PROCESAMIENTO DEL LENGUAJE NATURAL - CLASIFICACION DE TEXTOS

El objetivo de este caso práctico es clasificar una serie de tuits en función de su tendencia política. Cada uno de los tuits ha sido escrito por algún miembro perteneciente a alguno de los siguientes partidos políticos: PSOE, PP, VOX, Unidas Podemos o Ciudadanos.

IMPORTAMOS LIBREARIAS

In [1]:
```
import pandas as pd

import spacy

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB, BernoulliNB

from sklearn.linear_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score,precision_score, recall_score, f1_score

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

import numpy as np

import itertools

from sklearn.metrics import confusion_matrix
```

DEFINICION DE FUNCIONES

In [78]:
```
def normalize(sentenses):
    """normalizamos la lista de frases y devolvemos la misma lista de frases normalizada"""
    for index, sentense in enumerate(sentenses):
        # Tokenizamos el tweets realizando los puntos 1,2 y 3.
        sentense = nlp(sentense.lower().replace('.', ' ').replace('#', ' ').strip())
        # Puntos 4,5,6,7 y 8
        sentenses[index] = " ".join([word.lemma_ for word in sentense if (not word.is_punct)
                                     and (len(word.text) > 2) and (not word.is_stop)
                                     and (not word.text.startswith('@'))
                                     and (not word.text.startswith('http'))
                                     and (not ':' in word.text)])
    return sentenses


def evaluation(model, name, X_train, y_train, X_test, y_test):
    """
    Función de devuelve en un diccionario las métricas de evaluación de
    Accuracy, Precision, Recall y F1 para los conjuntos de datos de entrenamiento y test
        model: modelo a evaluar
        name: nombre del modelo
        X_train: Variables de entrada del conjunto de datos de entrenamiento
        y_train: Variable de salida del conjunto de datos de entrenamiento
        X_test: Variables de entrada del conjunto de datos de test
        y_test: Variable de salida del conjunto de datos de test
        return: diccionario con el nombre del modelo y el valor de las métricas
    """
    model_dict = {}
    model_dict['name'] = name
    y_pred_train = model.predict(X_train)
    y_pred_test = model.predict(X_test)
    model_dict['accuracy_train'] = accuracy_score(y_true=y_train, y_pred=y_pred_train)
    model_dict['accuracy_tests'] = accuracy_score(y_true=y_test, y_pred=y_pred_test)
    model_dict['precision_train'] = precision_score(y_true=y_train, y_pred=y_pred_train, ave
rage='weighted')
    model_dict['precision_tests'] = precision_score(y_true=y_test, y_pred=y_pred_test, avera
ge='weighted')
    model_dict['recall_train'] = recall_score(y_true=y_train, y_pred=y_pred_train, average=
'weighted')
    model_dict['recall_tests'] = recall_score(y_true=y_test, y_pred=y_pred_test, average='we
ighted')
    model_dict['f1_train'] = f1_score(y_true=y_train, y_pred=y_pred_train, average='weighte
d')
    model_dict['f1_tests'] = f1_score(y_true=y_test, y_pred=y_pred_test, average='weighted')

    return model_dict




# Definimos el heatmap de la matriz de confusión
def plot_confusion_matrix(cm, classes, title, cmap=plt.cm.Greens):
    """
    This function prints and plots the confusion matrix.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], 'd'), horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

Cargamos los datos

In [162]:
```
politica = pd.read_csv("tweets_politica_5p.csv", sep = "::::")

politica
```

```
<ipython-input-162-56113e846a83>:1: ParserWarning: Falling back to the 'python' engine becaus
e the 'c' engine does not support regex separators (separators > 1 char and different from
'\s+' are interpreted as regex); you can avoid this warning by specifying engine='python'.
  politica = pd.read_csv("tweets_politica_5p.csv", sep = "::::")
```

Out[162]:

	cuenta	partido	timestamp	tweet
0	populares	pp	1.560158e+09	▶ @TeoGarciaEgea explica las diferencias de nu...
1	populares	pp	1.560158e+09	▶ “Somos la alternativa al Gobierno, y cierro ...
2	populares	pp	1.560158e+09	▶ “El PSOE ha inaugurado el outlet de los pact...
3	populares	pp	1.560157e+09	🔴 EN DIRECTORueda de prensa de @TeoGarciaEgea....
4	populares	pp	1.560155e+09	📡 Rueda de prensa de @TeoGarciaEgea. En direct...
...	...	...	...	...
99640	NoALaIdDeGenero	vox	1.596060e+09	"La mujer necesita al feminismo como un pez un...
99641	NoALaIdDeGenero	vox	1.596060e+09	@lastfirstime Muchos de nuestros compatriotas ...
99642	NoALaIdDeGenero	vox	1.596035e+09	Este hombre es nefasto. Sin fisuras. Nefasto i...
99643	pedro_fhz	vox	1.596283e+09	Una vez más, Corea del Sur nos demuestra cómo ...
99644	pedro_fhz	vox	1.596207e+09	Todo mi apoyo a las movilizaciones del sector ...

99645 rows × 4 columns

In [163]:
```
# buscamos nulos

politica.isnull().sum()  # 13 nulos en 3 columnas. Los vamos a borrar
```

Out[163]:
```
cuenta        0
partido      13
timestamp    13
tweet        13
dtype: int64
```

In [164]:
```
politica = politica.dropna()

politica.isnull().sum()
```

Out[164]:
```
cuenta       0
partido      0
timestamp    0
tweet        0
dtype: int64
```

In [165]:
```
# Nos quedamos solo con las los tweets publicados por determinadas personalidaddes de cada p
artido

CUENTAS = ["PSOE", "sanchezcastejon",
           "populares", "pablocasado_",
           "vox_es", "Santi_ABASCAL", "Jorgebuxade",
           "PODEMOS", "PabloIglesias", "MiguelUrban", "pnique", "TeresaRodr_",
           "CiudadanosCs", "InesArrimadas"]

politica = politica[politica["cuenta"].isin(CUENTAS)]
```

In [166]:
```
politica.shape
```

Out[166]:
```
(37677, 4)
```

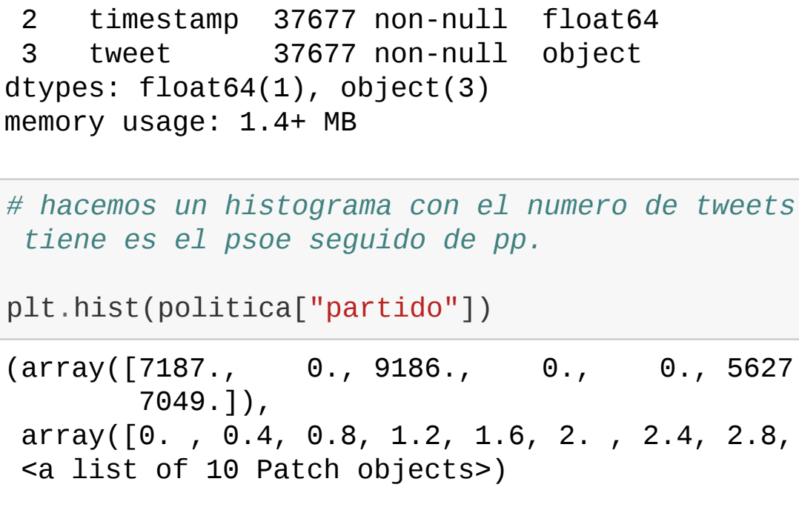
In [167]:
```
politica.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 37677 entries, 0 to 99557
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   cuenta     37677 non-null  object
 1   partido    37677 non-null  object
 2   timestamp  37677 non-null  float64
 3   tweet      37677 non-null  object
dtypes: float64(1), object(3)
memory usage: 1.4+ MB
```

In [168]:
```
# hacemos un histograma con el numero de tweets de cada partido. Podemos ver que el que mas
 tiene es el psoe seguido de pp.

plt.hist(politica["partido"])
```

Out[168]:
```
(array([7187.,    0., 9186.,    0.,    0., 5627.,    0., 8628.,    0.,
        7049.]),
 array([0. , 0.4, 0.8, 1.2, 1.6, 2. , 2.4, 2.8, 3.2, 3.6, 4. ]),
 <a list of 10 Patch objects>)
```



NORMALIZACION

Para normalizar los tweets realizaremos las siguientes acciones:

Pasamos las frases a minúsculas.

Eliminamos los signos de puntuación.

Eliminamos las palabras con menos de 3 caracteres.

Eliminamos las Stop-Words.

Eliminamos las palabras que empiecen por '@' o 'http'.

Pasamos la palabra a su lema

In [169]:
```
# cargamos el lenguaje de interpretacion, en este caso el castellano

nlp = spacy.load('es_core_news_sm')
```

In [170]:
```
# Divido el df en dos partes, los tweets y la target

X = politica["tweet"]

y = politica["partido"]
```

In [172]:
```
# convertimos X a lista

X_lista = list(X)
```

In [173]:
```
# normalizamos los tweets(X)

X_norm = normalize(X_lista)
```

In [175]:
```
len(X_norm)
```

Out[175]:
```
37677
```

In [ ]:

BOLSA DE PALABRAS

In [176]:
```
# Creamos el vectorizador donde vamos a indicar que entren en la bolsa de palabras, solo las
1000 palabras mas habituales
# y que para entrar, deben aparecer al menos 10 veces

vectorizer = CountVectorizer(max_features=10, min_df=2)

# creamos la bolsa de palabras con la variable X(tweets)

X_vector =  vectorizer.fit_transform(X_norm)
```

Particionado de Datos (Train y Test)

Vamos a particionar los datos en conjunto de Train y Test.

Para este ejemplo nos vamos a quedar con:

80% de datos de entrenamiento

20% de datos de test

In [177]:
```
X_vector.shape
```

Out[177]:
```
(37677, 10)
```

In [178]:
```
len(y)
```

Out[178]:
```
37677
```

In [179]:
```
X_train, X_test, y_train, y_test = train_test_split(X_vector, y, test_size=0.2, random_state
=0)
```

Creacción de Modelos y Evaluación (Accuracy)

Vamos a crear y evaluar una serie de modelos para ver cual es que obtiene mejores resultados.

Los modelos que vamos a crear y evaluar son los siguientes:

Multinomial Naive Bayes

Bernoulli Naive Bayes

Regresion Logistica

Support Vector Machine

Random Forest

In [181]:
```
mnb = MultinomialNB()
bnb = BernoulliNB()
lr = LogisticRegression(solver='lbfgs', multi_class='multinomial', max_iter=1000)
svm_lin = SVC(kernel='linear')
svm_rbf = SVC(kernel='rbf')
rf_50 = RandomForestClassifier(n_estimators=500, bootstrap=True, criterion='gini', max_depth
=50, random_state=0)

clasificadores = {'Multinomial NB': mnb,
                  'Bernoulli NB': bnb,
                  'Regresion Logistica': lr,
                  'SVM lineal': svm_lin,
                  'SVM Kernel rbf': svm_rbf,
                  'Random Forest d_50': rf_50}

# Ajustamos los modelos y calculamos el accuracy para los datos de entrenamiento
for k, v in clasificadores.items():
    print ('CREANDO MODELO: {clas}'.format(clas=k))
    v.fit(X_train, y_train)
```

```
CREANDO MODELO: Multinomial NB
CREANDO MODELO: Bernoulli NB
CREANDO MODELO: Regresion Logistica
CREANDO MODELO: SVM lineal
CREANDO MODELO: SVM Kernel rbf
CREANDO MODELO: Random Forest d_50
```

Evaluación del Modelo

Para cada uno de los modelos vamos a calcular las siguientes métricas de evaluación:

Accuracy

Precision

Recall

F1

In [182]:
```
# Calculamos las métricas de los modelos por separado
evaluacion = list()
for key, model in clasificadores.items():
    evaluacion.append(evaluation(model=model, name=key,
                                 X_train=X_train, y_train=y_train,
                                 X_test=X_test, y_test=y_test))

# Pasamos los resultados a un DataFrame para visualizarlos mejor
df = pd.DataFrame.from_dict(evaluacion)
df.set_index("name", inplace=True)
df
```

```
C:\Users\jaime\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: Undefined
MetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samp
les. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[182]:

	accuracy_train	accuracy_tests	precision_train	precision_tests	recall_train	recall_tests	f1_train	f1_tests
name								
Multinomial NB	0.316413	0.322718	0.391442	0.396293	0.316413	0.322718	0.276975	0.283544
Bernoulli NB	0.348794	0.357484	0.335311	0.340899	0.348794	0.357484	0.304345	0.313072
Regresion Logistica	0.349358	0.357749	0.334858	0.342109	0.349358	0.357749	0.306277	0.315300
SVM lineal	0.348396	0.354299	0.336880	0.341639	0.348396	0.354299	0.299097	0.306040
SVM Kernel rbf	0.363027	0.362128	0.419858	0.380062	0.363027	0.362128	0.330655	0.328093
Random Forest d_50	0.366444	0.354963	0.411464	0.365072	0.366444	0.354963	0.338040	0.324630

El que mejor resultado ha obtenido es Multinomial NB

Vamos a predecir estos tweets y ver como quedan

In [185]:
```
# vectorizamos un tweet a modo de prueba

tweet = ["Pedro Sanchez: dice que la monarquia es esencial"] # tweet llano

# normalizamos el tweet

tweet_norm = normalize(tweet)
```

In [188]:
```
# vectorizamos el tweet normalizado

vectorizer2 = CountVectorizer()

tweet_vector = vectorizer2.fit_transform(tweet_norm)
```

In [194]:
```
# predecimos a que partido pertenece el tweet

prediccion = mnb.predict(X_test)

prediccion[0:5]
```

Out[194]:
```
array(['psoe', 'podemos', 'psoe', 'psoe', 'podemos'], dtype='<U10')
```

In [195]:
```
y_test[0:5]
```

Out[195]:
```
74528    podemos
18217         pp
4419     podemos
93515       psoe
85484       psoe
Name: partido, dtype: object
```

In [ ]: