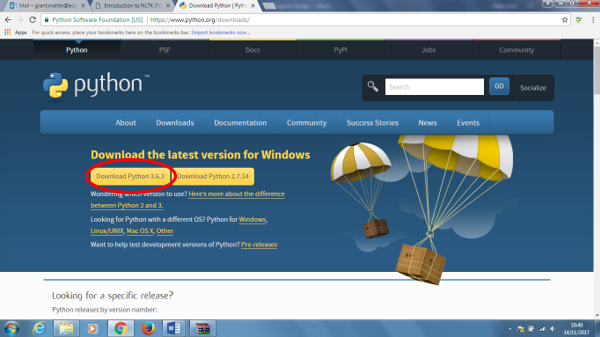
* **Computer Programs Needed and How to Install**
* **Software and Packages Needed and How to Install**
* **How to Set-Up Twitter Account (if user doesn’t already have one)**
* **How to Create an App on Twitter**
* **Overview of the Program we intend to Build**
* **How to Build a Sentiment Analysis Program Using Twitter- Code for Program with comments**
* **Common Mistakes and How to Investigate Bias of Classifier**
* **Self – Reflection**
* **Glossary**
* **References**

**Computer Programs Needed and How to Install**

For this project you will need to install the latest version of Python, which is an interpreted, object-oriented, high-level programming language and you will need to download Pycharm, which is a Python interpreter.

How to Install Python

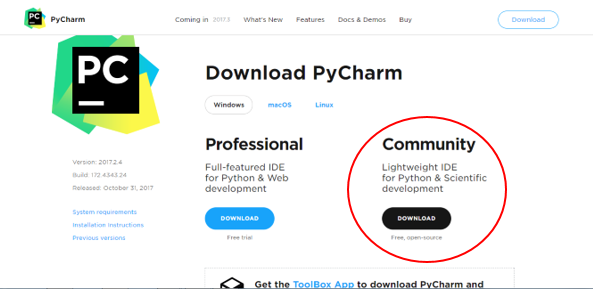
Download the latest version of Python from the internet, links to websites: <https://www.python.org/downloads/>



Click the button circled in red and download Python version 3.6.3

How to Install Pycharm

Download the free version of Pycharm from the internet, link to the website: [https://www.jetbrains.com/pycharm/download/#section=windows](https://www.jetbrains.com/pycharm/download/" \l "section=windows)

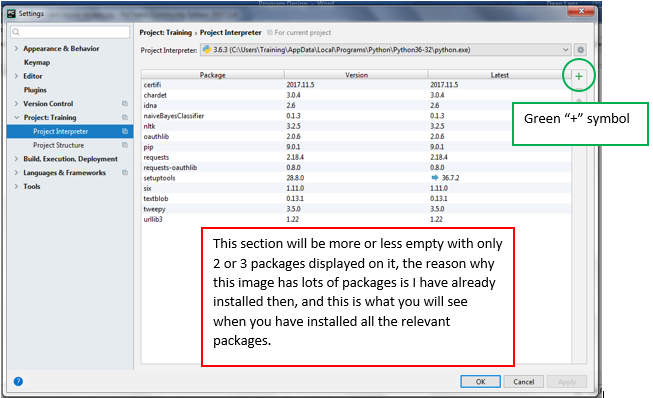
****

Click the Download button highlighted within the red circle

**Software and Packages Needed and How to Install**

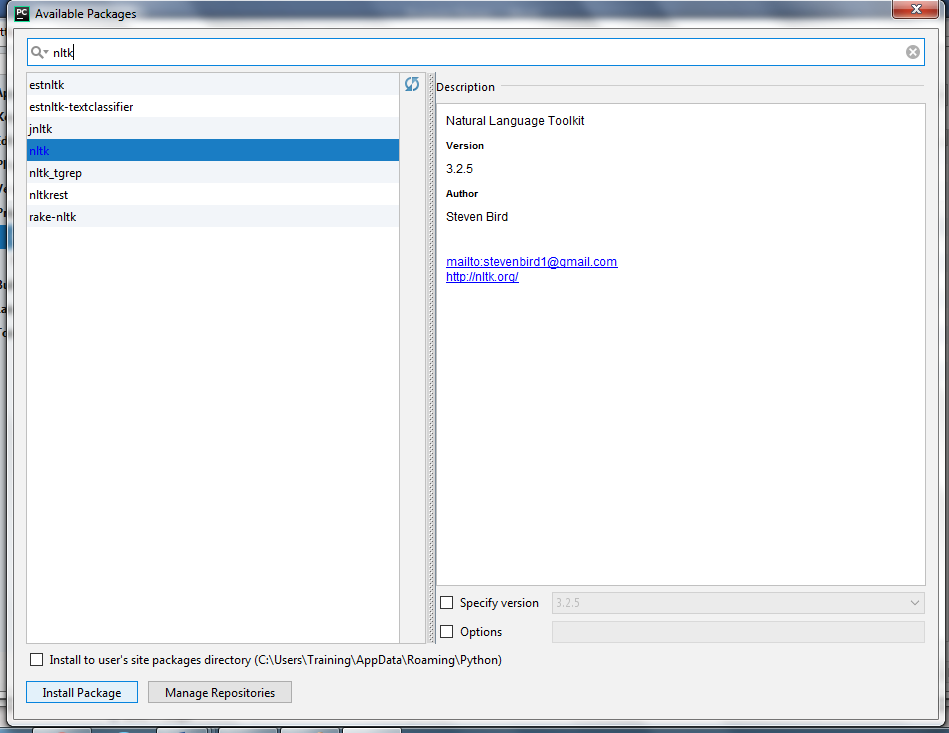
Open the program PyCharm and “create a new project” (keep file name short and simple). We are still not ready to write code, because we firstly need to install some software. Follow steps below:

* Click “file” at the top of the page
* Then from the drop menu scroll down to “settings” and click it



You will be presented with this menu:

* Click “Project Interpreter” which is already highlighted above in blue.
* To install packages, click the green “+” symbol



You will be presented with this menu:

* In the search bar at the top of the page input the name of the package you want to install. For building a sentiment analysis program we want to install: Tweepy, Natural Language Toolkit (NLTK), Naïve Bayes Classifier. In this example, I am searching for the NLTK.
* On the left-hand options will be displayed, in this case I click “NLTK” and then click “Install Package” at the bottom of the menu.
* Once the package is installed, click the cross at the top right-hand side and repeat the steps for installing Tweepy and Naïve Bayes Classifier.

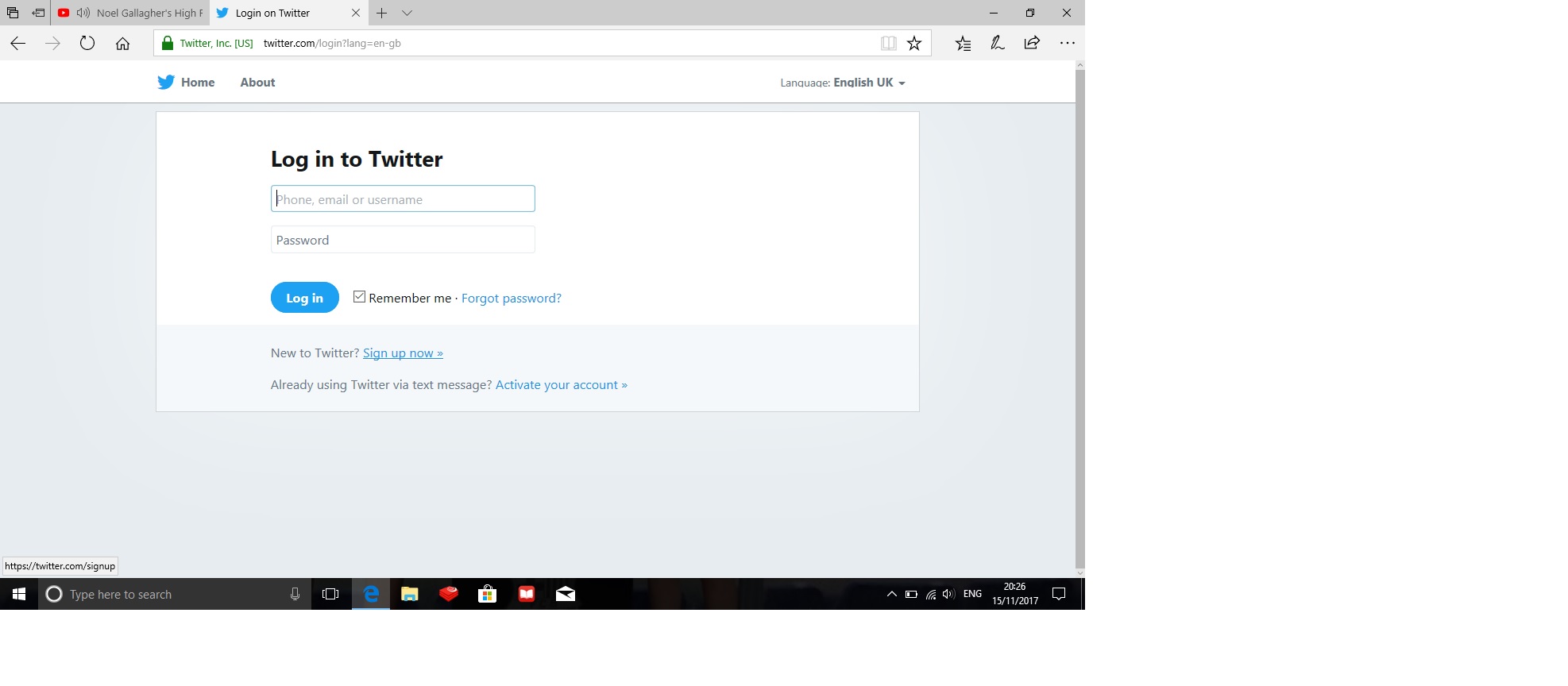
Now the packages are installed, your first two lines of code on Pycharm will be:



* And then click “Run” to download the NLTK. This will take some time to download and can take an hour. This download will download ‘Stopwords’ and all sorts of great packages for processing and analysing the text of the tweets. It also downloads data, that can be used to train and test the classifier of the program and improve the accuracy of the program.
* Once the NLTK has finished downloading, you are ready to start writing your code.

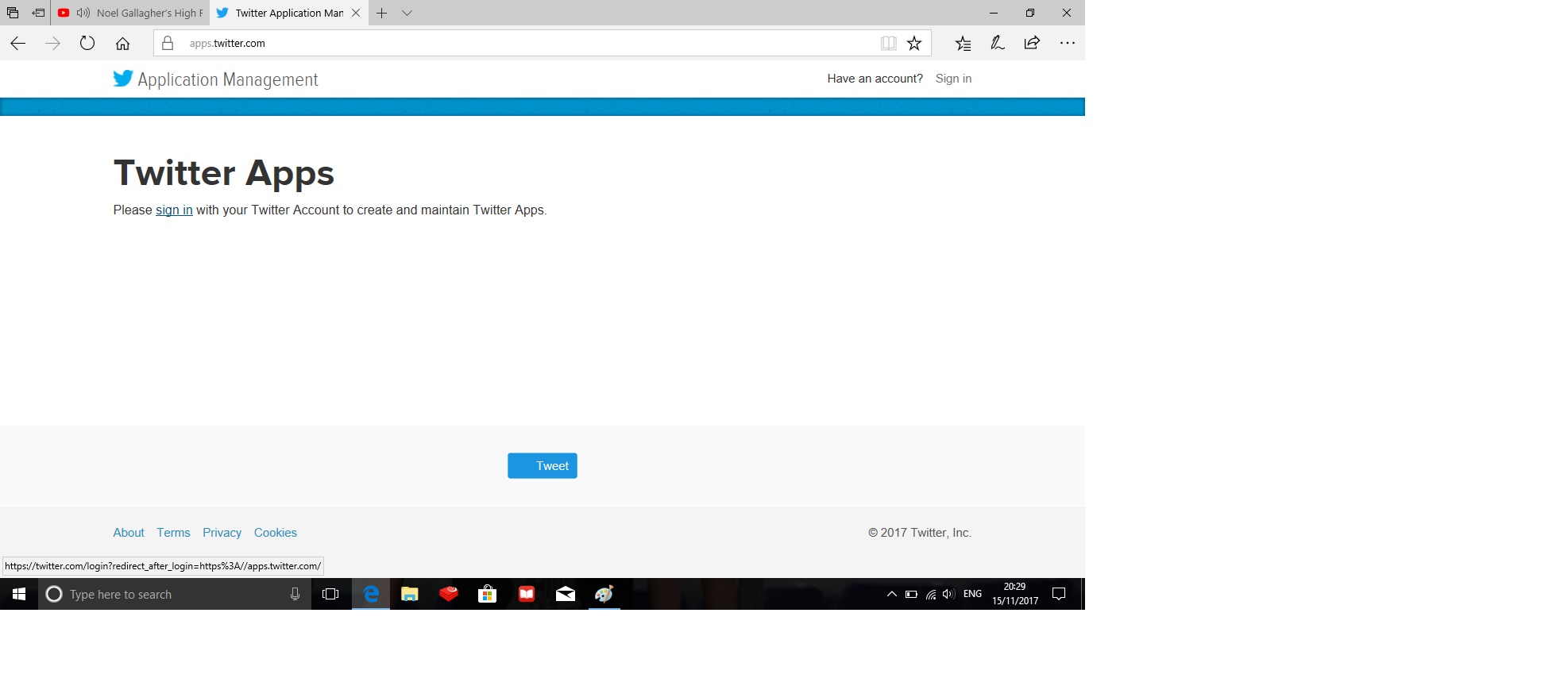
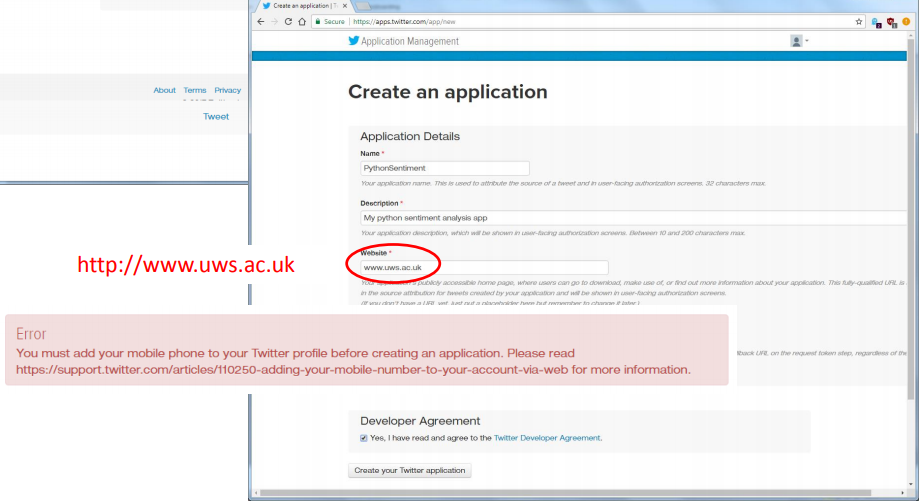
**How to Set-Up a Twitter Account (if you don’t already have one)**

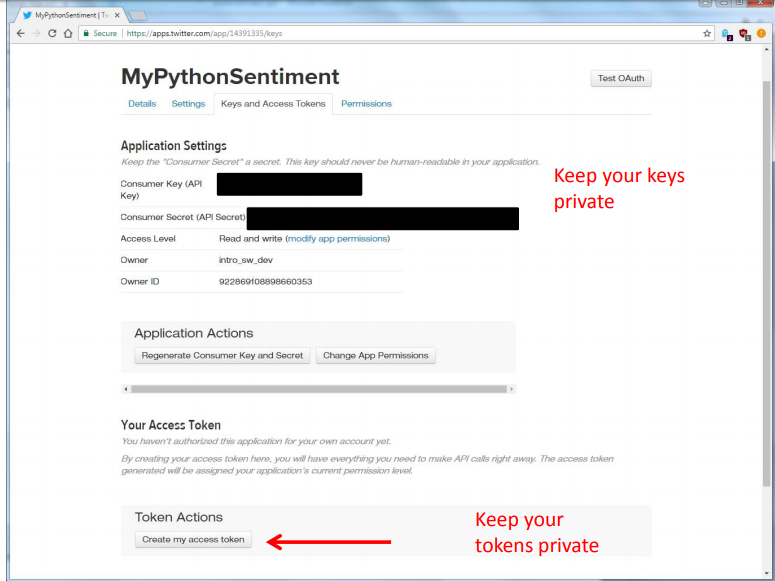
You will need a Twitter account to perform this sentiment analysis program, so if you don’t already have one you will have to create a Twitter account. I would also recommend that you don’t use your existing personal account. So, please make a Twitter account only for this program.

* To create a Twitter account, go to twitter.com and you will be presented with the page below.
* Then click on the “Sign up now” button and fill in the appropriate details

**How to create an App on Twitter**

Firstly, you need a Twitter account once you have one you are ready to create an app on Twitter and this is how you do that.

* Go to apps.twitter.com and click the “sign in” button
* You will then be presented with the page below



* On the next page you will receive your Consumer Key and Consumer Secret, please keep these pieces of information private
* Then you can click “create my access token” again write down these pieces of information and keep them private. You will need these pieces of information for your code later in the user and start-up guide.

Pip Install using the Command Prompt

Firstly, To access the Scikit-Learn, numpy, spicy and matplotlib you have to pip install them using the command prompt. Steps to install these programs:

* Click the start button and search ‘Command Prompt’ and click on it.
* After, that has installed type ‘pip install matplotlib’ and press the enter/return key on the keyboard.
* After, that has installed type ‘pip install scikit-learn’ and press the enter/return key on the keyboard.
* After, that has installed type ‘pip install numpy’ and press the enter/return key on the keyboard.
* After, that has installed type ‘pip install spicy’ and press the enter/return key on the keyboard.

**Overview of the Program we intend to Build**

When building your own software program, it is also important to plan your project and understand what pieces of code you need before you start, for this program I have drawn up a program design flow diagram informing us what we need to build this sentiment analysis program. See Diagram on the next page.

**Program Design - Flow Diagram**

Enter Key and Access Tokens from Twitter in Python code

JSON/Pickle files will be used to store and save data, which can be implemented into the code using strings

(twitter data is saved and formatted in the JSON file format)

Detailed Subproblem

Level 2

Subproblem

Level 1

Create a Twitter Account and Create an App with Twitter

Install Tweepy

Original Problem

Level 0

Install JSON

Or Pickle

Install Natural Language Tool Kit (NLTK)

Access Twitter API data

How to Build a Sentiment Analysis Program

Remove the ‘Stopwords’ from the text (example: ‘a’, ‘the’, ‘me’ etc)

Tokenise the text into words

Install Naïve Bayes Classifier, to perform classifications

Print the Classification result (positive or negative)

Make a ‘GET’ request to access tweets from Twitter.

**How to Build a Sentiment Analysis Program – Code**

**Index**

* **Introduction to the Execution of Code**
* **How to Download Your Training Data**
* **Code for Training the Naïve Bayes Classifier**
* **Code for Testing the Naïve Bayes Classifier**
* **Code for Creating a module for Sentiment Analysis**
* **Code for Sentiment Analysis to be carried out on live twitter data**
* **Code for Twitter Sentiment Analysis on live tweets**
* **Code for Graphing Live Twitter Sentiment Analysis**

**Introduction to the Execution of Code**

This section of the document will provide the code needed to build and run your very own sentiment analysis program. This sentiment analysis program will be able to flitter and determine whether the tweets about your particular query are positive or negative. It is important you run all of these pieces of code, you cannot just run the last piece of code and expect the program to work and have a fully operational sentiment analysis program. So, take your time and be patience as some piece of code take a little bit of time to run, so when that happens you can go and make yourself a cup of tea.

To start the project, you need to open PyCharm and ‘create new project’ name the project ‘sentiment analysis’ then click ‘file’-‘new’-‘python file’ and name it ‘naïve bayes classifier’.

**How to download Your Training Data**

To download the training data for the naïve bayes classifier go to: <https://pythonprogramming.net/static/downloads/short_reviews/>

And download both the ‘negative.txt’ file and the ‘positive.txt’ file and save them in the exact same location of as your pycharm project ‘sentiment analysis’ the location of this file will probably be within your C-Drive on your computer in the users folders in the pycharm projects folder.

**Code for Training the Naïve Bayes Classifier**

* In the ‘naïve bayes classifier’ file copy or type the below code into the file and then click ‘run’ at the top of the page.

import nltk

import random

import pickle

from nltk.tokenize import word\_tokenize

# I will be using the NLTK to assist me in building my naive bayes classifier

# import random will be used to shuffle my training and testing dataset of short movie reviews

# to make the classifier accurate and reliable when processing live tweets

# My dataset has already been labelled as positive and negative, making it possible to train and test with

# See the user guide for instructions on how to download the positive and negative.txt files for training and testing classifier.

# 2 two lines below open the text files and reads the text data contained within.

short\_pos = open("positive.txt", "r").read()

short\_neg = open("negative.txt", "r").read()

documents = []

# documents equals empty list

# r equals review, so for every review split them with a new line

for r in short\_pos.split('\n'):

documents.append((r, "pos"))

for r in short\_neg.split('\n'):

documents.append((r, "neg"))

random.shuffle(documents)

#This will mix up the positive and negative documents

all\_words = []

#all words equals empty list

# The code below will tokenize the words in the positive and negative text files.

short\_pos\_words = word\_tokenize(short\_pos)

short\_neg\_words = word\_tokenize(short\_neg)

for w in short\_pos\_words:

all\_words.append(w.lower())

for w in short\_neg\_words:

all\_words.append(w.lower())

all\_words = nltk.FreqDist(all\_words)

# The line of code above will form a list of the most common words in the text files.

word\_features = list(all\_words.keys())[:5000]

# The above line of code records the most common 5000 words from both text files.

def find\_features(document):

words = word\_tokenize(document)

features = {}

for w in word\_features:

features[w] = (w in words)

return features

# The line of code below does this to all documents, saving the feature existence booleans and the positive or negative categories

featuresets = [(find\_features(rev), category) for (rev, category) in documents]

random.shuffle(featuresets)

#This mixes up the positive and negative featuresets

# The below code follows the process of the supervised classification flow diagram that can be found on page ? of the user guide

# dataset I will train classifier with

training\_set = featuresets[:10000]

# dataset I will test classifier against

testing\_set = featuresets[10000:]

# The below code defines and trains my naive bayes classifier

classifier = nltk.NaiveBayesClassifier.train(training\_set)

# The 3 lines of code below open the previously saved pickle file to run in the code

#classifier\_f = open("naivebayes.pickle", "rb")

#classifier = pickle.load(classifier\_f)

#classifier\_f.close()

print("Classifier accuracy percent:",(nltk.classify.accuracy(classifier, testing\_set))\*100)

classifier.show\_most\_informative\_features(15)

# The 2 lines of code above give me a list of the 15 most informative words when the code is run and the accuracy of test data

# The code below allows me to save the naive bayes classifier process of running through the dataset

# By inserting import pickle at the top of the code, I can serialize the classifier and load it into my code, this saves time.

save\_classifier = open("naivebayes.pickle","wb")

pickle.dump(classifier, save\_classifier)

save\_classifier.close()

# The above 3 lines of code saved and stored the results of my code in a pickle file, to be accessed at any point in the future.

# Then I commented off the above code with the ‘#’ to allow me to carry out my next lines of code.

# which involved me uploading the saved data from the pickle file straight back into my code, See below for how it works.

# This opens up a pickle file, preparing to write in bytes some data.

# I used pickle.dump() to dump the data.

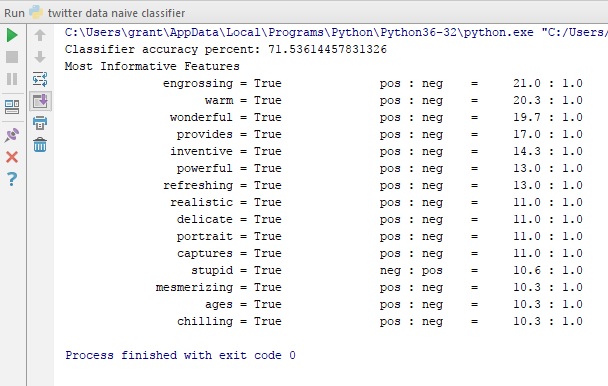
# The first parameter to pickle.dump() is what are you dumping.

# The second parameter is where are you dumping it.

# Close the file and now I have a pickle file saved.

# Reference for the code: https://pythonprogramming.net/naive-bayes-classifier-nltk-tutorial/

* What this code is doing is training the naïve bayes classifier to detect positive and negative sentiment from the dataset we just downloaded and then saving the classifier process in a pickle file.
* See below for what a result will look like on your computer after the code has run



**Code for Testing the Naïve Bayes Classifier**

* The code for testing the naïve bayes classifier is exactly the same as the code above with the only different being we are uncommenting some of the lines which are commented and commenting (#) some lines which are currently uncommented.

#classifier = nltk.NaiveBayesClassifier.train(training\_set)

classifier\_f = open("naivebayes.pickle", "rb")

classifier = pickle.load(classifier\_f)

classifier\_f.close()

print("Classifier accuracy percent:",(nltk.classify.accuracy(classifier, testing\_set))\*100)

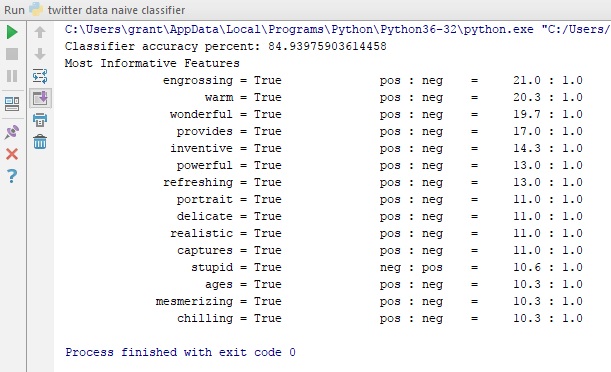
classifier.show\_most\_informative\_features(15)

#save\_classifier = open("naivebayes.pickle","wb")

#pickle.dump(classifier, save\_classifier)

#save\_classifier.close()

* Basically, the different between these two different sets of code is the code for the classifier at the bottom. In the code for training the naïve bayes classifier, the process of the naïve bayes classifier working through the Twitter sample dataset (10000 long) is being saved to a pickle file and in the above code for testing the naïve bayes classifier, I am using the exact same Twitter sample dataset and the code is telling the computer to open the saved classifier (which the previous code saved) and run it over the dataset and give me a result of the accuracy of the program. (see image of result below)



**Code for Creating a Module for Sentiment Analysis**

* For this you will need a new file, so again click ‘file’-‘new’-‘python file’ and save the file has ‘creating module for sentiment analysis’
* The code for creating a module for sentiment analysis is below, please copy and paste it into PyCharm or type it out yourself.

import nltk

import random

import pickle

from nltk.classify import ClassifierI

#from statistics import mode

from nltk.tokenize import word\_tokenize

# import nltk gives me access to the nltk libraries of data and programs for data analysis

# import random will be used to shuffle my training and testing dataset of short movie reviews

# to make the classifier accurate and reliable when processing live tweets

# My dataset has already been labelled as positive and negative, making it possible to train and test with

# import pickle will insert my previously saved and serialised file of my naive bayes classifier and most common 5000 words

# word\_tokenize will tokenizes the dataset, separating each word from the body of text as tokens

# I imported mode, this will choose the most popular classifier vote (this code was used when I had more classifiers in the code)

# Line classifierI is the classifier being used on the data

# The class below is for my classifier

# The classifier is called VoteClassifier and is inherting ClassifierI

# The classifiers well in this case the naive bayes classifier is programmed to pass through the class to self.classifier

# In the second function 'def classify' I define my classify process, so I can call on it later on.

# The functions below are passing through the classifier and classifying by features

# The classification is being processed as a vote (was more effective when I had more classifiers)

# Finally the class returns the the mode(vote), the most popular classifier (again better when you have more classifiers)

class VoteClassifier(ClassifierI):

def \_\_init\_\_(self, \*classifiers):

self.\_classifiers = classifiers

def classify(self, features):

votes = []

for c in self.\_classifiers:

v = c.classify(features)

votes.append(v)

return mode(votes)

def confidence(self, features):

votes = []

for c in self.\_classifiers:

v = c.classify(features)

votes.append(v)

choice\_votes = votes.count(mode(votes))

conf = choice\_votes / len(votes)

return conf

# confidence function is simply counting up the votes for each classifiers (again working better when you use more classifiers)

# See the user guide for instructions on how to download the positive and negative.txt files for training and testing classifier.

# 2 two lines below open the text files and reads the text data contained within.

short\_pos = open("positive.txt", "r").read()

short\_neg = open("negative.txt", "r").read()

all\_words = []

documents = []

# all\_words equals empty list

# documents equals empty list

# j is adjective, r is adverb, and v is verb

# allowed\_word\_types = ["J","R","V"]

allowed\_word\_types = ["J"]

# I am only looking for adjectives in the dataset

for p in short\_pos.split('\n'):

documents.append((p, "pos"))

words = word\_tokenize(p)

pos = nltk.pos\_tag(words)

for w in pos:

if w[1][0] in allowed\_word\_types:

all\_words.append(w[0].lower())

# The above if statement is saying if the word is an adjective I want to append that word

for p in short\_neg.split('\n'):

documents.append((p, "neg"))

words = word\_tokenize(p)

pos = nltk.pos\_tag(words)

for w in pos:

if w[1][0] in allowed\_word\_types:

all\_words.append(w[0].lower())

# The above if statement is saying if the word is an adjective I want to append that word

# Below I am saving the words in a pickle file

save\_documents = open("documents.pickle", "wb")

pickle.dump(documents, save\_documents)

save\_documents.close()

# The above 3 lines of code saved and stored the results of my code in a pickle file, to be accessed at any point in the future.

all\_words = nltk.FreqDist(all\_words)

# The line of code above will form a list of the most common words in the text files.

word\_features = list(all\_words.keys())[:5000]

# The above line of code records the most common 5000 words from both text files.

save\_word\_features = open("word\_features5k.pickle", "wb")

pickle.dump(word\_features, save\_word\_features)

save\_word\_features.close()

# The above 3 lines of code saved and stored the results of my code in a pickle file, to be accessed at any point in the future.

def find\_features(document):

words = word\_tokenize(document)

features = {}

for w in word\_features:

features[w] = (w in words)

return features

# The line of code below does this to all documents, saving the feature existence booleans and the positive or negative categories

featuresets = [(find\_features(rev), category) for (rev, category) in documents]

random.shuffle(featuresets)

#This mixes up the positive and negative featuresets

print(len(featuresets))

# The line of code above prints the length of the dataset (total number of positive and negative datasets)

# dataset I will test classifier against

testing\_set = featuresets[10000:]

# dataset I will train classifier with

training\_set = featuresets[:10000]

classifier = nltk.NaiveBayesClassifier.train(training\_set)

print("Original Naive Bayes Algo accuracy percent:", (nltk.classify.accuracy(classifier, testing\_set)) \* 100)

classifier.show\_most\_informative\_features(15)

# The above lines of code will print the percentage accuracy of the naive bayes classifier and the 15 most common words

save\_classifier = open("originalnaivebayes5k.pickle", "wb")

pickle.dump(classifier, save\_classifier)

save\_classifier.close()

# The above 3 lines of code saved and stored the results of my code in a pickle file, to be accessed at any point in the future.

# 'open' create a new pickle file

# 'wb' means write in bytes

# I used pickle.dump() to dump the data.

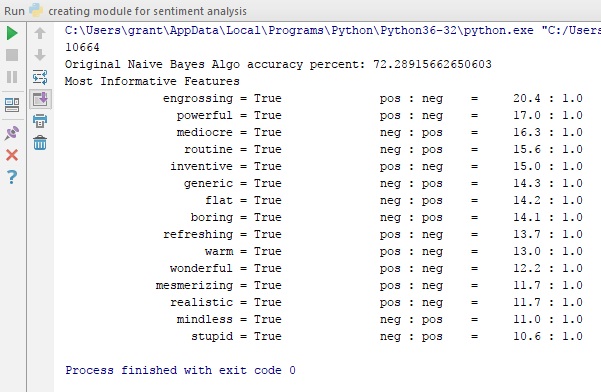
# The first parameter to pickle.dump() is what are you dumping.

# The second parameter is where are you dumping it.

# Close the file and now I have a pickle file saved.

# Reference for code used: https://pythonprogramming.net/sentiment-analysis-module-nltk-tutorial/

* To run the code again go to the top of the page click ‘run’ and then you need to click what file you want to run, click ‘creating module for sentiment analysis’
* See below for what an expected result will be



**Code for Sentiment Analysis to be carried out on live twitter data**

* You will need to create a new file again by clicking ‘file’-‘new’-‘python file’ and name the file ‘sentiment\_mod’
* Here is the code below

import random

import pickle

#from nltk.classify import ClassifierI

#from statistics import mode

from nltk.tokenize import word\_tokenize

# import random will be used to shuffle my training and testing dataset of short movie reviews

# to make the classifier accurate and reliable when processing live tweets

# My dataset has already been labelled as positive and negative, making it possible to train and test with

# import pickle will insert my previously saved and serialised file of my naive bayes classifier and most common 5000 words

# word\_tokenize will tokenizes the dataset, separating each word from the body of text as tokens

# The class below is for my classifier

# The classifier is called VoteClassifier and is inherting ClassifierI

# The classifiers well in this case the naive bayes classifier is programmed to pass through the class to self.classifier

# In the second function 'def classify' I define my classify process, so I can call on it later on.

# The functions below are passing through the classifier and classifying by features

# The classification is being processed as a vote (was more effective when I had more classifiers)

# Finally the class returns the the mode(vote), the most popular classifier (again better when you have more classifiers)

class VoteClassifier(ClassifierI):

def \_\_init\_\_(self, \*classifiers):

self.\_classifiers = classifiers

def classify(self, features):

votes = []

for c in self.\_classifiers:

v = c.classify(features)

votes.append(v)

return mode(votes)

def confidence(self, features):

votes = []

for c in self.\_classifiers:

v = c.classify(features)

votes.append(v)

choice\_votes = votes.count(mode(votes))

conf = choice\_votes / len(votes)

return conf

# confidence function is simply counting up the votes for each classifiers (again working better when you use more classifiers)

# The 3 lines of code below are simply opening the pickle file of the documents, I created in the last piece of code.

documents\_f = open("documents.pickle", "rb")

documents = pickle.load(documents\_f)

documents\_f.close()

# The 3 lines of code below are simply opening the pickle file of the word features, I created in the last piece of code.

word\_features5k\_f = open("word\_features5k.pickle", "rb")

word\_features = pickle.load(word\_features5k\_f)

word\_features5k\_f.close()

# The lines of code below are creating a function (def) to tokenize the words contained within the document

def find\_features(document):

words = word\_tokenize(document)

features = {}

for w in word\_features:

features[w] = (w in words)

return features

# The line of code below does this to all documents, saving the feature existence booleans and the positive or negative categories

featuresets = [(find\_features(rev), category) for (rev, category) in documents]

#This mixes up the positive and negative featuresets

random.shuffle(featuresets)

print(len(featuresets))

# The line of code above prints the length of the dataset (total number of positive and negative datasets)

# dataset I will test classifier against

testing\_set = featuresets[10000:]

# dataset I will train classifier with

training\_set = featuresets[:10000]

# The 3 lines of code below are opening the pickle file of the naive bayes classifier

# The pickle file has saved the classifier, this reduces the run time of the program especially if being used on large dataset

open\_file = open("originalnaivebayes5k.pickle", "rb")

classifier = pickle.load(open\_file)

open\_file.close()

voted\_classifier = VoteClassifier(

classifier)

# The last piece of code is the most important piece of code for the next python files

# The function called 'sentiment' is created and takes the text, analyses the features of the text using find\_features and returns the sentiment (positive or negative)

def sentiment(text):

feats = find\_features(text)

return voted\_classifier.classify(feats), voted\_classifier.confidence(feats)

# Reference for the code used: https://pythonprogramming.net/sentiment-analysis-module-nltk-tutorial/

* After, you have inserted the above code click ‘run’ then click ‘sentiment\_mod’
* Next step, is to test if the trained sentiment analysis program will work before trying it on live Twitter data (see results below)
* You don’t have to carry out this test you can skip to the next piece of code if you would like, but it is always good to test your code. To perform the next piece of code you need a new file, so click ‘file’-‘new’-‘python file’ and name it ‘test sentiment’
* The first line of code below imports the last piece of code I wrote above to carry out sentiment analysis on the two print statements below.

import sentiment\_mod as s

# This python file is just a test to make sure the 'sentiment\_mod' function I created in the last piece of code works

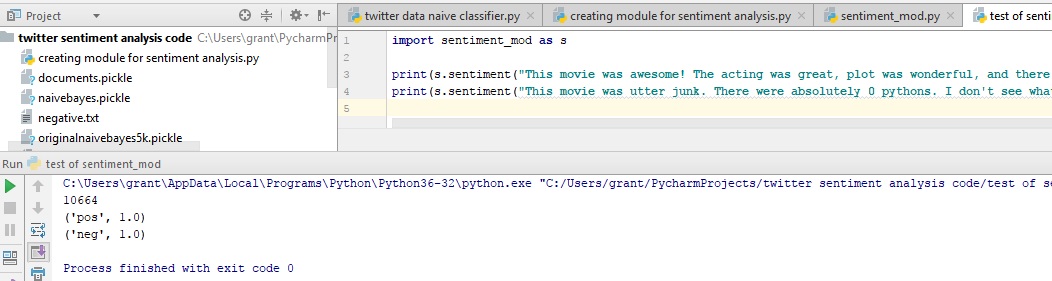
# import sentiment\_mod is being imported into the code as 's'

print(s.sentiment("This movie was awesome! The acting was great, plot was wonderful, and there were pythons...so yea!"))

print(s.sentiment("This movie was utter junk. There were absolutely 0 pythons. I don't see what the point was at all. Horrible movie, 0/10"))

# Reference for code used: https://pythonprogramming.net/sentiment-analysis-module-nltk-tutorial/

* See the expected result below



**Code for Twitter Sentiment Analysis on live tweets**

* Finally, to carry out sentiment analysis on Twitter data click ‘file’-‘new’-‘python file’ save the file as ‘twitter sentiment analysis’ and copy and paste or type the code below and insert in your Pycharm file

from tweepy import Stream

from tweepy import OAuthHandler

from tweepy.streaming import StreamListener

import json

import sentiment\_mod as s

# The above import sentiment\_mod as s line of code imports the sentiment analysis fuction from the last piece of code.

# tweepy is the python client for the official Twitter API to gather tweets through Twitter API

# To receive your own consumer key and secret, access token and secret

# You need to view my user guide for the instructions on how to do this

# Consumer keys and access tokens, used for OAuth (OAuthentication)

# It is used for security and to make sure you have permission to use the Twitter API data

consumer\_key = "use your own"

consumer\_secret = "use your own"

access\_token = "use your own"

access\_secret = "use your own"

# Line 4 of the code 'import json' allows me to use the json module to load the tweet data with the code below 'json.loads(data)'

# The line of code below 'tweet = all\_data["text"]' allows me to target the tweets specifically

# Once I have a tweet I can pass it through the sentiment\_mod

# In the code below the line 'output = open("twitter-out.txt", "a")' and the code that follows will output the tweets into a json file with its sentiment score

class listener(StreamListener):

def on\_data(self, data):

all\_data = json.loads(data)

tweet = all\_data["text"]

sentiment\_value, confidence = s.sentiment(tweet)

print(tweet, sentiment\_value, confidence)

if confidence \* 100 >= 80:

output = open("twitter-out.txt", "a")

output.write(sentiment\_value)

output.write('\n')

output.close()

return True

def on\_error(self, status):

print(status)

# OAuth process, using the keys and tokens

auth = OAuthHandler(consumer\_key, consumer\_secret)

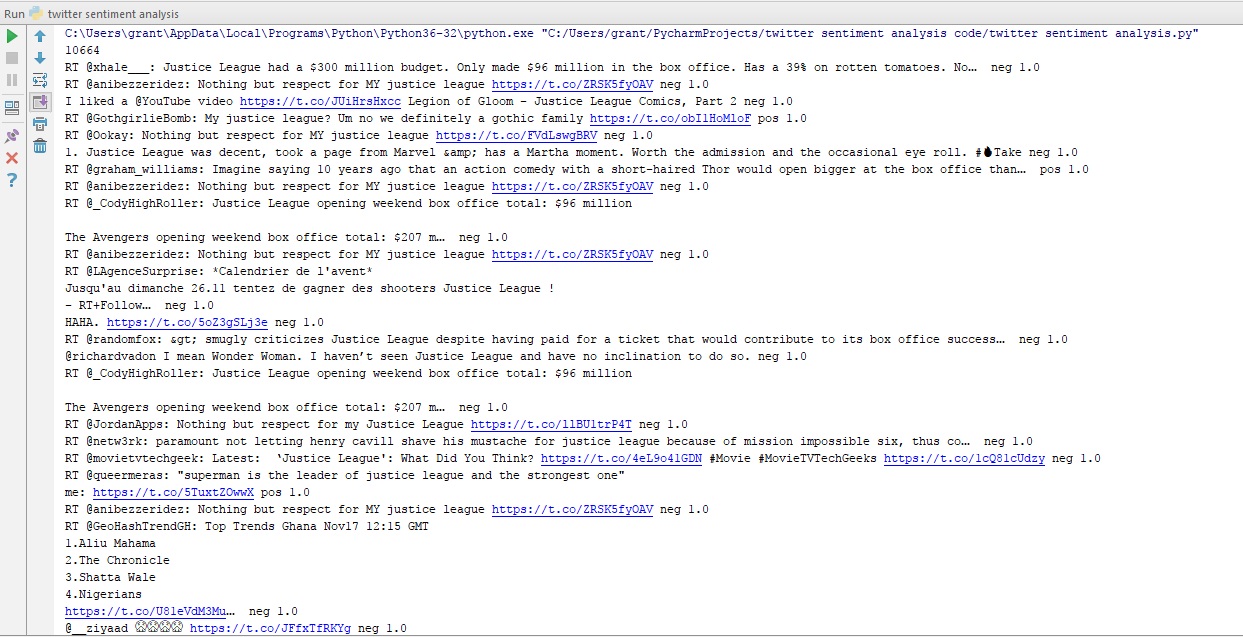
auth.set\_access\_token(access\_token, access\_secret)

twitterStream = Stream(auth, listener())

twitterStream.filter(track=["justice league"])

# The stream is retreiving all the live tweets about your chosen query, in this case justice league.

# Reference for the code used: https://pythonprogramming.net/twitter-sentiment-analysis-nltk-tutorial/

* After, you have inserted the code click ‘run’ then click ‘twitter sentiment analysis’
* Below is an image of expected results, remember to stop the code from run after a while because it can crash your computer if you leave the code running for long periods of time.

**Code for Graphing Live Twitter Sentiment Analysis**

* Finally, this last piece of code is optional, and it will allow you to quickly view and examine the results of your program.
* The steps to graph your sentiment, begin by creating new file by clicking ‘file’-‘new’-‘python file’ and name the file ‘graph’
* The code is below, please copy and paste or type the code into Pycharm and once it is inserted click ‘run’ and then ‘graph’

import matplotlib.pyplot as plt

import matplotlib.animation as animation

from matplotlib import style

import time

# import matplotlib allows me to present the live Twitter data in the form of and line graph, measuring the sentiment of the tweets

#

# style 'ggplot' just makes the graph look better and pleasing to the eye

style.use("ggplot")

fig = plt.figure()

ax1 = fig.add\_subplot(1, 1, 1)

# the pullData is the data I will use to construct the graph

# The pullData I will be using is the 'twitter-out.txt' file I created in the last piece of code

# lines equal to pullData split by new line

# 'xar' X array equals empty list

# 'yar' Y array equals empty list

# The line of code 'for l (line) in lines [-200:]:' is saving the graph data when it reaches 200 tweets

# The function below is constructing the graph

# x=0 and y=o are the starting points of the graph

# If the tweet is positive y = plus one and if the tweet is negative y= minus 1

def animate(i):

pullData = open("twitter-out.txt", "r").read()

lines = pullData.split('\n')

xar = []

yar = []

x = 0

y = 0

for l in lines[-200:]:

x += 1

if "pos" in l:

y += 1

elif "neg" in l:

y -= 1

xar.append(x)

yar.append(y)

ax1.clear()

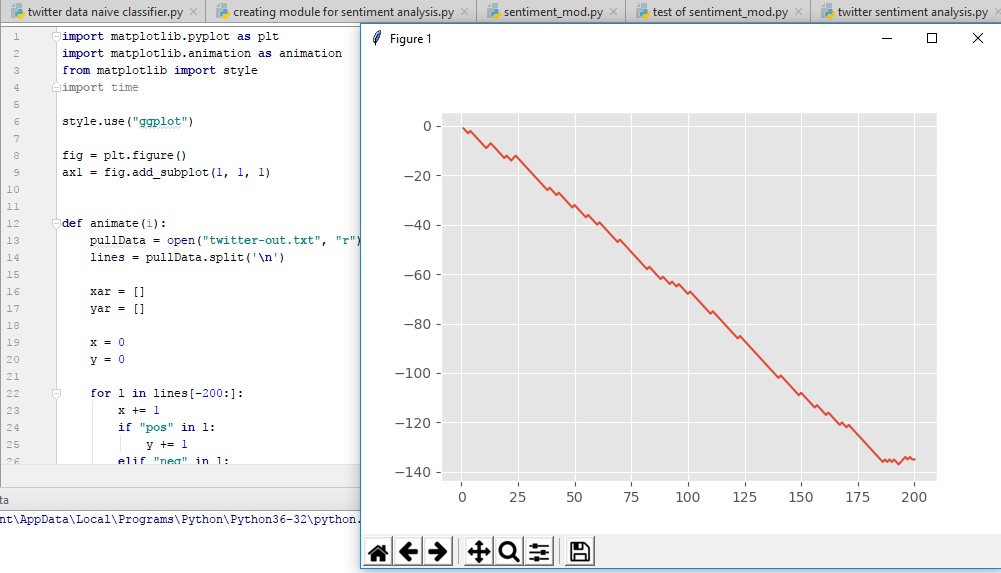
ax1.plot(xar, yar)

ani = animation.FuncAnimation(fig, animate, interval=1000)

plt.show()

# Reference for the code used: <https://pythonprogramming.net/graph-live-twitter-sentiment-nltk-tutorial/>

* See below for expected result and image of graph



**Common Mistakes and How to Investigate Bias of Classifier**

* Remember to download both the negative.txt and positive.txt files and save them in the same location as your pycharm project folder, where your sentiment analysis program code is.
* Remember to keep files the same name in your code and file location so the code can run the files.
* Plan your program before starting to write code
* This classifier is trained using short movie reviews, so if you try to run this classifier on tweets about anything other than movies the accuracy and reliable will not be as good. Always train your classifier with similar data you want to target on Twitter.

**How to Investigate Bias of Classifier**

* To investigate bias first comment off the random.shuffle line of code on the training and test data for the classifier.
* Then train against the first 9000 and test the classifier against the last 1000 positive data reviews
* If you have a low accuracy you have problems
* Then test against the first 1000 negative reviews and then train with the remaining 9000 reviews, if you have a high accuracy you have problems with bias.

**Self – Reflection**

In conclusion this project, how to build a sentiment analysis program has been difficult and challenging. The project would not have been possible if I didn’t do any background reading and planning, I believe one of the most successful parts of this project is the planning and understanding what software and packages I needed from the start to write my code. The first steps of my project started with pens and paper sketching out the program design flow diagram and supervised classification diagram, to fully understand how the components of the code were interacting and working. The majority of my time spend on this project was reading and watching video tutorial guides and reading how python and its code works.

Comments about the program and the code used within the program, I realized quickly that the naïve bayes classifier was not the most reliable giving me a wide range of accuracy results using the exact same dataset to train and test the program. So, I decided to use Scikit-Learn pip install to access a wider range of classifiers to use in my program to improve the reliability of the program. However, initial I only intended to use the naïve bayes classifier for machine learning of the program. I believe that using more than one classifier has resulted in my program becoming more accurate and reliable.

Another comment about the sentiment analysis program, when I began writing my code I was using the movie\_review corpus to train and test my classifiers, which I only used at the start of my project to make sure that my code was running and working probably, the problem with the movie\_reviews was the data was not like Twitter data has the movie\_reviews were too long in text. So, I began testing on a dataset like the data used on Twitter as in Twitter you are only allowed 140 characters, however, when completing this project, the character limit on Twitter was doubled to 280 characters.

Final comments about the program itself, the accuracy of the sentiment analysis program was never going to be 100%, so the challenge was to get as close to that figure as possible. To improve the accuracy of the classifiers in the future, I would train and test the classifiers with a bigger dataset with a bigger range of writing styles, emoji’s, sarcasm, slang, spelling mistakes and language. This would improve the performance and accuracy of the program. Sentiment analysis is still a relatively new program with massive potential for future development and improvement. However, the majority drawback is still the inability to recognize sarcasm, context and tone in the human language. This is because most sentiment analysis programmes are currently designed to recognise certain words as “positive” or “negative” sentiment. For example, words such as: “brilliant”, “amazing”, “great” are programmed as “positive”. However, words such as: “angry”, “sad”, “evil” are programmed as negative. Inaccuracy example, if someone writes on Twitter: “I am happy, I am skint for the week.” The programme will pick-up the word “happy” and will categorize the sentiment as “positive”, when it is “negative” and the twitter user is being sarcastic. This is the exact same problem I am experiencing with my sentiment analysis program and one of the many reasons why the accuracy is not 100%.

**Glossary**

**Tweepy** - is the python client for the official Twitter API to gather tweets through Twitter API, to do this we need to register an App through a twitter account.

**Natural Language Toolkit (NLTK)** - is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

**Tokenization** - Tokenizes the tweet, separates each word from the body of text as tokens.

**Supervised Classification** - works by separating the data into 2 categorizes training data and test data from the total dataset. Working example, the dataset total could be 2000 and this is split into 2 categorizes training data which has 1500 datasets and test data which has 500 datasets.

1500 (training data) + 500 (test data) = 2000 (total dataset)

**Random** - This is used to shuffle the dataset the naïve bayes classifier is being trained and tested with, as the dataset is separated into positive and negative is command mixes them up.

**Pickle** - Pickle is used as not all classifiers run in a few minutes some can take hours, so pickle allows us to run the classifier once and save the file as a serialized file that can be loaded into code and won’t take long to run your code, as the process and dataset of the classifier has already been carried out within the loaded pickle file.

**Write in bytes “wb”** - “wb” this is how it will appear in the code, the code is writing the data in binary

**Read in bytes “rb”** - “rb” this is how it will appear in the code; the code is being read in binary because it was in binary.

**Sentiment Analysis** - is the measure of positive and negative language or text. It is a method of gauging written or spoken language to determine if the statement is favourable, unfavourable or neutral and to what extent.

**JSON** - JSON files will be used to store and save data, which can be implemented into the code using strings (twitter data is saved and formatted in the JSON file format)

**Classes** - A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

Definition from: https://www.tutorialspoint.com/python/python\_classes\_objects.htm

**Matplotlib** – Matplotlib is a Python 2D plotting library that can generate tables and graphs displaying statistical data effectively and clearly.

**Matplotlib Animation** - Is used to graph live Twitter data sentiment directly onto a line graph within this program and code used.

**NumPy**: used for array processing for numbers, strings, records and objects

**SciPy:** Scientific library for python

**Function (def)** - The function is defined with 3 components: 1- header ‘def’ followed by name of the function or parameters. 2- optional comment which explains the function. 3- Body of the function which detail the procedures the function will carry out.

**This is the end of the User Guide and Start-Up Guide - How to Build a Sentiment Analysis Program. I hope you enjoyed it and that all your code is working correctly. Thank You.**

**References**

<https://www.tutorialspoint.com/python/python_classes_objects.htm>

<https://pythonprogramming.net/graph-live-twitter-sentiment-nltk-tutorial/>

<https://pythonprogramming.net/twitter-sentiment-analysis-nltk-tutorial/>

<https://pythonprogramming.net/sentiment-analysis-module-nltk-tutorial/>

<https://pythonprogramming.net/sentiment-analysis-module-nltk-tutorial/>

<https://www.python.org/downloads/>

<https://www.jetbrains.com/pycharm/download/#section=windows>

<https://apps.twitter.com>

<https://pythonprogramming.net/static/downloads/short_reviews/>

<https://pythonprogramming.net/naive-bayes-classifier-nltk-tutorial/>