Septiembre 2022
Universidad de Los Andes
Jesús David Barrios 201921887
Sergio Peñuela 201922873
Jhoan Diaz 201819861

Proyecto 1 Inteligencia de negocios – Etapa 1

Comprensión del negocio y enfoque analítico

El objetivo de este proyecto es apoyar a la detección de intentos de suicidio mediante información de comunidades de Reddit que sufren de depresión o han intentado suicidarse. Esto, se realizará mediante el análisis de texto de las publicaciones de Reddit y la creación de un modelo de clasificación que permita identificar a aquellos casos de intentos de suicidio. En este sentido, se considera un resultado exitoso el obtener un modelo con mínimos errores de clasificación.

Oportunidad/problema Negocio	Necesidad de detectar intentos de suicidios por parte de usuarios de Reddit dentro de comunidades de depresión en la plataforma.
Enfoque analítico (Descripción del requerimiento desde el punto de vista de aprendizaje de máquina)	Con el objetivo de detectar los posibles intentos de suicidio, y teniendo la clasificación inicial dada por los datos de Reddit, se propone un enfoque supervisado con una tarea de clasificación. Así, se podrá saber si un individuo puede tener tendencias suicidas.
Organización y rol dentro de ella que se beneficia con la oportunidad definida	Departamentos de investigación de organizaciones sociales dedicadas a la salud mental y prevención de suicidios pueden estar interesados en obtener una herramienta que permita detectar estos casos. Así mismo, el mismo Reddit puede estar interesado en obtener esta clasificación. Esto, con el objetivo de ayudar a prevenir situaciones de este tipo.
Técnicas y algoritmos para utilizar	Dado que se realizará clasificación, se utilizarán los siguientes 3 algoritmos: Árbol de Decisión, Random Forest, y KNN. Se busca comparar estos modelos, y escoger el que dé resultados óptimos para la compañía.

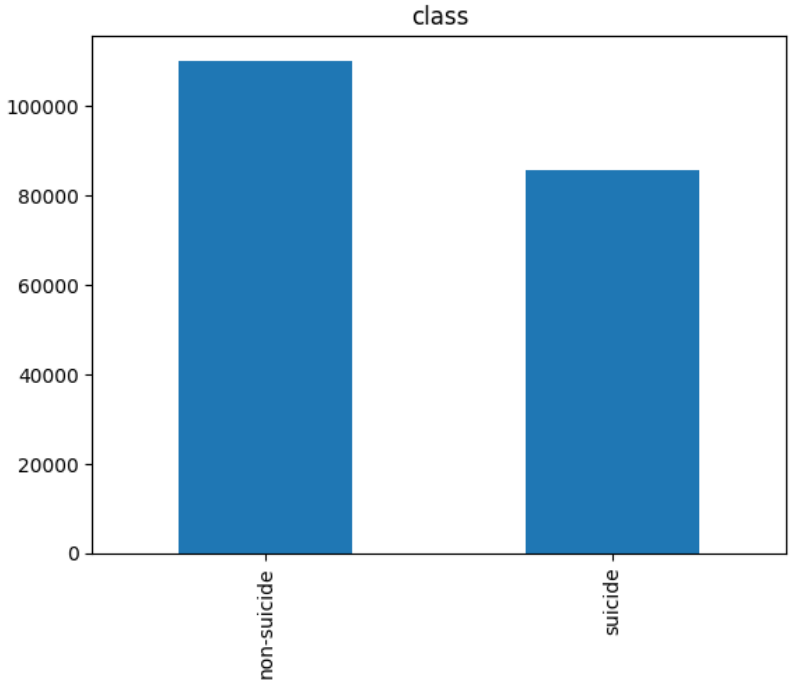
Entendimiento y Preparación de los datos

El proceso de entendimiento y preparación de datos se encuentra implementado y explicado en el notebook entendimiento_procesamientos_datos.ipynb. A continuación, se hace un resumen del procedimiento llevado a cabo.

Inicialmente se descargaron los datos entregados y se obtuvo información básica de estos. A continuación, se muestra la cantidad de datos y un ejemplo de ellos:

```
Número de filas:  195700
Número de columnas:  3
```

	Unnamed: 0	text	class
21738	135566	AAAA I'm literally the stupidest person in exi...	non-suicide
26627	17383	Is it just me or is the sound of rain One of t...	non-suicide
135198	36707	How close i amam tired of living, tired of bei...	suicide
140848	186151	Guess who kissed a girl? Not me.\n\n&#x200...	non-suicide
79670	255320	Should I delete my Reddit account? I mean shou...	non-suicide



Se observa que hay cerca de 200 mil registros, de los cuales alrededor de un 40% hacen referencia a intentos de suicidio. Posteriormente, se midieron dimensiones de calidad en los datos:

- Completitud: Se observa el número de valores nulos en los datos:

```
Número de filas con valores nulos:  0
Número de columnas con valores nulos:  0

Porcentaje de completitud de las columnas: 100.00%
```

En este caso no fue necesario eliminar ningún dato pues no había valores nulos en el archivo .csv que recibimos.

- Unicidad: Se ve el número de valores repetidos en la base

```
Número de filas duplicadas:  0
Número de filas con indice duplicado:  0
```

Tampoco fue necesario eliminar datos por unicidad pues no hubo ni un solo dato duplicado.

- Consistencia y validez

  La consistencia hace referencia a la integridad de datos entre fuentes y observaciones. Por su parte, la validez mide si los datos hacen sentido para el contexto específico. En este sentido, se considera que en términos generales se cumple con estas métricas.

Luego, se realizó la preparación de los datos. Para poder llevar a cabo la ejecución de los diferentes modelos primero tenemos que procesar los textos para que estén limpios y puedan ser tokenizados facilmente. Lo primero que se hizo fue eliminar todos los caracteres que no pertenecían al alfabeto (e.g. @, %, $, etc.). Después se pasaron todos los textos a minúsculas pues las letras mayúsculas no aportan ninguna información útil al modelo y queremos que todas las palabras estén iguales y no se vayan a diferenciar por una letra en mayúscula.

Posteriormente se hizo la tokenización de los textos y se utilizaron funciones para modificar los tokens de la siguiente manera: se quitaron aquellas palabras que no contuvieran ninguna vocal o “y”, pues sin ninguna de estas letras la palabra se considera un typo. Posteriormente se eliminaron las “stopwords”. Estas son palabras que no aportan significado por si solas y que se utilizan para cumplir con la sintaxis del lenguaje. Nosotros solo necesitamos palabras que nos aporten valor con respecto al tema que estamos buscando. Pronombres o conectores no aportan respecto al tema del suicidio. Así mismo, se eliminaron los tokens vacíos y aquellos que tuvieran una longitud menor a 2. Además, se utilizaron métodos para lematizar las palabras para obtenerlas de la manera en que se verían en un diccionario, y se eliminaron los prefijos y sufijos (stemming). Todos se pasaron a singular, los verbos se pasaron a infinitivo, etc.

Finalmente se volvieron a unir todos los tokens en un string para poder llevar a cabo el proceso de vectorización para cada modelo. La vectorización de datos fue realizada de manera independiente por cada estudiante en preparación de los datos para los respectivos modelos.

Modelado y evaluación

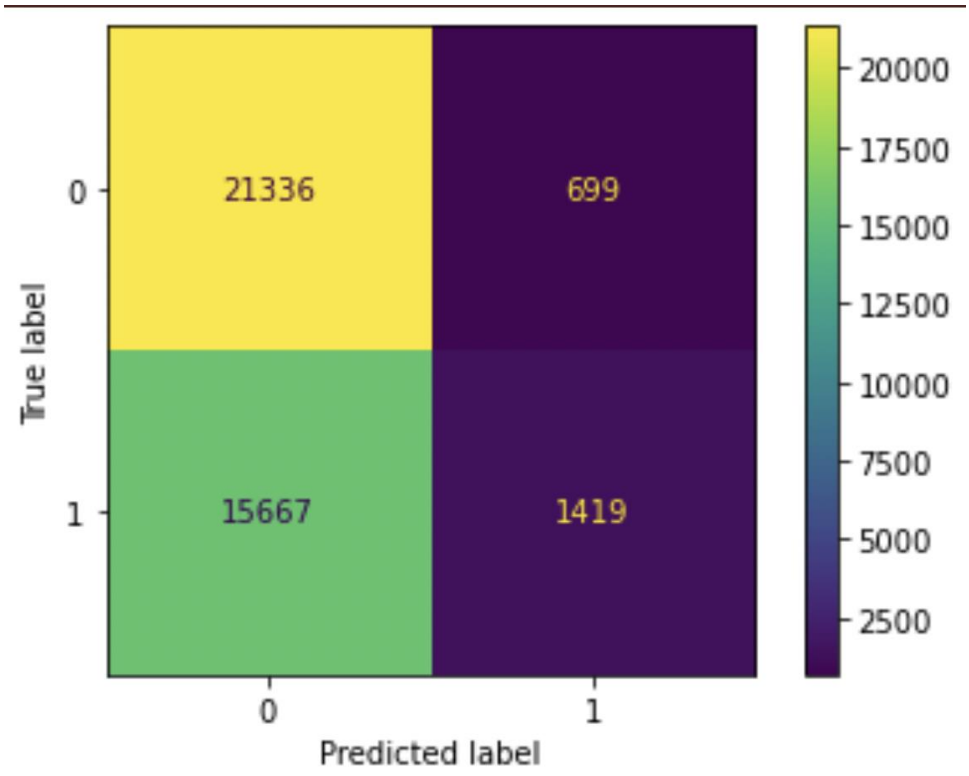
KNN

Estudiante encargado: Jhoan Diaz 201819861

En primer lugar, se implementó un algoritmo de KNN. Esto, dado su facilidad de implementación y su baja necesidad de datos de entrenamiento en comparación a otros algoritmos. La implementación de este algoritmo está plasmada en el notebook modeloknn.ipynb. En primera instancia, se clasifica el dataset entre las variables dependientes e independientes, en donde class hará referencia a los dependientes y los tokens la independiente, class es una variable binaria que adquirirá el valor de 1 si el texto coincide con un intente de suicidio.

Posteriormente se procede a vectorizar los tokens, para crear los datos de entrenamiento y prueba, Con los tokens vectorizados, se procedió a crear los datos de entrenamiento y prueba. En cuanto a métricas de desempeño, se obtuvieron los siguientes resultados.

	precision	recall	f1-score	support
0	0.58	0.97	0.72	22035
1	0.67	0.08	0.15	17086
accuracy			0.58	39121
macro avg	0.62	0.53	0.44	39121
weighted avg	0.62	0.58	0.47	39121



Se tiene una precisión de 62% y un recall del 53%. El f1-score obtenido fue de 44% lo cual indica que este modelo no hace un muy buen trabajo para poder identificar mensajes de suicidio

de los que no, el modelo logra identificar con una precisión de 62% aquellos mensajes que conllevan al suicidio, con lo que el modelo de todos los mensajes identificados como suicidas se equivocaran en el 38% de estos, con lo que el modelo podria identificar si el mensaje conlleva al suicidio o no, ya que estará en lo correcto el 62% de las veces.

```
Resultados:
Exactitud: 0.86
Recall: 0.7865829737151824
Precisión: 0.876541050974906
Puntuación F1: 0.8291291042924489
              precision    recall  f1-score   support

           0       0.85      0.91      0.88     16597
           1       0.88      0.79      0.83     12745

    accuracy                           0.86     29342
   macro avg       0.86      0.85      0.85     29342
weighted avg       0.86      0.86      0.86     29342
```

Modelo Árbol de decisión

Estudiante encargado: Sergio Peñuela 201922873

Aparte de implementar el algoritmo de KNN, se decidió implementar un modelo con árbol de decisión. Esto, dado que requiere de menor preprocesamiento de datos para obtener buenos resultados y tiene mejores tiempos de ejecución. Este modelo funciona dividiendo los datos a partir de "preguntas" con respecto a sus atributos. En este caso es respecto a la vectorización de los datos. Cada "pregunta" divide los datos a partir de los resultados y crea nodos nuevos en los cuales se hacen nuevas "preguntas" hasta llegar a los nodos hoja, los cuales son aquellos que no tienen hijos.

En este caso antes de correr el modelo se hizo un ajuste de hiperparámetros para poder tener con antelación el mejor modelo posible a partir de los datos obtenidos. Un árbol de decisión tiene 3 hiperparámetros diferentes. El primero es el criterio, que se utiliza para medir la calidad de una nueva división de datos. Después se encuentra la profundidad máxima, la cual indica que el número máximo de niveles que puede tener el árbol, entre más niveles más precisos y pequeños son las divisiones (Un número muy alto no es ideal, al igual que un número muy bajo). Por último, está el número mínimo de datos que debe haber en un nodo para poder llevar a cabo otra división de los datos. Para obtener los hiperparámetros se hizo uso de un grid search y se obtuvieron los siguientes resultados:

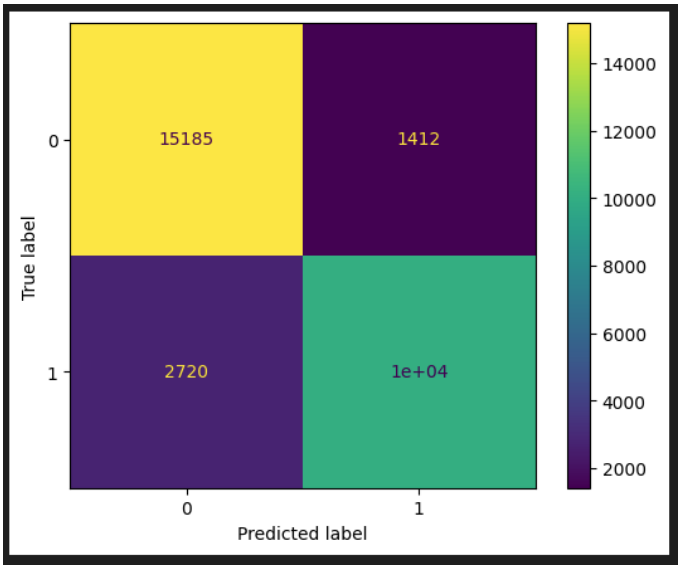
```
{'criterion': 'gini', 'max_depth': 8, 'min_samples_split': 2}
```

A partir de esos resultados se ejecutará el algoritmo utilizando los valores de los hiperparámetros obtenidos. Además, se fue modificando parámetros en el vectorizador para obtener los mejores resultados posibles. El min_df es un parámetro que nos deja decidir cuál es la cantidad mínima (Ya sea en términos de porcentaje o un número entero) de textos en la que debe aparecer un token para ser tenido en cuenta o no. En un principio se intentó con un valor de 0.5 (50% de los textos) pero esto no permitia que el modelo se pudiera ejecutar. Se bajó a 0.2 y por último hasta 0.1 donde se obtuvo el mejor resultado. En el caso del max_df es la cantidad máxima de textos en la que puede aparecer un token antes de que se deje de tomar en cuenta. En un principio se empezó con 0.95 (95% de los textos) y luego se bajó a 0.85 pero no afectó los datos, pero se prefirió utilizar el 0.85 (85% de los textos) para no caer en un problema de sobreajuste de los datos. A partir de eso se obtuvieron los siguientes resultados:

```
Resultados:
Exactitud: 0.86
Recall: 0.7865829737151824
Precisión: 0.876541050974906
Puntuación F1: 0.8291291042924489
              precision    recall  f1-score   support

           0       0.85      0.91      0.88     16597
           1       0.88      0.79      0.83     12745

    accuracy                           0.86     29342
   macro avg       0.86      0.85      0.85     29342
weighted avg       0.86      0.86      0.86     29342
```



Como se puede ver los resultados obtenidos son bastante buenos pues están arriba de 0.8. Se tiene una precisión de 86% y un recall del 85%. El f1-score obtenido fue de 85% lo cual indica que este modelo hace un buen trabajo al identificar mensajes que lleven al suicidio de los que no. El modelo identifica con una precisión del 86% aquellos mensajes que conllevan al suicidio. Por lo que el modelo de todos los mensajes identificados como suicidas se equivocará en el 14% de ellos, por lo que el modelo se podría utilizar para tener una idea acerca de si el mensaje conlleva al suicidio o no pues estará correcto el 86% de las veces.

Random Forest

Estudiante encargado: Jesús David Barrios 201922873

Finalmente, luego de obtener un f1 score de 0.74 en el árbol de decisión, se decide implementar el modelo con Random Forest. Este algoritmo agrupa múltiples árboles de decisión y cada árbol individual da una predicción de clase. La clase con más “votos” entre los árboles del bosque, se convierte en la predicción del modelo. Con esto, se busca aumentar la calidad de la clasificación obtenida.

La implementación de este algoritmo se puede observar en el notebook nombrado modelo_random_forest.ipynb. En primera medida, se separó el dataset entre variable

dependiente e independiente. Siendo la clase la dependiente y los tokens la independiente. Cabe recordar, que la clase es una variable binaria que toma valor de 1 si el texto es asociado con intento de suicidio.

Posteriormente, se procede a vectorizar los tokens con TfidfVectorizer, el cual permite asignar pesos a las palabras según sus apariciones y a su vez establecer umbrales de apariciones. En este caso, se incluyó un umbral superior dado que incluir tokens que tienen alta frecuencia no genera información útil para el modelo.

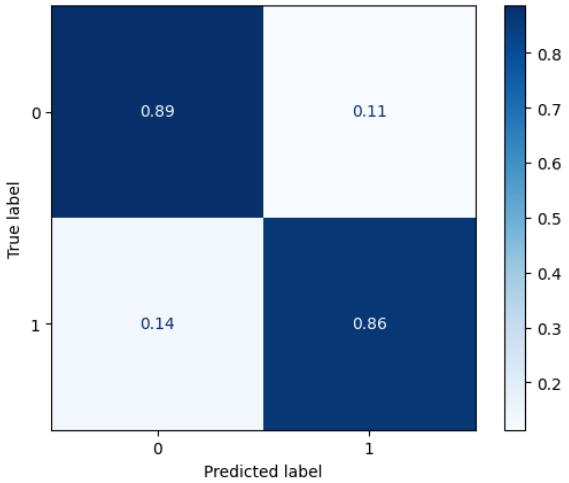
Con los tokens vectorizados, se procedió a crear los datos de entrenamiento y prueba, y a definir los hiperparametros para el grid search. Con esto, se creó el modelo, se obtuvo el óptimo con grid search, y se entrenó.

El modelo evidencia una mejoría frente al realizado con árbol de decisión. En cuanto a métricas de desempeño, se obtuvieron los siguientes resultados.

```
Accuracy:  0.876869200685054
F1:  0.8769471295397717
Precision:  0.8770717948176725
Recall:  0.876869200685054
              precision    recall  f1-score   support

           0       0.89      0.89      0.89     22035
           1       0.85      0.86      0.86     17086

    accuracy                           0.88     39121
   macro avg       0.87      0.88      0.88     39121
weighted avg       0.88      0.88      0.88     39121
```



Se observa que el modelo tiene un buen desempeño clasificando, teniendo pocos valores mal identificados. Con el f1-score, se concluye que el modelo obtenido clasifica correctamente el 88% de las publicaciones de Reddit que se le pasan. El modelo tiene valores similares en precisión y recall, por lo que a su vez se concluye que este es bueno identificando los casos de suicidios y los que no son de suicidio. Así, se les recomienda a los negocios interesados utilizar este modelo para detectar los casos de suicidio.

Resultados y conclusiones

Trabajo en equipo

- Jesús David Barrios

  El estudiante tuvo el rol de líder de datos y líder de negocio, por lo que se encargó de estar alineado con el problema de negocio a lo largo del proyecto y de gestionar la preparación y manejo de datos.

  En cuanto a actividades, el estudiante realizó el entendimiento y procesamiento de datos inicial e implementó el algoritmo de random forest. Para esto, descontando los tiempos

de ejecución, se tuvo una dedicación de alrededor de 3 horas y media. Respecto a retos encontrados en el proceso, el principal fueron los tiempos de ejecución.

La tokenización y la creación del modelo tomaban más de una hora en ejecutarse si se utilizaban todos los datos. Para esto, se decidió realizar un preprocesamiento previo a la tokenización, donde se quitaron mayúsculas y caracteres no alfabéticos. Con esto, el procesamiento en la tokenización fue menor. A su vez, para probar el funcionamiento de la implementación, inicialmente se trabajó solamente con una muestra aleatoria balanceada entre ambas clases. Ya cuando se probó que todo el flujo de trabajo funcionaba correctamente, se procedió a manejar todos los datos para poder entrenar mejor al modelo.

En cuanto a la repartición de 100 puntos por el trabajo, considero que estos podrían ser repartidos de manera equitativa entre los miembros del grupo. Para la siguiente entrega, considero que se debe mejorar la organización de tiempos, para no tener que trabajar con afán al final.

- Sergio Peñuela

  El estudiante tuvo el rol de líder de proyecto. En este caso en lugar de cuadrar reuniones todo se cuadró por medio del grupo de whatsapp de los integrantes del grupo. Por este medio se anunciaron todos los cambios hechos al proyecto, avances y dudas que alguno de los integrantes podía tener a lo largo del proyecto.
  En el caso de las actividades el estudiante realizó la implementación del modelo de árboles de decisión y gran parte del documento y la presentación. Para esto, sin tomar los tiempos de ejecución se dedicaron alrededor de 3 horas.
  En el proceso los retos encontrados fueron principalmente el de la mejoría de los resultados obtenidos en el modelo. Pues, como se explicó anteriormente se utilizó en un principio un grupo de solo 5 mil textos para poder verificar que el modelo se pudiera ejecutar cuando los datos procesados estuvieran listos. Al momento de ejecutar el modelo con los datos procesados fue necesario probar diferentes valores de max_df y min_df en el vectorizador “TfidfVectorizer” para poder hacer que corriera pues en un principio el min_df era muy alto. Ajustando el min_df finalmente se pudo lograr que el modelo corriera y obtener el mejor resultado.
  En cuanto a la repartición de los 100 puntos se considera que pueden ser repartidos de manera equitativa entre los 3 integrantes del grupo pues cada uno hizo correctamente su trabajo y aportó a la entrega del proyecto. Lo que se debería m

ejorar es el uso de los tiempos para poder probar diferentes valores e incluso diferentes modelos a los que se eligieron inicialmente para poder encontrar en verdad el mejor modelo posible.

- Jhoan Diaz tuvo el rol del líder de analítica. En este caso se encargó de gestionar las tareas de analítica del grupo, encargándose de verificar que los entregables cumplieran todos los estándares de análisis y con tal de seleccionar el "mejor modelo" dependiendo de las restricciones dictaminadas. En el caso de las actividades el estudiante realizo la implementación del modelo de KNN, su correspondiente parte de la presentación y del documento, se tuvo una dedicación de alrededor de 3 horas, con referente a retos encontrados en el proceso, el principal fueron los tiempos de ejecución.
-

Se encarga de gestionar las tareas de analítica del grupo. Se encarga de verificar que los entregables cumplen con los estándares de análisis y que se tiene el "mejor modelo" según las restricciones existentes.

Conclusión

A partir de los resultados obtenidos en los 3 modelos que se implementaron se recomienda utilizar el modelo de Random Forest pues es el que tiene un f1-score más alto por lo que es bueno identificando casos de suicido como casos de no suicidio, el árbol de decisión también se podría utilizar, pero tuvo un rendimiento menor de 3 puntos porcentuales. Además de eso se puede utilizar Random Forest para poder saber cuáles son las palabras que más influyen en la clasificación de suicidio. Por lo que este sería el modelo ideal para empresas que buscan identificar personas que puedan cometer un suicidio y evitar que lo hagan.

Referencias

Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.