COLEGIO DE CIENCIAS E INGENIERIA

INGENIERIA INDUSTRIAL

IIN-3007 Analítica de Datos

NRC: 2573

NOMBRE DEL ENTREGABLE: Reporte final

SEMESTRE:

Segundo Semestre 2022-2023

NOMBRE Y CÓDIGO DE ESTUDIANTE: Camilo Acosta 00323454, Sabrina Cárdenas 00216306

PROFESOR: Gabriela Baldeón

FECHA DE ENTREGA: 02-05-2023

• **Introducción breve del proyecto.**
Se utilizó una base de datos de un hospital público para desarrollar un modelo de predicción efectivo que pueda identificar a los pacientes con alta probabilidad de sufrir un accidente cerebrovascular. El objetivo es abordar uno de los problemas de salud más graves a nivel mundial. En el proyecto se realizará una limpieza de datos adecuada, se analizarán las variables predictivas y se implementarán algoritmos de predicción adecuados.

• **Explicar claramente todas las operaciones de preprocesamiento, EDA y visualizaciones de las variables predictivas y una discusión de su importancia.**

Base de Datos:

Varibables predictivas	Descripción
Hypertension (categórica)	1: Paciente tiene hipertensión 0: Paciente no tiene hipertensión
Hearth Disease (categórica)	1: Paciente tiene enfermedad cardiaca 0: Paciente no tiene enfermedad cardiaca
Married (categórica)	Yes: Paciente se ha casado No: Paciente no se ha casado
Work (categórica)	Children: Cuida niños Govt_jov: Trabaja en el gobierno Never_worked: Nunca ha trabajado Private: Trabaja en el sector privado Self-employed : Trabajador por cuenta propia
Residence (categórica)	Rural: Vive en área rural Urban: Vive en área urbana
Avg glucose level	Nivel de glucose promedio
Bmi	Body mass index
Smoking	Formerly smoked, never smoked, smokes, Unknown
Gender	Male, Female, Other
Age	Edad del paciente

Variable de respuesta	Descripción
Stroke	1: Paciente ha tenido ataque cerebral 0: Paciente no ha tenido ataque cerebral

Se realizó la limpieza de cada una de las variables:

- Determinar que no existan datos no pertenecientes a las clases que se muestran en la descripción para cada una de las variables, en caso de encontrarlos los eliminamos.

Ejemplo: en la variable “Work” existían datos que no correspondían a ninguna de las clases de la descripción, por lo tanto, se los eliminó.

- Determinar si existen valores NaN registrados, en caso de existir los eliminamos.

Ejemplo: en la variable “Bmi”, existían valores Nan, se los eliminó.

- Para las variables discretas se realizaron gráficos de boxplot y para las variables categóricas se realizaron gráficos de frecuencias acompañado de un gráfico de pastel.
- El histograma que se realizó anteriormente para las variables discretas tiene el objetivo de determinar la existencia de los valores atípicos, ayudado por tablas de los valores reales que pueden tomar las variables.

Ejemplo: la variable "Bmi" (índice de masa corporal) tiene un rango de valores que las personas pueden tener, en base a esto se analizaron los valores que eran imposibles en el ser humano y se los eliminaron. De esta manera, se realizó con cada caso similar en las variables que correspondiera.

- Por otro lado, la gráfica de frecuencias sirve para determinar características de la distribución de las variables categóricas, de esta manera nos dimos cuenta de la cantidad de veces que aparece cada categoría, comparación de frecuencias y patrones.

Ejemplo: en la variable "Gender" existía una clase llamada Other la cual se eliminó debido a que tenía un solo datos y no era representativa en nuestra variable.

Realizar el preprocesamiento de los datos es de suma importancia, debido a que la mayor parte del tiempo que se manipulan los datos se lo invierte en realizar este proceso. Dado que nos permite identificar errores y valores atípicos, se preparan los datos para el posterior análisis para lograr comunicar los resultados de manera efectiva en la toma de decisiones.

• **Explicar la correlación entre las variables predictivas y su importancia para la variable respuesta**.

Para explicar la correlación entre las variables se va a utilizar el coeficiente de correlación de Spearman, el cual se utiliza con datos que no siguen una distribución normal. El coeficiente de correlación de Spearman puede tener valores entre –1 y 1. Correspondiente a correlación negativa perfecta, correlación positiva perfecta respectivamente, es decir correlación fuerte. Por otro lado, valores cercanos a cero significa que la correlación es muy débil.

A continuación, se presenta una tabla con los rangos que se obtuvieron de aplicar el coeficiente en las variables:

Columna	'Stroke'
'Hypertension'	0,86470588
'Heart Disease'	0,9
'Avg_glucose_level'	0,80588235
'Bmi'	0,52647059
'Age'	0,79705882
'Married_Yes'	0,65588235
'Work_Never_worked'	-0,40588235
'Work_Private'	0,28529412
'Work_Self-employed'	0,73823529
'Work_children'	-0,45
'Residence_Urban'	-0,01470588
'Smoking_formerly smoked'	0,68529412
'Smoking_never smoked'	0,17058824
'Smoking_smokes'	0,31176471
'Gender_Male'	-0,08529412

De esta manera, las variables que se puede decir que tienen una correlación alta con la variable de respuesta son:
Hypertension, Hearth Disease, Avg glucose level, Bmi, Age, Married (Yes), Work (Self employed) y Smoking (formerly smoked). Si nos damos cuenta la correlación es positiva, por lo tanto, quiere decir que si algunas de estas variables crecen entonces nuestra variable de respuesta también va a crecer, en nuestro caso “Stroke”.
Las variables que no fueron mencionadas significan que tienen una correlación débil, por lo tanto, sabemos que no afectan en gran medida a nuestra variable de respuesta.

• Investigación sobre bases de datos no-balanceadas. Incluir referencias pertinentes en el texto.

Las bases de datos no-balanceadas presentan una desproporción en el número de elementos en cada clase dentro de un conjunto de datos. Podemos encontramos con dos clases. La clase minoritarias corresponden a los conjuntos de datos de los que tenemos menos muestras. Por otro lado, las clases mayoritarias son conjuntos de datos que están más representadas en este tipo de bases de datos.
La complejidad de las bases de datos no-balanceadas puede causar varios problemas en el aprendizaje automático, por ejemplo: la superposición ocurre cuando las muestras de diferentes clases se parecen demasiado, lo que dificulta la tarea de clasificación. La falta de datos representativos puede llevar a un modelo a aprender características irrelevantes o incluso a sobre ajustarse a los datos disponibles. Los pequeños disyuntos se refieren a casos donde las muestras de diferentes clases están separadas por regiones vacías, lo que hace que sea difícil para un modelo encontrar una frontera de decisión. Los desequilibrios dentro de las clases pueden hacer que el modelo tenga un sesgo hacia la clase mayoritaria y no preste suficiente atención a las clases minoritarias.
Existen distintas técnicas utilizadas para reducir los efectos negativos de las bases de datos no-balanceadas:
- Técnicas de remuestreo:
    1. Sobremuestreo

2. Submuestreo
3. Híbridos
4. Kernel Based
5. Active Learning Methods

- Aprendizaje costo-sensitivo: corresponde al uso de matrices de costos que describen los costos de clasificar erróneamente cualquier ejemplo de datos particular.
- Métodos de ensamble:
    1. Ensambles secuenciales: “Boosting”
    2. Ensambles paralelos: “Bagging”

• Prueba estadística para escoger la mejor estrategia de entrenamiento que reduce los problemas de bases de datos no balanceados y el análisis de los resultados.

Se realiza una validación cruzada en ambos modelos con los que se están trabajando de 6 iteraciones utilizando la métrica “Accuracy”.

Luego, se realiza una prueba de hipótesis en este caso utilizando la prueba t entre los dos modelos

Hipótesis nula (H0): No hay diferencia significativa entre los “Accuracy” medio de los dos modelos relacionadas.

Hipótesis alternativa (H1): Hay diferencia significativa entre los “Accuracy” medio de los dos modelos relacionadas.

Se utilizan técnicas de validación cruzada, pruebas de hipótesis: Estadístico t = -24.985, Valor p = 0.000

Finalmente, los resultados de la prueba t arrojaron que se rechaza la hipótesis nula, por lo tanto, hay una diferencia significativa entre los “Accuracy” medio de los dos modelos relacionados. Por lo tanto, los procesos posteriores a este análisis se van a realizar con el modelo 2 con un “Accuracy” de 0.979 lo que quiere decir que nuestro modelo realiza una mayor cantidad de predicciones correctas en relación al número total de predicciones realizadas en comparación al modelo 1.

• División de la base de la base de datos con su respectiva referencia.

De acuerdo con un estudio en el que se realizó analítica de datos en una aplicación médica se reflejaban los siguientes porcentajes de división en la base de datos:

Set de Entrenamiento	70%	3428
Set de Prueba	15%	735
Set de Validación	15%	735

Referencia: Python Machine Learning, Sebastian Raschka.

• Mencionar y justificar los 4 algoritmos seleccionados. Explicar en qué modelos e hiperparámetros se utiliza un modelo de optimización. Debidamente justificar los valores de los otros hiperparámetros. Explicar que ensamble se utiliza y el nuevo modelo implementado. Incluir una descripción del nuevo modelo.

1. **Modelo De Ensamble**

Se decidio usar un ensamble de ensamble al plantear dos modelos: Arboles de decisión y RandomForest y unirlos creando un modelo mas robusto.

Los parametros y sus valores seteados son los siguientes, el resto de hiperparametros no explicados aquí son los que estan por defecto, no se explican esos debido a que son muchos para poner en este informe:

- **criterion: 'gini'**

El criterio de Gini se utiliza por defecto en el modelo DecisionTreeClassifier de sklearn, ya que es una medida eficiente para calcular la impureza de un nodo (Breiman et al., 1984). La impureza de Gini mide la probabilidad de clasificación errónea de una muestra, siendo 0 la impureza mínima y 1 la máxima. La ventaja de usar el índice de Gini en lugar del criterio de entropía es que Gini es menos sensible a los cambios en las probabilidades de clase y, por lo tanto, tiende a producir árboles más equilibrados (Murthy, 1998).

- **splitter: 'best'**

El parámetro splitter, que determina la estrategia de selección del atributo en cada nodo, tiene como valor predeterminado 'best'. Esta estrategia selecciona el mejor atributo basado en el criterio de impureza seleccionado (Gini o entropía), eligiendo el que maximiza la reducción de impureza (Pedregosa et al., 2011). La otra opción disponible es 'random', que selecciona un atributo al azar en lugar del mejor. La ventaja de utilizar 'best' es que produce árboles más pequeños y precisos, mientras que 'random' puede ser útil para evitar el sobreajuste en conjuntos de datos ruidosos o con muchas características.

- **max_depth: None**

El parámetro max_depth controla la profundidad máxima del árbol. Cuando se establece en None, el árbol se expande hasta que todas las hojas sean puras o contengan menos muestras que el mínimo establecido en el parámetro min_samples_split (Pedregosa et al., 2011). Esta configuración predeterminada permite al árbol crecer lo suficiente para capturar patrones complejos en los datos, pero puede provocar sobreajuste. Establecer un valor máximo de profundidad puede ayudar a evitar el sobreajuste, pero puede resultar en un modelo subajustado si se elige un valor demasiado bajo.

2. Modelo GradientBoostingClassifier

Los parametros y sus valores seteados son los siguientes, el resto de hiperparametros no explicados aquí son los que estan por defecto:

- **learning_rate: 0.1**

El parámetro learning_rate controla la tasa de aprendizaje en el proceso de potenciación del gradiente (Friedman, 2001). El valor predeterminado es 0.1, lo que indica una tasa de aprendizaje moderada. Una tasa de aprendizaje más baja puede mejorar la precisión del modelo al permitir un ajuste más fino de los pesos de los predictores. Sin embargo, también puede aumentar el tiempo de entrenamiento y requerir un mayor número de árboles

(n_estimators) para converger. Una tasa de aprendizaje más alta puede permitir una convergencia más rápida, pero también puede aumentar el riesgo de sobreajuste o no converger a una solución óptima.

- **n_estimators: 100**

El parámetro n_estimators controla la cantidad de árboles de decisión en el modelo de Gradient Boosting (Friedman, 2001). El valor predeterminado es 100, lo que significa que se construyen 100 árboles. Utilizar más árboles puede mejorar la precisión del modelo al permitir la captura de patrones más complejos en los datos. Sin embargo, un mayor número de árboles también aumenta el tiempo de entrenamiento y el riesgo de sobreajuste. Un valor más bajo de n_estimators puede ser más eficiente en términos de tiempo y recursos, pero podría no proporcionar la misma precisión que un modelo con más árboles.

- **subsample: 1.0**

El parámetro subsample controla la proporción de muestras utilizadas para entrenar cada árbol en el proceso de Gradient Boosting (Friedman, 2002). El valor predeterminado es 1.0, lo que significa que se utilizan todas las muestras para entrenar cada árbol. Esto permite que cada árbol tenga acceso a toda la información en el conjunto de entrenamiento. Establecer un valor más bajo para subsample permite utilizar un subconjunto aleatorio de las muestras de entrenamiento para cada árbol, lo que puede mejorar la robustez del modelo al reducir el riesgo de sobreajuste. Esta técnica se conoce como "Stochastic Gradient Boosting" (Friedman, 2002). Sin embargo, un valor demasiado bajo puede resultar en un modelo subajustado al no permitir que el árbol capture suficientes patrones en los datos.

3. **Modelo Redes Neuronales Artificiales (ANN)**

Los parametros y sus valores seteados son los siguientes, el resto de hiperparametros no explicados aquí son los que estan por defecto:

- **alpha: 0.0001**

El parámetro alpha controla el término de regularización L2 (también conocido como "ridge regression" o "weight decay") en el modelo MLPClassifier (Hastie et al., 2009). El valor predeterminado es 0.0001, lo que indica una cantidad pequeña de regularización. La regularización L2 ayuda a prevenir el sobreajuste al penalizar los pesos grandes en el modelo, lo que fomenta la selección de modelos más simples con pesos más pequeños. Un valor más alto de alpha aumenta la cantidad de regularización, lo que podría mejorar la generalización del modelo al reducir la complejidad del modelo. Sin embargo, un valor demasiado alto puede resultar en un modelo subajustado al forzar a los pesos a ser demasiado pequeños y no permitir que el modelo capture patrones en los datos.

- **max_fun: 15000**

El parámetro max_fun controla el número máximo de llamadas a la función objetivo durante la optimización en el modelo MLPClassifier (Pedregosa et al., 2011). El valor predeterminado es 15000, lo que significa que se permiten hasta 15000 llamadas a la función objetivo antes de detener la optimización. Un valor más alto de max_fun permite más iteraciones para la optimización y puede mejorar la precisión del modelo al permitir una convergencia más completa hacia una solución óptima. Sin embargo, también puede

aumentar el tiempo de entrenamiento y el riesgo de sobreajuste si se permite demasiada optimización. Un valor más bajo de max_fun puede ser más eficiente en términos de tiempo y recursos, pero podría no permitir la convergencia completa hacia una solución óptima.

- **shuffle: True**

El parámetro shuffle controla si las muestras se barajan después de cada iteración durante el entrenamiento en el modelo MLPClassifier (Pedregosa et al., 2011). El valor predeterminado es True, lo que significa que se realiza una mezcla de las muestras. Barajar las muestras puede mejorar la convergencia del modelo y prevenir problemas de estancamiento en optimizaciones locales. También puede reducir el riesgo de sobreajuste al evitar que el modelo se adapte demasiado a un orden específico de las muestras de entrenamiento. Establecer shuffle en False puede acelerar ligeramente el entrenamiento al evitar la operación de mezcla, pero podría resultar en una convergencia más lenta o un mayor riesgo de sobreajuste.

Optimización de Hiperparametros

Por otra parte se optimizan las siguientes hiperparametros usando el GridSearchCV. Esta es una búsqueda exhaustiva en una cuadrícula de hiperparámetros. Entrena y evalúa un modelo para cada combinación de valores de los hiperparámetros en la cuadrícula Rejor rendimiento en términos de una métrica de evaluación específica RECALL.

Los siguientes son los parametros que se optimizaron:

- **'learning_rate_init' (0.01, 0.003)**
  El parámetro learning_rate_init controla la tasa de aprendizaje inicial en el modelo MLPClassifier (Pedregosa et al., 2011). El valor predeterminado es 0.01, lo que indica una tasa de aprendizaje moderada. Una tasa de aprendizaje más baja (como 0.003) puede mejorar la precisión del modelo al permitir un ajuste más fino de los pesos de los predictores, pero también puede aumentar el tiempo de entrenamiento y requerir un mayor número de iteraciones para converger. Por otro lado, una tasa de aprendizaje más alta puede permitir una convergencia más rápida, pero también puede aumentar el riesgo de sobreajuste o no converger a una solución óptima. El valor predeterminado de 0.01 es un buen punto de partida para la mayoría de los casos.
- **'momentum' (0.7, 0.8)**
  El parámetro momentum controla la cantidad de impulso utilizado en la actualización de los pesos durante el entrenamiento en el modelo MLPClassifier (Sutskever et al., 2013). El valor predeterminado es 0.9, lo que indica una cantidad considerable de impulso. Utilizar un valor más alto de impulso (como 0.8) puede acelerar la convergencia al ayudar a superar óptimos locales y suavizar las oscilaciones en la actualización de los pesos. Un valor más bajo de impulso (como 0.7) puede reducir el riesgo de oscilaciones y mejorar la estabilidad del entrenamiento, pero puede resultar en una convergencia más lenta. El valor predeterminado de 0.9 es una buena opción para la mayoría de los casos, ya que

proporciona un equilibrio entre aceleración de la convergencia y estabilidad del entrenamiento.

- **'activation' (tanh, relu)**

  El parámetro activation controla la función de activación utilizada en las capas ocultas del modelo MLPClassifier (Glorot et al., 2011). Las opciones incluyen 'tanh' y 'relu' (rectified linear unit). 'tanh' es una función de activación que mapea la entrada a un rango de -1 a 1 y puede proporcionar un mejor rendimiento en algunos casos. Sin embargo, 'relu' es la función de activación predeterminada porque es computacionalmente eficiente y puede mejorar la convergencia en redes neuronales profundas al evitar problemas de desaparición y explosión de gradientes. En general, 'relu' es una buena opción para la mayoría de los casos, pero 'tanh' puede ser útil en situaciones específicas.

- **Number of nodes 'hidden_layer_sizes' (16,32)**
  El parámetro hidden_layer_sizes controla el número de nodos en cada capa oculta del modelo MLPClassifier (Pedregosa et al., 2011). El valor predeterminado es (100,), lo que significa una única capa oculta con 100 nodos. Utilizar un número diferente de nodos (por ejemplo, 16 o 32) puede afectar la capacidad del modelo para aprender representaciones complejas de los datos. Un mayor número de nodos puede aumentar la capacidad del modelo para aprender relaciones no lineales, pero también puede aumentar el riesgo de sobreajuste y el tiempo de entrenamiento. Un menor número de nodos puede reducir el riesgo de sobreajuste y mejorar la eficiencia computacional, pero también puede limitar la capacidad del modelo para aprender relaciones complejas. El valor predeterminado de 100 nodos es una buena opción para la mayoría de los casos, pero es posible que otros valores funcionen mejor en situaciones específicas.
- **Network Optimizer 'solver' (adam, sdg)**
  El parámetro solver controla el optimizador de red utilizado para actualizar los pesos del modelo MLPClassifier (Pedregosa et al., 2011). Las opciones incluyen 'adam' y 'sgd' (descenso de gradiente estocástico). 'adam' (Adaptive Moment Estimation) es el optimizador predeterminado porque es computacionalmente eficiente y 'adam' adapta la tasa de aprendizaje para cada peso en función de estimaciones del primer y segundo momento de los gradientes, lo que permite una convergencia más rápida y una mejor adaptación a problemas no estacionarios. Por otro lado, 'sgd' es un optimizador más simple que puede ser más adecuado para problemas más pequeños o más simples.

Resultado de la Optimización

Según los resultados de GridSearchCV, los hiperparametros son:

- **'learning_rate_init' (0.003)**
- **'momentum' (0.7)**
- **'activation' (tanh)**
- **Number of nodes 'hidden_layer_sizes' (32)**

- **Network Optimizer 'solver' (adam)**

4. Modelo Regresion Logistica

Los parametros y sus valores seteados son los siguientes, el resto de hiperparametros no explicados aquí son los que estan por defecto:

- **random_state: None**

El parámetro random_state controla la semilla utilizada para la generación de números aleatorios en el modelo LogisticRegression (Pedregosa et al., 2011). El valor predeterminado es 42 y establece una semilla específica permite obtener resultados reproducibles en diferentes ejecuciones del modelo, lo que puede ser útil para la comparación de modelos y la depuración.

- **solver: 'lbfgs'**

El parámetro solver controla el algoritmo utilizado para la optimización de la función objetivo en el modelo LogisticRegression (Pedregosa et al., 2011). El valor predeterminado es 'lbfgs' (Limited-memory Broyden-Fletcher-Goldfarb-Shanno), que es un algoritmo de optimización de memoria limitada que utiliza aproximaciones de segundo orden para actualizar los parámetros del modelo (Nocedal & Wright, 2006). 'lbfgs' es un algoritmo eficiente que funciona bien en la mayoría de los casos y puede manejar problemas de optimización de gran tamaño. Otras opciones de solver incluyen 'newton-cg', 'sag', 'saga' y 'liblinear'. Cada solver tiene sus propias ventajas y desventajas en términos de eficiencia, precisión y aplicabilidad a diferentes tamaños y tipos de problemas. El solver 'lbfgs' es una opción razonable para la mayoría de los casos, pero es posible que otros solvers funcionen mejor en situaciones específicas.

Optimización de Hiperparametros

Por otra parte se optimizan las siguientes hiperparametros usando el GridSearchCV:

- El **parámetro 'penalty'** controla el tipo de regularización aplicada en el modelo LogisticRegression de sklearn (Pedregosa et al., 2011). Las opciones incluyen 'l1' (norma L1) y 'l2' (norma L2). La regularización ayuda a prevenir el sobreajuste al agregar una penalización a los coeficientes del modelo en función de su magnitud. La regularización L1 tiende a generar soluciones dispersas, lo que significa que algunos coeficientes se vuelven exactamente cero, lo que puede ayudar en la selección de características. Por otro lado, la regularización L2 tiende a generar soluciones más suaves y es menos propensa a generar coeficientes exactamente iguales a cero. El valor predeterminado es 'l2' porque tiende a funcionar bien en la mayoría de los casos y es más fácil de optimizar numéricamente. Sin embargo, 'l1' puede ser útil en situaciones específicas, como cuando se busca una solución dispersa o se necesita realizar selección de características.
- **El parámetro 'l1_ratio'** solo se aplica cuando se utiliza la regularización Elastic Net (una combinación de regularización L1 y L2) en el modelo LogisticRegression

de sklearn. Por lo tanto, no es aplicable en el contexto de los valores predeterminados de los hiperparámetros 'penalty' ('l1' y 'l2').

- **Alpha 'C': El parámetro 'C'** controla el parámetro de regularización inversa en el modelo LogisticRegression de sklearn (Pedregosa et al., 2011). Un valor más pequeño de 'C' (como 0.00001) impone una regularización más fuerte, lo que puede ayudar a prevenir el sobreajuste al reducir la complejidad del modelo. Sin embargo, una regularización demasiado fuerte también puede conducir a un subajuste. Un valor más grande de 'C' (como 0.01) impone una regularización más débil, lo que permite una mayor complejidad del modelo pero también puede aumentar el riesgo de sobreajuste. El valor predeterminado de 'C' es 1.0, lo que proporciona un equilibrio entre regularización y flexibilidad del modelo. Es posible que otros valores de 'C' funcionen mejor en situaciones específicas, y generalmente es una buena idea explorar diferentes valores de 'C' mediante validación cruzada.

• Incluir la matriz de confusión de cada algoritmo y curva ROC. Resumir mediante una tabla las métricas de evaluación de los algoritmos en el set de entrenamiento, validación y prueba. Explicar y justificar la selección de la métrica de evaluación y el mejor algoritmo.

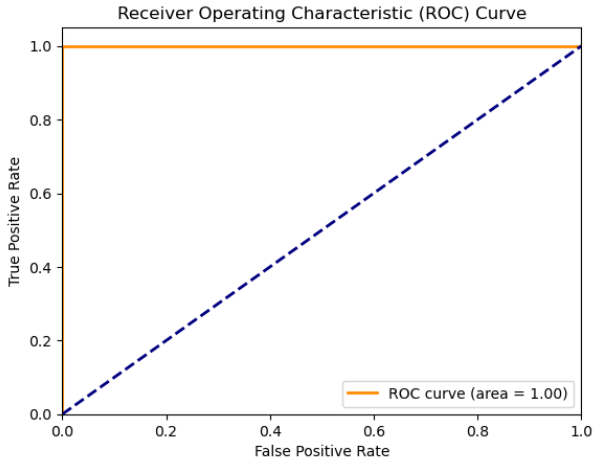
1. **Modelo de ensamble**

Validation		
	Prediction	
Actual	0	1
0	686	24
1	23	2
Test		
	Prediction	
Actual	0	1
0	670	31
1	29	5
Train		
	Prediction	
Actual	0	1
0	3279	0
1	0	32

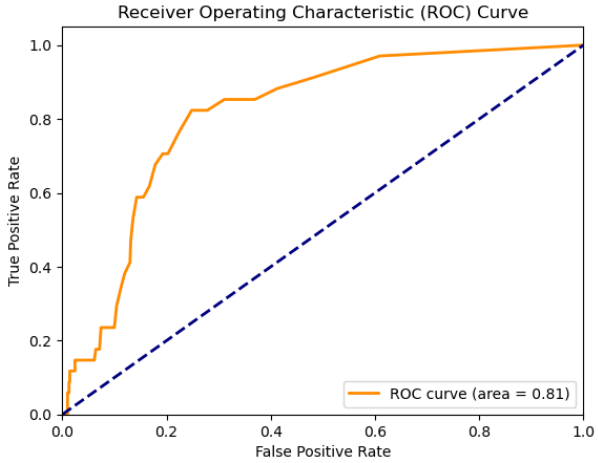
	Train	Test	Validation
Sensibilidad	1.0	0.1470	0.08
Especificidad	1.0	0.9557	0.9661

Exactitud	1.0	0.9183	0.9360
Precisión	1.0	0.1388	0.0769
AUC	1.0	0.8127	0.7276

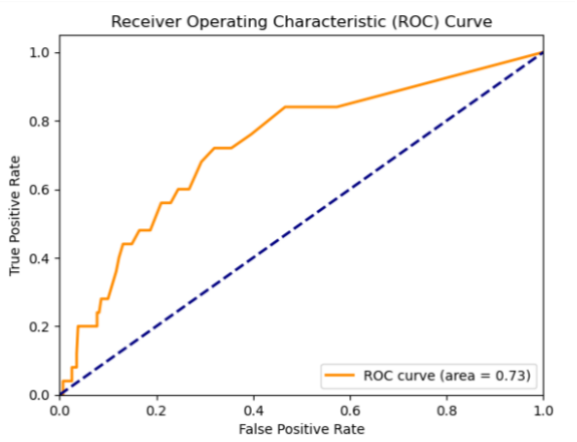
ROC Train



ROC Test



ROC Val



2. Gradient Boosting Classifier

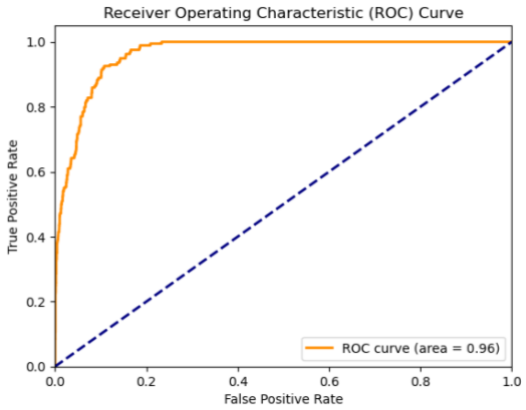
Validation		
	Prediction	
Actual	0	1
0	578	132
1	12	13

Test		
	Prediction	
Actual	0	1
0	570	131
1	11	23

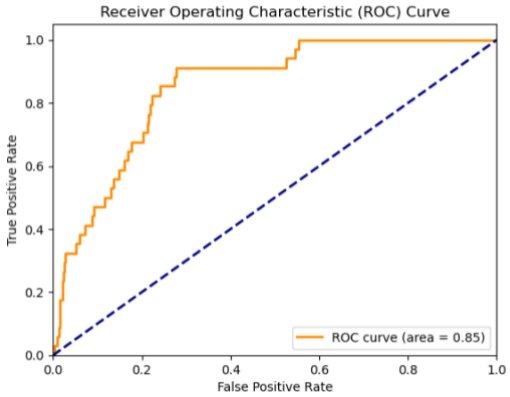
Train		
	Prediction	
Actual	0	1
0	2732	547
1	81	3198

	Train	Test	Validation
Sensibilidad	0.9752	0.6764	0.8140
Especificidad	0.8331	0.8131	0.52
Exactitud	0.9042	0.8068	0.8040
Precisión	0.8539	0.1493	0.0896
AUC	0.9633	0.8505	0.7494

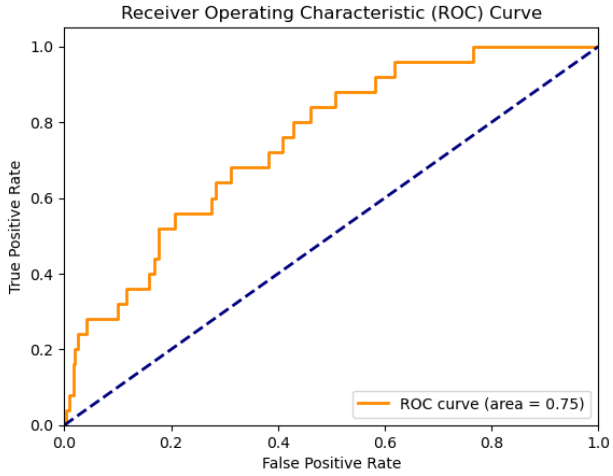
ROC Train



ROC Test



ROC Val



3. Redes neuronales

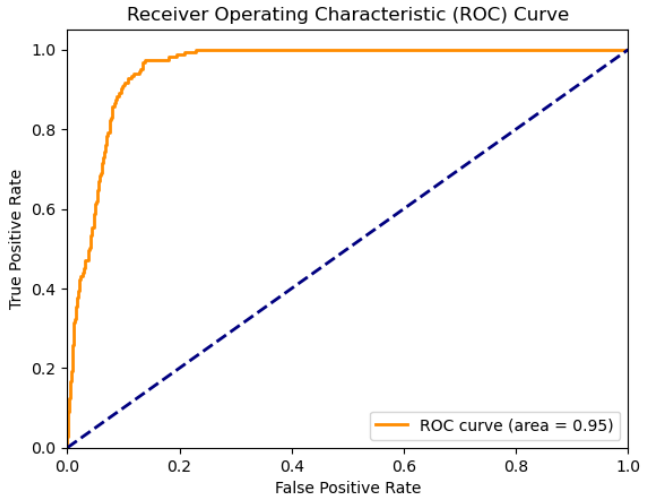
Validation		
	Prediction	
Actual	0	1
0	603	107
1	18	7

Test		
	Prediction	
Actual	0	1
0	590	111
1	16	18

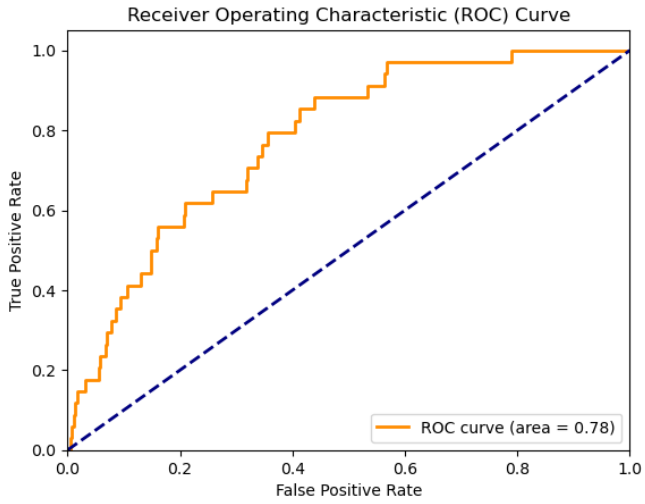
Train		
	Prediction	
Actual	0	1
0	2846	433
1	163	3116

	Train	Test	Validation
Sensibilidad	0.9502	0.5294	0.28
Especificidad	0.8679	0.8416	0.8492
Exactitud	0.9091	0.8272	0.8299
Precisión	0.8779	0.1395	0.0614
AUC	0.9547	0.7791	0.7597

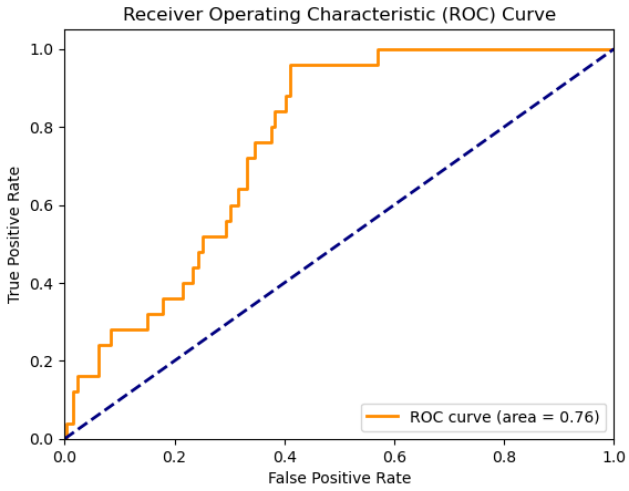
ROC Train



ROC Test



ROC Val



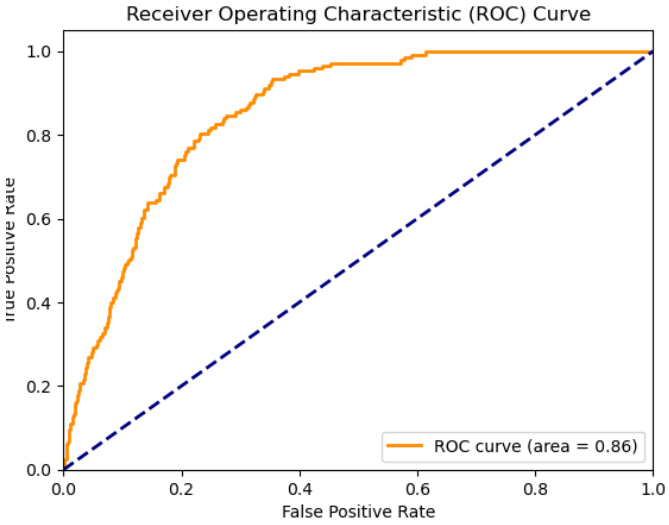
4. Regresión logística

Validation		
	Prediction	
Actual	0	1
0	524	186
1	7	18
Test		
	Prediction	
Actual	0	1
0	511	190
1	5	29
Train		
	Prediction	
Actual	0	1

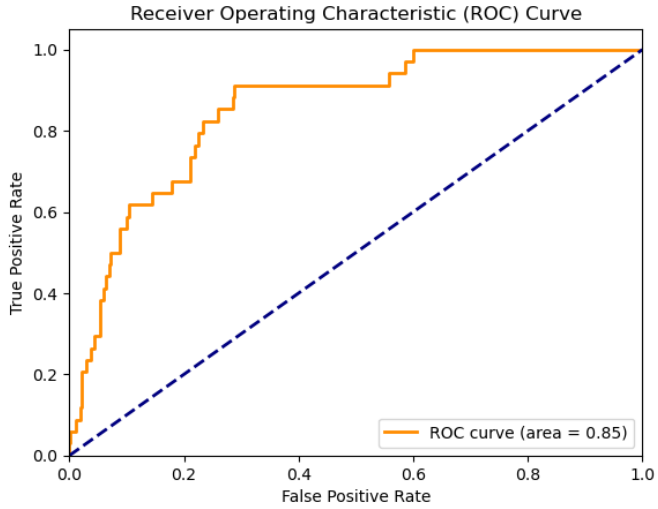
0	2408	871
1	553	2726

	Train	Test	Validation
Sensibilidad	0.8313	0.8529	0.72
Especificidad	0.7343	0.7289	0.7380
Exactitud	0.7828	0.7346	0.7374
Precisión	0.7578	0.1324	0.0882
AUC	0.8550	0.8526	0.8032

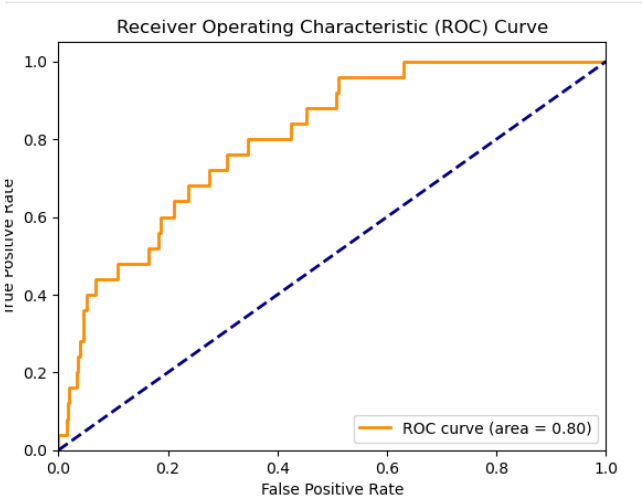
ROC Train



ROC Test



ROC Val



• Análisis de las variables predictivas más importantes para la predicción. Explicar el método basado en random forest utilizado para obtener estos resultados.

Se puede usar un modelo Random Forest obtener las importancias. la importancia de una característica se refiere a su capacidad para explicar la variabilidad en la variable objetivo.

La importancia de cada característica se calcula a través de la reducción de la impureza que resulta al dividir el conjunto de datos según esa característica. El cálculo de la importancia de las características se basa en la idea de que las características más importantes son aquellas que reducen la impureza del árbol de decisiones en mayor medida.
La importancia de las características se utiliza a menudo para identificar las variables predictoras más importantes o relevantes para el modelo. Esto puede ser útil para el análisis de datos y la toma de decisiones

Una vez implementado el modelo de RandomForest se crea un dataframe de importancias de características ordenado por la importancia decreciente. Para esto se utiliza la instancia *model.feature_importances_* . A continuación, se seleccionan el numero de variables predictoras más importantes que se desea (en nuestro ejemplo 5) y se entrena un nuevo modelo sea el que sea, con estas variables predictoras.
El modelo resultante se puede utilizar para predecir el riesgo de un ataque cerebral utilizando solo las variables predictoras más importantes.

De esta manera se obtuvo que las variables mas importantes son:

- Age
- Avg_glucose_level
- Bmi
- Hypertension
- Married_Yes

Y al ser mas significativas se puede repetir el proceso, con mejores resultados y con un menor tiempo de procesado de la maquina sobretodo en la parte de optimización de hiperparámetros.

• Costos de implementación del modelo. Ventajas y desventajas de utilizar el modelo propuesto en el hospital.

La importancia de la implementación de modelos de análisis de datos se basa en la disponibilidad de datos precisos y consistentes permite establecer diagnósticos precisos y tomar decisiones informadas sobre la atención del paciente. Además, las estadísticas sanitarias exactas y comparables son fundamentales para mejorar la planificación de los recursos y políticas de salud de atención médica en general.

De esta forma se puede entender que la implementación de modelos dedicados a la analítica de datos es prioridad y, por lo tanto, se dedican recursos económicos por parte de las empresas para poder implementarlos.

Tomando en cuenta la estructura de nuestro modelo los costos dependen de los siguientes recursos:

1. **Personal:** Es necesario contar con un equipo capacitado en análisis de datos para desarrollar y mantener el modelo. Por lo tanto, la persona encargada del desarrollo e implementación del modelo es el analista de datos. Según información de la página web de salarios (https://www.talent.com), un analista de datos cobra $11.95 por hora, por lo tanto, tomando en cuenta que el tiempo que tomó el desarrollo del modelo fue de 3 semanas, el costo por su servicio sería de $1434 dólares.
2. **Software:** El análisis de datos requiere de programas para la visualización, manejo de bases de datos y software de análisis estadístico. De los cuales pueden ser de pago, mientras que otras son de código abierto. Por lo tanto, en este caso se puede utilizar la herramienta Python el cual es utilizado a nivel mundial, por lo tanto, no hay un costo asociado al software.
3. **Hardware:** Dentro de los costos del hardware se encuentran: una infraestructura adecuada de hardware, como servidores, almacenamiento y equipos informáticos para soportar el modelo y los datos. Por lo tanto, el costo de una computadora para análisis de datos puede variar, pero para nuestro modelo el costo puede rondar los $1500 dólares. Este equipo incluye: procesador Intel Core i7, 16 GB de RAM y 512 GB de almacenamiento sólido.

Ahora, existen costos que deben ser implementados en la estructura del hospital:

4. Infraestructura de red: El costo de las redes depende del tamaño de la red y de la velocidad de transferencia necesaria. Un sistema de red básico puede costar alrededor de $2000 dólares. Pero según la empresa Gartner en año 2021, el costo promedio de una infraestructura de red de u hospital de tamaño mediano para implementar un modelo de analítica de datos es de aproximadamente $500000 dólares.
5. Almacenamiento: El costo del almacenamiento depende de la capacidad requerida y del tipo de almacenamiento utilizado. Para facilitar el manejo de estos datos se asume que el almacenamiento de la base de datos es en la nube. Por lo tanto, en promedio, se estima que el costo de almacenamiento de datos en la nube puede oscilar entre $0.03 a $0.15 USD por GB al mes, dependiendo del proveedor y los

niveles de servicio requeridos. Para un hospital con una gran cantidad de datos, esto podría traducirse en costos mensuales de varios miles de dólares.

Ventajas:

La aplicación de modelos de analítica de datos en un hospital puede brindar ventajas importantes, como la mejora en la toma de decisiones clínicas y administrativas, la detección temprana de enfermedades, la optimización del uso de recursos. Además, la implementación de modelos de analítica antes que la competencia va a lograr la actualización temprana de su servicio generará grandes beneficios en la elección de los clientes y la toma de decisiones.

Desventajas:

Anteriormente se mencionó que existen costos que van a tener que implementar el hospital en caso de que se decida utilizar el modelo, por ejemplo: invertir en una mejor estructura de red, contratar servicios de almacenamiento en la nube e invertir en recursos tecnológicos más potentes para facilitar la analítica de datos. Por lo tanto, a corto y mediano plazo va a ser un gasto muy fuerte para el hospital, tomando en cuenta que no fueron diseñados específicamente para realizar este tipo de operaciones. Pero, con el fin de mejorar la calidad de servicio a sus clientes y tomar mejores decisiones.

• Link al repositorio de Github.

https://github.com/acosmilo/ProyectoFinalAnalitica

• Conclusiones.

- La parte de preprocesamiento es la más importante y es clave para el correcto manejo de los datos en los siguientes pasos.
- Por medio de los gráficos podemos ver la distribución de datos y mucha información de medidas de tendencia central para cada variable.
- Existen muchos métodos para balancear el set de entrenamiento y muchos modelos a implementar, por lo que no se puede asegurar que el procedimiento hecho sea el óptimo. De los dos modelos implementados el de Ensamble es más robusto y mejor según la métrica Recall, ya que al se un modelo que predice una condición medica grave, se debe minimizar los falsos negativos, es decir tratar de que no existan personas propensas a tener el ataque y que no se los medique o controle.
- Se optimizaron los modelos de Redes Neuronales y de Regresión logistica, enfocandonos en maximisar el RECALL, como resultado se obtuvo los mejores valores en esta metrica con estos dos modelos en comparación a otros modelos.
- Sabiendo que una AUC ROC aleatoria tendría una puntuación de 0.5, lo que significa que el modelo no tiene habilidad de clasificación, y que para ningun modelo se obtuvo un valor cercano a 0.5, se puede afirmar que todos los modelos tanto para el set de validación como el de prueba tiene habilidad de clasificación.
- Por medio de los valores AUC en los sets para cada modelo se puede asegurar que el mejor puntuado, tanto en test como en validation, es el modelo de RegresiónLogistica, lo que indica que es el que tiene mejor habilidad de clasificación.
- Por medio del valor de sensibilidad, tanto para train como validación, se puede ver que el modelo de regresión logistica es el que mejor minimiza los falsos negativos.

• **Referencias**

- https://www.medigraphic.com/pdfs/conapeme/pm-2012/pm124g.pdf
- https://www.mayoclinic.org/es-es/diseases-conditions/prediabetes/diagnosis-treatment/drc-20355284
- https://repositorio.utp.edu.co/server/api/core/bitstreams/767003ef-6a5a-4b19-8d13-0c3a25b8f128/content
- https://repositorio.saludcapital.gov.co/bitstream/handle/20.500.14206/4034/SDS-SHAREPOINT2-0107.pdf?sequence=1
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
- Nocedal, J., & Wright, S. (2006). Numerical Optimization. Springer Series in Operations Research and Financial Engineering.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics.
- Friedman, J. H. (2002). Stochastic gradient boosting. Computational Statistics & Data Analysis, 38(4), 367-378
- Breiman, L. (1996). Bagging predictors. Machine Learning
- Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest
- Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks