

## **Second Assignment**

**Pertuz, Jose Ángel**

**Banda, Miguel Ángel**

### **Question**

Which political party does a user belong to on Twitter based on the tweets he/she posts?

### **Hypothesis**

People belonging to the same political party tend to express their ideas through tweets using similar lexicons and set of words.

### **Descriptive analysis**

We have a dataset that relates information of political Twitter users and their tweets. With which it is intended to make a classification program or algorithm that using certain methodologies or tools of natural language processing (NLP), can identify and classify which political parties they belong to.

The dataset is composed of the following information:

- Account: hashed name of the user writing the tweet
- Party: political party with which he/she identifies
- Timestamp: date of publication of the tweet
- Tweet: textual content of the tweet

## CONEXIÓN GOOGLE DRIVE - COLAB

```
In [2]: from google.colab import drive  
drive.mount('/content/drive', force_remount=True)
```

Mounted at /content/drive

```
In [3]: import sys  
sys.path.append('/content/drive/MyDrive/ColabNotebooks/')
```

## IMPORTAR LIBRERÍAS Y FUNCIONES

t)

```

except Exception as e:
    print('Error remove_patterns: {0}'.format(e))
return result

@staticmethod
def transformer(text: str, stopwords: bool = False):
    result = ''
    try:
        text_out = TextProcessing.proper_encoding(text)
        text_out = text_out.lower()
        text_out = re.sub("[\U0001f000-\U000e007f]", '[EMOJI]', text_out)
        text_out = re.sub(
            r'(?i)\b(?:https?://|www\d{0,3}[.][a-z0-9.-]+[.][a-z]{2,}
            )(?:[^\s()<>+|\\((\[^\s()<>+\\))\s]+|
            r'|(\[^\s()<>+\\))\s]+)+(?:\\((\[^\s()<>+\\))\s]+|(\[^\s()<>+\\))\s]+)*\\
            \s`!()\[\]{};:~".,<>?«»‘’"’])',
            '[URL]', text_out)
        text_out = re.sub("@([A-Za-z0-9_]{1,40})", '[MENTION]', text_out)
        text_out = re.sub("#([A-Za-z0-9_]{1,40})", '[HASTAG]', text_out)
        text_out = TextProcessing.remove_patterns(text_out)
        # text_out = TextAnalysis.Lemmatization(text_out) if Lemmatizer el
text_out
out
text_out = TextProcessing.stopwords(text_out) if stopwords else te

text_out = re.sub(r'\s+', ' ', text_out).strip()
text_out = text_out.rstrip()
result = text_out if text_out != ' ' else None
except Exception as e:
    print('Error transformer: {0}'.format(e))
return result

@staticmethod
def tokenizer(text: str):
    val = []
    try:
        text_tokenizer = TweetTokenizer()
        val = text_tokenizer.tokenize(text)
    except Exception as e:
        print('Error make_ngrams: {0}'.format(e))
    return val

@staticmethod
def make_ngrams(text: str, num: int):
    result = ''
    try:
        n_grams = ngrams(nltk.word_tokenize(text), num)
        result = [' '.join(grams) for grams in n_grams]
    except Exception as e:
        print('Error make_ngrams: {0}'.format(e))
    return result

```

```
def transformer(text: str, stopwords: bool = False):
    result = ''
    try:
        text_out = TextProcessing.proper_encoding(text)
        text_out = text_out.lower()
        text_out = re.sub("[\U0001f000-\U000e007f]", '[EMOJI]', text_out)
        text_out = re.sub(
            r'(?i)\b(?:https?://|www\d{0,3}[.][a-z0-9.-]+[.][a-z]{2,4})/?(?:[^\s()<>+|\\((([^\s()<>+]|'
            r'|(\([^\s()<>+\\)))*\\))+?:\\((([^\s()<>+|(\([^\s()<>+\\)))*\\'
            r'|[^\s`!()\\[\]{};:\\"'.,<>?«»‘’”])))',
            '[URL]', text_out)
        text_out = re.sub("@([A-Za-z0-9_]{1,40})", '[MENTION]', text_out)
        text_out = re.sub("#([A-Za-z0-9_]{1,40})", '[HASTAG]', text_out)
        text_out = TextProcessing.remove_patterns(text_out)
        # text_out = TextAnalysis.Lemmatization(text_out) if Lemmatizer el
se text_out
        text_out = TextProcessing.stopwords(text_out) if stopwords else te
xt_out
        text_out = re.sub(r'\s+', ' ', text_out).strip()
        text_out = text_out.rstrip()
        result = text_out if text_out != ' ' else None
    except Exception as e:
        print('Error transformer: {0}'.format(e))
    return result

def make_ngrams(text: str, num: int):
    result = ''
    try:
        n_grams = ngrams(nltk.word_tokenize(text), num)
        result = [' '.join(grams) for grams in n_grams]
    except Exception as e:
        print('Error make_ngrams: {0}'.format(e))
    return result
```

```
In [6]: import io
import sys
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import nltk
from collections import Counter
from nltk.tokenize import RegexpTokenizer
from sklearn.pipeline import FeatureUnion
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import RandomOverSampler
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split, cross_val_score, Shuffle
Split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, recall_sc
ore, log_loss
from sklearn.metrics import f1_score, accuracy_score, precision_score
```

/usr/local/lib/python3.7/dist-packages/sklearn/externals/six.py:31: FutureWarning: The module is deprecated in version 0.21 and will be removed in version 0.23 since we've dropped support for Python 2.7. Please rely on the official version of six (<https://pypi.org/project/six/>).

"(<https://pypi.org/project/six/>).", FutureWarning)

/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning: The sklearn.neighbors.base module is deprecated in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.neighbors. Anything that cannot be imported from sklearn.neighbors is now part of the private API.

warnings.warn(message, FutureWarning)

```
In [7]: class POSExtraction(BaseEstimator, TransformerMixin):
        def fit(self, x, y=None):
            return self

        def transform(self, list_messages):
            try:
                result = self.get_features(list_messages)
                return result
            except Exception as e:
                print('Error transform: {0}'.format(e))

        def get_features(self, list_messages):
            result = {}
            i = 0
            for row in list_messages:
                dict_pos = {}
                doc = nlp(str(row))
                pos = [token.pos_ for token in doc]
                for token in pos:
                    if token not in dict_pos:
                        dict_pos[token] = 1
                    else:
                        val = dict_pos[token]
                        dict_pos[token] = val + 1
                result[i] = dict_pos
                i += 1
            features = pd.DataFrame.from_dict(result, orient='index').fillna(0)
            return features.to_numpy()
```

```
In [8]: le = LabelEncoder()
```

## EXTRAER DATASET

```
In [38]: raw_data = pd.read_csv('/content/drive/MyDrive/Datasets/tweets_politica_kaggl
e.csv', sep=' ')
raw_data
```

Out[38]:

	cuenta	partido	timestamp	tweet
0	a34133350b0605cb24081843f63176ca	psoe	1.363973e+09	@vesteve3 @manubenas @ccoo_rm @desobediencia_ ...
1	a34133350b0605cb24081843f63176ca	psoe	1.364061e+09	"@kirovast: @Hugo_Moran muy fan de la "radical...
2	a34133350b0605cb24081843f63176ca	psoe	1.364117e+09	@ALTAS_PRESIONES Nuevos dueños para las renova...
3	a34133350b0605cb24081843f63176ca	psoe	1.364121e+09	@jumanjisolar @solartradex @josea_dolera El di...
4	a34133350b0605cb24081843f63176ca	psoe	1.364153e+09	"@cesarnayu: https://t.co/J4OTXj1x7w ... Por fav...
...	...	...	...	...
130024	2a5fcd1034beb5bd30bf5a1528008d81	psoe	1.633250e+09	Qué maravilla, visitar #LaRioja en #vendimia 🍷,...
130025	1e826d8471835f7feb02f8028b736ebb	pp	1.633250e+09	"Querido Pablo, nos complace tu voluntad de re...
130026	aeaa6ce266f823338e7d222032a9edd	psoe	1.633250e+09	Quiero reivindicar la buena política, la de fr...
130027	aeaa6ce266f823338e7d222032a9edd	psoe	1.633250e+09	🇪🇺 El pasado viernes se cumplió el aniversar...
130028	1e826d8471835f7feb02f8028b736ebb	pp	1.633250e+09	"Para nosotros la cuestión no es Europa sí o n...

130029 rows × 4 columns

```
In [39]: a=np.shape(raw_data)
b=a[0]
b
```

Out[39]: 130029

## EXTRACCIÓN DE CARACTERÍSTICAS

```
In [40]: cuenta=[]
partido=[]
tweet=[]
for i in range(b):
    cuenta.append(raw_data['cuenta'][i])
    partido.append(raw_data['partido'][i])
    tweet.append(raw_data['tweet'][i])
```



## TWEETS

```
In [41]: np.array(tweet)
```

```
Out[41]: array(['@vesteve3 @manubenas @ccoo_rm @desobediencia_ @ccoo @emparempar (Buen
á nimo para esta primavera que iniciamos).',
 '@kirovast: @Hugo_Moran muy fan de la "radicalidad social"' (Frente a
la devaluación democrática).',
 '@ALTAS_PRESIONES Nuevos dueños para las renovables. En ese momento ya
no serán un problema sino una apuesta magnífica.',
 ...,
 'Quiero reivindicar la buena política, la de frente al insulto y ocurr
encias, las propuestas y los acuerdos. Señora Ayuso, quien le diga que es pos
ible unos servicios públicos suficientes con una fiscalidad INJUSTA, es menti
ra. 🗣️@isauralealf #EspañaAvanza_ https://t.co/Wqa97Vva0c',
 '👩‍🦰\u200d♀️El pasado viernes se cumplió el aniversario de la aprobación
del voto femenino en España. 🇪🇸 Debemos proteger, cuidar y hacer crecer ese leg
ado, especialmente ahora que el machismo más reaccionario se vuelve a abrir p
aso en las instituciones. 🗣️\u200d♀️@Adrilastra #EspañaAvanza_ https://t.co/d
r332FzhNU',
 '"Para nosotros la cuestión no es Europa sí o no, sino cómo; cómo pode
mos hacer que Europa sea más fuerte, más eficiente y más poderosa".A Pablo Ca
sado: "Gracias por ser una voz fuerte en Europa, para una Europa del sentido
común". 🗣️ @sebastiankurz #CreemosElCambio https://t.co/xGTUdAZIgj'],
 dtype='<U800')
```

## NORMALIZACIÓN DE DATOS (PARTIDO POLÍTICO)

```
In [42]: y = le.fit_transform(raw_data['partido'])
y
```

```
Out[42]: array([3, 3, 3, ..., 3, 3, 2])
```

## EXTRACCIÓN DE CARACTERÍSTICAS

```
In [43]: corpus = [transformer(row, stopwords=True) for row in raw_data['tweet'].tolist
()]
bow = CountVectorizer(analyzer='word', ngram_range=(2, 3))
x = bow.fit_transform(corpus)
```

```
In [15]: np.shape(x)
```

```
Out[15]: (9998, 300764)
```

```
In [16]: #to visualize Bag of Words
#df = pd.DataFrame(x.toarray(),
#                  index=['content '+str(i) for i in range(1, 1+len(corpus))],
#                  columns=bow.get_feature_names())
#np.shape(df)
```

```
In [44]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=40)

In [45]: tfidf_vectoriser = TfidfVectorizer(max_features=5000, min_df=2, max_df=0.9, ngram_range=(1,3))
bow_vector = CountVectorizer(analyzer='word', ngram_range=(1, 3))
pos_vector = POSExtraction()

In [46]: preprocessor = FeatureUnion([('bow_vector', bow_vector),('pos_vector', pos_vector)])

In [47]: k_fold = ShuffleSplit(n_splits=1, test_size=0.20, random_state=32)

In [48]: ros_train = RandomOverSampler(random_state=1000)
x_train, y_train = ros_train.fit_resample(x_train, y_train)

/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function safe_indexing is deprecated; safe_indexing is deprecated in version 0.22 and will be removed in version 0.24.
  warnings.warn(msg, category=FutureWarning)

In [49]: print('**OverSample train:', sorted(Counter(y_train).items()))

**OverSample train: [(0, 23057), (1, 23057), (2, 23057), (3, 23057), (4, 23057)]

In [50]: ros_test = RandomOverSampler(random_state=1000)
x_test, y_test = ros_test.fit_resample(x_test, y_test)

/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function safe_indexing is deprecated; safe_indexing is deprecated in version 0.22 and will be removed in version 0.24.
  warnings.warn(msg, category=FutureWarning)

In [51]: print('**OverSample test:', sorted(Counter(y_test).items()))

**OverSample test: [(0, 5825), (1, 5825), (2, 5825), (3, 5825), (4, 5825)]

In [52]: softmax = LogisticRegression(multi_class="multinomial", solver="lbfgs", C=1)

In [53]: accuracies_scores = []
recalls_scores = []
precisions_scores = []
f1_scores = []
```

```
In [57]: for train_index, test_index in k_fold.split(x_train, y_train):
        data_train = x_train[train_index]
        target_train = y_train[train_index]

        data_test = x_train[test_index]
        target_test = y_train[test_index]

        softmax.fit(data_train, target_train)
        predict = softmax.predict(data_test)
        # Accuracy
        accuracy = accuracy_score(target_test, predict)
        accuracies_scores.append(accuracy)
        # Recall
        recall = recall_score(target_test, predict, average='macro')
        recalls_scores.append(recall)
        # Precision
        precision = precision_score(target_test, predict, average='weighted')
        precisions_scores.append(precision)
        # F1
        f1 = f1_score(target_test, predict, average='weighted')
        f1_scores.append(f1)
        print(1)
```

/usr/local/lib/python3.7/dist-packages/sklearn/linear\_model/\_logistic.py:940:  
 ConvergenceWarning: lbfgs failed to converge (status=1):  
 STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:  
<https://scikit-learn.org/stable/modules/preprocessing.html>  
 Please also refer to the documentation for alternative solver options:  
[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG)

1

```
In [55]: average_recall = round(np.mean(recalls_scores) * 100, 2)
        average_precision = round(np.mean(precisions_scores) * 100, 2)
        average_f1 = round(np.mean(f1_scores) * 100, 2)
        average_accuracy = round(np.mean(accuracies_scores) * 100, 2)
```

/usr/local/lib/python3.7/dist-packages/numpy/core/fromnumeric.py:3373: RuntimeWarning: Mean of empty slice.

out=out, \*\*kwargs)

/usr/local/lib/python3.7/dist-packages/numpy/core/\_methods.py:170: RuntimeWarning: invalid value encountered in double\_scalars

ret = ret.dtype.type(ret / rcount)

## PREDICTION

```
In [58]: y_predict = []
         for features in x_test:
             features = features.reshape(1, -1)
             value = softmax.predict(features)[0]
             y_predict.append(value)

         classification = classification_report(y_test, y_predict)
         confusion = confusion_matrix(y_predict, y_test)
```

```
In [33]: output_result1 = {'F1-score': average_f1, 'Accuracy': average_accuracy, 'Recall': average_recall, 'Precision': average_precision}
```

## RESULTS

```
In [34]: for item, val in output_result1.items():
         print('{0} {1}'.format(item, val))
```

```
F1-score 79.31
Accuracy 79.4
Recall 79.32
Precision 79.71
```

```
In [59]: output_result2 = {'Classification Report\n': classification, 'Confusion Matrix\n': confusion}
         for item, val in output_result2.items():
             print('{0} {1}'.format(item, val))
```

```
Classification Report
              precision    recall  f1-score   support

     0       0.71      0.54      0.61      5825
     1       0.69      0.64      0.67      5825
     2       0.64      0.56      0.60      5825
     3       0.60      0.67      0.63      5825
     4       0.54      0.72      0.62      5825

 accuracy          0.63      29125
 macro avg       0.64      0.63      0.63      29125
 weighted avg    0.64      0.63      0.63      29125
```

```
Confusion Matrix
[[3117  230  564  208  278]
 [ 392 3731  325  555  378]
 [ 721  301 3291  355  454]
 [ 584  729  748 3917  550]
 [1011  834  897  790 4165]]
```

```
In [ ]:
```

## BAG OF WORDS - SOFTMAX

```
In [ ]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=40)
softmax1 = LogisticRegression(multi_class="multinomial", solver="lbfgs", C=10, max_iter=200)
softmax1.fit(x_train, y_train)
```

```
In [17]: y_predict = softmax1.predict(x_test)
confusion_matrix(y_test, y_predict, labels=[0, 1, 2, 3, 4])
```

```
Out[17]: array([[2589, 319, 717, 403, 609],
 [ 261, 3252, 289, 609, 543],
 [ 720, 275, 2933, 576, 712],
 [ 368, 660, 500, 3699, 598],
 [ 393, 337, 508, 434, 3702]])
```

```
In [18]: classification = classification_report(y_test, y_predict)
print(classification)
```

	precision	recall	f1-score	support
0	0.60	0.56	0.58	4637
1	0.67	0.66	0.66	4954
2	0.59	0.56	0.58	5216
3	0.65	0.64	0.64	5825
4	0.60	0.69	0.64	5374
accuracy			0.62	26006
macro avg	0.62	0.62	0.62	26006
weighted avg	0.62	0.62	0.62	26006

## BAG OF N-GRAMS - SOFTMAX

```
In [ ]: corpus = [transformer(row) for row in raw_data['tweet'].tolist()]
bow = CountVectorizer(analyzer='word', ngram_range=(1, 3))
x = bow.fit_transform(corpus)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=40)
softmax2 = LogisticRegression(multi_class="multinomial", solver="lbfgs", C=10, max_iter=200)
softmax2.fit(x_train, y_train)
```

```
In [20]: y_predict = softmax2.predict(x_test)
confusion_matrix(y_test, y_predict, labels=[0, 1, 2, 3, 4])
```

```
Out[20]: array([[2785, 299, 654, 380, 519],
 [ 246, 3438, 283, 532, 455],
 [ 612, 249, 3258, 496, 601],
 [ 281, 543, 442, 3995, 564],
 [ 333, 356, 462, 389, 3834]])
```

```
In [21]: classification = classification_report(y_test, y_predict)
print(classification)
```

	precision	recall	f1-score	support
0	0.65	0.60	0.63	4637
1	0.70	0.69	0.70	4954
2	0.64	0.62	0.63	5216
3	0.69	0.69	0.69	5825
4	0.64	0.71	0.68	5374
accuracy			0.67	26006
macro avg	0.67	0.66	0.66	26006
weighted avg	0.67	0.67	0.67	26006

## CONEXIÓN GOOGLE DRIVE - COLAB

```
In [1]: from google.colab import drive  
drive.mount('/content/drive', force_remount=True)
```

Mounted at /content/drive

```
In [2]: import sys  
sys.path.append('/content/drive/MyDrive/ColabNotebooks/')
```

## IMPORTAR LIBRERÍAS Y FUNCIONES

t)



```

except Exception as e:
    print('Error remove_patterns: {0}'.format(e))
return result

@staticmethod
def transformer(text: str, stopwords: bool = False):
    result = ''
    try:
        text_out = TextProcessing.proper_encoding(text)
        text_out = text_out.lower()
        text_out = re.sub("[\U0001f000-\U000e007f]", '[EMOJI]', text_out)
        text_out = re.sub(
            r'(?i)\b(?:https?://|www\d{0,3}[.][a-z0-9.-]+[.][a-z]{2,}
            (?:[^\s()<>+|\\(([\^s()<>+|\\))*)\\))+(?:[^\s()<>+|\\(([\^s()<>+|\\))*)\\)
            \s`!()\[\]{};:\\".,<>?«»‘’"’])')',
            '[URL]', text_out)
        text_out = re.sub("@([A-Za-z0-9_]{1,40})", '[MENTION]', text_out)
        text_out = re.sub("#([A-Za-z0-9_]{1,40})", '[HASTAG]', text_out)
        text_out = TextProcessing.remove_patterns(text_out)
        # text_out = TextAnalysis.Lemmatization(text_out) if Lemmatizer el
text_out
    text_out = TextProcessing.stopwords(text_out) if stopwords else te
out

    text_out = re.sub(r'\s+', ' ', text_out).strip()
    text_out = text_out.rstrip()
    result = text_out if text_out != ' ' else None
except Exception as e:
    print('Error transformer: {0}'.format(e))
return result

@staticmethod
def tokenizer(text: str):
    val = []
    try:
        text_tokenizer = TweetTokenizer()
        val = text_tokenizer.tokenize(text)
    except Exception as e:
        print('Error make_ngrams: {0}'.format(e))
    return val

@staticmethod
def make_ngrams(text: str, num: int):
    result = ''
    try:
        n_grams = ngrams(nltk.word_tokenize(text), num)
        result = [' '.join(grams) for grams in n_grams]
    except Exception as e:
        print('Error make_ngrams: {0}'.format(e))
    return result

```

xt\_out

```
In [5]: import os
import spacy
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import svm
import matplotlib.pyplot as plt
from collections import Counter
from sklearn import preprocessing
from nltk.tokenize import RegexpTokenizer
from sklearn.pipeline import FeatureUnion
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import RandomOverSampler
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, cross_val_score, Shuffle
Split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import classification_report, confusion_matrix, recall_sc
ore, log_loss
from sklearn.metrics import f1_score, accuracy_score, precision_score
```

/usr/local/lib/python3.7/dist-packages/sklearn/externals/six.py:31: FutureWarning: The module is deprecated in version 0.21 and will be removed in version 0.23 since we've dropped support for Python 2.7. Please rely on the official version of six (<https://pypi.org/project/six/>).

"(<https://pypi.org/project/six/>).", FutureWarning)

/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning: The sklearn.neighbors.base module is deprecated in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.neighbors. Anything that cannot be imported from sklearn.neighbors is now part of the private API.

warnings.warn(message, FutureWarning)

```
In [6]: class POSExtraction(BaseEstimator, TransformerMixin):
        def fit(self, x, y=None):
            return self

        def transform(self, list_messages):
            try:
                result = self.get_features(list_messages)
                return result
            except Exception as e:
                print('Error transform: {0}'.format(e))

        def get_features(self, list_messages):
            result = {}
            i = 0
            for row in list_messages:
                dict_pos = {}
                doc = nlp(str(row))
                pos = [token.pos_ for token in doc]
                for token in pos:
                    if token not in dict_pos:
                        dict_pos[token] = 1
                    else:
                        val = dict_pos[token]
                        dict_pos[token] = val + 1
                result[i] = dict_pos
                i += 1
            features = pd.DataFrame.from_dict(result, orient='index').fillna(0)
            return features.to_numpy()
```

```
In [7]: le = LabelEncoder()
```

## EXTRAER DATASET

```
In [8]: raw_data = pd.read_csv('/content/drive/MyDrive/Datasets/tweets_politica_kaggl
e.csv', sep=' ')
raw_data
```

Out[8]:

	cuenta	partido	timestamp	tweet
0	a34133350b0605cb24081843f63176ca	psoe	1.363973e+09	@vesteve3 @manubenas @ccoo_rm @desobediencia_ ...
1	a34133350b0605cb24081843f63176ca	psoe	1.364061e+09	"@kirovast: @Hugo_Moran muy fan de la "radical...
2	a34133350b0605cb24081843f63176ca	psoe	1.364117e+09	@ALTAS_PRESIONES Nuevos dueños para las renova...
3	a34133350b0605cb24081843f63176ca	psoe	1.364121e+09	@jumanjisolar @solartradex @josea_dolera El di...
4	a34133350b0605cb24081843f63176ca	psoe	1.364153e+09	"@cesarnayu: https://t.co/J4OTXj1x7w ... Por fav...
...	...	...	...	...
130024	2a5fcd1034beb5bd30bf5a1528008d81	psoe	1.633250e+09	Qué maravilla, visitar #LaRioja en #vendimia 🍷,...
130025	1e826d8471835f7feb02f8028b736ebb	pp	1.633250e+09	"Querido Pablo, nos complace tu voluntad de re...
130026	aeaa6ce266f823338e7d222032a9edd	psoe	1.633250e+09	Quiero reivindicar la buena política, la de fr...
130027	aeaa6ce266f823338e7d222032a9edd	psoe	1.633250e+09	🇪🇺 El pasado viernes se cumplió el aniversar...
130028	1e826d8471835f7feb02f8028b736ebb	pp	1.633250e+09	"Para nosotros la cuestión no es Europa sí o n...

130029 rows × 4 columns

```
In [9]: a=np.shape(raw_data)
b=a[0]
b
```

Out[9]: 130029

## EXTRACCIÓN DE CARACTERÍSTICAS

```
In [10]: cuenta=[]
partido=[]
tweet=[]
for i in range(b):
    cuenta.append(raw_data['cuenta'][i])
    partido.append(raw_data['partido'][i])
    tweet.append(raw_data['tweet'][i])
```

## TWEETS

```
In [11]: np.array(tweet)
```

```
Out[11]: array(['@vesteve3 @manubenas @ccoo_rm @desobediencia_ @ccoo @emparempar (Buen
ánimo para esta primavera que iniciamos).',
 '@kirovast: @Hugo_Moran muy fan de la "radicalidad social"' (Frente a
la devaluación democrática).',
 '@ALTAS_PRESIONES Nuevos dueños para las renovables. En ese momento ya
no serán un problema sino una apuesta magnífica.',
 ...,
 'Quiero reivindicar la buena política, la de frente al insulto y ocurr
encias, las propuestas y los acuerdos. Señora Ayuso, quien le diga que es pos
ible unos servicios públicos suficientes con una fiscalidad INJUSTA, es menti
ra. 🗣️@isauralealf #EspañaAvanza_ https://t.co/Wqa97Vva0c',
 '👤\u200d♀️El pasado viernes se cumplió el aniversario de la aprobación
del voto femenino en España. 🇪🇸 Debemos proteger, cuidar y hacer crecer ese leg
ado, especialmente ahora que el machismo más reaccionario se vuelve a abrir p
aso en las instituciones. 👤\u200d♀️@Adrilastra #EspañaAvanza_ https://t.co/d
r332FzhNU',
 '"Para nosotros la cuestión no es Europa sí o no, sino cómo; cómo pode
mos hacer que Europa sea más fuerte, más eficiente y más poderosa".A Pablo Ca
sado: "Gracias por ser una voz fuerte en Europa, para una Europa del sentido
común". 🗣️ @sebastiankurz #CreemosElCambio https://t.co/xGTUdAZIgj'],
 dtype='<U800')
```

## NORMALIZACIÓN DE DATOS (PARTIDO POLÍTICO)

```
In [12]: y = le.fit_transform(raw_data['partido'])
y
```

```
Out[12]: array([3, 3, 3, ..., 3, 3, 2])
```

## EXTRACCIÓN DE CARACTERÍSTICAS

```
In [13]: corpus = [transformer(row, stopwords=True) for row in raw_data['tweet'].tolist
()]
bow = CountVectorizer(analyzer='word', ngram_range=(2, 3))
x = bow.fit_transform(corpus)
```

```
In [14]: np.shape(x)
```

```
Out[14]: (130029, 2905850)
```

```
In [25]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state
=40)
```

```
In [16]: #to visualize Bag of Words
#df = pd.DataFrame(x.toarray(),
#                  index=['content '+str(i) for i in range(1, 1+len(corpus))],
#                  columns=bow.get_feature_names())
#df
```

```
In [17]: tfidf_vectoriser = TfidfVectorizer(max_features=5000, min_df=2, max_df=0.9, ng
ram_range=(1,3))
bow_vector = CountVectorizer(analyzer='word', ngram_range=(2, 3))
pos_vector = POSExtraction()
```

```
In [18]: preprocessor = FeatureUnion([('bow_vector', bow_vector), ('pos_vector', pos_vec
tor)])
```

```
In [19]: #preprocessor.fit(x_train)
```

```
In [20]: k_fold = ShuffleSplit(n_splits=1, test_size=0.20, random_state=19)
```

```
In [21]: ros_train = RandomOverSampler(random_state=1000)
x_train, y_train = ros_train.fit_resample(x_train, y_train)

/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function safe_indexing is deprecated; safe_indexing is deprecated i
n version 0.22 and will be removed in version 0.24.
  warnings.warn(msg, category=FutureWarning)
```

```
In [22]: print('**OverSample train:', sorted(Counter(y_train).items()))

**OverSample train: [(0, 23057), (1, 23057), (2, 23057), (3, 23057), (4, 2305
7)]
```

```
In [23]: ros_test = RandomOverSampler(random_state=1000)
x_test, y_test = ros_test.fit_resample(x_test, y_test)

/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function safe_indexing is deprecated; safe_indexing is deprecated i
n version 0.22 and will be removed in version 0.24.
  warnings.warn(msg, category=FutureWarning)
```

```
In [24]: print('**OverSample test:', sorted(Counter(y_test).items()))

**OverSample test: [(0, 5825), (1, 5825), (2, 5825), (3, 5825), (4, 5825)]
```

```
In [ ]: clf = svm.SVC(kernel='poly', degree=3, C=4).fit(x_train, y_train)
```

```
In [ ]: accuracies_scores = []
recalls_scores = []
precisions_scores = []
f1_scores = []
```

```
In [ ]: for train_index, test_index in k_fold.split(x_train, y_train):
        data_train = x_train[train_index]
        target_train = y_train[train_index]

        data_test = x_train[test_index]
        target_test = y_train[test_index]

        clf.fit(data_train, target_train)
        predict = clf.predict(data_test)
        # Accuracy
        accuracy = accuracy_score(target_test, predict)
        accuracies_scores.append(accuracy)
        # Recall
        recall = recall_score(target_test, predict, average='macro')
        recalls_scores.append(recall)
        # Precision
        precision = precision_score(target_test, predict, average='weighted')
        precisions_scores.append(precision)
        # F1
        f1 = f1_score(target_test, predict, average='micro')
        f1_scores.append(f1)
```

```
In [ ]: average_recall = round(np.mean(recalls_scores) * 100, 2)
        average_precision = round(np.mean(precisions_scores) * 100, 2)
        average_f1 = round(np.mean(f1_scores) * 100, 2)
        average_accuracy = round(np.mean(accuracies_scores) * 100, 2)
```

## PREDICTION

```
In [ ]: y_predict = []
        for features in x_test:
            features = features.reshape(1, -1)
            value = clf.predict(features)[0]
            y_predict.append(value)

        classification = classification_report(y_test, y_predict)
        confusion = confusion_matrix(y_predict, y_test)
```

```
In [ ]: output_result1 = {'F1-score': average_f1, 'Accuracy': average_accuracy, 'Recall': average_recall,
                          'Precision': average_precision}
```

## RESULTS

```
In [ ]: for item, val in output_result1.items():
        print('{0} {1}'.format(item, val))
```

```
F1-score 82.02
Accuracy 82.02
Recall 82.4
Precision 89.9
```



```
In [33]: output_result2 = {'Classification Report\n': classification, 'Confusion Matrix\n': confusion}
```

```
In [34]: for item, val in output_result2.items():
         print('{0} {1}'.format(item, val))
```

#### Classification Report

	precision	recall	f1-score	support
0	0.65	0.60	0.63	4637
1	0.70	0.69	0.70	4954
2	0.64	0.62	0.63	5216
3	0.69	0.69	0.69	5825
4	0.64	0.71	0.68	5374
accuracy			0.67	26006
macro avg	0.67	0.66	0.66	26006
weighted avg	0.67	0.67	0.67	26006

#### Confusion Matrix

```
[[2785 246 612 281 333]
 [ 299 3438 249 543 356]
 [ 654 283 3258 442 462]
 [ 380 532 496 3995 389]
 [ 519 455 601 564 3834]]
```

## BAG OF WORDS - SOFTMAX

```
In [26]: softmax1 = LogisticRegression(multi_class="multinomial", solver="lbfgs", C=10,
         max_iter=200)
         softmax1.fit(x_train, y_train)
```

/usr/local/lib/python3.7/dist-packages/sklearn/linear\_model/\_logistic.py:940:  
ConvergenceWarning: lbfgs failed to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:  
<https://scikit-learn.org/stable/modules/preprocessing.html>  
Please also refer to the documentation for alternative solver options:  
[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)  
extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG)

```
Out[26]: LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
                             intercept_scaling=1, l1_ratio=None, max_iter=200,
                             multi_class='multinomial', n_jobs=None, penalty='l2',
                             random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                             warm_start=False)
```

```
In [27]: y_predict = softmax1.predict(x_test)
confusion_matrix(y_test, y_predict, labels=[0, 1, 2, 3, 4])
```

```
Out[27]: array([[2598, 341, 713, 388, 597],
 [ 246, 3284, 326, 528, 570],
 [ 552, 283, 3208, 517, 656],
 [ 265, 574, 518, 3807, 661],
 [ 317, 370, 600, 422, 3665]])
```

```
In [28]: classification = classification_report(y_test, y_predict)
print(classification)
```

	precision	recall	f1-score	support
0	0.65	0.56	0.60	4637
1	0.68	0.66	0.67	4954
2	0.60	0.62	0.61	5216
3	0.67	0.65	0.66	5825
4	0.60	0.68	0.64	5374
accuracy			0.64	26006
macro avg	0.64	0.63	0.64	26006
weighted avg	0.64	0.64	0.64	26006

```
In [29]: corpus = [transformer(row) for row in raw_data['tweet'].tolist()]
bow = CountVectorizer(analyzer='word', ngram_range=(1, 3))
x = bow.fit_transform(corpus)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=40)
softmax2 = LogisticRegression(multi_class="multinomial", solver="lbfgs", C=10,
max_iter=200)
softmax2.fit(x_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max\_iter) or scale the data as shown in:  
<https://scikit-learn.org/stable/modules/preprocessing.html>  
Please also refer to the documentation for alternative solver options:  
[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)  
extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG)

```
Out[29]: LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, l1_ratio=None, max_iter=200,
multi_class='multinomial', n_jobs=None, penalty='l2',
random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
warm_start=False)
```

```
In [30]: y_predict = softmax2.predict(x_test)
         confusion_matrix(y_test, y_predict, labels=[0, 1, 2, 3, 4])
```

```
Out[30]: array([[2785, 299, 654, 380, 519],
                [ 246, 3438, 283, 532, 455],
                [ 612, 249, 3258, 496, 601],
                [ 281, 543, 442, 3995, 564],
                [ 333, 356, 462, 389, 3834]])
```

```
In [31]: classification = classification_report(y_test, y_predict)
         print(classification)
```

	precision	recall	f1-score	support
0	0.65	0.60	0.63	4637
1	0.70	0.69	0.70	4954
2	0.64	0.62	0.63	5216
3	0.69	0.69	0.69	5825
4	0.64	0.71	0.68	5374
accuracy			0.67	26006
macro avg	0.67	0.66	0.66	26006
weighted avg	0.67	0.67	0.67	26006