NOMBRE: Bernardita Rosas

SECCIÓN:

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2613 — Inteligencia Artificial — 2' 2023

Tarea 4 – Respuesta Pregunta 1

1.1.1 La etiqueta a utilizar es el sentiment, el cual puede ser positivo o negativo.

1.1.2 Notamos que hay 50.000 filas de datos, tanto los datos de la columna review como sentiment corresponden a strings (objects en pandas como se puede ver al usar data.dtypes). Luego, pareciera que tanto positive como negative tienen la misma cantidad de datos (25.000 c/u) con su etiqueta, aunque cabe notar que solo hay 49582 valores distintos para review, lo que indica que existen datos que se repiten (hasta 5 veces), lo que no debiese ocurrir. En el caso de las categorías, solo existen dos.

1.2.1 Como se puede ver según el método visto en la ayudantía 8, este no cuenta con valores nulos en ninguna de sus dos columnas. Es importante eliminar valores nulos ya que de lo contrario, estaremos analizando datos que se encuentran incompletos y pueden confundir al modelo a la hora de hacer predicciones.

1.2.2 Las clases se encuentran efectivamente balanceadas, sin embargo, como se mencionó anteriormente, existen varios valores de review que se repiten que es necesario eliminar para asegurarnos de que realmente esten balanceados. Para ello, comenzaremos borrando datos repetidos para evitar el sobreajuste y veremos si las clases todavía se encuentran balanceadas. Una vez se borraron las entradas duplicadas, notamos que las clases ya no se encuentran balanceadas, teniendo 24884 datos etiquetados con positive y 24698 con negative. Si las clases se encuentran desbalanceadas, entonces el modelo no tendrá suficiente información sobre la clase sub-representada y no será capaz de clasificarla correctamente, generando un sesgo hacia la clase sobre representada. Para arreglar esto, se utiliza el método 3 de 8 tacticts to combat imbalances classes (Brownlee, 2020), según el cual se borran instancias de las clases sobre-representadas, en este caso habría que borrar 186 instancias etiquetadas con positive para que queden 24698. También se podrían agregar duplicados de las clases subrepresentadas pero no elegí tal opción al hacer el modelo más susceptible a sobreajuste, además de que se cuentan con suficientes datos para que undersampling sea efectivo. Para ahora borrar las filas necesarias, seleccionaremos tanto las filas positivas como negativas y tomaremos la muestra aleatoria correspondiente de las filas pero con una semilla para siempre obtener los mismos resultados. A continuación, en base a la recomendación del enunciado, reduciremos ambas clases a 5000 instancias.

1.2.3 Según What are Stopwords in Machine Learning?(Kharwal, 2020a), las stopwords corresponden a aquellas que no aportan información para clasificar clases al ser conectores o de uso frecuente, entonces solo distraen al modelo de las palabras importantes. Es necesario eliminarlas ya que, de lo contrario, el modelo pensará que son relevante para el análisis y tomará decisiones en base a ellas. Para remover las stop words, se utilizó código de Remove Stop Words from Text in DataFrame Column (datasnips, NA).

1.3.1 Bag of words Según Bag Of Words in Machine Learning with Python (Kharwal, 2020b), la representación bag of words consiste en remover la mayor parte de la estructura del texto a analizar y solo contar cuanto aparece cada palabra.

1.3.2 TF-IDF Según Bag Of Words in Machine Learning with Python (Kharwal, 2020b), el cuenteo de palabras según bag of words es demasiado básico y podrá tomar en cuenta palabras no significativas, su alternativa es calcular la frecuencia de palabras. Entonces TF(term frequency)-IDF(inverse document frequency) resalta las palabras que son más comunes en un documento pero no en todos los documentos (es decir, no toma palabras consideradas irrelevantes). También el conteo de palabras se ajusta según el largo del documento, a escala logarítmica y con frecuencia booleana según Understanding TF-IDF for Machine Learning (Simha, 2021)

1.3.3 SBERT Según Large Language Models: SBERT — Sentence-BERT (Efimov, 2023), BERT consiste primordialmente en el uso de transformadores encodificadores que representan el significado semántico de las palabras como vectores numéricos, lo que da una ventaja significativa en machine learning que no pueden trabajar con texto pero si con vectores de vectores. Luego, SBERT corresponde a un framework para computar incrustraciones (embeddings) de frases usando un modelo BERT, lo que significa que puede comprender el significado de una palabra en relación con las palabras que la rodean, el código se aplica según SBERT: How to Use Sentence Embeddings to Solve Real-World Problems (Sen, 2023).

1.5 Explicabilidad usando Lime

De acuerdo a la documentación oficial y Explaining Text classifier outcomes using LIME (Amami, 2020), esta librería se usa para explicar los clasificadores de texto y restringir explicaciones a palabras que se encuentren presentes en el documento. En particular, Lime (Local Interpretable Model-agnostic Explanations) se utiliza para explicar las predicciones hechas por los diferentes modelos y vectorizaciones. De esta manera, podemos establecer si creemos que los criterios utilizados por el clasificador son válidos para la situación, a través de ejemplos entendibles para cualquier persona, y así confiar o no en predicciones realizadas por el modelo.

1.6 Cálculo de métricas

De acuerdo a las métricas calculadas:

Figura 1: Métricas

	Modelo	Vectorización	Accuracy	Precisión	Recall	F1 Score
0	Árbol de decisión	Bag of Words	0.7110	0.710974	0.710981	0.710977
1	Árbol de decisión	TF-IDF	0.7090	0.708984	0.708921	0.708935
2	Árbol de decisión	SBERT	0.6285	0.628471	0.628478	0.628473
3	Random Forest	Bag of Words	0.8380	0.837996	0.837964	0.837977
4	Random Forest	TF-IDF	0.8350	0.835042	0.835064	0.834999
5	Random Forest	SBERT	0.7625	0.763070	0.762241	0.762233
6	SVM	Bag of Words	0.8695	0.871116	0.869599	0.869378
7	SVM	TF-IDF	0.8720	0.873387	0.871697	0.871805
8	SVM	SBERT	0.8210	0.821274	0.820842	0.820897

Es evidente que modelo de svm con la vectorización TF-IDF fue el que obtuvo los mejores resultados al tener las mejores métricas en cada una de estas. Por su parte, el clasificador svm también fue el que mejor funcionó al obtener los mejores resultados dada una vectorización. Luego, las vectorizaciones de TF-IDF (accuracy promedio = 0.805) y bag of words (accuracy promedio = 0.806) resultaron muy similares en cuanto a rendimiento, por lo que tendría que investigarse más respecto a cual realmente es mejor que otra aunque uno esperaría que fuese TF-IDF dada su mayor consideración de factores.
Para una mejor legibilidad se calculo el promedio de precisión, recall y f1-score para ambas clases y no por separado, pero tales valores se encuentran en la ejecución de cada modelo en el informe de clasificación.

1.7. Análisis de desempeño

Explica porqué un clasificador obtuvo un mejor desempeño que el otro y porqué cierta vectorización provocó que los clasificadores obtuvieran un mejor rendimiento. Para ello, basa tu respuesta en material bibliográfico.

En cuanto a que no haya existido mayor diferencia entre TF-IDF y Bag of words, según la información bibliográfica (Saket, 2020), efectivamente bag of words debiese tener desventajas significativas, pero pareciera que en este caso no ocurrió posiblemente debido a la simplicidad de los modelos (si hubo un mayor incremento en SVM de TF-IDF) o que simplemente los datos no eran suficientes para que la ventaja fuese extremadamente evidente y por azar se dio que bag of words funcionó relativamente parecido. Otra posibilidad es que las stopwords hayan sacado gran parte de las palabras que podrían haber hecho una mayor diferencia entre bow y tf-idf. Luego, según la tesis Clasificador documental sin glosario para documentos en español (Gabarre, 2021), efectivamente SBERT, incluso al ser mejor en teoría al considerar una mayor cantidad de factores, en la práctica no siempre funciona mejor que métodos más tradicionales como tf-idf o bow. Además, es probable que no se haya elegido el modelo más correcto de SBERT para utilizarse ya que eso requeriría de una mayor investigación. En 1.8, probé un modelo diferente 'paraphrase-multilingual-mpnet-base-v2', según (Gabarre, 2021), en el cual efectivamente logré aumentar la accuracy de SVM con SBERT de 0.8210 a 0.8315, lo que demuestra que quizás la elección del modelo específico tiene un gran impacto por lo que requeriría un mayor testeo llegar a los niveles de accuracy que logran bow y tf-idf. Luego, según documentación de sbert, al ser cada tarea de clasificación distinta, no existe una vectorización que siempre sea mejor, lo que puede significar que sbert simplemente no se adecua bien a estos datos específicos.
Como conclusión en cuanto a vectorizaciones, la principal razón por la que una vectorización funcionó mejor que otra tiene plenamente que ver con los datos específicos con los que se trabaja y no hay una formula mágica que permita determinar de manera absoluta la mejor de forma no empírica.

Luego, en cuanto a los modelos, no existe ninguna sorpresa en que random forest haya funcionado mejor que los árboles de decisión, ya que según lo que se vio en clase y en bibliografía (Sharma, 2023), estos permiten una menor cohesión entre los datos que disminuye el sobreajuste y permite un mejor rendimiento a costo de tiempo de ejecución e interpretabilidad.

Ahora, en cuanto a SVM en relación a random forest, según bibliografía (Pragati, 2021), ocurre que SVM funciona mejor cuando se trata de datos simples de clasificar (sin muchas categorías) ya que resulta relativamente fácil encontrar un hiperplano que separe ambas clases por lo que SVM tiende a tener mejor rendimiento que random forest.

Como conclusión, a la hora de eligir un clasificador, es extremadamente importante tener en consideración el tipo y cantidad de datos con los que se está trabajando, así como la necesidad de rendimiento y/o interpretabilidad, para así elegir el que más se adecue a sus necesidades.

1.8 Mejoramiento de los modelos

Según bibliografía, Hashing Vectorizer puede funcionar mejor ya que tanto TF-IDF como BOW son susceptibles a la creación de diccionarios demasiados extensos, y una de las teorías de porqué falla tiene que ver con que resulta demasiado complejo. También se probará la vectorización one-hot y n-gramas que según Análisis y aplicación de técnicas de aprendizaje automático para clasificación de reseñas en redes sociales puede ser muy útil "ya que en vez de construir el vocabulario en base a palabras, se forma en base a n-gramas, lo que permite emplear el modelo de bagof-words pero dándole a las palabras cierto contexto, lo cuál mejora los resultados en clasificación de texto a costa de tiempo de cómputo."(De la Fuente, 2019). Por un lado, se utilizará también la vectorización word2vec que, según bibliografía (Sarwade, 2023, puede funcionar mejor al capturar el significado semántico de las palabras, es decir, puede asociar palabras relacionadas a un contexto. Para esto, se utilizará un modelo de SVM y un modelo de regresión lineal. Por otro lado,

aumentar la cantidad de datos, también debiese ayudar a mejorar los resultados evidentemente al permitir entrena al modelo en una mayor diversidad de datos.

Se intentaran dos métodos en cuanto a clasificadores para mejorar las métricas: el clasificador XGBClassifier que goza de " facilidad de implementación, sus buenos resultados y porque está predefinido en un montón de lenguajes" (Sanz, 2023) y el ensamblaje de los mejores modelos que obtuvimos (SVM con tf-idf, SVM con bow y random forest con bow) ya que "Los métodos de ensamble de modelos o métodos combinados intentan ayudar a mejorar el rendimiento de los modelos de Machine Learning al mejorar su precisión" (AprendeIA, 2023).

Para la implementación de código, no se aumentará los datos utilizados plenamente por falta de recursos en el computador.

En cuanto a resultados, Hash Vectorizer funcionó muy mal, por lo que se descarta la teoría de que la simplicidad funcionaría mejor. Luego, no pudo ejecutarse one-hot por falta de recursos (se dejó horas y no corrió). En cuanto a modelos, a pesar de que XGboost está de moda, nuevamente pudimos comprobar, que no todo modelo es adecuado a un problema por lo que este tuvo un rendimiento inferior a SVM. Finalmente, en cuando al ensamble, este tiene el mismo nivel que SVM-tfidf por si solo, ya que pareciera que simplemente toma las mismas decisiones que este modelo que lo compone.

	Modelo	Vectorización	Accuracy	Precisión promedio clases	Recall promedio clases	F1 Score promedio clases
0	Árbol de decisión	Bag of Words	0.7110	0.710974	0.710981	0.710977
1	Árbol de decisión	TF-IDF	0.7090	0.708984	0.708921	0.708935
2	Árbol de decisión	SBERT	0.6285	0.628471	0.628478	0.628473
3	Random Forest	Bag of Words	0.8380	0.837996	0.837964	0.837977
4	Random Forest	TF-IDF	0.8350	0.835042	0.835064	0.834999
5	Random Forest	SBERT	0.7625	0.763070	0.762241	0.762233
6	SVM	Bag of Words	0.8695	0.871116	0.869599	0.869378
7	SVM	TF-IDF	0.8720	0.873387	0.871697	0.871805
8	SVM	SBERT	0.8315	0.821274	0.820842	0.820897
9	SVM	HashVectorizer	0.5415	0.541355	0.541139	0.540709
10	XGboost	Tf-idf	0.8215	0.822538	0.821207	0.821249
11	ensemble	Svm-Tf-idf, Svm-bow, forest-bow	0.8720	0.873387	0.871697	0.871805
12	Logistic regression	word2vec	0.7745	0.774594	0.774526	0.774490
13	SVM	word2vec	0.7695	0.769691	0.769538	0.769475

Figura 2: Nuevas métricas

Conclusiones

En cuanto a conclusiones, podemos observar que SVM con vectorización tf-idf prevalece sobre el resto, incluso en comparación al ensamble de los mejores modelos. Luego, la simplicidad de hash vectorizer jugó en contra y se lograron los peores rendimientos, mientras que la vectorización word2vec no funcionó tan bien, en relación a bow, SBERT o tf-idf, pero cabe mecionar que esta se utiliza más para establecer relaciones entre palabras que para problemas de calificación por lo que se puede deber a ello.

En base a la explicabilidad usando lime, podemos observar que, a través de todos los modelos, palabras como boring, bad, awful son claves para que una reseña termine siendo negativa mientras que palabras como wonderful, love, great, favorite terminan siendo clave para que un modelo determine la reseña con el sentimiento positive.

Como conclusión final, tal y como se ha mencionado a lo largo de la tarea, no existe un solo clasificador y/o vectorizador que funcione mejor para cada uno de los casos por lo que es necesario evidencia empírica para realizar las mejores predicciones posibles en base a los recursos que se tengan disponibles.

Bibliografía

·Brownlee, J. (2020). 8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset. https://machinelearningmas
to-combat-imbalanced-classes-in-your-machine-learning-dataset/.

· Kharwal, A. (2020a). What are Stopwords in Machine Learning? https://thecleverprogrammer.com/2020/10/27/what-
are-stopwords-in-machine-learning/#: :text=Stop%20words%20are%20commonly%20used,or%E2%80%9D%20and%20%E2%

· Kharwal, A. (2020b). Bag Of Words in Machine Learning with Python. Bag Of Words in Machine Learning with Python

· Datasnips. (NA). Remove Stop Words from Text in DataFrame Column. Remove Stop Words from Text in DataFrame Column.

· Simha, A. (2021). Understanding TF-IDF for Machine Learning. Understanding TF-IDF for Machine Learning.

· Efimov, V. (2023). Large Language Models: SBERT — Sentence-BERT. Large Language Models: SBERT — Sentence-BERT

· Sen, A. (2023). SBERT: How to Use Sentence Embeddings to Solve Real-World Problems. SBERT: How to Use Sentence Embeddings to Solve Real-World Problems

· Amami, M. (2020). Explaining Text classifier outcomes using LIME. Explaining Text classifier outcomes using LIME.

· Rojas, L. (2022). Una comparación empírica de algoritmos de aprendizaje automático versus aprendizaje profundo para la detección de noticias falsas en redes sociales. Una comparación empírica de algoritmos de aprendizaje automático versus aprendizaje profundo para la detección de noticias falsas en redes sociales

· Gabarre, A. (2021). Clasificador documental sin glosario para documentos en español. Clasificador documental sin glosario para documentos en español

· Sharma, A. (2023). Random Forest vs Decision Tree — Which Is Right for You? Random Forest vs Decision Tree — Which Is Right for You?

· Pragati. (2021). SVM AND RANDOM FOREST: A Case Study. SVM AND RANDOM FOREST: A Case Study

· De la Fuente, O. (2019). Análisis y aplicación de técnicas de aprendizaje automático para clasificación de reseñas en redes sociales Análisis y aplicación de técnicas de aprendizaje automático para clasificación de reseñas en redes sociales.

· Sanz, F. (2023). Cómo funciona el algoritmo XGBoost en Python. Cómo funciona el algoritmo XGBoost en Python

· AprendeIA. (2023). Métodos de ensamble de modelos/ machine learning ensemble methods. Métodos de ensamble de modelo

· Sarwade, S. (2023). NLP Simplified Part 2 – Types of Vectorization Techniques. NLP Simplified Part 2 – Types of Vectorization Techniques