Funciones auxiliares para el preprocesamiento del DataSet

[17]:
```python
def create_prep_dataset(index_path, n_elements):
    X = []
    y = []
    indexes = parse_index(index_path, n_elements)
    for i in range(n_elements):
        print("\rParsing email: {0}".format(i+1), end = '')
        mail, label = parse_email(indexes[i])
        X.append(" ".join(['Subject']) + " ".join(mail['body']))
        y.append(label)
    return X, y
```

[18]:
```python
# Leer únicamente un subconjunto de 1000 correos electrónicos
X_train, y_train = create_prep_dataset("datasets/datasets/trec07p/full/index", 1000)
X_train
```

```
Parsing email: 1000
```

[18]:
```
['Subject\n\n\n\n\n\n\ndo feel pressur perform rise occasion\n\n\n\n\n\ntri viagra\nyour anxieti thing past will\nb back old self\n\n\n',
 'Subjecthi ive updat gulu i check mirrors\nit seem littl typo debianreadm file\n\nexample\nhttpgulususherbrookecadebianreadme\nftpftpfrdebia
norgdebianreadme\n\ntest lenni access releas diststest the\ncurr test develop snapshot name etch packag which\nhav test unstabl pass autom te
st propog to\nthi release\n\netch replac lenni like readmehtml\n\n\n\n \nyan morin\nconsult en logiciel libre\nyanmorinsavoirfairelinuxcom\n5
149941556\n\n\n \nto unsubscrib email debianmirrorsrequestlistsdebianorg\nwith subject unsubscrib troubl contact listmasterlistsdebianorg\n
\n',
 'Subjectmega authenticv i a g r a discount pricec i a l i s discount pricedo miss it click here\nhttpwwwmoujsjkhchumcom\n\n authent viagra\n
\nmega authenticv i a g r a discount pricec i a l i s discount pricedo miss it click here\n\n',
 'Subject\nhey billi \n\nit realli fun go night \nand talk said felt\ninsecur manhood i notic toilets\ny quit small area \nworri websit i tel
l \nmi secret weapon extra 3 inch trust girls\nlov bigger one ive 5 time mani chick \nsinc i use pill year ago the packag i used\nwa 6 month
suppli one worth everi \ncent websit httpctmaycom \nring weekend go drink \nagain let know secret \nlater dude brad\n\n',
 'Subject\nsystem home it capabl link far i \nknow i within part with respect the\n affect technolog societi scienc ad agenc cashin\ng commer
ci and\nphotograph paint electron canvas still seem like silenc \n white black light it didnt happen although far p\nerfect especi preclud va
st explan i can\nt understand peopl reli so\navantgard art world addit writer lawyer yet re\nach full potenti i imagin futur comput screen ho
\nw would percept artifici imag busi apart there t\nhe computer sign shop the\nb filter unnecessari labor technolog comp\nani monopoli servic
as servic root tre\nnd western cultur sinc dark age possibl unimag\nin human think \n\nfriendli not industri isnt welcom new peopl thought c
```

Aplicar vectorización a los Datos

[19]:
```python
vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
```

Aplicar vectorización a los Datos

[19]:
```python
vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
```


[20]:
```python
print(X_train.toarray())
print("\Features", len(vectorizer.get_feature_names_out()))
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
\Features 21970
```

[21]:
```python
import pandas as pd

pd.DataFrame(X_train.toarray(), columns=[vectorizer.get_feature_names_out()])
```

[21]:

	00	000	0000	000000	000002	000048000000000	000099	0000ff	000115000000000	0001171749	...	绰tel	绰٩ϵ₸ctsƽ	肾əvă	铿ėvłq	侬jwk	鸫x	닭ôglgwă	뾊ᶰϵ	쨊ƹư	춘ã
0	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	0
2	0	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	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
996	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
997	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
998	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
999	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

1000 rows × 21970 columns

[22]:
```
y_train
```

[22]:
```
['spam',
 'ham',
 'spam',
 'spam',
 'spam',
 'spam',
 'spam',
 'spam',
 'spam',
 'ham',
 'spam',
 'spam',
 'spam',
 'spam',
 'spam',
 'spam',
 'spam',
 'spam',
 'spam'
```

Entrenamiento del algoritmo de REgresión Logistica con el DataSet Preprocesado

[23]:
```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression()
clf.fit(X_train, y_train)
```

[23]:
```
LogisticRegression()
```

4.- Predicción

[24]:
```
# Lectura de un DataSet de correos nuevos.

# Leer 1500 correos de nuestro DataSet y quedarnos unicámente con los 500 ultimos correos electrónicos, los cuales no se han utilizado para en
X, y = create_prep_dataset("datasets/datasets/trec07p/full/index", 150)
X_test = X[100:]
y_test = y[100:]
```

```
Parsing email: 150
```

Preprocesamiento de los correos electrónicos con el vectorizado creado anteriormente

[25]:
```
X_test = vectorizer.transform(X_test)
```

[26]:
```
y_pred = clf.predict(X_test)
y_pred
```

[26]:
```
array(['spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam',
       'spam', 'spam', 'ham', 'spam', 'spam', 'spam', 'spam', 'spam',
       'ham', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'ham',
       'spam', 'spam', 'ham', 'spam', 'spam', 'ham', 'spam', 'spam',
       'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam',
       'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam',
       'spam', 'spam'], dtype='<U4')
```

[27]:
```
print("Predicción\n", y_pred)
print("\nEtiquetas reales", y_test)
```

```
Predicción
 ['spam' 'spam' 'spam' 'spam' 'spam' 'spam' 'spam' 'spam' 'spam' 'spam'
 'ham' 'spam' 'spam' 'spam' 'spam' 'spam' 'ham' 'spam' 'spam' 'spam'
 'spam' 'spam' 'spam' 'ham' 'spam' 'spam' 'ham' 'spam' 'spam' 'ham' 'spam'
 'spam' 'spam' 'spam' 'spam' 'spam' 'spam' 'spam' 'spam' 'spam' 'spam'
 'spam' 'spam' 'spam' 'spam' 'spam' 'spam' 'spam' 'spam' 'spam']

Etiquetas reales ['spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'ham', 'spam', 'spam', 'spam', 'spam', 'spa
m', 'ham', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'ham', 'spam', 'spam', 'ham', 'spam', 'spam', 'ham', 'spam', 'spam', 'spam', 'spam
', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam', 'spam']
```

Evaluación de resultados

[28]:
```
from sklearn.metrics import accuracy_score
print("Accuracy: {:.3f}".format(accuracy_score(y_test, y_pred)))
```

```
Accuracy: 1.000
```

5.- Aumentando el DataSet

[29]:
```
# Leer 20,000 correos electrónicos
X, y = create_prep_dataset("datasets/datasets/trec07p/full/index", 20000)
```

```
Parsing email: 20000
```

[30]:
```
# Utilizamos 15,000 correos para entrenar el algoritmo y 5,000 para realizar pruebas
X_train, y_train = X[:15000], y[:15000]
X_test, y_test = X[15000:], y[15000:]
```

[31]:
```
vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
```

[32]:
```
clf = LogisticRegression()
clf.fit(X_train, y_train)
```

[32]:
```
LogisticRegression()
```

[33]:
```
X_test = vectorizer.transform(X_test)
y_pred = clf.predict(X_test)
```

[34]:
```
print("Accuracy: {:.3f}".format(accuracy_score(y_test, y_pred)))
```

```
Accuracy: 0.989
```