Sentiment Analysis of Tweets

Based on "2.5 Machine Learning Text Classification", adapted and improved by Raúl Martínez.

In this problem we wil be training several models to classify tweets based on sentiment.

They will be three categories, Positive, Negative and Neutral.

We will be using a open dataset loaded below.

1.- Loading the dataset

We open the dataset from a CSV file and replace null values with NA string.

We print our data.

```
import pandas as pd

tweets = pd.read_csv('https://raw.githubusercontent.com/marrrcin/ml-twitter-sentiment-analysis/develop/data/train.csv', na_values=['NA']);

tweets
```

	Id	Category	Tweet
0	635769805279248384	negative	Not Available
1	635930169241374720	neutral	IOS 9 App Transport Security. Mm need to check...
2	635950258682523648	neutral	Mar if you have an iOS device, you should down...
3	636030803433009153	negative	@jimmie_vanagon my phone does not run on lates...
4	636100906224848896	positive	Not sure how to start your publication on iOS?...
...	...	...	...
5965	639016598477651968	neutral	@YouAreMyArsenal Wouldn't surprise me if we en...
5966	640276909633486849	neutral	Rib injury for Zlatan against Russia is a big ...
5967	640296841725235200	neutral	Nooooooo! I was hoping to see Zlatan being Zlat...
5968	641017384908779520	neutral	Not Available

2.- Cleaning the data

Remove all the rows that contain a "Not Available" tweet because they are not real tweets, they are issues on our data. [1]

This raises our score by 4 percentage points.

We need to reset the index numbers to avoid errors below. [2]

We print again.

```
tweets = tweets.drop(tweets[tweets.Tweet == "Not Available"].index)

tweets = tweets.reset_index(drop=True)

tweets
```

	Id	Category	Tweet
0	635930169241374720	neutral	IOS 9 App Transport Security. Mm need to check...
1	635950258682523648	neutral	Mar if you have an iOS device, you should down...
2	636030803433009153	negative	@jimmie_vanagon my phone does not run on lates...
3	636100906224848896	positive	Not sure how to start your publication on iOS?...
4	636176272947744772	neutral	Two Dollar Tuesday is here with Forklift 2, Qu...
...	...	...	...
5417	638445576212754433	positive	Ok ed let's do this, Zlatan, greizmann and Lap...
5418	638531837313306624	neutral	Goal level: Zlatan 90k by Friday? = Posting e...
5419	639016598477651968	neutral	@YouAreMyArsenal Wouldn't surprise me if we en...
5420	640276909633486849	neutral	Rib injury for Zlatan against Russia is a big ...

Delete the Id collumn from our dataset as this does not contain useful information. [3]

We print again to see our result.

```
del tweets['Id']

tweets
```

	Category	Tweet
0	neutral	IOS 9 App Transport Security. Mm need to check...
1	neutral	Mar if you have an iOS device, you should down...
2	negative	@jimmie_vanagon my phone does not run on lates...
3	positive	Not sure how to start your publication on iOS?...
4	neutral	Two Dollar Tuesday is here with Forklift 2, Qu...
...	...	...
5417	positive	Ok ed let's do this, Zlatan, greizmann and Lap...
5418	neutral	Goal level: Zlatan 90k by Friday? = Posting e...
5419	neutral	@YouAreMyArsenal Wouldn't surprise me if we en...
5420	neutral	Rib injury for Zlatan against Russia is a big ...
5421	neutral	Noooooo! I was hoping to see Zlatan being Zlat...

5422 rows × 2 columns

Remove urls from Tweets as they do not have information about the sentiment of the tweet. [4]

We print our result.

```
# Example of tweet before removing urls.
print(tweets.loc[3]['Tweet'])
```

```
Not sure how to start your publication on iOS? We'll be live helping with ask me anything sessions today and Friday http://t.co/KPqqGjjh3x
```

```
tweets['Tweet'] = tweets['Tweet'].str.replace('http\S+|www.\S+', '', case=False)
```

```
# Example of tweet after removing urls.
print(tweets.loc[3]['Tweet'])
```

```
Not sure how to start your publication on iOS? We'll be live helping with ask me anything sessions today and Friday
```

Remove twitter handles using Regex, as they do not affect sentiment of a tweet. [5] [6]

```
# Example of tweet before removing twitter handles.
print(tweets.loc[2]['Tweet'])
```

```
@jimmie_vanagon my phone does not run on latest IOS which may account for problem the other day .. time it was replaced
```

```
tweets['Tweet'] = tweets['Tweet'].str.replace('\B@\w+', '', case=False)
```

```
# Example of tweet after removing twitter handles.
print(tweets.loc[2]['Tweet'])
```

```
 my phone does not run on latest IOS which may account for problem the other day .. time it was replaced
```

Replace double spaces with single spaces and saving the clean dataset in a new variable.

```
tweets['Tweet'] = tweets['Tweet'].str.replace('  ', ' ', case=False)

tweets_clean = tweets
```

We can print only the Tweet collumn by using this syntax.

```
tweets_clean['Tweet']
```

```
0       IOS 9 App Transport Security. Mm need to check...
1       Mar if you have an iOS device, you should down...
2        my phone does not run on latest IOS which may...
3       Not sure how to start your publication on iOS?...
4       Two Dollar Tuesday is here with Forklift 2, Qu...
                              ...
5417    Ok ed let's do this, Zlatan, greizmann and Lap...
5418    Goal level: Zlatan 90k by Friday? = Posting ev...
5419     Wouldn't surprise me if we enquired.He can't ...
5420    Rib injury for Zlatan against Russia is a big ...
5421    Noooooo! I was hoping to see Zlatan being Zlat...
Name: Tweet, Length: 5422, dtype: object
```

3.- First approach to classification

We split our collumns into X and Y and perform a train test split with 20% of the records going to testing and 80% to training.

```
from sklearn.model_selection import train_test_split
X = tweets['Tweet']
Y = tweets['Category']
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=1005)
```

We reindex our data to get index numbers that start from 0 and go secuentially.

```
X_train = X_train.reset_index(drop=True)
X_test = X_test.reset_index(drop=True)

Y_train = Y_train.reset_index(drop=True)
Y_test = Y_test.reset_index(drop=True)
```

We extract features from text files and print its shape.

4337 samples with 9973 diffent words in them.

```
from sklearn.feature_extraction.text import CountVectorizer

count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
X_train_counts.shape
```

```
(4066, 9581)
```

For example, the first tweet contains the following words this number of times:

```
print(X_train_counts[0,:])
```

Now we start our Term frequency - Inverse document frequency algorithm that works as explained on lectures, by assigning variable weights on terms depending on its rarity.

```
from sklearn.feature_extraction.text import TfidfTransformer

tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_train_tfidf.shape
```

```
(4066, 9581)
```

This TF-IDF algorithm generates this weights for each of the words.

```
print(X_train_tfidf[0,:])
```

```
  (0, 9471)	0.23083673217385792
  (0, 9385)	0.10001067621412355
  (0, 8831)	0.16629181626209993
  (0, 8517)	0.05374210727430672
  (0, 8104)	0.1789048679921987
  (0, 8090)	0.3071022480426049
  (0, 7560)	0.2679510132050171
  (0, 7257)	0.3071022480426049
  (0, 7230)	0.24974672570212947
  (0, 6862)	0.1938459763607244
  (0, 3855)	0.16951746301835324
  (0, 2742)	0.28240056909450734
  (0, 2586)	0.2926526921531146
  (0, 1816)	0.2926526921531146
  (0, 1783)	0.3071022480426049
  (0, 1006)	0.213054189116479
  (0, 622)	0.3071022480426049
```

4.- The classifier

Machine Learning

Training the classifier on training data.

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
```

```
clf = MultinomialNB().fit(X_train_tfidf, Y_train)
```

Building a pipeline:

We can write less code and do all of the above, by building a pipeline as follows:

The names ‘vect’ , ‘tfidf’ and ‘clf’ are arbitrary but will be used later.

We will be using the 'text_clf' going forward.

```
from sklearn.pipeline import Pipeline

text_clf = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTransformer()), ('clf', MultinomialNB())])
text_clf = text_clf.fit(X_train, Y_train)
```

Performance of the classifier:

```
import numpy as np

predicted = text_clf.predict(X_test)
np.mean(predicted == Y_test)
```

```
0.5663716814159292
```

We got a 56,6% score, that is an improvement from the random baseline that would be 33% (as they are 3 classes to choose between).

We can do better than this.

```
print("Real:")
print(Y_test[801])
print(Y_test[802])
print(Y_test[803])

print("\nPredictions:")
print(predicted[801])
print(predicted[802])
print(predicted[803])
```

```
Real:
positive
neutral
negative

Predictions:
positive
neutral
positive
```

We can confirm that our output series is the same lenght as our predicted series. A non equal shape will be an indicator of bad code and would lead to incorrect answers.

```
print("Y_test shape:")
print(Y_test.shape)

print("\nPredicted Shape:")
print(predicted.shape)
```

```
Y_test shape:
(1356,)

Predicted Shape:
(1356,)
```

5.- Using SVM

Training Support Vector Machines - SVM and calculating its performance.

SGD is a optimization method used in machine learning models that defines a loss function, and the optimization method maximizes or minimizes it.

```
from sklearn.linear_model import SGDClassifier
text_clf_svm = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                         ('clf-svm', SGDClassifier(loss='hinge', penalty='l2',alpha=1e-3, random_state=1004))])

text_clf_svm = text_clf_svm.fit(X_train, Y_train)
predicted_svm = text_clf_svm.predict(X_test)
np.mean(predicted_svm == Y_test)
```

```
0.5759587020648967
```

We get a 58% by using SGD Classifier

We can do grid search for SVM to explore the best parameters

```
from sklearn.model_selection import GridSearchCV
parameters_svm = {'vect__ngram_range': [(1, 1), (1, 2)], 'tfidf__use_idf': (True, False),'clf-svm__alpha': (1e-2, 1e-3)}

gs_clf_svm = GridSearchCV(text_clf_svm, parameters_svm, n_jobs=-1)
gs_clf_svm = gs_clf_svm.fit(X_train, Y_train)


print(gs_clf_svm.best_score_)
print(gs_clf_svm.best_params_)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_split.py:1978: FutureWarning: The default value of cv will change from 3 to 5 in versior
  warnings.warn(CV_WARNING, FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_split.py:657: Warning: The least populated class in y has only 1 members, which is too f
  % (min_groups, self.n_splits)), Warning)
0.5459911460895229
{'clf-svm__alpha': 0.001, 'tfidf__use_idf': True, 'vect__ngram_range': (1, 1)}
```

```
gs_clf_svm.best_score_
```

```
0.5459911460895229
```

We get as ouput the best parameters.

Now we will use the Natural Language Toolkit to remove stop words, these are words without real meaning.

We will choose the stop word list from English language.

```
from sklearn.pipeline import Pipeline
text_clf = Pipeline([('vect', CountVectorizer(stop_words='english')), ('tfidf', TfidfTransformer()),
                     ('clf', MultinomialNB())])
```

Stemming is the process of producing variants of a base word. [7]

```
import nltk
nltk.download('stopwords')
```

```
from nltk.stem.snowball import SnowballStemmer
stemmer = SnowballStemmer("english", ignore_stopwords=True)

class StemmedCountVectorizer(CountVectorizer):
    def build_analyzer(self):
        analyzer = super(StemmedCountVectorizer, self).build_analyzer()
        return lambda doc: ([stemmer.stem(w) for w in analyzer(doc)])

stemmed_count_vect = StemmedCountVectorizer(stop_words='english')

text_mnb_stemmed = Pipeline([('vect', stemmed_count_vect), ('tfidf', TfidfTransformer()),
                             ('mnb', MultinomialNB(fit_prior=False))])

text_mnb_stemmed = text_mnb_stemmed.fit(X_train, Y_train)

predicted_mnb_stemmed = text_mnb_stemmed.predict(X_test)

np.mean(predicted_mnb_stemmed == Y_test)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
0.5870206489675516
```

This gets us a 59% of accuracy agains our testing dataset.

6.- Fixing Imbalanced Dataset

In the full dataset we have detected an issue, the distribution of the classess is the following:

- Positive: 2889/5970 = 48,4%
- Neutral: 2127/5970 = 35,6%
- Negative: 959/5970 = 16,1%

Our dataset has more samples of positive and neutral tweets than negative ones, we must take this into account. [13]

```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=42)
X_resampled, Y_resampled = ros.fit_resample(X_train.values.reshape(-1, 1), Y_train)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/externals/six.py:31: DeprecationWarning: The module is deprecated in version 0.21 and will be removed in
  "(https://pypi.org/project/six/).", DeprecationWarning)
```

We import our oversampler and we initiate it giving providing the X and Y input variables. [8]

```
X_train_OS = pd.Series(X_resampled[:, 0])
Y_train_OS = pd.Series(Y_resampled)
```

We have converted the output into pandas series. [9]

Here we can see the before:

```
print(X_train);
print(Y_train);
```

```
0       Tuesday Raw Roundtable: Sting, Dudleys, Seth R...
1        make Blossom your comeback single in February...
2       Reminder: Madonna's show originally scheduled ...
3       Just happened its 8:33 am and I just stopped p...
4       Is the Pope a Catholic? Do they drive PU truck...
                              ...
4061    Tom Cruise movie marathon wasn't in my plan li...
4062    Messi and Ronaldo both go up a rating in FUT 1...
4063               Nike trying to change the online game
4064    Take the 1st step to purchase your dream Lexus...
4065     i just went to metlife! i may go to a philly ...
Name: Tweet, Length: 4066, dtype: object
0       positive
1       positive
2        neutral
3        neutral
4        neutral
          ...
4061     neutral
4062    positive
4063     neutral
4064    positive
4065    positive
Name: Category, Length: 4066, dtype: object
```

And its imbalanced shape:

```
print(tweets[tweets.Category == "positive"].shape)

print(tweets[tweets.Category == "neutral"].shape)
```

```
print(tweets[tweets.Category == "negative"].shape)
```

```
(2599, 2)
(1953, 2)
(869, 2)
```

Now, after the oversampling we observe the same number of rows in each category, as expected.

```
print(Y_train_OS[Y_train_OS == "positive"].shape)

print(Y_train_OS[Y_train_OS == "neutral"].shape)

print(Y_train_OS[Y_train_OS == "negative"].shape)
```

```
(1938,)
(1938,)
(1938,)
```

We can print one category to show if they are all of the expected category:

```
Y_train_OS[Y_train_OS == "neutral"]
```

```
2       neutral
3       neutral
4       neutral
8       neutral
14      neutral
          ...
7747    neutral
7748    neutral
7749    neutral
7750    neutral
7751    neutral
Length: 1938, dtype: object
```

Also we can check the actual oversampled neutral tweets:

```
X_train_OS[Y_train_OS == "neutral"]
```

```
2       Reminder: Madonna's show originally scheduled ...
3       Just happened its 8:33 am and I just stopped p...
4       Is the Pope a Catholic? Do they drive PU truck...
8       Scum???.... He votes Tory. Reads the sun, and ...
14      Trump reiterates Hillary Clinton is worst sec ...
                              ...
7747    Microsoft's Windows 10 testing resumes with a ...
7748    LIVE Stream: Monsanto scientist Dr. Frederick ...
7749    Mansbridge interview with Trudeau may not have...
7750    1st he smashed someone's phone now ran over an...
7751    Weird, but signs of UKIP votes going back to t...
Length: 1938, dtype: object
```

Now we initiate the same CountVectorizer, TFIDF and process explained in the chapters before.

```
# Extracting features from text files
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.tree import DecisionTreeClassifier
test_count_vect = CountVectorizer()
test_X_train_counts = test_count_vect.fit_transform(X_train_OS)
```

```
# TF-IDF
from sklearn.feature_extraction.text import TfidfTransformer
test_tfidf_transformer = TfidfTransformer()
test_X_train_tfidf = test_tfidf_transformer.fit_transform(test_X_train_counts)


# Machine Learning
# Training Naive Bayes (NB) classifier on training data.
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
test_clf = MultinomialNB().fit(test_X_train_tfidf, Y_train_OS)


# Building a pipeline: We can write less code and do all of the above, by building a pipeline as follows:
# The names ‘vect’ , ‘tfidf’ and ‘clf’ are arbitrary but will be used later.
# We will be using the 'text_clf' going forward.
from sklearn.pipeline import Pipeline
#test_text_clf = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTransformer()), ('clf', MultinomialNB())])
#test_text_clf = test_text_clf.fit(X_train_OS, Y_train_OS)

text_clf_svm = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                         ('clf-svm', SGDClassifier(loss='hinge', penalty='l2',alpha=1e-3, random_state=1004))])
text_clf_svm = text_clf_svm.fit(X_train_OS, Y_train_OS)
```

```
# Performance of NB Classifier
import numpy as np


test_predicted = text_clf_svm.predict(X_test)
print(np.mean(test_predicted == Y_test))
```

```
0.7160766961651918
```

Wow, this is our highest score yet by using SVM and by fixing the imbalanced dataset problem.

We can compare this accuracy agains a random predicted sample by shuffling the testing set.

```
test_predicted = text_clf_svm.predict(X_test)
print(np.mean(test_predicted == Y_test.sample(frac=1)))
```

```
0.3635693215339233
```

We can observe that a fully trained model gets 70,4% of accuracy choosing between 3 classes and below the expected output from a random model that just tries without clue, that is called the baseline.

Baseline: 37,1%

Our model: 70,4%

7.- Bagging

Using techniques like Boosting or Baggin can help to increase the robustness and decrease the variance of the model. By combining multiple models we can decrease variance thus producing a more reliable classification than a single model could provide. [10] [11]

Bagging consists on aggregating a group of models into a common output, that could consist on averaging the answers of each model to produce consensus. This is a simple but very powerful ensemble method.

Boosting consists in grouping models outputs by utilizing weighted averages to make weak learners into stronger learners.

These techniques decrease the variance of your single estimate as they combine several estimates from different models.

In this practice we will try bagging by aggregating three models built with subsets of the train dataset.

The method to build these subsets will be sampling with replacement, as this is the recommended method by the available literature. [12]

```
print(tweets_clean)
```

```
print(tweets_clean)
```

```
       Category                                              Tweet
0       neutral  IOS 9 App Transport Security. Mm need to check...
1       neutral  Mar if you have an iOS device, you should down...
2      negative   my phone does not run on latest IOS which may...
3      positive  Not sure how to start your publication on iOS?...
4       neutral  Two Dollar Tuesday is here with Forklift 2, Qu...
...         ...                                                ...
5417   positive  Ok ed let's do this, Zlatan, greizmann and Lap...
5418    neutral  Goal level: Zlatan 90k by Friday? = Posting ev...
5419    neutral   Wouldn't surprise me if we enquired.He can't ...
5420    neutral  Rib injury for Zlatan against Russia is a big ...
5421    neutral  Noooooo! I was hoping to see Zlatan being Zlat...

[5422 rows x 2 columns]
```

```
tweets_clean_train = tweets_clean[0:4337]
tweets_clean_test = tweets_clean[4337:5422]
```

```
print(tweets_clean_train)

print(tweets_clean_test)
```

```
     Category                                              Tweet
0     neutral  IOS 9 App Transport Security. Mm need to check...
1     neutral  Mar if you have an iOS device, you should down...
2    negative   my phone does not run on latest IOS which may...
3    positive  Not sure how to start your publication on iOS?...
4     neutral  Two Dollar Tuesday is here with Forklift 2, Qu...
...       ...                                                ...
4332  neutral  Is Tiger Woods form finally returning or is it...
4333  neutral   Hi jamie I heard Tiger Woods was at -13 Under...
4334  neutral  Belgian Grand Prix, 100m Final, Super Sunday, ...
4335 positive  Tiger Woods 2 strokes behind leader Jason Gore...
4336 positive  "Tiger Woods doesn't move the needle. He is th...

[4337 rows x 2 columns]
     Category                                              Tweet
4337  neutral  Vintage 2008 Tiger Woods on master sunday on t...
4338 positive  Bolt wins an epic. Imagine if Tiger Woods wins...
4339  neutral  2015 Wyndham Championship: Tee times, pairings...
4340 positive  A Sunday with Tiger Woods in contention for a ...
4341 positive  Hello Tiger family. Do the #Tigertwirlchalleng...
...       ...                                                ...
5417 positive  Ok ed let's do this, Zlatan, greizmann and Lap...
5418  neutral  Goal level: Zlatan 90k by Friday? = Posting ev...
5419  neutral   Wouldn't surprise me if we enquired.He can't ...
5420  neutral  Rib injury for Zlatan against Russia is a big ...
5421  neutral  Noooooo! I was hoping to see Zlatan being Zlat...

[1085 rows x 2 columns]
```

```
# We are extracting our data with replacement
# We shuffle our dataset in each extraction by using the sample method of pandas
# We also reset the index

# Notice that in each bag we are selecting 40% of the rows, this is not a problem
# as we are using replacement so 3 x 40% = 120% is not a problem.

bag1_tweets = tweets_clean_train.sample(frac=0.40).reset_index(drop=True)
print(bag1_tweets)
#print(bag1_tweets_train[bag1_tweets_train.Category == "positive"].shape)
#print(bag1_tweets_train[bag1_tweets_train.Category == "negative"].shape)

bag2_tweets = tweets_clean_train.sample(frac=0.40).reset_index(drop=True)
#print(bag2_tweets)
#print(bag2_tweets_train[bag2_tweets_train.Category == "positive"].shape)
#print(bag2_tweets_train[bag2_tweets_train.Category == "negative"].shape)
```

```
bag3_tweets = tweets_clean_train.sample(frac=0.40).reset_index(drop=True)
#print(bag3_tweets)
#print(bag3_tweets_train[bag3_tweets_train.Category == "positive"].shape)
#print(bag3_tweets_train[bag3_tweets_train.Category == "negative"].shape)

# We got 1735 rows in each bag
```

```
      Category                                              Tweet
0     positive  TOP REASONS JOE BIDEN SHOULD RUN FOR PRESIDENT...
1     positive  Omw to work bumping this 90's Mariah Carey. Be...
2     positive  Looks like I have a 1/3 shot at getting ticket...
3      neutral   SCOTUS has interpreted equal rights under the...
4     positive   congratulations on pumping Taylor Swift mate ...
...        ...                                                ...
1730  positive  Chelsea may have 33 out on loan; Wait till you...
1731  positive   You can resort to the old Madonna hit "Holida...
1732  positive  The sun, earth, the Lakers, Jay-Z, Oprah, and ...
1733  positive  Jan runs into Michelle Obama on her way to a c...
1734  positive   Taylor Swift is going to be in Houston TX on ...

[1735 rows x 2 columns]
```

```
# Extracting features from text files
from sklearn.feature_extraction.text import CountVectorizer
bag1_count_vect = CountVectorizer()
bag1_X_train_counts = bag1_count_vect.fit_transform(bag1_tweets['Tweet'])
bag2_count_vect = CountVectorizer()
bag2_X_train_counts = bag2_count_vect.fit_transform(bag2_tweets['Tweet'])
bag3_count_vect = CountVectorizer()
bag3_X_train_counts = bag3_count_vect.fit_transform(bag3_tweets['Tweet'])

# TF-IDF
from sklearn.feature_extraction.text import TfidfTransformer
bag1_tfidf_transformer = TfidfTransformer()
bag1_X_train_tfidf = bag1_tfidf_transformer.fit_transform(bag1_X_train_counts)
bag2_tfidf_transformer = TfidfTransformer()
bag2_X_train_tfidf = bag2_tfidf_transformer.fit_transform(bag2_X_train_counts)
bag3_tfidf_transformer = TfidfTransformer()
bag3_X_train_tfidf = bag3_tfidf_transformer.fit_transform(bag3_X_train_counts)

# Machine Learning
# Training Naive Bayes (NB) classifier on training data.
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import SGDClassifier
```

```
# Building a pipeline: We can write less code and do all of the above, by building a pipeline as follows:
# The names ‘vect’ , ‘tfidf’ and ‘clf’ are arbitrary but will be used later.
# We will be using the 'text_clf' going forward.
from sklearn.pipeline import Pipeline
bag1_text_clf = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTransformer()), ('clf', RandomForestClassifier())])
bag1_text_clf = bag1_text_clf.fit(bag1_tweets['Tweet'], bag1_tweets['Category'])
bag2_text_clf = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTransformer()), ('clf', MultinomialNB())])
bag2_text_clf = bag2_text_clf.fit(bag2_tweets['Tweet'], bag2_tweets['Category'])
#bag3_text_clf = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTransformer()), ('clf', DecisionTreeClassifier())])
#bag3_text_clf = bag3_text_clf.fit(bag3_tweets['Tweet'], bag3_tweets['Category'])

bag3_text_clf = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                         ('clf-svm', SGDClassifier(loss='hinge', penalty='l2',alpha=1e-3, random_state=1004))])
bag3_text_clf = bag3_text_clf.fit(bag3_tweets['Tweet'], bag3_tweets['Category'])

# Performance of NB Classifier
import numpy as np

bag1_predicted = bag1_text_clf.predict(tweets_clean_test['Tweet'])
print(np.mean(bag1_predicted == tweets_clean_test['Category']))

bag2_predicted = bag2_text_clf.predict(tweets_clean_test['Tweet'])
print(np.mean(bag2_predicted == tweets_clean_test['Category']))

bag3_predicted = bag3_text_clf.predict(tweets_clean_test['Tweet'])
print(np.mean(bag3_predicted == tweets_clean_test['Category']))
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of n_estimators will change from 10 in version 0
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
0.4368663594470046
0.432258064516129
0.48110599078341015
```

This custom function was created in an attempt to aggregate the output from our 3 bags, it does so by querying each model then adding the outputs and finally making a decision based on a numeric range that can be adjusted.

Finally it outputs an array just as the normal prediction functions remaining compatible with non-bagging code.

```
def bagged_predict(input_tweet):
    prediction_1 = bag1_text_clf.predict(input_tweet)
    prediction_2 = bag2_text_clf.predict(input_tweet)
    prediction_3 = bag3_text_clf.predict(input_tweet)
    output = []
```

```
    for i in range(len(prediction_1)):
      #print(i, prediction_1[i])

      recuento = 0  # Possible values: -3, -2, -1, 0, 1, 2, 3
      result = "neutral"

      if prediction_1[i] == "positive":
        recuento = recuento + 1
      if prediction_1[i] == "negative":
        recuento = recuento - 1
      if prediction_2[i] == "positive":
        recuento = recuento + 1
      if prediction_2[i] == "negative":
        recuento = recuento - 1
      if prediction_3[i] == "positive":
        recuento = recuento + 1
      if prediction_3[i] == "negative":
        recuento = recuento - 1
      if recuento <= -2:
        result = "negative"
      if recuento >= 3:
        result = "positive"
      output.append(result)

    #print(output)
    return(output)

#print( bagged_predict(["dont","a"]) )
```

```
bagged_predicted = bagged_predict(tweets_clean_test['Tweet'])
print(np.mean(bagged_predicted == tweets_clean_test['Category']))
```

```
0.46728110599078343
```

We obtain a result above the baseline but its not the one that we expected.

8.- Bibliography

[1] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.drop.html

[2] https://www.geeksforgeeks.org/python-pandas-dataframe-reset_index/

[3] https://stackoverflow.com/questions/13411544/delete-column-from-pandas-dataframe

[4] https://stackoverflow.com/questions/45395676/remove-a-url-row-by-row-from-a-large-set-of-text-in-python-panda-dataframe

[5] https://stackoverflow.com/questions/20282452/regex-to-match-word-beginning-with

[6] https://stackoverflow.com/questions/13851535/delete-rows-from-a-pandas-dataframe-based-on-a-conditional-expression-involving

[7] https://www.geeksforgeeks.org/python-stemming-words-with-nltk/

[8] https://imbalanced-learn.readthedocs.io/en/stable/over_sampling.html

[9] https://stackoverflow.com/questions/33246771/convert-pandas-data-frame-to-series

[10] https://becominghuman.ai/ensemble-learning-bagging-and-boosting-d20f38be9b1e

[11] https://machinelearningmastery.com/how-to-create-a-random-split-cross-validation-and-bagging-ensemble-for-deep-learning-in-keras/

[12] https://www.youtube.com/watch?v=2Mg8QD0F1dQ

[13] https://towardsdatascience.com/machine-learning-multiclass-classification-with-imbalanced-data-set-29f6a177c1a

[ ] https://www.geeksforgeeks.org/ml-bagging-classifier/

[ ] https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html

9.- Conclusion

Features included in the system:

- Invalid tweet removing (Dropping Not Available tweets)
- Tweet bloat removing (Urls, twitter handles, double spaces and IDs)
- Initial model using CV, TF-IDF and Multinomial Naive Bayes implementation
- Improved model using SVM - SDG
- Stopwords remover
- Fixing imbalanced dataset using RandomOverSampler
- Bagging approach using custom made bagging function

In this practice we have learned that by using TF-IDF we can approach text classification machine learning problems just as any number based classification problem.

Also we've learned that its important to clean our dataset to obtain significant accuracy gains, that simple models like MultinomialNB work relatively well but they can be improved by adding some advancements as SVM, SDG, removing stopwords and doing some tweaks as Oversampling classess with less rows to achieve even better results agains our baseline.

We have been introduced to bagging as a method to aggregate outputs from diffent models trained using subsets of our train data.

My skills related to machine learning have clearly increased as I knew only some theory when we started this course and now im able to perform some approximations to real problems such as this one.

I've assigned about 6 to 10 hours to this practice, part of this time has been researching on the internet to understand how everything worked and then I've tried my best to implement them and also contribute with my own skills. This calculation does not include the lectures that have been important to understand our task.

In conclusion, we have obtained a maximum accuracy of 70% by using SVM model with balanced datasets. That is 34 percentage points above our baseline of 36% so I consider it a success.

The problems with this tasks in my opinion were related to the dataset, it contained few tweets for our models to understand the difference between classes, as text classification is much harder problem because the input comes from humans writing.