El algoritmo SVM (máquina de vectores de soporte) para la clasificación de texto

Ahora vamos a utilizar el algoritmo SVM para intentar clasificar nuestro texto. Este algoritmo empieza a ser familiar porque en los capítulos anteriores hemos podido conocer su funcionamiento y optimización.

Al igual que el algoritmo bayesiano ingenuo, vamos a utilizar una canalización para llevar a cabo las distintas tareas de análisis de los mensajes antes de su aprendizaje:

from sklearn.pipeline import Pipeline

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfTransformer

**from sklearn.svm import SVC**

etapas\_aprendizaje = Pipeline([('frequence',

CountVectorizer()),

                                 ('tfidf', TfidfTransformer()),

                                 ('algoritmo',

**SVC(kernel='linear', C=2))**])

Ahora hacemos el aprendizaje:

modelo = etapas\_aprendizaje.fit(X\_train,y\_train)

from sklearn.metrics import classification\_report

print(classification\_report(y\_test, modelo.predict(X\_test),

digits=4))

|  | **Precisión** | **Recall** | **F1-score** | **F1-score** |
| --- | --- | --- | --- | --- |
| 0 | 0.7514 | 0.6376 | 0.6898 | 218 |
| 1 | 0.8803 | 0.9266 | 0.9266 | 627 |
| Micro avg | 0.8521 | 0.8521 | 0.8521 | 845 |
| Macro avg | 0.8158 | 0.7821 | 0.7964 | 845 |
| Weight avg | **0.8470** | 0.8521 | 0.8479 | 845 |

Veamos ahora si podemos obtener mejores resultados optimizando el algoritmo.

Para ello, vamos a buscar el mejor valor para el parámetro C usando la función GridSearchCV, que tomará como parámetro la canalización que hemos creado previamente.

from sklearn.model\_selection import GridSearchCV

parameters = {'algoritmo\_\_C':(1,2,4,5,6,7,8,9,10,11,12)}

clf = GridSearchCV(etapas\_aprendizaje, parameters,cv=2)

clf.fit(X\_train,y\_train)

print(clf.best\_params\_)

Entonces obtenemos la siguiente respuesta:

{'algoritmo\_\_C': 1}

Si no está familiarizado con el parámetro C del algoritmo máquina de vectores de soporte, le recomendamos que vaya al capítulo Clasificar bien no es una opción para consultar las explicaciones.

Ahora probamos este parámetro nuevo:

etapas\_aprendizaje = Pipeline([('frequence',

CountVectorizer()),

                                 ('tfidf', TfidfTransformer()),

                                 ('algoritmo',

svm.SVC(kernel='linear', **C=1**))])

modelo = etapas\_aprendizaje.fit(X\_train,y\_train)

from sklearn.metrics import classification\_report

print(classification\_report(y\_test, modelo.predict(X\_test),

digits=4))

Entonces obtenemos una precisión de clasificación del 85 %.

weighted avg     **0.8507**    0.8556    0.8501       845