Máster SIANI

Recomendador de revistas científicas

Ciencia de Datos en Ingeniería

Leopoldo Lopez Reveron

Trabajo de Curso 2021/2022

Fecha de Entrega: 24 de diciembre de 2021

Objetivos del trabajo:

- La parte obligatoria se basará en la aproximación clásica que, a partir de los datos obtenidos de las revistas, se obtendrá la matriz de términos-documentos. Una vez obtenida la matriz se procederá a entrenar y validar un/os modelo/s de clasificación de documentos.

- La parte optativa se basará en la aplicación de técnicas redes neuronales para la obtención del recomendador de revistas científicas, siendo el estudiante el que proponga la solución que considere más adecuada bajo este paradigma.

Formato de entrega:

Se deberá entregar en un fichero comprimido lo siguiente:

- Memoria descriptiva.
- Código en Python generado durante la realización del trabajo.
- Conjuntos de datos utilizados en la realización del trabajo.

Memoria

Inicialmente se obtuvieron los datos de las fuentes de datos designadas, en concreto son:

0- Applied Ergonomics
1- Chemical Engineering Journal
2- Computer Vision and Image Understanding
3- Expert Systems with Applications

Quedando de igual manera asignadas las clases de cada una de las fuentes de datos. En este caso se escogió el formato de .bib.

Una vez se obtuvieron los datos se desarrollo un parseador del formato .bib, para transformar a .csv, En este cambio se tuvieron las siguientes consideraciones:

> Para evitar la problemática de tener texto plano en el contenido en un csv, y tener añadir los tokens contextuales del tipo del campo que se está leyendo, por ejemplo, el uso de " para la indicación de que se trata de un campo de tipo String y su limitación. Se utilizo el carácter "¡" debido a las características propias de los datos, ya que el tener como idioma el inglés, podemos presuponer que no se hará uso de este carácter.

Una vez resuelta la carga de los datos se procedió a separarlos en los distintos sets de train y test. Para ello se utilizó train_test_split de SKLEARN, además se utilizó el parámetro de 'stratify' para obtener un split balanceado entre las clases. Por ultimo se utilizo la semilla = 42 para obtener resultados reproducibles.

Después de obtener las distintas separaciones se procedió a entrenar los clasificadores utilizado en siguiente pipeline:

```
('vect', CountVectorizer()),
('tfidf', TfidfTransformer()),
('clf',    Clasificador())
```

Y se obtuvieron los siguientes resultados:

```
mean prediction naive bayes:                0.7058823529411765
mean prediction SVM:                        0.8627450980392157
mean prediction SVC:                        0.6862745098039216
mean prediction AdaBoostClassifier:         0.7254901960784313
mean prediction RandomForestClassifier:  0.6862745098039216
mean prediction DecisionTreeClassifier:     0.7450980392156863
mean prediction KNeighborsClassifier:       0.9411764705882353
mean prediction MLPClassifier:              0.8627450980392157
```

En vista de los resultados se procedió a revisar las métricas de los clasificadores, teniendo un caso interesante en SVM:

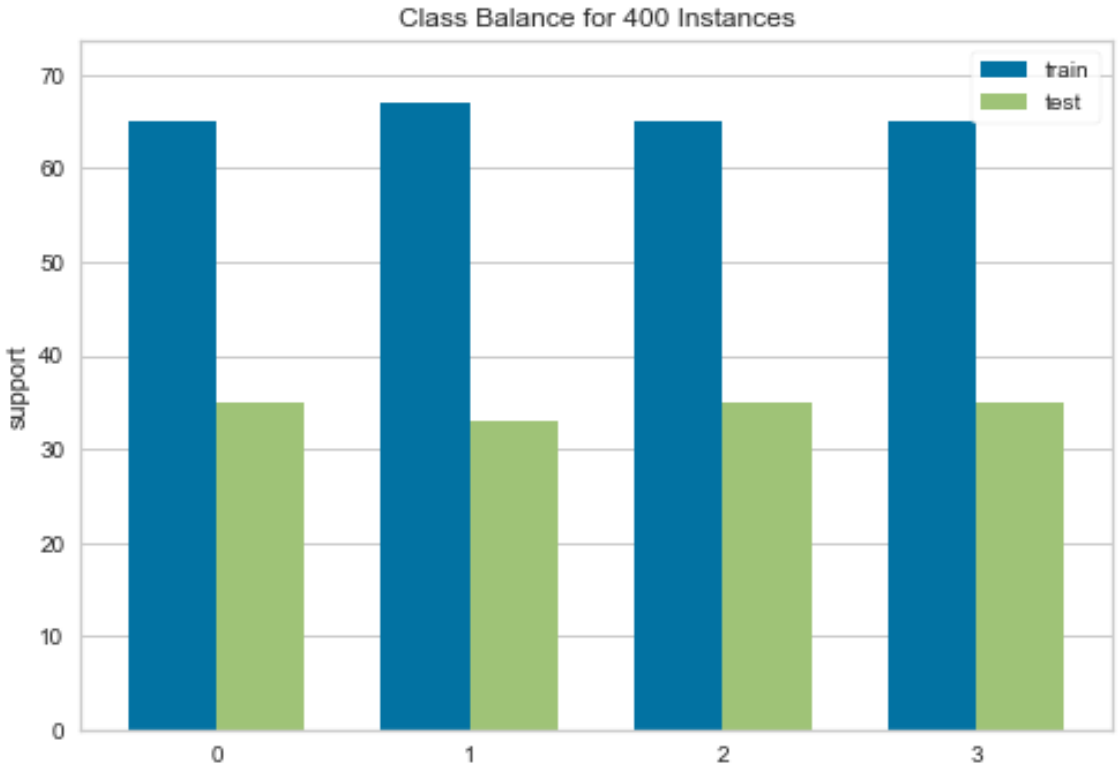
```
SVM:
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1
1	0.97	1.00	0.99	35
2	0.00	0.00	0.00	3
3	0.79	0.92	0.85	12
accuracy			0.92	51
macro avg	0.69	0.73	0.71	51
weighted avg	0.87	0.92	0.90	51

Como podemos observar en la clase 2 el clasificador obtiene un 0. Por lo

que se procedió a realizar un análisis exploratorio de los datos.

En este caso vemos como las clases de los datos están desbalanceados por lo que se procedió a balancearlos, obteniendo el siguiente resultado:



Para este balance se realizaron dos Upsampling diferentes, uno para la test y otro para el train, evitando de esta forma que datos aumentados cayeran en ambas particiones falseando los resultados.

Una vez realizado este ajuste se volvió a entrenar los clasificadores obteniendo los siguientes resultados:

mean prediction naive bayes:	0.8043478260869565
mean prediction SVM:	0.5869565217391305
mean prediction SVC:	0.2391304347826087
mean prediction AdaBoostClassifier:	0.3333333333333333
mean prediction RandomForestClassifier:	0.37681159420289856
mean prediction DecisionTreeClassifier:	0.427536231884058
mean prediction KNeighborsClassifier:	0.7391304347826086
mean prediction MLPClassifier:	0.4492753623188406

A partir de estos resultados se volvió a comprobar las métricas de entrenamiento en esta ocasión si se obtuvieron resultados satisfactorios

naive bayes:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	35
1	1.00	0.94	0.97	33
2	0.73	0.46	0.56	35
3	0.58	0.83	0.68	35
accuracy			0.80	138
macro avg	0.83	0.81	0.80	138
weighted avg	0.82	0.80	0.80	138

Como podemos ver ahora todas las clases tienen su correspondiente precisión, en este caso concreto de naive bayes vemos que su debilidad es la clasificación de la clase 3, lo cual seguramente pueda ser compensado por otro clasificador distinto, pudiéndose incluso crear un clasificador fuerte compuesto de clasificadores débiles.

En conclusión, la extracción de datos y su correcta interpretación será la mayor carga del trabajo, debido a sus características estocásticas, los métodos de clasificación ya están ampliamente probados e implementados, dejando la carga del trabajo en la correcta aplicación de estos y su parametrización.

Además, todo el trabajo se puede encontrar en un repositorio en:

https://github.com/qwerteleven/CDI_science_journal_recomendator