Primera Entrega Proyecto

Inteligencia de Negocios

Universidad de los Andes

Sergio Pardo
Nathalia Quiroga
Juan Diego Calixto

Análisis de comentarios de películas

Descripción del problema

Para este proyecto se busca hacer un análisis de sentimientos sobre los comentarios en español de una película y se espera poder clasificarlos en 'positivo' o negativo'. El conjunto de datos suministrados es un csv con dos columnas, la primera es el comentario en sí, y la segunda es la clasificación que ya se le dio.

Entendimiento del negocio y enfoque analítico

- *Oportunidad / Problema de negocio*
  La oportunidad que se presenta para este caso es poder definir según un comentario, si se está haciendo una crítica positiva o negativa acerca de la película.

- *Enfoque analítico*
  El requerimiento desde el punto de vista del aprendizaje automático es clasificar cada comentario en una de dos categorías, positivo o negativo. Para ello se debe utilizar un enfoque de aprendizaje supervisado puesto que los datos ya vienen etiquetados con sus respectivas categorías. De esta forma, se puede entrenar con dichas etiquetas al modelo. Con el fin de lograr este objetivo se limpian los datos, se entrena el modelo y se prueba.

- *Organización y rol dentro de ella que se beneficia con la oportunidad definida*
  La organización que se beneficia con la oportunidad definida es la industria del entretenimiento, particularmente las empresas dedicadas a hacer producciones cinematográficas. El rol de los productores es quien más gana en el asunto, pues les será más fácil filtrar aquellos comentarios negativos y entender qué se podría mejorar o mirar los comentarios positivos para ver qué aspectos fueron los que más le gustaron al público y replicarlos para siguientes producciones. En general, el poder separar los comentarios positivos de los negativos permite un mejor entendimiento de que aspectos de le película fueron los más impactantes para replicarlos o no en próximas películas.

- *Técnicas y algoritmos utilizados*
  Primeramente, se utilizará el algoritmo de procesamiento de lenguaje natural Naive Bayes, puesto que es el más común para este tipo de tareas. Asimismo, se emplearán otros dos algoritmos de clasificación, Random Forest y KNN.

1. Importacion de librerias

Se cargan librerías para el preprocesamiento (spacy), para la transformación de los datos (FunctionTransformer), para vectorizar (TfidfVectorizer, CountVectorizer), los modelos de scikit learn (MultinomialNB, RandomForest, KNeighbors), las métricas, y el pipeline.

2. Carga de datos

Se cargan los datos utilizando pandas.

3. Entendimiento de los datos

Lo primero es ver la cantidad de datos que hay para entrenar el modelo, puesto que es importante comprender que entre más datos se tengan, mejor será la aproximación que se obtenga. Para este caso se tiene una cantidad considerablemente buena de datos.

```
(5000, 3)
```

Se debe revisar el tipo de datos que hay para cada columna.

```
review_es      object
sentimiento    object
```

Para rectificar que efectivamente la clasificación sea de 'positivo' y 'negativo' se revisan los valores que hay en la columna categórica.

```
negativo    2500
positivo    2500
```

También se revisa la cantidad de nulos que pueda haber en el data set para ver cuántos datos se pierden al tener que eliminarlos. Para este caso no hay nulos.

4. Creación de los pipelines para cada algoritmo

Transformadores

Lo primero que se crea es un transformador para el preprocesamiento de los datos. Debido a que estamos tratando con frases en español se quiere estandarizar todos los comentarios aplicándoles ciertos cambios. De esta forma, se pasa todo a minúsculas, se eliminan las tildes para evitar discriminaciones incorrectas por errores de ortografía, se eliminan los números, signos de puntuación y se eliminan las palabras vacías. Estas son palabras que no aportan significado al contexto, como lo son conectores, artículos y demás. Para ello se utilizó la librería spacy.

4.1 Naive Bayes (Juan Diego Calixto)

Pipeline

La creación del pipeline se hace únicamente con dos procesos.

El primero es el transformador de los datos que estandariza los comentarios como se especificó anteriormente, utilizando un TFIDFVECTORIZER. Este transformador lo que hace es tokenizar cada comentario separándolo por palabras y asignándole un puntaje a cada palabra según su número de apariciones en todos los comentarios, pero aquellas palabras que estén presentes en todos los comentarios y no sirvan para identificar si son positivos o negativos el TFIDF le baja al puntaje para que no afecten el

entrenamiento del modelo. Asimismo, se delimitó el número de palabras significantes a 1800 puesto que luego de hacer pruebas fue el número máximo que dio mejor resultado.

El segundo proceso ya es la creación del modelo utilizando Naive Bayes. Se le paso por parámetro un alpha de 1 para sumarle a todos los puntajes +1. De esta forma aquellas palabras que tenían un puntaje de 0 y que podían afectar la probabilidad, ya no sean un problema.

5.1 Ejecución y análisis Naive Bayes

Entrenamiento del modelo

Se hace una separación de los datos en entrenamiento y prueba. El X son las variables de decisión y el Y lo que se busca predecir. La separación se hace 80% para entrenamiento y 20% para prueba.

Análisis de métricas

Según las métricas podemos ver que el modelo se ajustó bastante bien a los datos, pero no hizo un sobreajuste. Si bien son un poco más altas las métricas entrenamiento no son exageradas con respecto a las del conjunto de prueba. Entonces, se puede concluir que el modelo se ajustó considerablemente bien a los datos y puede predecir con un porcentaje de acierto de al rededor del 80% la clasificación de un comentario para esa película.

```
Metricas del conjunto de entrenamiento:
Accuracy: 0.869
Precision: 0.8592023065833734
Recall: 0.8855869242199108
F1 score: 0.8721951219512195

Metricas del conjunto de prueba:
Accuracy: 0.81
Precision: 0.7963340122199593
Recall: 0.8128898128898129
F1 score: 0.8045267489711934
```

4.2 Random Forest (Sergio Pardo)

Pipeline

La creación del pipeline se hace únicamente con dos procesos.

El primero es el transformador que estandariza los comentarios con el mismo proceso que el modelo de Naive Bayes con la función 'clean_text'. Sin embargo, este utiliza el método de COUNTVECTORIZER que únicamente crea una tabla con las palabras tokenizadas y sus repeticiones a lo largo de todo el conjunto de datos, aquí no se

asignan puntajes a las palabras, únicamente se pone la cantidad de veces que aparece.

El segundo proceso ya es la creación del modelo utilizando Random Forest. Se emplea este algoritmo pues es una tarea de clasificación y por experiencia consideramos que resultaría adecuada para este proyecto.

5.2 Ejecución y análisis Random Forest

Entrenamiento del modelo

Se hace una separación de los datos en entrenamiento y prueba. El X son las variables de decisión y el Y lo que se busca predecir. La separación se hace 80% para entrenamiento y 20% para prueba.

Análisis de métricas

Según las métricas podemos ver que el modelo se ajustó bastante bien a los datos de entrenamiento al punto de generar un sobreajuste. No obstante, los resultados de evaluación son buenos (se acercan al 80%) Entonces se puede concluir que el modelo se ajustó considerablemente bien a los datos y puede predecir con un porcentaje de acierto de al rededor del 80% la clasificación de un comentario para esa película.

```
Metricas del conjunto de entrenamiento:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 score: 1.0

Metricas del conjunto de prueba:
Accuracy: 0.775
Precision: 0.7549800796812749
Recall: 0.7879417879417879
F1 score: 0.7711088504577822
```

4.2 KNN (Nathalia Quiroga)

Pipeline

La creación del pipeline se hace únicamente con dos procesos.

El primero, es el transformador que estandariza los comentarios con el mismo proceso que los dos modelos anteriores con la función 'clean_text'. Sin embargo, este al igual que Bayes utiliza el método de TFIDVECTORIZER ya que es una técnica de vectorización en el procesamiento de lenguaje natural que tiene en cuenta tanto la frecuencia de ocurrencia de una palabra en un documento como su frecuencia de ocurrencia en todo el conjunto de documentos. Esto permite que las palabras que aparecen con frecuencia en un documento específico, pero raramente en el resto de los documentos tengan una puntuación más alta, lo que las hace más significativas

para la representación de ese documento. TF-IDF Vectorizer también tiene un mejor rendimiento que CountVectorizer en la tarea de clasificación de documentos, ya que las palabras únicas en un documento pueden ser más importantes para la clasificación.

El segundo proceso, como se dijo anteriormente, ya es la creación del modelo y el pipeline usando KNN, además de esto, más adelante se explica cómo se reduce la diemensionalidad del KNN para unas mejores métricas. Se usó este algoritmo ya que puede ser utilizado para la clasificación de texto, donde cada documento (o conjunto de palabras que forman una unidad de texto) se representa como un vector de características y se compara con los documentos de entrenamiento existentes. El documento se clasifica según la clase predominante de los vecinos más cercanos.

TruncatedSVD es una técnica que reduce la dimensionalidad de los datos de alta dimensión al comprimirlos en un espacio de menor dimensión, al mismo tiempo que conserva la información relevante. Esta técnica se utiliza especialmente para manejar entradas dispersas o "sparse input". En el procesamiento de lenguaje natural, TruncatedSVD se utiliza comúnmente para reducir la dimensionalidad de los vectores de características que representan palabras o documentos, como aquellos obtenidos con CountVectorizer o TfidfVectorizer. Al reducir la dimensionalidad de los datos, TruncatedSVD puede mejorar la eficiencia computacional y disminuir el tiempo de entrenamiento de los modelos de aprendizaje automático, y además puede ayudar a prevenir el sobreajuste y reducir el ruido en los datos.

Asimismo, se delimitó el número de palabras significantes a 1800 puesto que luego de hacer pruebas fue el número máximo que dio mejor resultado.

Por último, "n_components" es un parámetro que especifica el número de dimensiones que se deben mantener después de aplicar la técnica de reducción de dimensionalidad. Es decir, "n_components" determina el tamaño de la matriz resultante después de aplicar TruncatedSVD.

Este parámetro se utiliza para controlar el equilibrio entre la cantidad de información que se mantiene en los datos y la cantidad de reducción de dimensionalidad deseada.

Para escoger el "n_components" se realizó un tanteo de valores entre 5-50 para evaluar el cambio de las métricas en cada caso y así lograr el más acertado.

5.2 Ejecución y análisis KNN

Entrenamiento del modelo

Se hace una separación de los datos en entrenamiento y prueba. El X son las variables de decisión y el Y lo que se busca predecir. La separación se hace 80% para entrenamiento y 20% para prueba.

Análisis de métricas

Según las métricas podemos ver que el modelo se ajustó bien a los datos de entrenamiento y el sobreajuste no es exagerado. Sin embargo, los resultados de evaluación son considerablemente inferiores respecto a los otros modelos

implementados (no superan el 70%). Por lo que, puede afirmarse que este no es un buen modelo para resolver el problema actual

```
Metricas del conjunto de entrenamiento:
Accuracy: 0.80525
Precision: 0.7938388625592417
Recall: 0.8296186230807331
F1 score: 0.8113344635504965

Metricas del conjunto de prueba:
Accuracy: 0.691
Precision: 0.6641221374045801
Recall: 0.7234927234927235
F1 score: 0.6925373134328358
```

7. Comparación de los modelos

	Algoritmo	Accuracy	Precision	Recall	F1 score
0	Naive Bayes	0.810	0.796334	0.812890	0.804527
1	Random Forest	0.775	0.754980	0.787942	0.771109
2	KNN	0.691	0.664122	0.723493	0.692537

Conclusiones

Con el fin de comprender cual fue el algoritmo que dio mejores resultados, primero hay que entender que evalúan cada una de las métricas que se calcularon.

- *Accuracy:*
  La exactitud es una medida que indica la proporción de predicciones correctas del modelo en relación con el total de predicciones. Se calcula dividiendo el número de predicciones correctas por el número total de predicciones. Una alta exactitud generalmente indica que el modelo está prediciendo correctamente la mayoría de los ejemplos.

- *Precicion:*
  La precisión es la proporción de predicciones positivas verdaderas con respecto a todas las predicciones positivas (verdaderas y falsas). Se calcula dividiendo el número de predicciones positivas verdaderas por la suma de las predicciones positivas verdaderas y falsas. Una alta precisión indica que el modelo tiene pocos falsos positivos, es decir, pocas instancias negativas se predicen incorrectamente como positivas.

- *Recall:*
  La sensibilidad es la proporción de predicciones positivas verdaderas con respecto a todas las instancias verdaderamente positivas. Se calcula dividiendo el número de predicciones positivas verdaderas por la suma de las predicciones positivas verdaderas y falsas negativas. Una alta sensibilidad indica que el

modelo tiene pocos falsos negativos, es decir, pocas instancias positivas se predicen incorrectamente como negativas.

- *F1*
  El valor F1 es una medida que combina la precisión y la sensibilidad en una sola métrica. Se calcula como la media armónica de la precisión y la sensibilidad, y proporciona una medida equilibrada del rendimiento del modelo en términos de tanto los falsos positivos como los falsos negativos. Un valor F1 alto indica un buen equilibrio entre la precisión y la sensibilidad.

Según los resultados, el modelo que tuvo mejores métricas fue el de Naive Bayes. Empezando por la exactitud, se aprecia que empleando este algoritmo se logra predecir correctamente el 80% de los comentarios, en otras palabras, se tiene la certeza de que, para cada 5 comentarios, 4 de ellos serán correctamente clasificados como positivos o negativos. De esta manera, se les asegura a los productores de las películas con un alto grado de confiabilidad que sabrán si la crítica es buena o mala para su respectivo análisis.

Por otro lado, al observar la precisión, se nota que el 80% de las predicciones positivas realizadas por el modelo son verdaderas, en relación con el total de predicciones positivas (verdaderas y falsas). En otras palabras, de cada 100 predicciones positivas hechas por el modelo, aproximadamente 80 son correctas.

En general, al ver que todas las métricas arrojaron valores aproximados del 80%, se puede concluir que el modelo de Naive Bayes resultó bastante adecuado y sus predicciones son considerablemente buenas.