Laboratorios
Practica nº2

Jose Javier Calvo Moratilla

Aplicaciones de Lingüística Computacional
Curso 2021/2022

1. Introducción

En la presente memoria se describe el trabajo realizado en la práctica 2 de la asignatura de aplicaciones de lingüística computacional, que consiste en la utilización de modelos de clasificación para la tarea de análisis de sentimientos en tuits.

En primer lugar se limpian y tokenizan los datos para obtener el vector de características utilizando diferentes metodologías de vectorización, por ello se utilizan los vectorizadores *counter, hashing y tfid.*

El primer vectorizador de tipo 'count' convierte los tuits utilizados en una matriz de recuento de tokens, el vectorizador de tipo 'hashing' los convierte en una matriz de ocurrencia de tokens y en último lugar el vectorizador de tipo 'tfid' convierte el texto de entrada en una matriz de características TF-IDF, (Frecuencia de términos y Frecuencia inversa del documento), que identifica bien las palabras que son relevantes para clasificar una frase.

Cuando se han pre procesado los datos se utiliza un lexicón para añadir al vector de características inicial dos columnas, donde se recoge el contador de positividad, negatividad, que indica de una frase dada, cuantas palabras se identifican como positivas y negativas.

Una vez diseñado el vector final de características se prueban los clasificadores svc, svc linear, gaussian, gradient boosting, sgd y knn, con diferentes parámetros relevantes en cada uno de ellos, obteniendo así el modelo de mayor rendimiento junto a sus parámetros de entrada. *average F1.*

Después de obtener el modelo de mayor rendimiento para la tarea, se prueba con el conjunto de datos de *test* y se recogen los resultados obtenidos en una tabla.

2. Experimentación

Se realiza la experimentación para cada modelo con diferentes métodos de vectorización *counter, hashing y tfid.* Dicha metodología es relevante para la obtención de buenos resultados.

Cada modelo tiene sus parámetros relevantes, por ello se prueban combinaciones diferentes, en base a las especificaciones de la librería *Scikit-Learn* para estudiar si dichos cambios afectan significativamente al rendimiento del modelo.

Se ha propuesto una variedad de modelos para que las pruebas puedan ser lo más amplias posible y explorar el funcionamiento de los modelos para la tarea concreta de la detección del sentimiento en tuits. Los modelos utilizados son importados mediante el uso de la librería *Scikit-Learn.*

Una vez probado los parámetros más significativos se utilizan para la prueba final, obteniendo la tabla de resultados finales, dónde se observa la precisión, el recall, el f1-score y el

support para cada una de las clases.

Se utiliza el entorno de *Google Colab Pro* para la ejecución de los experimentos para la realización de la práctica.

2.1. Modelo SVC

Los modelos de tipo *suport vector machines, SCV* forman parte de la familia de los algoritmos de aprendizaje automático supervisado y se utilizan para múltiples tareas, como clasificadores, para regresión y detección de *outliers* o valores atípicos.

Para la discriminación se utilizan unos puntos de entrenamiento llamados vectores de soporte, seleccionados por ser los datos que mejor definen la frontera de clasifiación entre las clases.

Para la utilización del modelo SVC se seleccionan como parámetros de experimentación los kernels {rbf, linear, poly, sigmoid}, y para cada uno de ellos el valor 'C' con el rango de [0.1-10.0].

En primer lugar se realiza la experimentación con el vectorizador 'counter' obteniendo los siguientes resultados:

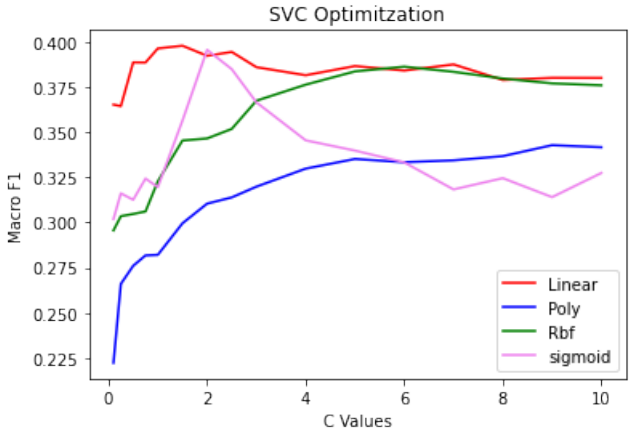

Figura 1: Resultados SVC Counter Vectorizer

Kernel	C	Macro Avg F1
linear	1.5	0.3978
poly	9.0	0.3428
rbf	6.0	0.3862
sigmoid	2.0	0.3955

En segundo lugar se realiza el experimento con el vectorizador de tipo 'hashing' obteniendo los siguientes resultados:

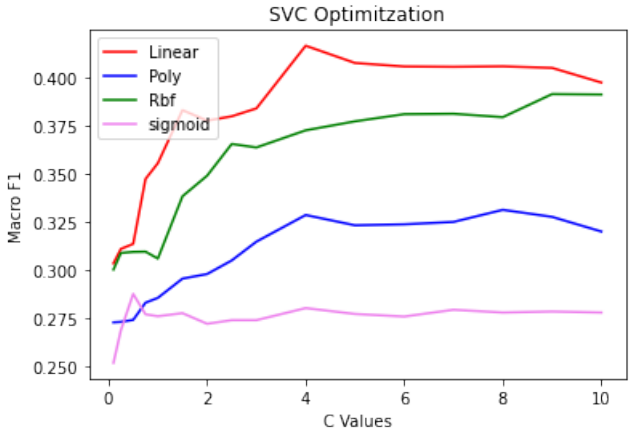

Figura 2: Resultados SVC HashingVectorizer

Kernel	C	Macro Avg F1
linear	4.0	0.4166
poly	8.0	0.3311
rbf	9.0	0.3914
sigmoid	0.5	0.2874

En último lugar se realiza el experimento con el vectorizador de tipo 'tfid' obteniendo los siguientes resultados:

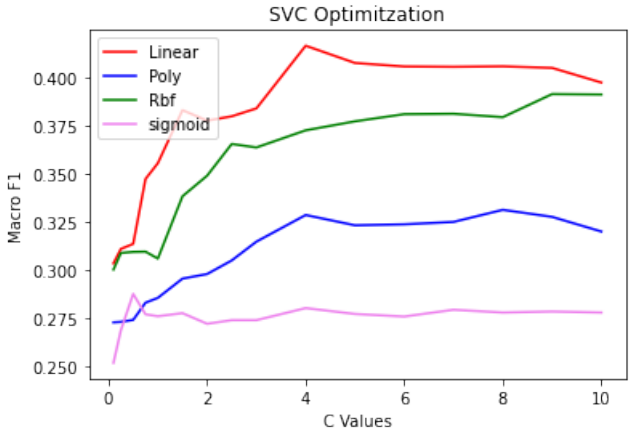

Figura 3: Resultados SVC Tfid Vectorizer

Kernel	C	Macro Avg F1
linear	2.5	0.4281
poly	6.0	0.3263
rbf	6.0	0.4081
sigmoid	5.0	0.2853

Se concluye que el mejor modelo con vectorización 'counter' es kernel 'linear', valor de C = 1.5 obteniendo un Macro Avg F1 = 0.3978

Se concluye que el mejor modelo con vectorización 'hashing' es kernel 'linear', valor de C = 4.0 obteniendo un Macro Avg F1 = 0.4166

Se concluye que el mejor modelo con vectorización 'tfid' es kernel 'linear', valor de C = 2.5 obteniendo un Macro Avg F1 = 0.4281

Mejor Modelo	Macro Avg F1
Tfid, linear, C= 2.5	0.4281

Después de analizar los resultados se concluye que el mejor modelo se basa en la vectorización 'tfid' de Kernel 'linear', con un valor de C = 2.5 obteniendo una Macro Avg F1 = 0.4281.

	precision	recall	f1-score	support
N	0.63	0.62	0.62	219
NEU	0.17	0.13	0.15	69
NONE	0.31	0.40	0.35	62
P	0.58	0.60	0.59	156
accuracy			0.52	506
macro avg	0.42	0.44	0.43	506
weighted avg	0.51	0.52	0.52	506

Figura 4: Resultados Finales Tfid, linear, C= 2.5

2.2. Modelo Linear SVC

Para la utilización del modelo Linear SVC, con kernel 'linear', pero a diferencia del modelo descrito en el punto 2.1 se implementa en término de liblinear y no en libsvm, otorgando al modelo mayor flexibilidad a la hora de elegir las penalizaciónes y las funciones de pérdida o *loss*.

Se seleccionan como parámetros de experimentación las funciones de pérdida {hinge, squared hinge}, y para cada uno de ellos el valor 'C' con el rango de [0.1-10.0].

En primer lugar se realiza la experimentación con el vectorizador 'counter'.

Al realizar los experimentos el loss de tipo 'hinge' da como resultado un vector de diferente tamaño a la longitud de los parámetros de C, por ello solo se ha podido realizar el experimento con el valor de loss de 'Squared hinge', obteniendo los siguientes resultados:

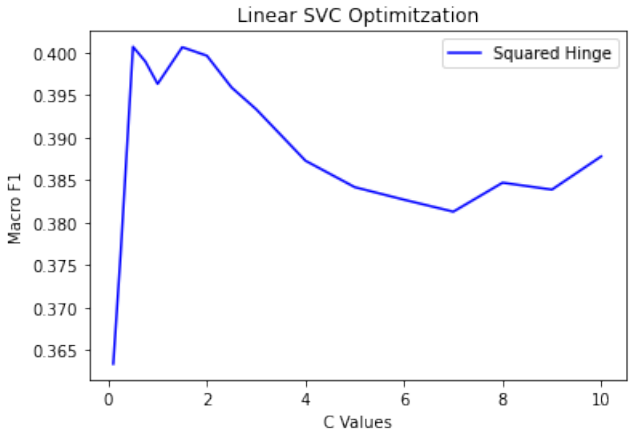

Figura 5: Resultados Linear SVC Counter Vectorizer

Loss	C	Macro Avg F1
Squared Hinge	0.5	0.4007

En segundo lugar se realiza el experimento con el vectorizador de tipo 'hashing' obteniendo los siguientes resultados:

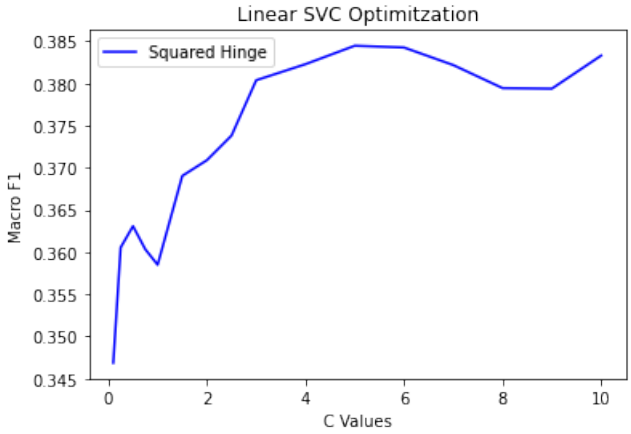

Figura 6: Resultados Linear SVC HashingVectorizer

Loss	C	Macro Avg F1
Squared Hinge	5.0	0.3845

En último lugar se realiza el experimento con el vectorizador de tipo 'tfid' obteniendo los siguientes resultados:

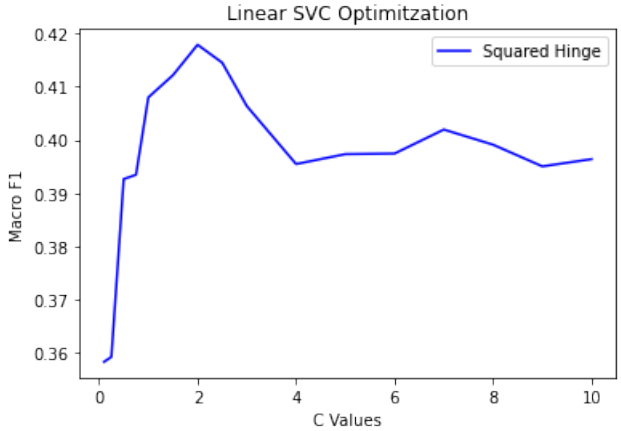

Figura 7: Resultados Linear SVC Tfid Vectorizer

Loss	C	Macro Avg F1
Squared Hinge	2.0	0.4179

Se concluye que el mejor modelo con vectorización 'counter' es loss 'Squared hinge', valor de C = 0.5 obteniendo un Macro Avg F1 = 0.4007

Se concluye que el mejor modelo con vectorización 'hashing' es loss 'Squared hinge', valor de C = 5.0 obteniendo un Macro Avg F1 = 0.3845

Se concluye que el mejor modelo con vectorización 'tfid' es loss 'Squared hinge', valor de C = 2.0. obteniendo un Macro Avg F1 = 0.4179

Mejor Modelo	Macro Avg F1
Tfid, Squared Hinge, C= 2.0	0.4179

Después de analizar los resultados se concluye que el mejor modelo basado en vectorización 'counter' con Loss = 'Squared Hinge' y un valor de C = 2.0, obteniendo una Macro Avg F1 = 0.4179

	precision	recall	f1-score	support
N	0.62	0.63	0.62	219
NEU	0.17	0.13	0.15	69
NONE	0.30	0.32	0.31	62
P	0.58	0.60	0.59	156
accuracy			0.52	506
macro avg	0.42	0.42	0.42	506
weighted avg	0.50	0.52	0.51	506

Figura 8: Resultados Finales Tfid, Squared Hinge, C= 2.0

2.3. Modelo Gaussian

El modelo Gaussian Naive Bayes también se engloba dentro de los algoritmos de aprendizaje automático supervisado y se basa en una variante de Naive Bayes que sigue una distribución normal gaussiana y permite admitir datos continuos.

El modelo asume que cada clase del modelo de clasificación sigue una distribución gaussiana y que las características son totalmente independientes.

Para la utilización del modelo Gaussian se seleccionan como parámetro de experimentación los valores de *smoothing* con el rango de [1e-14, 1.2].

En primer lugar se realiza la experimentación con el vectorizador 'counter' obteniendo los siguientes resultados:

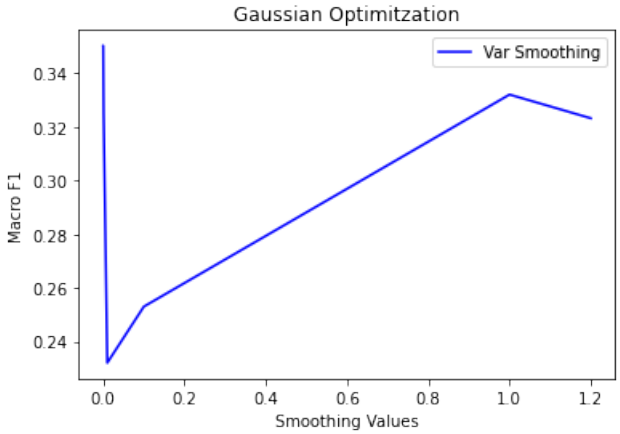

Figura 9: Resultados Gaussian Counter Vectorizer

Smoothing	Macro Avg F1
1e-14	0.3502

En segundo lugar se realiza el experimento con el vectorizador de tipo 'hashing' obteniendo los siguientes resultados:

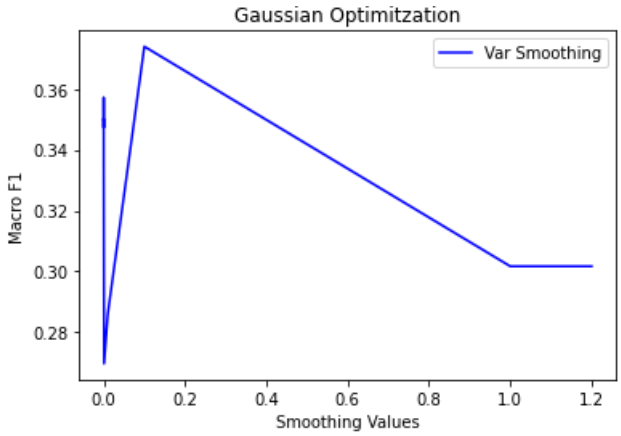

Figura 10: Resultados Gaussian Hashing Vectorizer

Smoothing	Macro Avg F1
0.1	0.374

En último lugar se realiza el experimento con el vectorizador de tipo 'tfid' obteniendo los siguientes resultados:

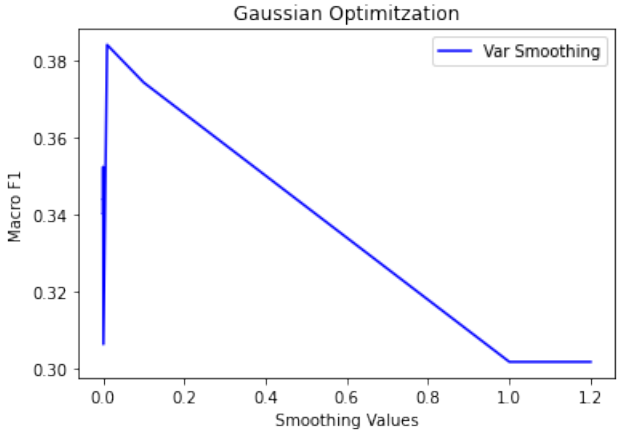

Figura 11: Resultados Gaussian Tfid Vectorizer

Smoothing	Macro Avg F1
0.01	0.384

Se concluye que el mejor modelo con vectorización 'counter' con Smoothing 1e-14 obteniendo un Macro Avg F1 = 0.3502

Se concluye que el mejor modelo con vectorización 'counter' con Smoothing 0.1 obteniendo un Macro Avg F1 = 0.374

Se concluye que el mejor modelo con vectorización 'tfid' con Smoothing 0.01 obteniendo un Macro Avg F1 = 0.384

Mejor Modelo	Macro Avg F1
Smoothing 0.01, tfid	0.384

Después de analizar los resultados se concluye que el mejor modelo se basa en vectorización 'tfid' es Smoothing = 0.01, obteniendo una Macro Avg F1 = 0.384

	precision	recall	f1-score	support
N	0.80	0.26	0.39	219
NEU	0.16	0.36	0.22	69
NONE	0.26	0.60	0.36	62
P	0.59	0.53	0.56	156
accuracy			0.40	506
macro avg	0.45	0.44	0.38	506
weighted avg	0.58	0.40	0.42	506

Figura 12: Resultados Finales Smoothing 0.01, Tfid

2.4. Modelo Gradient Boosting

El modelo Gradient Boosting o *Gradient Boosted Decision Trees (GBDT)* se utiliza generalmente en tareas de clasificación y regresión, por ello se decide probar el modelo en la presente memoria.

Para la utilización del modelo Gradient Boosting se seleccionan como parámetro de experimentación los valores de *learning rate* con el rango de [1e-5, 10.0].

En primer lugar se realiza la experimentación con el vectorizador 'counter' obteniendo los siguientes resultados:

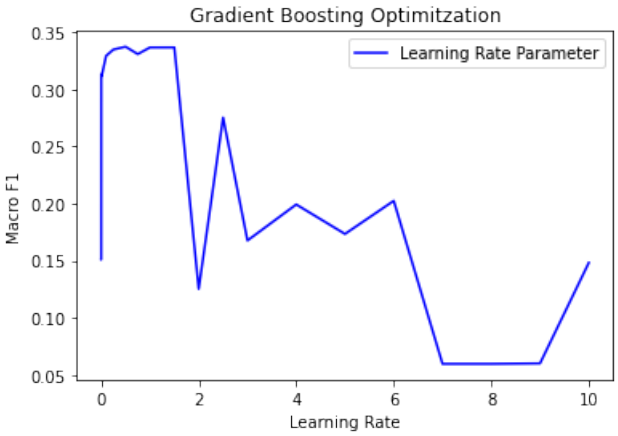

Figura 13: Resultados Gradient Boosting Counter Vectorizer

Learning Rate	Macro Avg F1
0.5	0.3372

En Segundo lugar se realiza la experimentación con el vectorizador 'hashing' obteniendo los siguientes resultados:

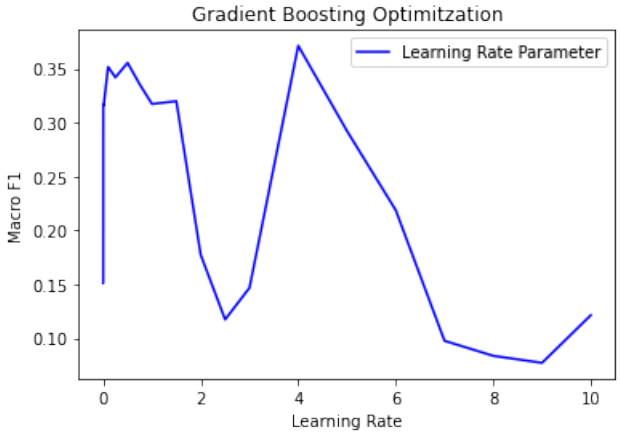

Figura 14: Resultados Gradient Boosting Hashing Vectorizer

Learning Rate	Macro Avg F1
4.0	0.3713

En último lugar se realiza la experimentación con el vectorizador 'tfid' obteniendo los siguientes resultados:

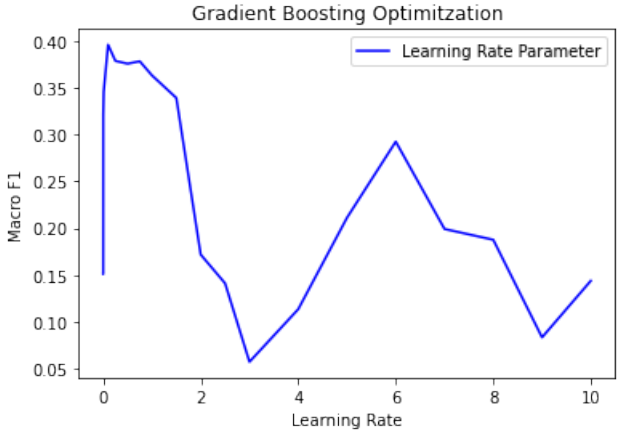

Figura 15: Resultados Gradient Boosting Tfid Vectorizer

Learning Rate	Macro Avg F1
0.1	0.3956

Se concluye que el mejor modelo con vectorización 'counter' es Learning Rate = 0.5 obteniendo un Macro Avg F1 = 0.3372

Se concluye que el mejor modelo con vectorización 'hashing' es Learning Rate = 4.0 obteniendo un Macro Avg F1 = 0.3713

Se concluye que el mejor modelo con vectorización 'tfid' es Learning Rate = 0.1 obteniendo un Macro Avg F1 = 0.3956

Mejor Modelo	Macro Avg F1
Learning Rate 0.1, Tfid	0.3956

Después de analizar los resultados se concluye que el mejor modelo se basa en vectorización 'tfid' con Learning Rate = 0.1, obteniendo una Macro Avg F1 = 0.3956

	precision	recall	f1-score	support
N	0.60	0.67	0.64	219
NEU	0.20	0.16	0.18	69
NONE	0.24	0.19	0.22	62
P	0.55	0.56	0.55	156
accuracy			0.51	506
macro avg	0.40	0.40	0.40	506
weighted avg	0.49	0.51	0.50	506

Figura 16: Resultados Finales Learning Rate 0.1, Tfid

2.5. Modelo SGD

Los modelos basados en el descenso por gradiente estocástico son entrenados con funciones de pérdida convexas, como los SVM lineales y la regresión logística.

Se utilizan ampliamente dentro del campo del aprendizaje automático para tareas de clasificación del lenguaje natural, dado que los datos disponibles son reducidos.

Para la utilización del modelo SGD se seleccionan como parámetro de experimentación los valores de *learning Rate* {constant, optimal, invscaling} y losses con {log, squared hinge, perceptrón} con el parámetro eta0 = 1e-4, seleccionado después de la realización de pruebas aleatorias.

En primer lugar se realiza la experimentación con el vectorizador 'counter' obteniendo los siguientes resultados:

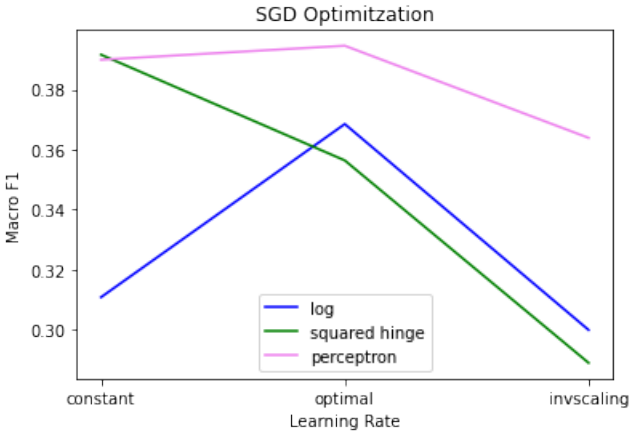

Figura 17: Resultados SGD Counter Vectorizer

loss	learning rates	Macro Avg F1
log	optimal	0.3978
squared hinge	constant	0.3917
perceptron	constant	0.4053

En segundo lugar se realiza la experimentación con el vectorizador 'hashing' obteniendo los siguientes resultados:

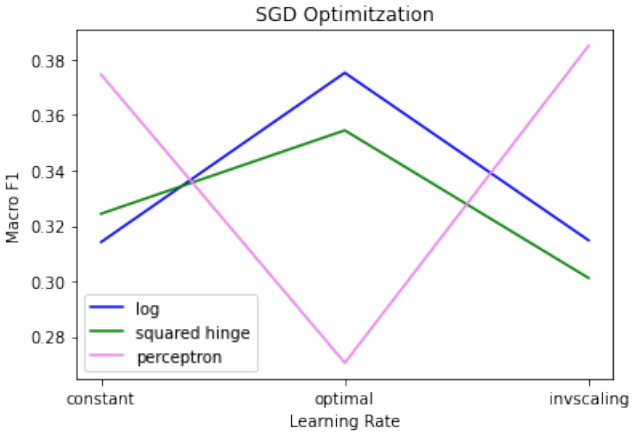

Figura 18: Resultados SGD Hashing Vectorizer

loss	learning rates	Macro Avg F1
log	optimal	0.3753
squared hinge	optimal	0.3545
perceptron	invscaling	0.385

En último lugar se realiza la experimentación con el vectorizador 'tfid' obteniendo los siguientes resultados:

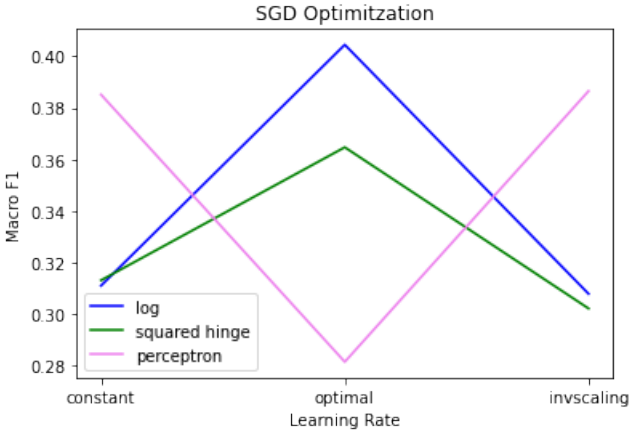

Figura 19: Resultados SGD Tfid Vectorizer

loss	learning rates	Macro Avg F1
log	optimal	0.4044
squared hinge	optimal	0.3647
perceptron	invscaling	0.3865

Se concluye que el mejor modelo con vectorización 'counter' es loss = perceptron y learning rate = constant obteniendo un Macro Avg F1 = 0.4053

Se concluye que el mejor modelo con vectorización 'hashing' es loss = perceptron y learning rate = invscaling obteniendo un Macro Avg F1 = 0.385

Se concluye que el mejor modelo con vectorización 'tfid' es loss = log y learning rate = optimal obteniendo un Macro Avg F1 = 0.4044

Mejor Modelo	Macro Avg F1
loss = Perceptron, LR = constant, counter	0.4053

Después de analizar los resultados se concluye que el mejor modelo se basa en la vectorización 'counter' con loss 'perceptrón' y con un Learning Rate 'constant', obteniendo una Macro Avg F1 = 0.4053

	precision	recall	f1-score	support
N	0.60	0.63	0.61	219
NEU	0.14	0.10	0.12	69
NONE	0.25	0.39	0.30	62
P	0.61	0.51	0.55	156
accuracy			0.49	506
macro avg	0.40	0.41	0.40	506
weighted avg	0.50	0.49	0.49	506

Figura 20: Resultados Finales loss 'Perceptrón' LR 'constant', Counter

2.6. Modelo K-NN

Los modelos de tipo K-NN se utilizan para tareas de clasificación dónde los propios datos se identifican en base al 'voto' mayoritario de los datos más cercanos en su representación en el espacio de dimensiones.

Es uno de los métodos de clasificación clásica más utilizados cuando las dimensiones son más pequeñas, dado que a grandes dimensiones se producen problemas de dimensionalidad o *curse of dimensionality*.

Para la utilización del modelo KNN se seleccionan como parámetro de experimentación los valores de *algoritmos* {auto, ball tree, kd_tree, brute} y vecinos más cercanos con rango [1,10].

En primer lugar se realiza la experimentación con el vectorizador 'counter' obteniendo los siguientes resultados:

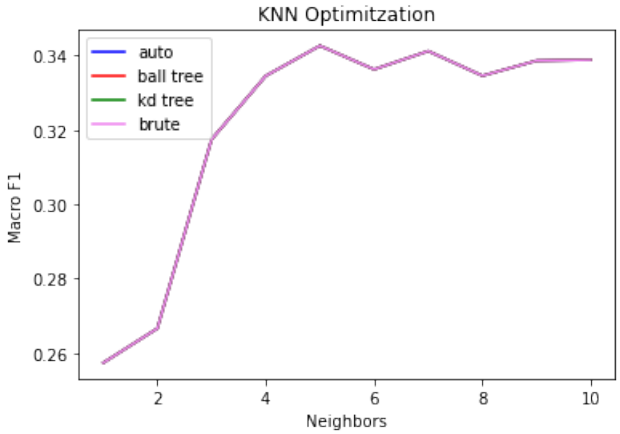

Figura 21: Resultados KNN Counter Vectorizer

Algoritmo	Vecinos	Macro Avg F1
auto	5	0.3426
ball tree	5	0.3426
kd tree	5	0.3426
brute	5	0.3426

En primer lugar se realiza la experimentación con el vectorizador 'hashing' obteniendo los siguientes resultados:

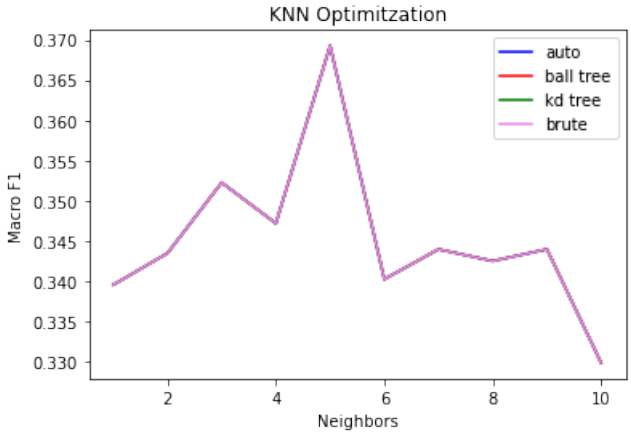

Figura 22: Resultados KNN Hashing Vectorizer

Algoritmo	Vecinos	Macro Avg F1
auto	5	0.3693
ball tree	5	0.3693
kd tree	5	0.3693
brute	5	0.3693

En último lugar se realiza la experimentación con el vectorizador 'Tfid' obteniendo los siguientes resultados:

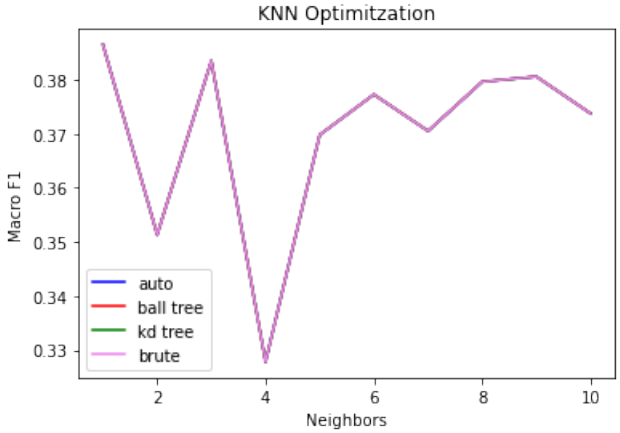

Figura 23: Resultados KNN Tfid Vectorizer

Algoritmo	Vecinos	Macro Avg F1
auto	1	0.3865
ball tree	1	0.3865
kd tree	1	0.3865
brute	1	0.3865

Se concluye que el mejor modelo con vectorización 'counter' es cualquier algoritmo utilizado y 5 vecinos más próximos, obteniendo un Macro Avg F1 = 0.3426

Se concluye que el mejor modelo con vectorización 'hashing' es cualquier algoritmo utilizado y 5 vecinos más próximos, obteniendo un Macro Avg F1 = 0.3693

Se concluye que el mejor modelo con vectorización 'Tfid' es cualquier algoritmo utilizado y 1 vecinos más próximos, obteniendo un Macro Avg F1 = 0.3865

Mejor Modelo	Macro Avg F1
Todos los algoritmos, Tfid, 1 vecino	0.3865

Después de analizar los resultados se concluye que el mejor modelo se basa en vectorización 'Tfid' con cualquier algoritmo y 1 vecino más cercano, obteniendo una Macro Avg F1 = 0.3865

	precision	recall	f1-score	support
N	0.61	0.53	0.56	219
NEU	0.17	0.14	0.16	69
NONE	0.23	0.37	0.28	62
P	0.54	0.54	0.54	156
accuracy			0.46	506
macro avg	0.39	0.40	0.39	506
weighted avg	0.48	0.46	0.47	506

Figura 24: Resultados Finales Todos los algoritmos, Tfid, 1 vecino

2.7. Resultados experimentación

Una vez realizado todos los experimentos se recoge el orden de todos los mejores modelos en orden descendente, seleccionando el mejor para la realización de la evaluación final, con sus parámetros correspondientes

Mejores Modelos	Macro Avg F1
SVC, Tfid, Linear, C = 2.5	0.4281
Linear SVC, Tfid, Squared Hinge, C=2.0	0.4179
SGD, Counter, Perceptron, constant	0.4053
Gradient Boosting, Tfid, LR = 0.1	0.3956
KNN, Tfid, 1NN	0.3865
Gaussian, Tfid, Smoothing = 0.01	0.384

Por ello se decide utilizar el modelo *SVC, Tfid, Linear, C = 2.5* para la evaluación final.

3. Conclusiones

Después de haber realizado la experimentación se ha observado que no hay un criterio fijo para la utilización de cada tipo de clasificador, ya que se han combinado diferentes parámetros, obteniendo resultados diferentes.

La vectorización de tipo 'Tfid' demuestra que es la que funciona en la tarea de análisis de sentimiento, en la mayoría de modelos utilizados, como se observa en la tabla final de resultados. Se utiliza mayoritariamente en todos los sistemas de recuperación de la información *Retrieval Information*, ya que es capaz de identificar las palabras más relevantes, que identifican a un texto, penalizando las palabras que aparecen en todos, no relevantes para la clasificación.

Si se observa la vectorización de tipo 'Counter' ha funcionado de manera óptima para los modelos de tipo SGD, Perceptrón y ocupa el tercer puesto en los resultados de la experimentación, ocupando un lugar junto al resto de modelos que han mostrado un buen rendimiento con el vectorizador de tipo 'Tfid.

La vectorización de tipo 'hashing' ha mostrado un gasto de memória más elevado, no pudiendo realizar el experimento inicialmente para el modelo de tipo Gaussian, pudiendo ser realizado ampliando las características de la cuenta de textitGoogle Colab, que la cuenta básica cuenta con limitaciones, por ello con la utilización de una cuenta Pro se ha podido ejecutar. En el caso del *Gradient Boosting* el problema viene dado por el tiempo de ejecución del experimento.

Se ha observado que los algoritmos de cálculo de los K-NN vecinos dan un resultado similar para un número de K vecinos dado y que para la tarea del laboratorio el mejor resultado se obtiene con 1 vecino más cercano.