Ryan Amoudi (1742970) - Sulaiman Thagafi (1741804)
Abdulrahman Damanhori (1742171) - Faisal Khairallah (1744096)

TWITTER SENTIMENT ANALYSIS

Artificial Intelligence I Project (CPCS-331)

Table of Content

Team members. (Section GE) 3

What’s the project about. 3

Project Description. 4

Project Steps. 4

Data Processing. 5

Classification. 8

Output Comparison of Algorithms. 10

Team members. (Section GE)

Student Name	Student ID	Contribution
Ryan Amoudi (Leader)	1742970	25%
Sulaiman Thagafi	1741804	25%
Abdulrahman Damanhori	1742171	25%
Faisal Khairallah	1744096	25%

What's the project about.

This project is Twitter Sentiment Analysis for Arabic Tweets. Tweets are divided into Positive or Negative. We collect around 2000 labelled tweets (1000 positive tweets and 1000 negative ones) on various topics such as: politics and arts. These tweets include opinions written in both Modern Standard Arabic and the Jordanian dialect.

The selected tweets convey some kind of feelings (positive or negative) and the objective of our model is to extract valuable information from such tweets in order to determine the sentiment orientation of the inputted text. The months-long annotation process of the tweets is manually conducted mainly by two human experts (native speakers of Arabic). If both experts agree on the label of a certain tweet, then the tweet is assigned this label. Otherwise, a third expert is consulted to break the tie.

Project Description.

Instances & Attributes	
Number of Instances	2000
Number of Attributes	2

Attribute Information	
Tweet	String Vector
Class	Positive Polarity and Negative Polarity

Dataset used in the code is tweets.csv, original version of the dataset consists of txt form of all tweets divided in two folders. Both in the package, consists of two attributes: the class (Negative or Positive) and the tweet itself in string. No missing values in the dataset as it's text based. The dataset is categorical.

This project is made in Python using website kaggle.com with libraries pandas, numpy, sklearn, os, re, string, collections, nltk, and matplotlib.

In the files of the project there's the code file in py and ipynb. The original dataset and the transformed csv version of the dataset (tweets.csv) is also in the files.

Project Steps.

This project is divided into three parts:

1. Data-Preprocessing.
2. Classification.
3. Comparison of Algorithms Decision Tree and Naïve Bayes.

The code written in the project will be thoroughly explained and will provide the information needed to understand it.

Data Processing.

This process is done in 5 steps in order to make the data ready for classification.

1. Upload the data into Pandas.
2. Cleaning the data.
3. Tokenization.
4. Stemming. (if needed)
5. Feature Extraction.

Starting with the first step, uploading the data into pandas is simple.

In [91]:

```
import numpy as np
import pandas as pd

tweets = pd.read_csv("../input/tweets/tweets.csv", header = None)
tweets.columns = ['Sentiment', 'Tweet']

import collections
collections.Counter(tweets['Sentiment'])

tweets.head()
```

Out[91]:

	Sentiment	Tweet
0	Negative	أنا متأكده أنو هذولي مدفوعلهم علشان يجون يغنو...
1	Negative	يا ثقل دمه الاصلع\n
2	Negative	سوبر طقاع ..... اول واحد جايته زبيريه \n
3	Negative	كله على راس المواطن ما يثير السخرية رفع اسعار ...
4	Negative	الا يكفي ان نسبة مرضى السرطان في الاردن اعلى ...

Then Step 2 is to clear the data from anything other than the text like punctuations, mentions, and URLs.

In [92]:
```
import re
import string

def removeMentionsUrls(text):
    tweetOut = re.sub(r'@[A-Za-z0-9]+','',text)
    re.sub('https?://[A-Za-z0-9./]+','',tweetOut)
    return tweetOut

def removeNonAlphaNumeric(text):
    textOut = "".join([char for char in text if char not in string.punctuation])
    return textOut

tweets["No Mention No URL"] = tweets['Tweet'].apply(lambda x: removeMentionsUrls(x))

tweets["No Punctuation"] = tweets["No Mention No URL"].apply(lambda x: removeNonAlphaNumeric(x)
)

tweets.head()
```

Out[92]:

	Sentiment	Tweet	No Mention No URL	No Punctuation
0	Negative	أنا متأكده أنو هذولي مدفوعلهم علشان يجون يغنو...	أنا متأكده أنو هذولي مدفوعلهم علشان يجون يغنو...	أنا متأكده أنو هذولي مدفوعلهم علشان يجون يغنو...
1	Negative	\nيا ثقل دمه الاصلع	\nيا ثقل دمه الاصلع	\nيا ثقل دمه الاصلع
2	Negative	\n سوبر طقاع ..... اول واحد جايته زبيريه	\n سوبر طقاع ..... اول واحد جايته زبيريه	\n سوبر طقاع اول واحد جايته زبيريه
3	Negative	كله على راس المواطن ما يثير السخرية رفع اسعار ...	كله على راس المواطن ما يثير السخرية رفع اسعار ...	كله على راس المواطن ما يثير السخرية رفع اسعار ...
4	Negative	الا يكفي ان نسبة مرضى السرطان في الاردن اعلى ...	الا يكفي ان نسبة مرضى السرطان في الاردن اعلى ...	الا يكفي ان نسبة مرضى السرطان في الاردن اعلى ...

Step 3 is the tokenization part, where the text from the tweet is divided into a list of words.

In [96]:
```
def tokenization(text):
    tokens = re.split('\W+', text)
    return tokens

tweets["Tokenized"] = tweets["No Punctuation"].apply(lambda x: tokenization(x.lower()))

tweets.head()
```

Out[96]:

	Sentiment	Tweet	No Mention No URL	No Punctuation	Tokenized	Stemmed
0	Negative	أنا متأكده أنو هذولي مدفوعلهم علشان يجون يغنو...	أنا متأكده أنو هذولي مدفوعلهم علشان يجون يغنو...	أنا متأكده أنو هذولي مدفوعلهم علشان يجون يغنو...	[أنا, متأكده, أنو, هذولي, مدفوعلهم, علشان, ي...]	[أنا, متأكده, أنو, هذولي, مدفوعلهم, علشان, ي...]
1	Negative	\nيا ثقل دمه الاصلع	\nيا ثقل دمه الاصلع	\nيا ثقل دمه الاصلع	[, يا, ثقل, دمه, الاصلع]	[, يا, ثقل, دمه, الاصلع]
2	Negative	\n سوبر طقاع ..... اول واحد جايته زبيريه	\n سوبر طقاع ..... اول واحد جايته زبيريه	\n سوبر طقاع اول واحد جايته زبيريه	[, سوبر, طقاع, اول, واحد, جايته, زبيريه]	[, سوبر, طقاع, اول, واحد, جايته, زبيريه]
3	Negative	كله على راس المواطن ما يثير السخرية رفع اسعار ...	كله على راس المواطن ما يثير السخرية رفع اسعار ...	كله على راس المواطن ما يثير السخرية رفع اسعار ...	[كله, على, راس, المواطن, ما, يثير, السخرية, رف...]	[كله, على, راس, المواطن, ما, يثير, السخرية, رف...]
4	Negative	الا يكفي ان نسبة مرضى السرطان في الاردن اعلى ...	الا يكفي ان نسبة مرضى السرطان في الاردن اعلى ...	الا يكفي ان نسبة مرضى السرطان في الاردن اعلى ...	[الا, يكفي, ان, نسبة, مرضى, السرطان, في, الا...]	[الا, يكفي, ان, نسبة, مرضى, السرطان, في, الا...]

Step 4 is stemming the text.

In [95]:

```
import nltk
ps = nltk.PorterStemmer()

def stemming(text):
    outText = [ps.stem(word) for word in text]
    return outText

tweets["Stemmed"] = tweets["Tokenized"].apply(lambda x: stemming(x))

tweets.head()
```

Out[95]:

	Sentiment	Tweet	No Mention No URL	No Punctuation	Tokenized	Stemmed
0	Negative	أنا متأكده أنو هذولي مدفوعلهم علشان يجون يغنو...	أنا متأكده أنو هذولي مدفوعلهم علشان يجون يغنو...	أنا متأكده أنو هذولي مدفوعلهم علشان يجون يغنو...	[أنا, متأكده, أنو, هذولي, مدفوعلهم, علشان, ي...	[أنا, متأكده, أنو, هذولي, مدفوعلهم, علشان, ي...
1	Negative	يا ثقل دمه الاصلع\n	يا ثقل دمه الاصلع\n	يا ثقل دمه الاصلع\n	[يا, ثقل, دمه, الاصلع,]	[يا, ثقل, دمه, الاصلع,]
2	Negative	سوبر طقاع ..... اول واحد جايته زبيريه \n	سوبر طقاع ..... اول واحد جايته زبيريه \n	سوبر طقاع اول واحد جايته زبيريه \n	[سوبر, طقاع, اول, واحد, جايته, زبيريه,]	[سوبر, طقاع, اول, واحد, جايته, زبيريه,]
3	Negative	كله على راس المواطن ما يثير السخرية رفع اسعار ...	كله على راس المواطن ما يثير السخرية رفع اسعار ...	كله على راس المواطن ما يثير السخرية رفع اسعار ...	[كله, على, راس, المواطن, ما, يثير, السخرية, رف...	[كله, على, راس, المواطن, ما, يثير, السخرية, رف...
4	Negative	الا يكفي ان نسبة مرضى السرطان في الاردن اعلى ...	الا يكفي ان نسبة مرضى السرطان في الاردن اعلى ...	الا يكفي ان نسبة مرضى السرطان في الاردن اعلى ...	[الا, يكفي, ان, نسبة, مرضى, السرطان, في, الا...	[الا, يكفي, ان, نسبة, مرضى, السرطان, في, الا...

Step 5 is Feature Extraction. We need to pass two data sources when creating the bag of words (ListOfUniqueWords). Two sets of data 1. Unique List of words 2. Dataset Document which will contain sentences. CounterVectorizer method needs a list of strings and not tokens.

In [109]:

```
import itertools
ListOfUniqueWords = set(list(itertools.chain.from_iterable(tweets["Stemmed"])))

def join_tokens(tokens):
    document = " ".join([word for word in tokens if not word.isdigit()])

    return document

tweets["Dataset Documents"] = tweets["Stemmed"].apply(lambda x: join_tokens(x))

#print("A Sample of Unique Words of the Data\n")
#print(list(ListOfUniqueWords)[:20])
print(len(ListOfUniqueWords))
```

```
7071
```

Classification.

Importing CounterVectorizer and initializing DataFrame with the two inputs we created.

In [112]:

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(ListOfUniqueWords, max_features=5000)

countvector = cv.fit_transform(tweets["Dataset Documents"])

countvectorDF = pd.DataFrame(countvector.toarray())
countvectorDF.columns = cv.get_feature_names()

countvectorDF.head(10)
```

Out[112]:

	ساعة14	aes92	principl	أثر	أثناء	أجل	أجمعين	أجمل	أجنبي	أحد	...	نفسه	هدمت	والشيعه	وشردت	وطني	يا	يابن	ياحمار	يشيل
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
8	0	2	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0

10 rows × 5000 columns

Then we split the dataset in features and target variables. As well as configuring train and test sets. (30% Test 70% Train)

In [132]:

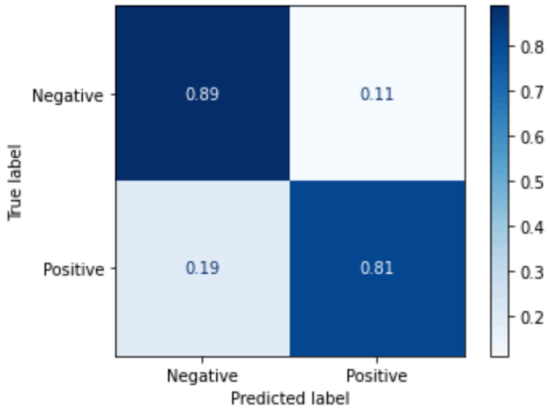
```
# Split Data
X = countvectorDF
y = tweets["Sentiment"]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)
```

Next, importing Logistic Regression module to create classifier. It's used to predict the probability of a categorical dependent variable. A confusion matrix is made to evaluate the performance of a classification model. Classification Report shows the following.

In [131]:

```
# Classification with Logistic Regression
from sklearn import metrics
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()

loreg = logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)

print("Classifier Report\n",classification_report(y_test, y_pred))
accCLF = metrics.accuracy_score(y_test,y_pred)
print("Accuracy:", accCLF)

disp = metrics.plot_confusion_matrix(loreg, X_test, y_test, display_labels=set(y), normalize='t
rue', cmap=plt.cm.Blues)
print(disp)
```

```
Classifier Report
              precision    recall  f1-score   support

    Negative       0.83      0.89      0.86       309
    Positive       0.87      0.81      0.84       289

    accuracy                           0.85       598
   macro avg       0.85      0.85      0.85       598
weighted avg       0.85      0.85      0.85       598

Accuracy: 0.8478260869565217
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7f669f8fb978>
```



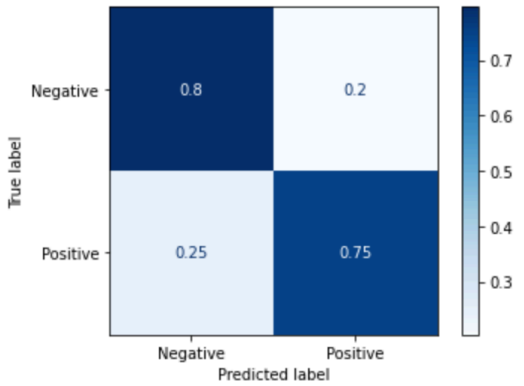
Output Comparison of Algorithms.

Decision Tree classifier is made, evaluated and plotted showing 77% Accuracy.

In [134]:

```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
y_predDTC = dtc.predict(X_test)

print("Decision Tree Report\n",classification_report(y_test,y_predDTC))
accDTC = metrics.accuracy_score(y_test,y_predDTC)
print("Accuracy : ",accDTC)

disp = metrics.plot_confusion_matrix(dtc, X_test, y_test, display_labels=set(y), normalize='tru
e', cmap=plt.cm.Blues)
print(disp)
```

```
Decision Tree Report
              precision    recall  f1-score   support

    Negative       0.77      0.80      0.78       309
    Positive       0.78      0.75      0.76       289

    accuracy                           0.77       598
   macro avg       0.77      0.77      0.77       598
weighted avg       0.77      0.77      0.77       598

Accuracy :  0.774247491638796
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7f66a396eba8>
```

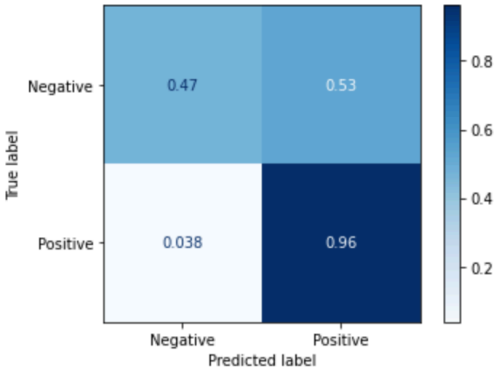


Naïve Bayes is implemented, evaluated, and plotted. Resulting in 70% Accuracy.

In [136]:

```
# Naive Bayes
from sklearn.naive_bayes import BernoulliNB
bnb = BernoulliNB()
bnb.fit(X_train, y_train)
y_predNB = bnb.predict(X_test)

print("Bayes Naïve Report\n",classification_report(y_test,y_predNB))
accNB = metrics.accuracy_score(y_test,y_predNB)
print("Accuracy : ",accNB)

disp = metrics.plot_confusion_matrix(bnb, X_test, y_test, display_labels=set(y), normalize='tru
e', cmap=plt.cm.Blues)
print(disp)
```

```
Bayes Naïve Report
              precision    recall  f1-score   support

    Negative       0.93      0.47      0.63       309
    Positive       0.63      0.96      0.76       289

    accuracy                           0.71       598
   macro avg       0.78      0.72      0.69       598
weighted avg       0.79      0.71      0.69       598

Accuracy :  0.7090301003344481
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7f66957b96a0>
```



Proving that Decision Tree is a better algorithm in this project than Naïve Bayes