Schneider Electric Hackathon

Equipo 64

Por Víctor García Huerta y Marcos Gonzalo López

Planteamiento del problema

El reto consistía en un problema de clasificación. Tras unificar el training set de sus diversas fuentes (e importando de los archivos pdf con el script *pdf_to_json.py)*, hemos observado que contenía multitud de features no numéricas de difícil interpretación. Hemos descartado las features que probablemente no podían contener información relevante para el modelo.

```
df_total.head()
```

	countryName	eprtrSectorName	EPRTRAnnexIMainActivityLabel	FacilityInspireID	facilityName	City	targetRelease	pollutant
0	Germany	Mineral industry	Installations for the production of cement cli...	https://registry.gdi-de.org/id/de.ni.mu/062217...	Holcim (Deutschland) GmbH Werk Höver	Sehnde	AIR	Carbon dioxide (CO2)
1	Italy	Mineral industry	Installations for the production of cement cli...	IT.CAED/240602021.FACILITY	Stabilimento di Tavernola Bergamasca	TAVERNOLA BERGAMASCA	AIR	Nitrogen oxides (NOX)
2	Spain	Waste and wastewater management	Landfills (excluding landfills of inert waste ...	ES.CAED/001966000.FACILITY	COMPLEJO MEDIOAMBIENTAL DE ZURITA	PUERTO DEL ROSARIO	AIR	Methane (CH4)

```
df_total = df_total.drop(['facilityName','FacilityInspireID','targetRelease',
                          'CONTINENT','REPORTER NAME','City','CITY ID','DAY',''],axis=1)
```

Procesado de datos

Hemos obtenido ‘dummy variables’ para las features *'countryName','reportingYear','MONTH','EPRTRAnnexIMainActivityCode'* y *'EPRTRSectorCode'.* Además, para las dos últimas, hemos completado los valores nulos que aparecían en los archivos csv del training set. Las features *SectorCode* y *ActivityCode* estaban correlacionados con *pollutant*, lo que reforzó la idea de tomar dummy variables.

df_total_dummies

	pollutant	max_wind_speed	avg_wind_speed	min_wind_speed	max_temp	avg_temp	min_temp	DAY WITH FOGS	countryName_Belgium	countryName_Bulgaria
0	1	15.118767	14.312541	21.419106	2.864895	4.924169	9.688206	2	0	0
1	0	19.661550	19.368166	21.756389	5.462839	7.864403	12.023521	1	0	0
2	2	12.729453	14.701985	17.103930	1.511201	4.233438	8.632193	2	0	0
3	0	11.856417	16.122584	17.537184	10.970301	10.298348	15.179215	0	0	0

Correlaciones

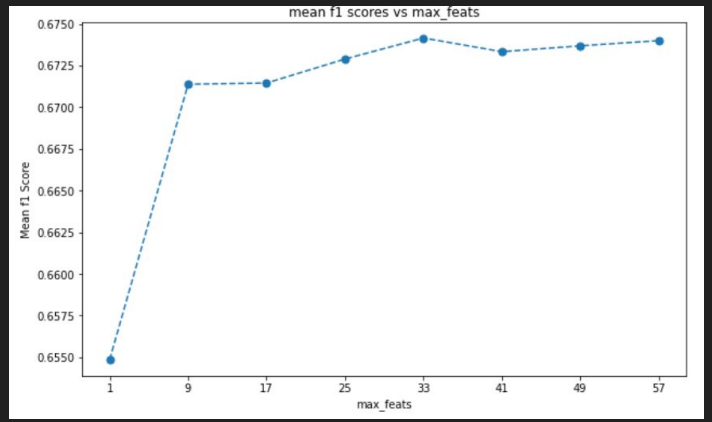
	pollutant_0	code_0
0	0.324325	5(d)
1	0.175109	3(e)
2	0.111929	7(a)(ii)
3	0.105354	1(c)
4	0.078349	6(b)
5	0.075095	5(c)

Modelo

El modelo con mejor desempeño ha sido el Random Forest. Hemos realizado un split en el training set para evaluar su rendimiento antes de hacer las predicciones. El f1-score tiene un valor decente, aunque no brillante.

Hemos tratado de optimizar el hiperparámetro *max_feats*, pero no hemos apreciado una tendencia clara, por lo que lo hemos dejado por defecto.

	precision	recall	f1-score	support
0	0.608	0.671	0.638	7800
1	0.586	0.550	0.567	6899
2	0.931	0.859	0.893	4990
accuracy			0.676	19689
macro avg	0.708	0.693	0.700	19689
weighted avg	0.682	0.676	0.678	19689



Otros vectores de ataque y futuros pasos

	precision	recall	f1-score	support
(CO2)	0.66	0.56	0.61	8175
(CH4)	0.82	0.94	0.87	4405
(NOX)	0.57	0.62	0.60	7154
accuracy			0.67	19734
macro avg	0.41	0.42	0.42	19734
weighted avg	0.66	0.67	0.66	19734

En paralelo al modelo principal, exploramos la idea de usar NLP para extraer información de los campos de texto. Usamos la librería Sparknlp en un docker y encodeamos los campos eprtrSectorName y EPRTRAnnexIMainActivityLabel gracias a un modelo BERT preentrenado (ver trainnlp.py) para posteriormente realizar una clasificación con uno del los clasificadores DL que nos ofrece la librería. Quedó pendiente un mayor preprocesado de las sentencias (tokenización)

Este modelo resultó devolver unos scores f1 de similar, aunque peores. Sin embargo, dado mas tiempo hubiéramos estudiado componer un ensemble de los dos modelos de tipo stacking. Suponiendo que la diferencia de los modelos y los datos de entreno nos permitiría mejorar el score de los modelos por separado. Esto queda pendiente.