Universidad de los Andes

Ingeniería de Sistemas

Inteligencia de Negocios

Proyecto 1 – Etapa 2

Nathalia Quiroga - 202013212

Juan Diego Calixto - 202020774

Sergio Pardo - 202025720

Índice

1. Entendimiento del negocio y enfoque analítico

Oportunidad/problema 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 (Descripción del requerimiento desde el punto de vista de aprendizaje automático)	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 a utilizar	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.

2. Algoritmos implementados

Para empezar, es necesario entender que, al ser una tarea de clasificación, fue necesario emplear técnicas de aprendizaje supervisado en las que se utilicen etiquetas para que el modelo pueda corroborar si sus predicciones son correctas. Dado esto, el primer paso fue realizar un entendimiento de los datos proporcionados, en los que se revisó la estructura y calidad de los mismos; la cantidad de registros, los tipos de datos presentados y los posibles valores de las etiquetas; 'positivo' o 'negativo' en este caso. Posteriormente, se revisó la presencia datos nulos en caso de que fuera necesario realizar algún tipo de imputación o reemplazo, lo cual no fue necesario.

Ya que el objetivo consistía en el análisis del lenguaje natural, se optó por tokenizarlo, es decir, dividir cada reseña/texto en las unidades más sencillas con significado propio que lo conforman, las palabras. A grandes rasgos, se toma una palabra, y observando la etiqueta de la reseña, se pueden empezar a trazar relaciones entre determinadas palabras y un tipo de comentario negativo o positivo. Para la simplificación del modelo, se tomó la decisión de pasar el texto a minúsculas, eliminar los números, signos de puntuación y tildes. Idealmente, estas últimas no deberían eliminarse ya que pueden alterar el significado del texto en general, pero en esta situación se priorizó la simplicidad. Para tokenizar los comentarios se utilizó el TFIDVectorizer. 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. Con los tokens en mano, fue posible empezar a entrenar distintos modelos para conseguir el objetivo.

En primer lugar, se utilizó el algoritmo Naive Bayes ya que es conocido por ser utilizado en teoría de probabilidad y minería de datos particularmente. Este se basa en el teorema de Bayes, que calcula la probabilidad de un evento aleatorio dado un evento inicial y la asunción de que no existe una correlación entre la aparición las palabras, es decir, las toma como independientes entre sí. Si se toma esta hipótesis inicial y se aplica al modelo, es posible simplificar la fórmula de probabilidad condicional planteada por Bayes y reducir el costo y tiempo invertido en realizar los cálculos de incidencia de cada palabra.

En segundo lugar, se escogió el algoritmo de clasificación Random Forest por varias razones. Para empezar, ya se había utilizado para otros proyectos y pese al No Free Lunch Theorem se consideró, por su factor de aleatoriedad, que podría ser un buen algoritmo a utilizar para poder comparar su rendimiento con el de los demás. Adicional a lo anterior, parte del modelo Decision Tree, que busca inferir reglas de decisión e importancia de cada característica a partir de los datos proporcionados.

Es decir, que ejecuta muchos de estos modelos para consolidar un único árbol, lo cual le da densidad y lo aleja de ser el resultado de una simple casualidad.

Por último, se escogió el algoritmo KNN ya que puede ser utilizado para la clasificación de texto, donde cada documento (o conjuno 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. Para este caso, fue necesario reducir la dimensionalidad de los datos.

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.

3. Resultados de los algoritmos

	Algoritmo	Accuracy	Precision	Recall	F1 score
0	Naive Bayes	0.8690	0.859202	0.885587	0.872195
1	Random Forest	1.0000	1.000000	1.000000	1.000000
2	KNN	0.8085	0.801637	0.824666	0.812988

Tabla 1. Resultados de entrenamiento de los algoritmos

	Algoritmo	Accuracy	Precision	Recall	F1 score
0	Naive Bayes	0.810	0.796334	0.812890	0.804527
1	Random Forest	0.795	0.780488	0.798337	0.789311
2	KNN	0.682	0.663984	0.686071	0.674847

Tabla 2. Resultados de evaluación de los algoritmos

Como es posible observar, el algoritmo que presentó los mejores resultados en el caso de estudio fue el de Naive Bayes ya que sus resultados de entrenamiento y evaluación no difieren en gran medida entre sí y al momento de clasificar si un comentario será positivo o negativo su predicción es acertada el 80% de las veces. Más específicamente, debido al puntaje de accuracy obtenido, es posible afirmar que el 81 de verdaderos positivos y verdaderos negativos se identifican correctamente. Acudiendo a la precisión, se observa que el 79.6% de los verdaderos positivos son identificados correctamente, mientras que el % restante corresponde a comentarios negativos erróneamente clasificados como positivos (falsos positivos). El recall permite concluir que el 81.2% de los comentarios positivos fueron identificados con éxito, mientras que el restante fue erróneamente clasificado como comentarios negativos. Por último, el F1 Score nos da un balance entre la precisión y el recall, dando pie a afirmar que el 80.4% de las veces, el modelo de Naive Bayes es capaz de predecir los comentarios positivos correctamente.

Por otro lado, se encuentra el modelo KNN, que en pruebas iniciales presentaba un alto sobreajuste, por lo que se redujo su dimensionalidad y en las pruebas de entrenamiento obtuvo resultados parecidos a los resultados de evaluación del modelo entrenado con Naive Bayes. Sin embargo, en los resultados de evaluación, sus métricas se ven considerablemente afectadas cayendo por debajo del 70%, por lo que es posible afirmar que este no es un buen modelo para solucionar el problema presentado.

Finalmente, el algoritmo de random forest presentó un alto sobreajuste, mostrando una adaptación perfecta a los datos de entrenamiento, pero con resultados de evaluación inferiores al modelo Naive Bayes. Se considera la posibilidad de reducir la dimensionalidad del modelo, así como con KNN y se pondera con el riesgo de que le suceda lo mismo, un menor sobreajuste, pero resultados inferiores en general. Por lo que se llega a la conclusión de que el mejor modelo para este caso de estudio es el que se entrena con el algoritmo Naive Bayes.

Con todo esto en mente, se le recomienda a la compañía cinematográfica hacer uso de nuestro modelo desarrollado a partir del algoritmo Naive Bayes para identificar las palabras más influyentes en los comentarios tanto positivos como negativos. Esto les dará la capacidad de identificar las palabras más decisivas en la clasificación de un comentario. Dichas palabras le darán la capacidad al director de entender qué se hizo bien y qué se puede mejorar sin tener que leer cada una de las críticas que se hagan a su producción. Le será posible establecer puntos clave que deben ser cubiertos en sus producciones. De esta manera, podrá realizar películas más exitosas, que tengan mejor recibimiento y generen más ingresos a la industria que permitan financiar nuevos proyectos. Un factor importante es entonces, que la película debe ganar suficiente popularidad para poder hacer una recolección de bastantes registros que le proporcionen al modelo suficiente información acerca de la opinión del público.

4. Desarrollo de la aplicación

En el lado del servidor se utilizó Node.js para crear el back-end de la aplicación y se utilizó FastAPI para manejar las solicitudes HTTP. También se importó Pandas para transformar los datos de entrada en un formato que el modelo de Machine Learning pudiera leer. Finalmente, se cargó el archivo Joblib y se utilizó el modelo para realizar predicciones. Para esta entrega decidimos solo cargar el modelo de Naive Bayes puesto que fue el que mejores resultados presentó en la primera entrega.

Para el front-end, se crearon dos componentes en React: uno para el cuadro de texto donde el usuario ingresa el comentario de la película, y otro para el botón que desencadena la solicitud de predicción. Al hacer clic en el botón, se envía el texto del cuadro de texto al back-end para su análisis y se muestra la respuesta del modelo en la página.

En resumen, se utilizó una combinación de Python, Node.js, FastAPI y React para crear una aplicación que utiliza Machine Learning para analizar la polaridad de los comentarios de las películas. El resultado final es una aplicación web interactiva que permite a los usuarios ingresar comentarios y recibir una respuesta en tiempo real.

Además de la implementación del front-end y el back-end, se utilizó Postman para probar el funcionamiento de la API. Se enviaron solicitudes HTTP al servidor para verificar que el modelo de Machine Learning pudiera manejar diferentes tipos de comentarios de películas y que la API pudiera enviar respuestas precisas y rápidas.

Postman es una herramienta muy útil para realizar pruebas y verificar que nuestra API esté funcionando correctamente antes de integrarla con otras aplicaciones. En resumen, el uso de Postman nos permitió asegurarnos de que nuestra aplicación estaba funcionando correctamente y entregando resultados precisos antes de integrarla con el front-end y ponerla en producción.

5. Justificación

El usuario/rol de la organización que va a utilizar la aplicación es un analista de películas que necesita consultar de manera rápida y precisa los comentarios de las películas para poder obtener información valiosa sobre la percepción del público en general. La existencia de esta aplicación es de gran importancia para este rol ya que le permite obtener información eficientemente, facilitándole la toma decisiones informadas sobre la percepción del público con respecto a una película en particular.

En cuanto a la aplicación web o móvil para interactuar con el resultado del modelo, se desarrolló una interfaz de usuario que permite a los analistas ingresar un comentario sobre una película y recibir una predicción del modelo indicando si el comentario es positivo o negativo. Se trabajó en colaboración con el compañero de

estadística asignado, quien proporcionó valiosos aportes en cuanto a la interpretación de los resultados y en la mejora de la calidad del modelo.

La aplicación web resultante se ha probado y ha demostrado que puede ser una herramienta valiosa para el análisis de películas y puede mejorar significativamente la eficiencia del proceso de análisis de películas en la organización.

6. Roles por estudiantes

Líder de proyecto: Juan Diego Calixto

Líder de datos: Nathalia Quiroga

Líder de analítica: Sergio Pardo

7. Tiempos y número de horas

Backend:

Juan Diego Calixto: 5 horas

Nathalia Quiroga: 1 horas

Sergio Pardo: 5 horas

Frontend:

Juan Diego Calixto: 1 horas

Nathalia Quiroga: 5 horas

Sergio Pardo: 1 horas

Documento, presentación y organización entregables:

Juan Diego Calixto: 1 horas

Nathalia Quiroga: 1 horas

Sergio Pardo: 2 horas

8. Retos enfrentados

Desde el principio surgieron varios retos con respecto a lo desarrollado en esta etapa. Para iniciar, tokenizar las reseñas requería de varis decisiones técnicas y de negocio importantes, como qué métodos utilizar para simplificar las palabras o qué herramienta utilizar para ejecutar la tokenización según estas características. Todo

esto requirió varias sesiones de debate y revisión después de ser implementado, ya que un cambio mínimo podría afectar enormemente los resultados obtenidos.

Posteriormente, el reto surgió en la escogencia de los algoritmos a implementar en los modelos, debido a la gran variedad que se puede encontrar y a las características individuales de cada uno. Aquí se optó por implementar algunos viejos conocidos, pero ya que sus resultados no fueron los mejores, se planteó la interrogante de qué pasaría si se implementan otros algoritmos que no hayamos utilizado, pero que tal vez se adapten mejor a la situación como fue el caso de Naive Bayes.

Por último, el reto estuvo en la extracción, entendimiento y explicación de los resultados. Esto, ya que una vez ejecutado cada algoritmo era necesario no solo visualizar los resultados sino saber explicarlos, para lo que fue necesario repasar el significado de las métricas obtenidas y estudiar la manera en que se aplicarían a esta situación. Además, fue necesario proponer la manera en que esto puede ser beneficioso más allá de una mera curiosidad y presentar el caso a nuestros potenciales clientes.

Durante el proyecto, uno de los mayores retos fue la conexión entre el back-end y el front-end para mostrar la respuesta del modelo en la página web. Fue necesario asegurarnos de que la información fluyera correctamente entre las diferentes partes de la aplicación y de que la respuesta del modelo se pudiera mostrar de manera clara y concisa para el usuario final.

En conclusión, creíamos que los mayores retos estarían en la implementación de los algoritmos y la escritura de su código, pero nos estrellamos con la realidad de que los mayores retos se encuentran en la toma de decisiones, la parte teórica-conceptual, la planeación de lo que se hará y la presentación de conclusiones concretas.

9. Reflexión sobre repartición de tareas

La repartición de tareas funcionó mayormente bien para esta etapa del proyecto, ya que cada integrante cumplió con su parte y el trabajo se completó satisfactoriamente. Sin embargo, se puede mejorar la coordinación de las partes grupales mejor, debido a que en algunas ocasiones uno de los integrantes terminaba haciendo un poco más que los otros.

10. Trabajo interdisciplinario

La colaboración con nuestra compañera de estadística fue relevante para la realización de este proyecto. Desde la entrega 1 como para esta segunda entrega del proyecto que nos reunimos para discutir los detalles técnicos y establecer un plan de trabajo que nos permitiera cumplir con los objetivos del proyecto. Durante

las reuniones, ella nos brindó una valiosa ayuda en la parte estadística, ayudándome a elegir las métricas adecuadas y a seleccionar los algoritmos más apropiados para nuestro modelo de Machine Learning. Además, su aporte en la revisión de los resultados y en la interpretación de estos fue crucial para asegurarnos de que nuestro modelo estuviera funcionando correctamente.

11. Bibliografía

1. Cardellino, F. (2021) *Cómo funcionan los clasificadores Naive Bayes: con ejemplos de código de Python*. Recuperado de https://www.freecodecamp.org/espanol/news/como-funcionan-los-clasificadores-naive-bayes-con-ejemplos-de-codigo-de-python/
2. IBM (2023) *¿Qué es el algoritmo de k vecinos más cercanos?* Recuperado de https://www.ibm.com/mx-es/topics/knn#:~:text=El%20algoritmo%20de%20k%20vecinos%20m%C3%A1s%20cercanos%2C%20tambi%C3%A9n%20conocido%20como,un%20punto%20de%20datos%20individual.
3. Scikit-Learn (S.A*) sklearn.ensemnle.RandomForestClassifier*. recuperado de https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html