Juan Sebastián Ramírez 201923800

Andrés Santiago Triana 201923265

Gabriela García 201912531

Proyecto 1

El notebook asociado a esta documentación se puede encontrar en **Notebook/main.ipynb**

Comprensión del negocio y enfoque analítico

Oportunidad/ Problema Negocio	El problema principal es que muchas veces los doctores no tienen mucho tiempo disponible para dedicar a cada paciente, especialmente esto se ve en países de menores recursos, donde la cantidad de expertos es mucho menor que en los países desarrollados. Además de esto, los ensayos clínicos de cáncer pueden ser bastante costosos y en el caso en el que alguien no tenga nada, entre más exámenes se le hagan, más dinero se está desperdiciando. No solo eso, muchas veces los pacientes corren riesgos al hacer dichos exámenes, sobre todo los que requieren cualquier tipo de radiación como lo son los rayos x. Sería de gran utilidad que los doctores tuviesen un diagnóstico inicial de si un paciente pudiese requerir ensayos clínicos para cáncer, puesto que ahorraría tiempo, esfuerzo y dinero, algo muy valioso en todo el mundo, pero sobre todo en los países en vía de desarrollo.
Descripción del requerimiento desde el punto de vista de aprendizaje de máquina	Lo que la maquina debe hacer es análisis de lenguaje, dado que la información que se tiene únicamente son dos columnas: En primer lugar, se tiene una columna con un label que dice 1 si el paciente no requiere ningún tipo de ensayo clínico y en segundo lugar una columna con dos frases separadas con un punto, la que va antes del punto indica el estudio realizado, mientras que la que va después tiene una descripción del tipo de cáncer que podría tener. La máquina debe hacer análisis del lenguaje con las dos frases que se le dan. Lo que debe hacer es intentar entender el tipo de cáncer por medio del lenguaje y palabras que pueden ser similares, una vez lo comprende, entender si el estudio puede implicar que el paciente va a requerir de los ensayos clínicos.

Comprensión del negocio

El negocio que se nos presenta es el de un modelo de salud para enfermedades complejas como lo puede ser el cáncer. Hay que tener en cuenta que este modelo se puede presentar de muchas maneras, la salud varía según si las organizaciones son privadas o públicas, según las leyes de los países, los seguros de los usuarios, etc. Sin embargo, en términos generales, lo que se busca para este caso específico es poder tener buenas predicciones acerca de si un paciente requiere un ensayo clínico o no. Esto se debe a que no hacer los ensayos a quienes lo necesitan tiene muchas implicaciones, se pierde dinero de la organización, el usuario pierde su tiempo de manera innecesaria, se genera estrés en él, se

puede poner en riesgo innecesario al usuario y finalmente, se pierde el tiempo de los doctores (que también representa dinero) que en muchos países de bajos recursos es vital.

Objetivo Principal

Apoyar las decisiones de los médicos acerca de si un paciente va a requerir ensayos clínicos para cáncer.

Objetivos Secundarios

- Mejorar la comprensión acerca de qué palabras en el lenguaje son las que puede implicar que se requieran hacer ensayos clínicos o no.
- Proveer un modelo relativamente veloz, dado que lo ideal sería permitir reducciones en los tiempos que un doctor debe tomar para poder saber si un paciente va a requerir un ensayo clínico.
- Tener porcentajes de acierto relativamente altos, dado que es un tema de salud, los errores implican afectación en muchas personas.

Detalles de la actividad de minería de datos

Tipo aprendizaje	**Tarea de aprendizaje**	**Algoritmo e hiperparámetros utilizados (con la justificación respectiva)**
Supervisada	Árboles de decisión (clasificación)	DecisionTreeClassifier, entropía (Explicación y detalles en notebook)
Supervisada	Redes neuronales (clasificación)	MLPClassifier(Explicación y detalles en el notebook)
Supervisada	KNN (clasificación)	KNeighborsClassifier, n_neighbors=49(Explicación y detalles en notebook)

Comprensión de los datos y preparación de los datos

Para poder hacer correctamente la limpieza de los datos, antes se requieren de varias cosas. En primer lugar, los datos que se tienen no son fáciles de analizar en ese estado, dado que solo se tienen dos columnas de Strings, lo que dificulta mucho cualquier tipo de análisis estadístico. Es por esto que, primero se dividen los datos de study_and_condition, se hace un parse para dejar una columna con las frases de study y otra columna con las frases de condition, esto no solo permite que se entiendan mucho mejor los datos, sino que además más adelante permitirá tomar estadísticas más relevantes que si simplemente nos quedáramos con el original.

También se modificó la columna de labels, dado que nos parecieron bastante anti intuitivos en su estado, puesto que los que tenían que hacer ensayos tenían label_0 y los que no tenían label_1, una notación muy poco utilizada, dados estándares generales. Es por esto,

que cambiamos para que los que se les hicieran exámenes tuviesen 1 y los que no tuviesen 0, de paso, convirtiendo la columna en numérica.

Una vez hecho el parse, se toman las frases y se crean nuevas columnas numéricas con información relevante de estas, específicamente se crean las columnas conteo_study, min_study, max_study, conteo_condition, min_condition y max_condition. Las columnas conteo lo que hacen es que dicen la cantidad de letras en la frase, las columnas min, dicen la cantidad de letras de la palabra con menos letras y la columna max dice la cantidad de letras de la palabra con más letras.

Cuando ya se crean las columnas, se utiliza el método de ProfileReport, el cual provee datos muy importantes acerca de las diferentes columnas

- **Overview**
  - **General**
    - **Cantidad de datos:** Se tiene un total de 12000 filas
    - **Duplicados:** Solo se tienen 12 duplicados en total, representando el 0.01% de los datos totales
    - **Tipos de datos:** Después de las modificaciones, se tienen 4 filas de String y 5 filas numéricas
- **Variables**
  - **Label:** De label no hay tantos datos relevantes, solo se tienen dos tipos de variables y esto causa claramente que sus datos no sean únicos, además no tiene nulos. Se tienen la misma cantidad de entradas en las que se requieren ensayos y en las que no.
  - **Study:** Una columna que nos sorprendió bastante, creíamos que los datos de esta iban a ser bastante únicos, sin embargo, nos dimos cuenta de que nuestra suposición no era correcta, dado que solo tiene un 13.8% de Unique. Esto se traduce en que los datos de study en su mayoría están repetidos, algo importante, pues es más fácil que el modelo encuentre patrones si se pueden agrupar con dicha información. Solo hay 1660 variaciones de study y se evidencia que las más repetidas tienen que ver con estudios en anticuerpos.
  - **Condition:** La historia es muy diferente a study, en conditon se puede ver un porcentaje alto de únicos (97.4%), algo que tiene sentido, pues hay muchas maneras de armar una frase para expresar la condición que puede tener un paciente. También se evidencia que los conditions más repetidos son los de lymphoma, a pesar de que no sean tantas repeticiones como en study, podrían tener peso en los futuros modelos.
  - **Conteo_study:** Para el conteo de las letras en las frases de study se pueden observar datos relevantes como una media de 38.7 letras, con un máximo de 142 letras y un mínimo de 25.
  - **Max_study:** La cantidad máxima de letras por palabra en promedio para study fue de 13.3, con una mínima cantidad de 13 letras y una máxima de 59
  - **Min_study:** El mínimo de palabras de study es tan poco variado que el programa lo toma como categórico, dado que solo están palabras de mínimo 1, 2 o 3 letras, la gran mayoría teniendo en promedio 3 letras.
  - **Conteo_condition:** La cantidad promedio del conteo de letras en condition fue de 131.89, como se puede ver, se tienen en promedio 3 veces más letras que en study. La cantidad mínima de letras fue de 0 y la máxima de 1042

- **Max_condition:** El promedio de letras de la palabra más larga de las condiciones fue de 12.36, un poco más bajo que el de study. La cantidad mínima de letras fue de 0 y la máxima de 32
- **Min_condition:** El promedio de las palabras más cortas fue de 2.2, siendo el máximo 17 letras y el mínimo 2 letras.

Nulos

Haciendo un print de los nulos, se puede ver como no hay ninguno en ninguna de las columnas

Transformaciones principales de los datos

Si bien se crearon una gran cantidad de datos numéricos para entender los datos, estos no van a ser los que se utilizan para el modelo, para ello, primero se requieren hacer ciertas transformaciones. En primer lugar, se crea una lista llamada words, para las palabras que estan en las frases de condition, esto se debe, a que, como se revisó antes hay demasiados únicos como para intentar convertir la variable a categórica, es por esto, que se mandan las palabras a una lista. También se crea una lista de word_study para las palabras de study, esto se debe a que se hizo el intento hacer categorización para study, pero salían errores muy complejos al intentarlo.

A ambas columnas se les corrigen las contracciones con un método y se les hace tokenizer para poder convertirlas en columnas. Al hacer este proceso, también se le hace un preproceso, que es un método que corre múltiples métodos que corrigen cosas como caracteres que no sean ASCII, las manda todas a minúscula, remueve cualquier signo de puntuación, reemplaza también todos los números y finalmente, cambia los stopwords. Cuando ya se hace esto, se corren otros 3 métodos que lo que hacen es que dejan solo las raíces de las palabras y luego lematizan los verbos que se tienen entre estas listas.

Una vez hecho el proceso de transformación de las palabras, es necesario convertirlas a cada una en una columna binaria que indica si la palabra está o no, una vez hecho este encoding, se generan al final el equivalente a 9206 columnas binarias de condition y 3174 de study.

Modelado y evaluación

- ### Árbol de decisión (por: Santiago Triana 201923265)

Los pasos detallados y las decisiones tomadas se explican mejor en el notebook, sin embargo, se retoman las ideas principales a continuación.

Para poder utilizar los datos en el árbol de decisión debemos asegurarnos que se encuentran en un tipo numérico, por lo que debemos desplegar los tipos de datos para cada una de las columnas y reemplazar los objects por tipos int.

Se crearon las variables X y Y del árbol, donde las primeras contienen las variables de decisión y la segunda la objetivo.

Además, dividmos nuestros **X** y **Y** en sets de entrenamiento y prueba, con una relación de 20-80 y creamos nuestro árbol utilizando como criterio la entropia, con un random_state de 0.

Alimentamos dicho modelo con nuestros X y Y de entrenamiento y buscamos las predicciones de Y usando los test de X.

Una vez se tiene el árbol generado, verificamos qué tanta influencia tiene cada una de las columnas (atributo) en la predicción del modelo. En este caso se trata de qué palabra tiene más influencia en la toma de decisión del árbol.

Los resultados de la influencia se pueden ver a continuación:

Primero: neuroginic
Segundo: twin
Tercero: systematic
Cuarto: narcotics_sedatives
Quinto: pelvic

Finalmente intentamos aplicar GridSearchCV pero pudimos ver que el promedio total disminuyó al aplicarlo y la brecha entre las presiciones y recalls a aumentado. Esto significa que nuestro modelo ha presentado un sobreajuste y se ha adaptado fuertemente a los datos con los que se entrenó. Esto se puede deber a una alta cantidad de profundidad.

- **KNN (por: Gabriela Garcia 201912531)**

Se implementa el algoritmo KNN con el objetivo de clasificar los pacientes que requieren ensayos clínicos y los pacientes que no. Para esto se uso como variable objetivo 'label'. Esta columna ya indica si el pacientes es elegible o no.

Para esto se separa dos conjuntos Y con la variable objetivo y X con las columnas sin la variable objetivo. Asimismo, se normaliza el conjunto X para que tanto X_train como X_test queden normalizados.

El algoritmo KNN tiene como hiperparametros el numero de vecinos. Con el objetivo de asignarle un valor con el que se obtengan buenos resultados se usa la cantidad de valores en el conjunto Y_test. Se obtiene la raíz de esta y se usa su aproximación como numero de vecinos.

Se obtuvo 48.969 por lo que se aproxima a 49 para la construcción de modelo.

Se construye el modelo de KNN con n_neighbors=49, p=2,metric='euclidean'. A partir de esta implementación se obtienen los resultado de las métricas para evaluar que tan efectivo es el algoritmo para obtener la clasificación correcta.

- **Redes Neuronales (por: Juan Sebastián Ramírez 201923800)**

Se hace una implementación de redes neuronales dadas diferentes características de las mismas, al igual que en los anteriores casos, la variable de entrada es el label, para el cual se quería predecir si iba a ser 0 o 1. También se hace una transformación de los datos para poder tenerlos en númerico todos, dado que había una columna tomada como objeto.

Se spearan los datos en dos grupos, X Y para hacer el entrenamiento, teniendo en cuenta que se usa 80% para entrenar y 20% para testear. Los hiperparámetros que se usaron fueron los que se tienen por defecto, suena como algo ilógico, pero el modelo era bastante pesado y las redes neuronales tienen la peculiaridad de tener muchos hiperparámetros bastante complejos como lo pueden ser el alpha, beta1, beta2, epsilon, etc. Realmente con tantos parametros hacer un GridSearch habría podido tomar muchisimas horas. Los valores estándar son perfectos porque en general encajan para muchos modelos. El único hiperparámetro completamente elegido fue el del solver, dada la sugerencia de utilizar “lbfgs” cuando el grupo de datos no era demasiado grande.

Los parámetros desafortunadamente no se pueden analizar como se podría hacer en un árbol, dado que las redes neuronales tienen un funcionamiento diferente y no tienen métodos existentes para poder gráficar la importancia de las variables.

Análisis de resultados

Resultados de árbol de decisión:

	precision	recall	f1-score	support
0	0.80	0.76	0.78	1245
1	0.75	0.80	0.78	1153
accuracy			0.78	2398
macro avg	0.78	0.78	0.78	2398
weighted avg	0.78	0.78	0.78	2398

Resultados de redes neuronales

	precision	recall	f1-score	support
0	0.83	0.80	0.82	1245
1	0.80	0.82	0.81	1153
accuracy			0.81	2398
macro avg	0.81	0.81	0.81	2398
weighted avg	0.81	0.81	0.81	2398

Resultado de KNN:

	precision	recall	f1-score	support
0	0.86	0.09	0.17	1245
1	0.50	0.98	0.66	1153
accuracy			0.52	2398
macro avg	0.68	0.54	0.42	2398
weighted avg	0.69	0.52	0.41	2398

Tanto las métricas obtenidas en el algoritmo de árbol de decisión y del algoritmo de redes neuronales obtuvieron unos buenos resultados de clasificación en ambos labels. Mientras que el algoritmo de KNN solo logro obtener buenos resultados en un label a comparación del otro. Por lo que se puede concluir que KNN no es un algoritmo clasificación para resolver lo que busca el negocio.

Por lo que nos enfocamos principalmente en los algoritmos de árbol de decisión y de redes neuronales. Para este ultimo algoritmo observamos resultados por encima

de 0.8 mientras que en el árbol de decisión los resultados oscilan entre 0.75 y 0.8. A partir de esta observación, se escoge primordialmente el algoritmo de redes neuronales ya que esta realizo una correcta clasificación tanto para los pacientes elegibles para los casos clínicos y para aquellos no son elegibles.

Trabajo en equipo

Líder de negocio: Juan Sebastián Ramirez

Algoritmo trabajado:

Tareas realizadas:

1. Redacción de toda la sección de Comprensión del negocio y enfoque analítico, incluyendo todo lo que tiene que ver con procesamiento de los datos (3 horas)
2. Creación de algoritmo individual (1 hora)
3. Investigación para intentar correr los modelos con GPU (0.5 hora)
4. Documentaciones propias del algoritmo (1 hora)
5. Tableros de control (1 hora)

Reflexión repartición de puntos: Todos trabajamos como lo establecimos, por ello creo que lo mejor es distribuir equitativamente los puntos para todos

Puntos a mejorar para la siguiente entrega: Iniciar la entrega con más antelación, sin embargo, es entendidible, dado que la actividad no se había visto por ninguno de los integrantes dado que tocaba ingresar a un grupo para poder verla.

Líder de datos: Santiago Triana

Algoritmo trabajado: DecisionTreeClassifier con entropia

Tareas realizadas:

1. Creación del repositorio, documento y presentación compartidas. Establecer plantillas para cada uno de estos elementos (1 hora)
2. Importación de librerias, solución de errores al importar y análisis de datos (1 hora)
3. Perfilamiento y preparación de los datos usados a lo largo de todo el proyecto (3 horas)
4. Creación de algoritmo individual trabajado (1h)
5. Evaluación del modelo trabajado (0.5h)
6. Introducir contenido a la presentación (1h)

Reflexión repartición de puntos: Todos cumplimos con las responsabilidades pactadas, por lo que considero lo mejor es un 33.3 para cada uno de los integrantes.

Puntos a mejorar para la siguiente entrega: Iniciar la entrega con tiempo, mejor comunicación entre los integrantes y repartición de tareas a lo largo de la realización del proyecto.

Líder de analítica: Gabriela García

Algoritmo trabajado: KNN

Tareas realizadas:

1. Parte de documento y presentación (1 hora)
2. Algoritmo individual (1 hora)
3. Evaluación de algoritmo trabajado(1 hora)
4. Comparación de algoritmos (1 hora)

Reflexión repartición de puntos: Todos cumplimos con las responsabilidades pactadas, por lo que considero lo mejor es lo mismo para todos 33.3.

Puntos a mejorar para la siguiente entrega: Mejor manejo del tiempo y mejorar la comunicación entre todos los integrantes.