†

BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

Sistemas de Apoyo a la Decisión

Informe Modo Tándem

Departamento:
Lenguajes y Sistemas Informáticos

Titulación:
Grado en Informática de Gestión y Sistemas de Información

3º Curso (2º Cuatrimestre)

24 de Abril de 2023

Alvaro Velasco Prieto
Alberto Arostegui García
Unai Bermudez Osaba
David Elorza Gabilondo

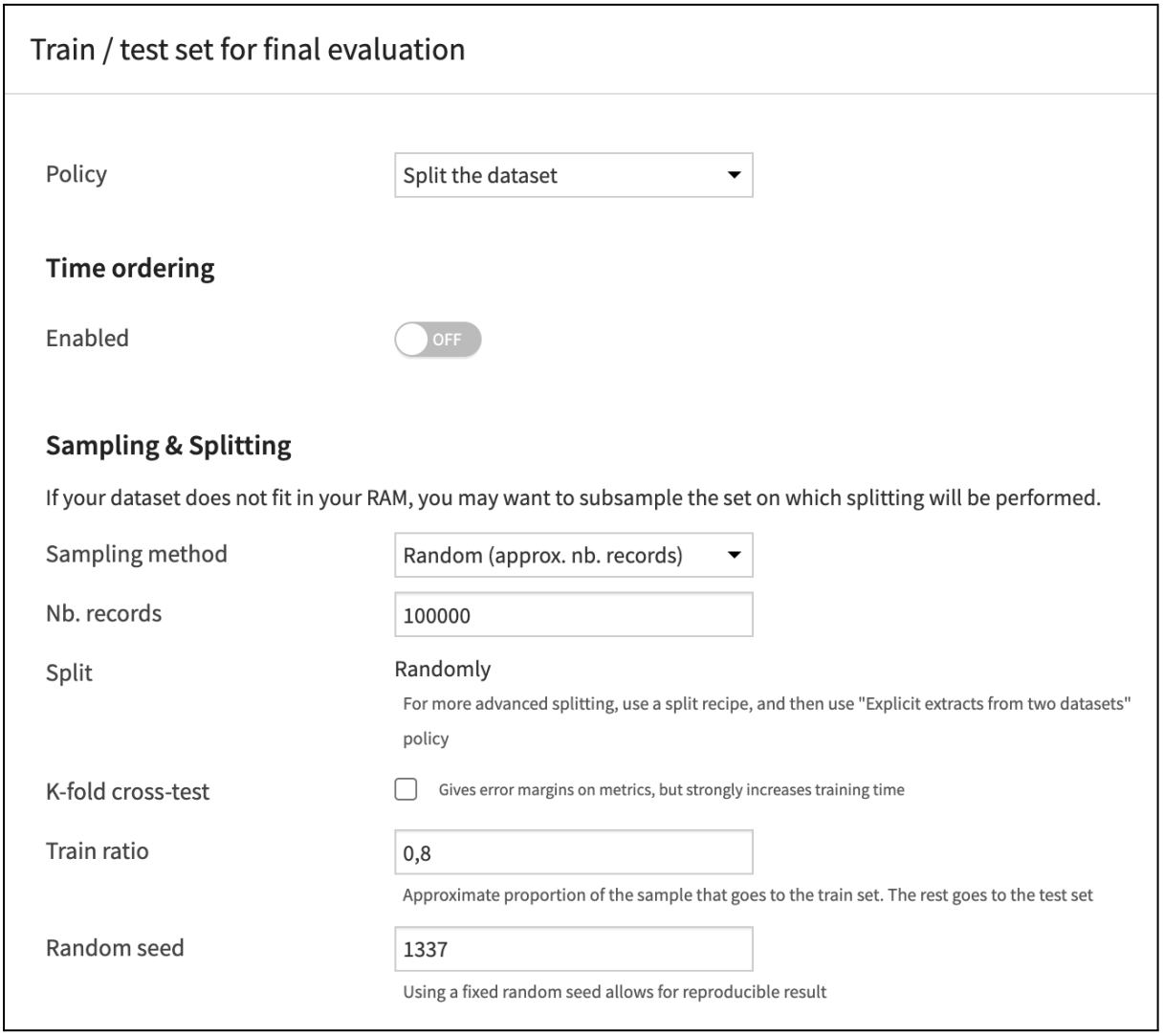
Índice

Indice de figuras

1. Datos: Análisis y Preproceso

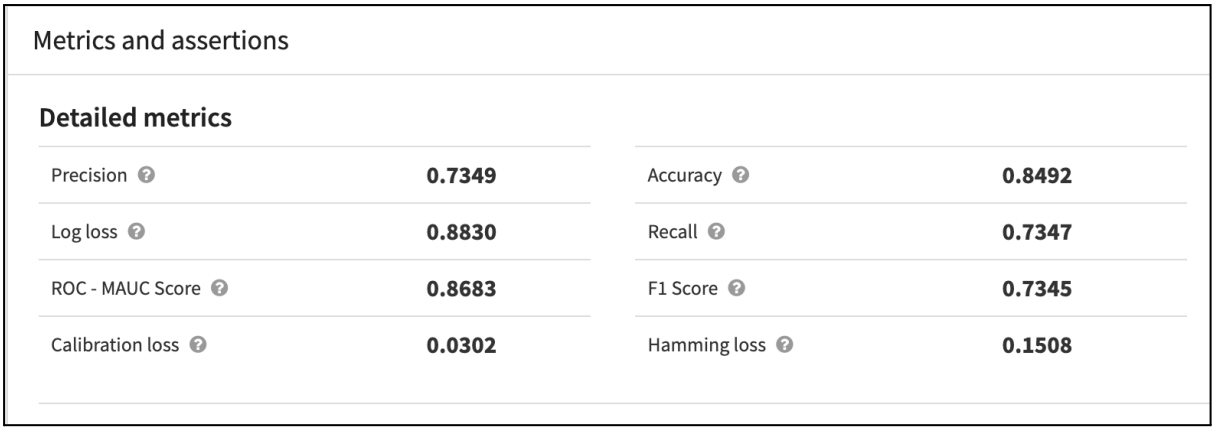
1.1. Baseline (Dataiku)

Previo al desarrollo completo de nuestro modelo de predicción, hemos utilizado Dataiku DataScienceStudio para hacer una primera aproximación, utilizando un modelo KNN.



1. Imagen: Train / test set for final evaluation (Dataiku)

Utilizando una randomización de las filas que utilizamos.



2. Imagen: Metrics and assertion (Dataiku)

Primeras métricas que obtenemos, vemos que obtenemos un F1-score de 0,7345, este será nuestro punto de partida.

1.2. División entre Train y Dev

Conjunto De Datos	% de instancias	Número de instancias
Train	80	9599
Dev	20	2400

3. Tabla: División Train y Dev

1.3. Distribución de las clases en cada conjunto

Conjunto de datos	Clase Negativa	Clase Neutra	Clase Positiva
Train	5827	2134	1638
Dev	1457	530	413

4. Tabla: Distribución Train y Dev

1.4. Descripción del preproceso

Después de dividir nuestro conjunto de datos en train y dev, el siguiente paso es el preproceso de los datos para poder obtener una mejor predicción. Para eso utilizamos el método .normalize() que hace los siguientes pre procesos a la columna “text” de nuestro conjunto de datos:

- demojize(): cambiar los emoticonos por su equivalente en texto, por ejemplo cambiar “👍” por “thumbs_up”.
- remove_non_ascii(): quitar todas las letras o símbolos que no estén en formato ascii.
- to_lowercase(): convertir todo el texto a minúsculas.
- remove_punctuation(): quitar todo tipo de signos de puntuación.

- replace_numbres(): sustituir los números por su equivalente en texto (5 = five).
- remove_stopword(): eliminar palabras vacías que no aportan al texto.
- lemmatize_verbs(): reducir los verbos a su forma no compuesta.

1.5. Primeros resultados

Los primeros resultados se obtuvieron llevando a cabo el mínimo preproceso indispensable para poder entrenar el modelo, es decir, procesó el texto para conseguir vectorizarlo mediante el TfidfVectorizer(), los resultados obtenidos tras esta primera prueba fueron los siguientes:

Algoritmo	Combinación de hiperparametros	Precisión	Recall	F-Score
Multinomial Naive Bayes	-	0,7161	0,98	0,8275
Decision Tree	Profundidad = 10 , minSLeaf=1 , minSSplit=2	0,6835	0,9279	0,7871

5. Tabla: resultados primitivos

Estos resultados superan el baseline establecido con dataiku, no obstante, como se detalla más adelante, se trata de mejorar los resultados mediante el preproceso de los datos y la incorporación de nuevos hiperparámetros extraídos de los metadatos de los tweets.

1.6. Descripción del Proceso de Submuestreo o Sobremuestreo

Uso de oversampling

Sin oversampling

	precision	recall	f1-score	support
negative	0.74	0.96	0.84	1457
neutral	0.69	0.38	0.49	530
positive	0.86	0.44	0.58	413
accuracy			0.74	2400
macro avg	0.76	0.59	0.63	2400
weighted avg	0.75	0.74	0.71	2400

```
Without using oversampling

F1-Score:                0.8351713859910581
Precision score:         0.738145416227608
Recall score:            0.9615648592999314
```

Con oversampling

	precision	recall	f1-score	support
negative	0.70	0.77	0.73	1457
neutral	0.65	0.64	0.65	1457
positive	0.76	0.69	0.72	1457
accuracy			0.70	4371
macro avg	0.70	0.70	0.70	4371
weighted avg	0.70	0.70	0.70	4371

```
Using oversampling

F1-Score:                0.7293503101534443
Precision score:         0.6955168119551681
Recall score:            0.7666437886067261
```

No solo no mejora, sino que empeora el resultado.

2. Algoritmos, link a la documentación y nombre de los hiperparámetros empleados

2.1. Algoritmos empleados: Breve Descripción

Hemos empleado los siguientes algoritmos con los siguientes hiper-parámetros.
Multinomial Naive Bayes: ideal para la clasificación con atributos discretos, como el conteo de palabras en clasificación de textos.

- Hiperparámetros: alpha = True (por defecto)
- Link: Sklearn MultinomialNB
- Guía de usuario

Decision Tree: consiste en la utilización de un árbol de decisión (en el que el modelo consiste en entrenar ese árbol según los parámetros utilizados) para lograr una conclusión

- Hiperparámetros: Profundidad = 18, minSLeaf = 1, minSSplit = 1/2
- Link: Sklearn DecisionTreeClassifier

2.2. Resultados sobre el Development

2.2.1. Optimizando los resultados de la clase negativa

Algoritmo	Combinación de hiperparametros	Precisión	Recall	F-Score
Multinomial Naive Bayes	Alpha = True	0,74	0,96	0,84
Decision Tree	Profundidad = 18 , minSLeaf=1 , minSSplit=2	0,73	0,96	0,83

6. Tabla: resultados optimizados sobre la clase negativa (development)

*Los datos de precisión, recall y f-score corresponden a la clase negativa

2.2.2. Discusión

Para obtener los resultados y tratar de optimizar los correspondientes a la clase negativa se han realizado diferentes pruebas y se han comparado los valores de precisión, recall y F-score que se han obtenido en cada una de ellas.

En primer lugar, se ha analizado cómo afectan los diferentes preprocesos del campo text al resultado final. Para ambos algoritmos se ha probado cada uno de los preprocesos de manera independiente y se ha comprobado que todos los preprocesos mejoran el resultado final. Además, se ha probado la diferencia entre lemmatization y stemming y se ha comprobado que la lematización ofrece un mejor resultado en este caso.

En segundo lugar, se ha realizado un escaneo de los parámetros del vectorizador TF-IDF, en concreto los parámetros max_features (Sólo tiene en cuenta las X features con

mayor valor de frecuencia), min_df (a la hora de construir el vocabulario se ignoran las palabras con frecuencia de aparición menor a este threshold) y max_df (a la hora de construir el vocabulario se ignoran las palabras que aparecen en más del X por ciento de los documentos). Tras realizar el escaneado se ha determinado que estos son los valores que ofrecen un mejor resultado: max_features=1600, min_df=10,max_df=0.8

Por otro lado, se ha comprobado si la adición de más features extraídas de los metadatos de los tweets mejoran el resultado. En este caso, se han obtenido diferentes resultados para cada algoritmo. En el caso del Multinomial Naive Bayes, la incorporación del número de retweets y la fecha de publicación del mismo ofrecen una pequeña mejora en el resultado final. En el caso del Decision Tree, esto no es así, ya que los mejores resultados se obtienen entrenando el modelo tan solo con la vectorización de los tweets.

Finalmente, en el caso del Decision Tree se ha llevado a cabo un barrido de los siguientes hiperparámetros: Profundidad (número máximo de niveles que puede tener el árbol), minSLeaf (número mínimo de instancias requeridas para ser nodo hoja) y minSSplit (número mínimo de instancias requeridas para partir un nodo interno).

2.2.3. Sin optimizar ninguna clase en particular

Algoritmo	Combinación de hiperparametros	Precisión	Recall	F-Score
Multinomial Naive Bayes	Alpha=True	0,7381	0,9615	0,8351
Decision Tree	Profundidad = 13 , minSLeaf=1 , minSSplit=2	0,7001	0,9375	0,8016

7. Tabla: resultados sin optimizar sobre una clase

2.2.4 Discusión

Para optimizar el modelo de manera global sin optimizar ninguna clase en concreto se ha empleado el mismo proceso que se ha descrito en el apartado 2.2.2.

2.3. Conclusión

El análisis de diferentes preprocesos, hiperparámetros y técnicas de muestreo son indispensables a la hora de entrenar un modelo con el que se quiere optimizar un tipo de resultado. Se debe analizar la métrica que se desea optimizar para ver cómo cada uno de los cambios afectan a dicha métrica. Al hacer pruebas se comprueba que puede diferir mucho un resultado, al utilizar oversampling, el resultado empeora considerablemente, tanto de manera global, como buscando la optimización para una clase en concreto.

3. Anexo

3.1 Llamadas para probar el clasificador

'python tweetDecisionTree.py (-hpmfa -n -o)'
h: help
p: PATH donde se encuentran el csv y el modelo
f: Nombre del archivo csv
a: algoritmo a utilizar (NaiveBayes o DecisionTree)
o: uso de oversampling si o no

'python clasificarItemsNuevos.py (-pmfh)'
m: Nombre del modelo que quieras utilizar (.sav)