Proyecto 1– BI

María Paula González Escallón
Jessica A. Robles Moreno
Juan Esteban Vergara

Tabla de contenido

1. El equipo

Nuestro equipo está conformado por tres integrantes:

- María Paula González Escallón:
  - María Paula es la encargada liderar el proyecto. Se encarga de manejar las preentregas de cada miembro del equipo, asignar tareas y de resolver conflictos. También ella hace parte del liderazgo de analítica, ya que fue la encargada de verificar que la limpieza y los datos limpios fueran los adecuados para cada algoritmo y así obtener los mejores resultados.
  - Las horas de trabajo fueron 10.
  - Debido a su colaboración en el equipo y a las tareas realizadas se ha decidido que a María Paula se le asignan 35 de los 100 puntos en total.
  - María Paula desarrolló el algoritmo de Linear SVC.

- Jessica A. Robles Moreno:
  - Jessica es la líder del negocio, la encargada de verificar que se cumplan con las necesidades del negocio, que el modelo escogido sea el que mejor responde a las necesidades del negocio y de comunicarle al negocio sugerencias sobre el proyecto. También es una de las encargadas de la analítica del proyecto, revisó que los 3 modelos escogidos fueran los más eficaces para resolver el problema que plantea la empresa.
  - Las horas de trabajo fueron 10.
  - Debido a su colaboración en el equipo y a las tareas realizadas se ha decidido que a Jessica se le asignan 35 de los 100 puntos en total.

    - Jessica desarrolló el algoritmo de Naive Bayers.

- Juan Esteban Vergara
    - Juan es el encargado de liderar el manejo de datos. El hizo la limpieza de estos para asegurar que los datos iban a ser entendidos por los modelos y así dar mejores resultados.
    - Las horas trabajadas fueron 7.
    - Debido a su colaboración en el equipo y a las tareas realizadas se ha decidido que a Juan Esteban se le asignan 30 de los 100 puntos en total.
    - Juan realizó el algoritmo de SGD Classification.

Nuestro equipo realizó las siguientes reuniones:

- Reunión de lanzamiento y planeación: Esta reunión se llevó a cabo a través de zoom el día 11 de octubre del 2022. En esta reunión se definieron los roles de cada integrante, se leyó el enunciado del proyecto y se realizó la lluvia de ideas para tener la base de cómo empezar el proyecto.
- Reunión de seguimiento: Esta reunión se llevó a cabo de manera presencial el día 12 de octubre del 2022. En esta reunión, por un lado, se realizó el perfilamiento y limpieza de los datos. Por otro lado, se realizó la implementación del primer algoritmo de analítica de textos.
- Reunión de seguimiento: Esta reunión se llevó a cabo a través de zoom el día 16 de octubre del 2022. En esta reunión, se realizó la implementación de los algoritmos de analítica de textos. Además, se realizaron cambios y pruebas del funcionamiento de los algoritmos implementados.
- Reunión de seguimiento: Esta reunión se llevó a cabo a través de zoom el día 18 de octubre del 2022. En esta reunión, por un lado, se realizó el documento con toda la información solicitada en el enunciado del proyecto. Por otro lado, se realizó la presentación y el video solicitado.
- Reunión de finalización: Esta reunión se llevó a cabo a través de zoom el día 19 de octubre del 2022. En esta reunión, se verificó que todos los puntos solicitados en el enunciado estén implementados y subidos al repositorio, con enlace en la wiki, con el fin de realizar la entrega pertinente en Bloque Neón.

2. Comprensión del negocio y enfoque analítico

Oportunidad/Problema Negocio	La empresa quiere poder clasificar de una manera automatizada si una persona es suicida o no suicida en base a los comentarios encontrados en comunidades en Reddit que sufren de depresión o han intentado suicidarse
Enfoque analítico (Descripción del requerimiento desde el punto de vista de aprendizaje de máquina)	Lo que se le pide al algoritmo realizado es poder determinar si una persona es suicida o no a partir de un comentario que haga en un foro de Reddit. Para esto se entrena al

	algoritmo con comentarios etiquetados (suicidas o no suicidas) provenientes de una comunidad en Reddit de personas que sufren de depresión o han intentado suicidarse
Organización y rol dentro de ella que se beneficia con la oportunidad definida	La organización “Suicide Awareness Voices of Education” (SAVE) se beneficia del modelo y de la automatización de la clasificación que este genera ya que ayuda a determinar de una manera más rápida las personas que tienden a ser suicidas y de esta manera poder actuar con tiempo y lograr salvar más vidas. Cuando el modelo se implementé ayudará a que las “hotlines” sean menos concurridas, porque se va a ayudar a más personas suicidas antes de que estas decidan que su situación es crítica y necesiten llamar a esta línea.
Técnicas y algoritmos a utilizar	Para obtener un modelo que logre clasificar a las personas suicidas a partir de textos se va a implementar tres modelos de análisis de texto con su respectiva limpieza de datos para un mejor resultado. En la limpieza de datos se va a tokenizar y eliminar caracteres de puntuación, caracteres especiales y volver las letras todas a minúscula. Los modelos que se van a utilizar son LinearSVC, Naive Bayers y SGD Classification.

3. Entendimiento y preparación de los datos

Los datos que nos brinda la organización para generar el modelo es un csv con tres columnas y 195.700 registros. La primera columna hace referencia al número asociado al texto que no nos interesa para hacer nuestro modelo ya que no aporta información relevante. La segunda columna (Text) hace referencia al comentario hecho por una persona en la comunidad de Reddit. Finalmente, la última columna (class) asocia el comentario de Reddit a dos etiquetas, non-suicide y suicide, haciendo referencia a si el comentario demuestra o no comportamientos o intenciones suicidas.

Al revisar la calidad de estos datos nos encontramos con que ningún registro es nulo, por lo tanto, no es necesario hacer ajustes ni eliminar registros nulos. También al revisar la cantidad de etiquetas de cada clase observamos que el número de textos etiquetados como ‘non-suicide’ es de 110165, y los etiquetados con ‘suicide’ son 85535. Pese a que los datos tienden a tener más registros con textos con tendencias no suicidas, la cantidad de etiquetas suicidas esta medianamente balanceada con la otra clase, por ende, no es necesario generar registros con SMOTE.

Sin embargo, se encontró que en la columna textos hay varios comentarios que cuentan con caracteres especiales como enter, signos de puntuación y espacios. Para esto se decidió hacer una limpieza de esta columna eliminando todos estos caracteres especiales y de paso convirtiendo todas las letras a minúscula con el fin de homogenizar el texto y que el algoritmo no clasifique erróneamente palabras con significado igual por un mal entrenamiento con palabras minúsculas y mayúsculas.

Otra observación sobre los datos administrados por la organización es que hay muchos comentarios con palabras repetidas, las cuales al momento de tokenizar se ignoran. Esto debido a que para nuestros modelos se generan predicciones a partir de las palabras encontradas y su significado, no por la cantidad de veces que estas aparezcan en el comentario, ya que puede generar problemas en el algoritmo al momento de identificar que comentarios son suicidas y cuales no lo son.

Nuestros algoritmos implementaron el método CountVectorizer como método para volver las palabras números y que sean mejor entendidas por los algoritmos. Lo que hace esta función es crear vectores donde cada posición representa la cantidad de repeticiones tienen cada palabra de nuestro vocabulario. En nuestro caso debido a la limpieza hecha previamente, estos vectores van a estar rellenos de números binarios.

4. Modelado y evaluación

Para resolver la problemática de clasificación de textos planteada por la organización “Suicide Awareness Voices of Education” (SAVE) decidimos implementar los siguientes algoritmos:

- LinearSVC:

  Para este algoritmo se utilizó Linear Support Vector Classifier. Lo que hace este algoritmo es implementar un kernel lineal para encontrar el mejor hiperplano que divide y categoriza los datos dados en el entrenamiento. De esta manera, cuando ingresen datos sin clasificar ya tendrá una referencia de como categorizar los nuevos datos. (sklearn.svm.LinearSVC, s. f.)

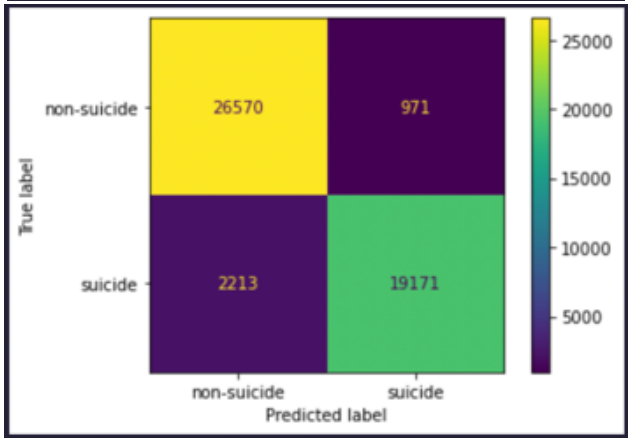
  Al momento de implementar el algoritmo utilizamos los parámetros que vienen por defecto. Entre estos se debe resaltar loss='squared_hinge', max_iter=10000 y penalty='l2'. Squared_hinge representa la información utilizada para la función loss en el algoritmo. ‘l2’ representa la penalización, y esta es utilizada ya que la mayoría de nuestros datos no son dispersos y queremos que la penalización sea constante. De manera similar, se utiliza squared_hinge ya que este es utilizado con l2 (en datos no dispersos). Aumentamos el número de iteraciones debido a que debido a la cantidad de datos se necesitan más iteraciones para obtener resultados más exactos.

  (Escala del parámetro de regularización para las SVC — documentación de scikit-learn - 0.24.1, s. f.).

(6.1.2. scikits.learn.svm.LinearSVC — scikits.learn 0.7.1 documentation, s. f.)

Al ejecutar el modelo Linear SVC, los resultados obtenidos son muy buenos. Los mejores resultados obtenidos son los relacionados al algoritmo después de hecha la limpieza, donde “Train accuracy” nos da un valor de 99.7234%, y “Test accuracy” un valor de 93.4921%. En este algoritmo se puede demostrar que la limpieza antes de hacer los modelos es importante para mejorar sus resultados, ya que sin limpieza los valores de “Train accuracy” y “Test accuracy” son de 99.7200% y 93.4410% respectivamente.

	precision	recall	f1-score	support
non-suicide	0.92	0.96	0.94	27541
suicide	0.95	0.90	0.92	21384
accuracy			0.93	48925
macro avg	0.94	0.93	0.93	48925
weighted avg	0.94	0.93	0.93	48925



Con respecto a la precision (VP/(VP+FP)) se puede observar que el porcentaje de casos positivos detectados es del 95% y el de negativos es de 92%. Esto nos indica que el algoritmo identifica mejor los casos que pueden representar comportamientos suicidas. Con recall (VP/(VP+FN)) los resultados nos informan la proporción de casos positivos que fueron correctamente identificados. En este caso para suicide es del 90% y para non-suicide del 96%, demostrando mejores resultados al identificar textos que no representan comportamientos suicidas.
En general podemos observar que este algoritmo identifica mejor los casos no suicidas que los suicidas a partir del f1-score.

- Naive Bayes:

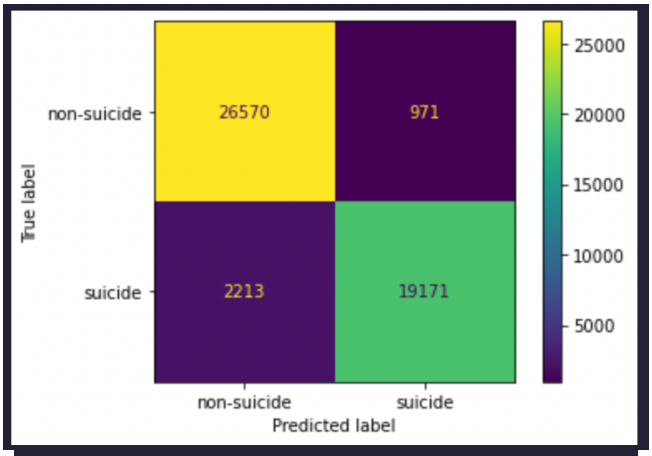
  Este algoritmo de aprendizaje supervisado de clasificación de textos utiliza el teorema de Bayes con una suposición Naive. Para esto implementamos el multinomial Bayes, este es adecuado para clasificación de variables

discretas. El modelo multinomial funciona de manera adecuada en casos de recuentos de palabras para variables de texto, tal como nos lo pide nuestro problema a resolver. (1.9. Naive Bayes, s. f.)

Para este modelo se utilizaron los parametros que vienen por defecto en la funcion MultinomialNB() de scikit-learn, las cuales son alpha = 1, fit_prior = True y class_prior = None. Alpha representa una variable en la ecuacion que puede ayudar a mejorar el resultado del algoritmo cuando no hay muchas etiquetas distintas (How does ,the Mutlinomial Bayes's alpha parameter, affects the text classification task?, 2018), fit_prior hace referencia a si el modelo aprende las clases antes de las probabilidades o no, y class_prior nos representa las probabilidades previas de las clases (sklearn.naive_bayes.MultinomialNB, s. f.)

Utilizamos dos técnicas distintas para la creación del modelo. En primer lugar, se utilizó un Pipeline con TfidfVectorizer(), que es parecido a CountVectorizer, a diferencia de que esta función convierte la colección de documentos sin procesar en una matriz de funciones TF-iDF. Al revisar la exactitud de este modelo nos da un resultado de 92,2160% para el train y 90,9351 % parta el test con limpieza. Mientras que sin utilizar el Pipeline y utilizar CountVectorizer() nos da un resultado de 93,2073% para el train y del test 92,4619% con limpieza. Podemos concluir que este modelo funciona mejor sin Pipeline, sin embargo, la diferencia no es significativamente grande.

A pesar de en los resultados del entrenamiento sean mejores sin limpieza (Con pipeline 92.2507% y sin pipeline 93.2625%) que los que ya tienen limpieza, decidimos ignorar esto para hacer la conclusión de cuál modelo es mejor. Esto porque lo que nos pide la empresa es crear un algoritmo que tenga mejores predicciones con datos nuevos, no sobre los entrenados. Como la prueba de test se hace con datos nuevos, el resultado de esta prueba es la que mejor representa el comportamiento que tendrá el modelo en producción.



	precision	recall	f1-score	support
non-suicide	0.96	0.91	0.93	27541
suicide	0.89	0.95	0.92	21384
accuracy			0.92	48925
macro avg	0.92	0.93	0.92	48925
weighted avg	0.93	0.92	0.92	48925

Al momento de revisar los resultados obtenidos por el modelo MultinomialNB la precision (VP/(VP+FP)) es mejor para non-suicide que para suicide. Con un valor de 96% y 89% respectivamente. Esto nos indica que el algoritmo identifica mejor los casos que pueden representar comportamientos no suicidas. Con recall (VP/(VP+FN)) los resultados nos informan la proporción de casos positivos y negativos que fueron correctamente identificados. En este caso para suicide es del 91% y para non-suicide del 95%, demostrando mejores resultados al identificar textos que representan comportamientos suicidas. En general nos damos cuenta de que este algoritmo es mejor identificando los casos no suicidas, ya que del f1-score es mayor para esta etiqueta con 93%, mientras que para las etiquetas de suicidio son del 92%.

- SGD Classifier:

  El método que utiliza este algoritmo para predecir es encontrar de manera iterativa la gradiente que mejor se adapta a los datos de entrenamiento mediante varias funciones matemáticas. Esto se hace movilizándose hacia el punto con mayor pendiente y así encontrar un mínimo local (Dot CSV, 2018b). Sin embargo, este mínimo local no siempre va a ser el mismo, ya que este depende del número de iteraciones y de los parámetros que se pongan en la función.

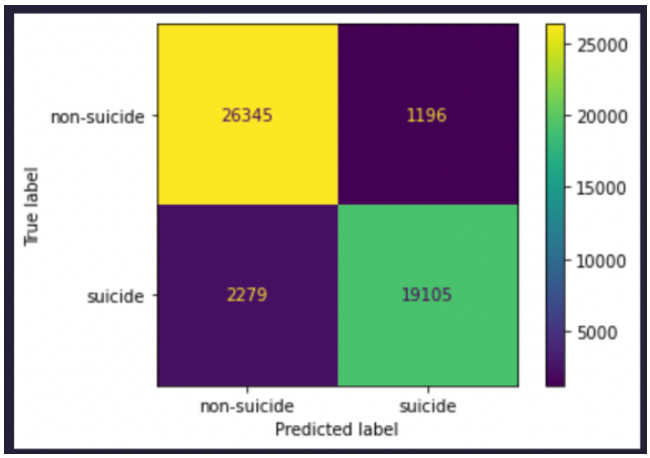
  Para este modelo algunos de los parámetros utilizados son loss='hinge', penalty='l2' y max_iter=5. De manera similar que en el modelo de Linear Support Vector Classifier el parámetro loss indica la función loss que se va a utilizar, y en nuestro caso la función ‘hinge’ nos brinda un SVM (Support vector machine), que es una version menos penalizada de squared_hinge.

Penalty hace referencia al termino de regulación que se va a utilizar, y utilizamos 'l2' ya que es el definido para los casos donde se genera un SVM. Finalmente, max_iter nos indica el número de iteraciones que se van a hacer sobre los datos de entrenamiento para generar el modelo, en nuestro caso son 5.

Utilizamos dos técnicas distintas para la creación del modelo. En primer lugar, se utilizó un Pipeline con CountVectorizer(). Nos dimos cuenta que la exactitud de este modelo daba 89.4301% para "Train accuracy" y 90.0991% para "Test accuracy", esto sin haberle hecho la limpieza a los datos. Por el otro lado, al haber hecho la limpieza, la exactitud para "Train accuracy" y para "Test accuracy" daba 89.3708% y 90.0623%, respectivamente.

En segundo lugar, no se utilizó el Pipeline, dándonos una exactitud de 93.1358% para "Train accuracy" y 92.5376% para "Test accuracy", esto sin haberle hecho la limpieza a los datos. Por el otro lado, al haber hecho la limpieza, la exactitud para "Train accuracy" y para "Test accuracy" daba 93.3020% y 92.8973%, respectivamente.

En este modelo podemos observar que se comporta mejor sin pipeline y con la limpieza de datos. Esto se puede observar ya que esta exactitud de test (92.8973%) y de entrenamiento (93.3020%) son las más altas para el algoritmo SGDClassifier.



	precision	recall	f1-score	support
non-suicide	0.92	0.96	0.94	27541
suicide	0.94	0.89	0.92	21384
accuracy			0.93	48925
macro avg	0.93	0.92	0.93	48925
weighted avg	0.93	0.93	0.93	48925

Con respecto a la precision (VP/(VP+FP)) se puede observar que el porcentaje de casos positivos detectados es del 94% y el de negativos es de 92%. Esto nos indica que el este algoritmo identifica mejor los casos que pueden representar comportamientos suicidas. Con recall (VP/(VP+FN)) los resultados nos indican la proporción de casos positivos que fueron

correctamente identificados. Para suicide es del 89% y para non-suicide del 96%, demostrando mejores resultados al identificar textos que no representan comportamientos suicidas por con una diferencia del 6%. En general podemos observar que este algoritmo identifica mejor los casos no suicidas que los suicidas a partir del f1-score ya que este valor para los no suicidas es de 94%, y el de los textos clasificados como suicidas es del 92%.

5. Resultados

Podemos concluir que los modelos realizados para la ejecución del proyecto en general dan muy buenos resultados, esto sucede ya que el f1-score es mayor a 90% para todos los algoritmos. También nos damos cuenta que el f1-score de los datos etiquetados como no suicidas, en todos los modelos, es mejor que el f1-score de los etiquetados como suicidas. Debido al contexto, la organización busca obtener una buena clasificación de las personas que tienden a tener actitudes suicidas, sin embargo, podemos observar que en los 3 casos el f1-score de los datos etiquetados como suicidas da 92%. Gracias a esto no se puede concluir solamente con este parámetro, por lo cual, decidimos que el mejor modelo para este caso es el que utiliza LinearSVC, ya que este nos da una exactitud de test de 93.4921%, el más alto de todos los modelos. Por esta razón, le recomendamos a la organización utilizar este algoritmo con el fin de clasificar mejor aquellos textos con comentarios que representen actitudes con tendencias suicidas.

En general, todos los modelos funcionan mejor con una limpieza de datos previa, además observamos que el uso de Pipelines no mejora significativamente los resultados obtenidos en los modelos. Es por esto que se le recomienda a la organización realizar una limpieza previa de los datos, que elimine signos de puntuación, caracteres especiales y palabras repetidas con técnicas de tokenizacion, también recomendamos utilizar la técnica de bag of words (CountVectorizer en scikit-learn) para que de esta manera los algoritmos funcionen mejor.

Referencias

- *sklearn.svm.LinearSVC*. (s. f.). scikit-learn. Recuperado 18 de octubre de 2022, de https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
- *1.9. Naive Bayes*. (s. f.). scikit-learn. Recuperado 18 de octubre de 2022, de https://scikit-learn.org/stable/modules/naive_bayes.html
- Dot CSV. (2018b, febrero 4). ¿Qué es el Descenso del Gradiente? Algoritmo de Inteligencia Artificial | DotCSV [Vídeo]. YouTube. Recuperado 18 de octubre de 2022, de https://www.youtube.com/watch?v=A6FiCDoz8_4
- *Escala del parámetro de regularización para las SVC — documentación de scikit-learn - 0.24.1*. (s. f.). Recuperado 19 de octubre de 2022, de

https://qu4nt.github.io/sklearn-doc-es/auto_examples/svm/plot_svm_scale_c.html

- *6.1.2. scikits.learn.svm.LinearSVC — scikits.learn 0.7.1 documentation*. (s. f.). Recuperado 18 de octubre de 2022, de https://scikit-learn.sourceforge.net/0.7/modules/generated/scikits.learn.svm.LinearSVC.html
- *How does ,the Mutlinomial Bayes's alpha parameter, affects the text classification task?* (2018, 18 abril). Data Science Stack Exchange. Recuperado 19 de octubre de 2022, de https://datascience.stackexchange.com/questions/30473/how-does-the-mutlinomial-bayess-alpha-parameter-affects-the-text-classificati
- *sklearn.naive_bayes.MultinomialNB*. (s. f.). scikit-learn. Recuperado 19 de octubre de 2022, de https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html