Autor: Jofre Gómez González

NIU: 1526889

Cas Kaggke: Topic Labeled News Dataset

Introducció

En aquest document veurem, analitzarem i compararem diferents classificadors sobre una base de dades de Kaggle.

Els objectius principals son:

- Analitzar la base de dades
- Aplicar diferents classificadors
- Analitzar els resultats obtinguts dels classificadors
- Extreure conclusions

La base de dades de kaggle utilitzada és la de "Topic Labeled News Dataset". Aquesta BD és una recopilació d'articles de notícies.

Analitzar base de dades

A continucació començarem veient i analitzant la base de dades.

In [1]:

```
# Importem llibreries que utilitzarem al llarg del document
import ipywidgets as widgets
from sklearn.datasets import make_regression
import numpy as np
import pandas as pd
%matplotlib notebook
from matplotlib import pyplot as plt
import scipy.stats
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import numpy as np #importem la llibreria
import random

from gensim.parsing.preprocessing import remove_stopwords
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB

import plotly.graph_objs as go
from plotly.offline import iplot

from sklearn.pipeline import Pipeline
from sklearn.linear_model import SGDClassifier
```

In [2]:

```
# Funcio per a llegir dades en format csv
def load_dataset(path):
    dataset = pd.read_csv(path,sep = None,engine='python')
    return dataset

# Carreguem dataset asignat
dataset = load_dataset('labelled_newscatcher_dataset.csv')
data = dataset.values
```

In [3]:

```
dataset.head()
```

Out[3]:

	topic	link	domain	published_date	
0	SCIENCE	https://www.eurekalert.org/pub_releases/2020-0...	eurekalert.org	2020-08-06 13:59:45	A c lo v splif sola
1	SCIENCE	https://www.pulse.ng/news/world/an-irresistibl...	pulse.ng	2020-08-12 15:14:19	irresi m lo sv
2	SCIENCE	https://www.express.co.uk/news/science/1322607...	express.co.uk	2020-08-13 21:01:00	Ar intellig warnir will
3	SCIENCE	https://www.ndtv.com/world-news/glaciers-could...	ndtv.com	2020-08-03 22:18:26	Gla (Scu Va
4	SCIENCE	https://www.thesun.ie/tech/5742187/perseid-met...	thesun.ie	2020-08-12 19:54:36	Pe m sh Wha and h

In [4]:

```
dataset.describe()
```

Out[4]:

	topic	link	domain	published_date	title	lang
count	108774	108774	108774	108774	108774	108774
unique	8	106130	5164	68743	103180	1
top	NATION	https://www.google.com/	dailymail.co.uk	2020-08-04 01:00:00	US tops 5 million confirmed virus cases, to Eu...	en
freq	15000	19	1855	41	21	108774

Com es pot observar, la BD està formada per 6 atributs:

- **topic:** topic en el que s'ha classificat l'article, segons el Kaggle n'hi han 8 tipos(BUSINESS, ENTERTAINMENT, HEALTH, NATION, SCIENCE, SPORTS, TECHNOLOGY i WORLD)
- **link:** link d'on trobar l'article
- **domain:** domini de la pàgina en que va ser publicat l'article
- **published_date:** data en que l'article va ser publicat
- **title:** títol de l'article
- **lang:** llengua en el que està escrit l'article, podem observar com tots els articles estàn en Anglés

Ara continuem analitzant la base de dades.

In [5]:

```
print(dataset.dtypes)
```

```
topic             object
link              object
domain            object
published_date    object
title             object
lang              object
dtype: object
```

In [6]:

```
print(dataset.isnull().sum())
```

```
topic             0
link              0
domain            0
published_date    0
title             0
lang              0
dtype: int64
```

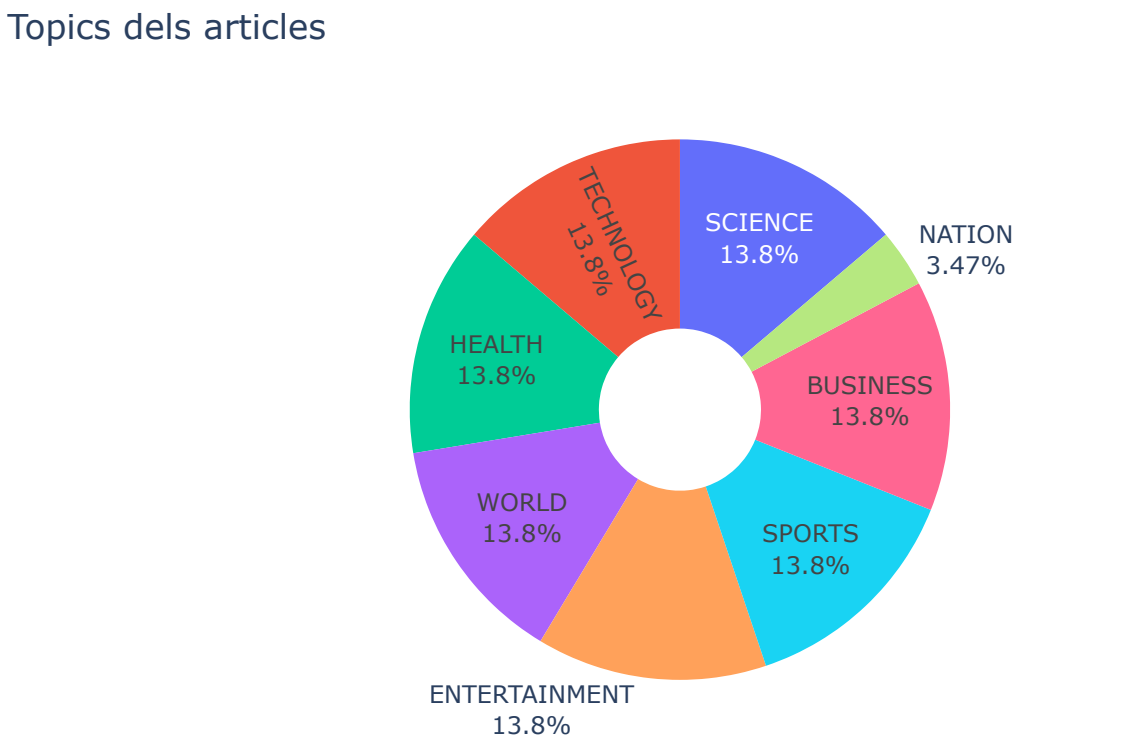
In [7]:

```
print (dataset['published_date'].min())
print (dataset['published_date'].max())
```

```
2012-09-16 04:44:50
2020-08-18 05:49:00
```

In [8]:

```
#contTopic = dataset['topic'].value_counts()
#contTopic

labels = pd.unique(dataset['topic'])
values = dataset['topic'].value_counts()

trace = go.Pie(labels=labels,
               values=values,
               textinfo='label+percent',
               hole=0.3)

data = [trace]

figure = go.Figure(data)
figure.update_layout(title="Topics dels articles", width=800, height=450)

iplot(figure)
```



Observacions importants:

- no tenim nungún atribut numéric
- no tenim ninguna entrada / fila en la base de dades amb algún valor a null
- podem observar com hi han 8 topics diferents, 7 d'ells consten de 15000 entrades en la BD i un d'ells (el de SCIENCE) 3774
- la data de publicació dels articles més antiga és del 2012-09-16 i la més recent del 2020-08-18
- l'atribut de "title" pot contenir paraules que no aporten cap informació útil per decidir en quina categoria s'ha de classificar un text ("stopwowrds") com per exemple articles, preposicions, etc...

Per a solucionar el problema dels "stopwords", utilitzem el métode "remove_stopwords" de la llibreria "gensim.parsing.preprocessing" en que si li pases un text per paràmetre el retorna sense "stopwords" i altres caràcters que no vulguis que contingui.

In [9]:

```
#eliminem els stopwords
dataset['title'] = dataset['title'].apply(lambda x: remove_stopwords(x))
```

In [10]:

```
#mirem si ha cannviat
dataset['title'].head()
```

Out[10]:

```
0    A closer look water-splitting's solar fuel pot...
1    An irresistible scent makes locusts swarm, stu...
2    Artificial intelligence warning: AI know bette...
3     Glaciers Could Have Sculpted Mars Valleys: Study
4    Perseid meteor shower 2020: What time huge bri...
Name: title, dtype: object
```

Finalment, he decidit entrenar i comparar classificadors que a partir del títol (camp "title" en la BD) faci una classificació d'un article segons el topic (camp objectiu "topic"). Descarto la llengua ja que tots els articles estàn escrits en la mateixa llengua, el link, el domini i la data de publicació perque no aportarien ninguna informació sobre el tópic de l'article.

Classificadors

A continuació, el primer classificador que entrenarem serà un classificador Naive Bayes. Com s'ha comentat, en la classificació el comp objectiu serà el de "topic" i es farà a partir del "title".

In [11]:

```
X = dataset['title']
Y = dataset['topic'] #objectiu

particions = [0.1, 0.2, 0.4] #porcentatge de dades que s'utilitzaran en test

for part in particions:
    # Dividim dades en part d'entrenament i test
    x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=part, random_state=

    #Creem un pipeline, que serveix per aplicar seqüencialment una llista de transformades
    #estimador final (l'estimador final ha de portar implementat "fit").
    #
    #La funció de "CountVectorizer" s'utilitza per convertir un text en un vector de recomp
    #retorna un diccionari de paraules i el seu recompte de vagades que ha sortit cada para
    #
    #La funció de "TfidfTransformer" transforma el diccionari / recompte de la funció "Coun
    #en una representació normalitzada tf (frequéncia) o tf-idf (frequéncia multiplicada pe
    #això fa que obtinguem la frequéncia d'aparició d'una paraula.
    #
    #Finalment, tenim la funció "MultinomialNB" que és el classificador multinomial Naive B
    #és adequat per a la classificació amb característiques discretes (p. ex., recompte de
    #de text)

    naiveBayes = Pipeline([('countVect', CountVectorizer()),
                           ('tfidfTransf', TfidfTransformer()),
                           ('clf', MultinomialNB()),
    ])

    #entrenem
    naiveBayes = naiveBayes.fit(x_train, y_train)

    #test
    predicted = naiveBayes.predict(x_test)

    print ("Correct classification Naive Bayes with stopwords      ", part, "% of the data:
```

```
Correct classification Naive Bayes with stopwords       0.1 % of the data:
0.793252436109579
Correct classification Naive Bayes with stopwords       0.2 % of the data:
0.7942541944380602
Correct classification Naive Bayes with stopwords       0.4 % of the data:
0.7823948517582165
```

Com podem observar, el millor accuracy que s'ha obtingut ha sigut l'entrenat amb el 80% de les dades amb un accuracy del 79,42%. També, comentar que per molt que utilitzem més porcentatge d'entrenament, no tenim una gran millora, la millora obtinguda es molt baixa(1%).

Ara anem a entrenar un classificador SVM (Support Vector Machines) i compararem els resultats amb el de Naive Bayes.

In [12]:

```
X = dataset['title']
Y = dataset['topic'] #objectiu

particions = [0.1, 0.2, 0.4] #porcentatge de dades que s'utilitzaran en test

for part in particions:
    # Dividim dades en part d'entrenament i test
    x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=part, random_state=

    #Creem un pipeline, que serveix per aplicar seqüencialment una llista de transformades
    #estimador final (l'estimador final ha de portar implementat "fit").
    #
    #La funció de "CountVectorizer" s'utilitza per convertir un text en un vector de recomp
    #retorna un diccionari de paraules i el seu recompte de vagades que ha sortit cada para
    #
    #La funció de "TfidfTransformer" transforma el diccionari / recompte de la funció "Coun
    #en una representació normalitzada tf (frequéncia) o tf-idf (frequéncia multiplicada pe
    #això fa que obtinguem la frequéncia d'aparició d'una paraula.
    #
    #Finalment, tenim la funció "SGDClassifier que és el classificador SVM

    svmMulti = Pipeline([('vect', CountVectorizer()),
                         ('tfidf', TfidfTransformer()),
                         ('clf-svm', SGDClassifier(alpha=0.001, penalty='l2')),
     ])

    #entrenem
    svmMulti.fit(x_train, y_train)

    #test
    #predicted_svm = svmMulti.predict(x_test)
    #això és per les curves roc i precision-recall
    predicted_svm_proba = svmMulti.decision_function(x_test)

    #print ("Correct classification SVM with stopwords      ", part, "% of the data: ", np.
    print ("Correct classification SVM with stopwords      ", part, "% of the data: ", svmM
```

```
Correct classification SVM with stopwords       0.1 % of the data:  0.735337
378194521
Correct classification SVM with stopwords       0.2 % of the data:  0.734084
1185934268
Correct classification SVM with stopwords       0.4 % of the data:  0.732567
2259250747
```

En el classificador SVM, s'han fet proves amb diferents porcentatges de dades que s'utilitzaven per entrenar el model i testejar. També s'han provat diferents paràmetres ("alpha", "penalty") amb l'intenció de buscar els millors. Això es pot observar en l'apartat de resultats.

Resultats

Classificador Naive Bayes	% train	% test	accuracy
	90	10	79,32%
	80	20	79,42%
	60	40	78,23%

Classificador SVM	% train	% test	accuracy
alpha=0.01, penalty='l2'	90	10	73,44%
	80	20	73,27%
	60	40	73,02%
alpha=0.1, penalty='l2'	90	10	23,99%
	80	20	25,65%
	60	40	13,98%
alpha=0.5, penalty='l2'	90	10	13,78%
	80	20	13,97%
	60	40	25,63%
alpha=0.001, penalty='l2'	90	10	73,32%
	80	20	73,34%
	60	40	73,24%
alpha=0.01, penalty='l1'	90	10	14,20%
	80	20	13,74%
	60	40	13,87%
alpha=0.1, penalty='l1'	90	10	13,96%
	80	20	13,96%
	60	40	13,96%

Com podem observar en les 2 taules de més amunt, el classificador Naive Bayes ens està donant un accuracy màxim de 79,42% quen entrenem amb el 80% de les dades, i si observem els resultats amb els altres porcentatges de dades que , observem que no varia gaire, tant amb un 60% com amb un 90% de dades d'entrenamest utilitzades la diferéncia és només d'un 1% (si en un futur la BD fos ampliada i s'afegissin millons de dades, s'hauria de veure si val la pena el cost computacional per entrenar el model a canvi d'un 1% de millora i si al treballar amb una BD més gran l'accuracy dona també més elevat).

Si observem la taula del classificador SVM, observem que la millor acccuracy ens dona amb els paràmetres "alpha=0.01" i "penalty:l2" amb un accuracy de 73,44% (inferior al del classificador Naive Bayes). Les conclusions que es poden treure sobre el classificador SVM son:

- amb el paràmetre "penalty:l1" (fent referéncia a que s'utilitzarà una regularització L1) l'accuracy es redueix dràsticament fins a un 14% aprox.
- amb el paràmetre "alpha" (segons sklearn, és una constant que multiplica el terme de regularització, com més gran sigui el valor, més forta serà la regularització) si utilitzem una alpha inferior a 0.1 obtindrem un molt millor accuracy en el model entrenat (fins a 73%) que no pas amb un alpha major o igual a 0.1 (que obtenim un accuracy d'entre 14% i 25% aprox.).

Comparant els dos models entrenats, el classificador Naive Bayes obté significativament un millor accuracy (fins a un 6% més), I en quant a temps / cost computacional per entrenar els dos classificadors, encara que no

(fins a un 6% més). Pel quant a temps / cost computacional per entrenar els dos classificadors, encara que no ha sigut cronometrat els dos triguen pràcticament el mateix, uns pocs segons, si en un futur s'afegissin milions de dades a la base de dades, hauriem de tornar a observar si n'hi han un model que requereixi molt més temps / cost computacional per entrenar-lo i si val la pena utilitzar-lo.

Conclusions

Podem conloure dient que amb la BD actual, el classificador Naive Bayes ens està donant un accuracy significativament superior (6%) respecte el classificador SVM alhora de classificar els articles segons el tópic en un temps molt similar.

Encara que el classificador Naive Bayes no és un classificador perfecte i té bastant marge de millora, en un futur s'hauria de provar amb una base de dades que contingués molts més articles (milions) i més classes a classificar, així el nostre classificador podria observar una major diverisitat de paraules en els títols dels articles i donar classificacions més precises.

També, mencionar la importància d'escollir uns bons paràmetres en els diferents classificadors ja que com hem pogut observar, al escollir uns paràmetres dolents en el classificador SVM com una alpha massa gran on una regularitazció l1 ens ha arrribat a reduir l'accuracy d'un 73% aproximadament a un 13-25% aprox.

Finalment, remarcar l'útil que ha sigut aquesta pràctica per posar en pràctica els coneixements adquirits en l'assignatura sobre una base de dades real i observar com es comporten diferents classificadors sobre ella.

Pots veure i descarregar el codi si accedeixes directament al github de la pràctica:
https://github.com/JofreGGonzalez/Kaggle_Topic_Labeled_News_Dataset
(https://github.com/JofreGGonzalez/Kaggle_Topic_Labeled_News_Dataset)