ACCIDENTE CEREBROVASCULAR: ANÁLISIS ESTADÍSTICO DE LOS FACTORES DE RIESGO Y CREACIÓN DE MODELOS PREDICITIVOS MEDIANTE MACHINE LEARNING

0523 Máster en Bioestadística y Bioinformática

Pablo Vercet Llopis

17/06/2024

CEMP CENTRO EUROPEO DE MÁSTERES Y POSGRADOS

UCAM UNIVERSIDAD CATÓLICA DE MURCIA

Índice

ABSTRACT

El accidente cerebrovascular (ACV), también conocido como ictus, representa una de las principales causas de discapacidad y mortalidad a nivel mundial. El ictus se produce cuando se interrumpe bruscamente el flujo sanguíneo a una parte del cerebro. Sin el riego sanguíneo, las células cerebrales mueren gradualmente y se produce una discapacidad que depende de la zona del cerebro afectada. Es importante conocer los factores de riesgo para poder tratarlos en una fase temprana y reducir la tasa de mortalidad. En este trabajo de investigación, con la ayuda del machine learning (ML), se desarrollan y evalúan varios modelos para predecir la probabilidad de que se produzca un ictus en el cerebro. Se ha evaluado los siguientes modelos: Random Forest, Support Vector Machine y Naive Bayes. Los resultados experimentales muestran que Naive Bayes tuvo la mejor sensibilidad, 0,98, pero los peores datos en especificidad y AUC, 0,33 y 0,65, respectivamente. Por el contrario, tanto Random Forest como SVM obtuvieron datos muy parejos en las 3 métricas estudiadas, en sensibilidad: 0,77 y 0,82; en especificidad: 0,75 y 0,71; y en AUC: 0,75 y 0,76. Datos que demuestran que son los dos modelos estudiados más óptimos para la predicción de ictus con esta base de datos.

Stroke is one of the leading causes of disability and mortality worldwide. Stroke occurs when blood flow to a part of the brain is abruptly interrupted. Without the blood supply, brain cells gradually die and disability results, depending on the area of the brain affected. It is important to know the risk factors in order to treat them at an early stage and reduce the mortality rate. In this research work, with the help of machine learning (ML), several models are developed and evaluated to predict the probability of stroke in the brain. The following models have been evaluated: Random Forest, Support Vector Machine and Naive Bayes. Experimental results show that Naive Bayes had the best sensitivity, 0.98, but the worst data in specificity and AUC, 0.33 and 0.65, respectively. In contrast, both Random Forest and SVM obtained very similar data in the 3 metrics studied, in sensitivity: 0.77 and 0.82; in specificity: 0.75 and 0.71; and in AUC: 0.75 and 0.76. These data demonstrate that the two models studied are the most optimal for stroke prediction with this database.

INTRODUCCIÓN

El accidente cerebrovascular (ACV), también conocido como ictus o stroke, representa una de las principales causas de discapacidad y mortalidad a nivel mundial. El ictus es una infección no contagiosa y se desencadena cuando el flujo sanguíneo que va al cerebro se interrumpe. (Benjamin et al., 2019).

El ictus puede ser isquémico o hemorrágico y se puede catalogar de leve a muy grave, con daños permanentes o temporales. Los hemorrágicos no son muy frecuentes y consisten en la rotura de un vaso sanguíneo que provoca una hemorragia cerebral. En cambio, los ictus isquémicos, que son los más comunes, implican el cese del flujo sanguíneo a una zona del cerebro debido al estrechamiento u obstrucción de una arteria (Dritsas & Trigka, 2022).

Se estima que cada año, alrededor de 15 millones de personas sufren un ACV, de las cuales aproximadamente 5 millones mueren y otras tantas quedan con discapacidad permanente (Feigin et al., 2015). Las consecuencias del ACV no solo afectan la calidad de vida de los individuos, sino que también imponen una carga significativa en los sistemas de salud y en la sociedad en general.

Esta enfermedad está aumentando rápidamente en países en vías de desarrollo como China, que posee la mayor carga de ictus, y Estados Unidos. En los últimas cinco décadas ha aumentado 10 veces el número de fallecidos a causa del ictus. La tasa de mortalidad por ictus, según la OMS, ocupa el puesto 84 en el mundo. Se espera que la tasa de mortalidad y el número de personas afectadas por esta enfermedad crezcan con la población mundial. Pero esta tasa de mortalidad puede prevenirse con un tratamiento y una predicción precoces. (Chen et al., 2023).

Los factores de riesgo asociados al ACV son diversos y pueden clasificarse en modificables y no modificables. Entre los factores de riesgo no modificables se encuentran la edad, el sexo y la historia familiar de ACV. La edad es uno de los factores de riesgo más importantes, ya que el riesgo de ACV aumenta significativamente con la edad, siendo más común en personas mayores de 65 años. Asimismo, los hombres tienen un mayor riesgo de ACV que las mujeres, aunque las mujeres tienen un riesgo más alto de ACV durante el embarazo y en las primeras semanas después del parto.

Además, la historia familiar de ACV se asocia con un mayor riesgo de padecer esta enfermedad, lo que sugiere una predisposición genética que puede aumentar la susceptibilidad individual al ACV (O'Donnell et al., 2016).

Por otro lado, los factores de riesgo modificables son aquellos que pueden ser controlados o modificados a través de intervenciones médicas o cambios en el estilo de vida. Entre los principales factores de riesgo modificables se encuentran la hipertensión arterial, la diabetes, el tabaquismo, la obesidad, la fibrilación auricular, la hiperlipidemia y el consumo excesivo de alcohol (O'Donnell et al., 2016). La hipertensión arterial es el factor de riesgo más importante y comúnmente asociado con el ACV, ya que aumenta la presión en las arterias y puede dañar los vasos sanguíneos en el cerebro, lo que aumenta el riesgo de hemorragia o bloqueo de las arterias cerebrales.

La diabetes mellitus también se asocia con un mayor riesgo de ACV, ya que puede causar daño a los vasos sanguíneos y aumentar la formación de coágulos sanguíneos. El tabaquismo es otro factor de riesgo importante, ya que los componentes tóxicos del humo del cigarrillo pueden dañar los vasos sanguíneos y aumentar la formación de placa en las arterias, lo que aumenta el riesgo de obstrucción de las arterias cerebrales. La obesidad y el sobrepeso se asocian con un mayor riesgo de ACV, especialmente cuando están acompañados de otros factores de riesgo como la hipertensión arterial y la diabetes. La fibrilación auricular, un tipo común de trastorno del ritmo cardíaco, también aumenta el riesgo de ACV, ya que puede causar la formación de coágulos sanguíneos en el corazón que pueden viajar al cerebro y causar un ACV.

Además, la hiperlipidemia, caracterizada por niveles elevados de colesterol y triglicéridos en la sangre, y el consumo excesivo de alcohol también se ha asociado con un mayor riesgo de ACV. La hiperlipidemia puede contribuir a la formación de placas en las arterias, mientras que el consumo excesivo de alcohol puede aumentar la presión arterial y causar daño a los vasos sanguíneos en el cerebro. La gestión y el control de estos factores de riesgo son fundamentales para reducir la probabilidad de sufrir un ACV y mejorar la salud cerebrovascular a largo plazo.

El ictus además, progresa rápidamente y sus síntomas pueden variar, es decir, que algunos pueden desarrollarse lentamente y otras rápidamente. Un ictus se produce con la aparición repentina de uno o varios síntomas. Los principales son: la parálisis de brazos o piernas (normalmente en un lado del cuerpo), entumecimiento de brazos o piernas o a veces de la cara, dificultad para hablar, dificultad para caminar, mareos, disminución de la visión, dolor de cabeza y vómitos y caída del ángulo de la boca (boca torcida). Por último, en los ictus graves, el paciente pierde el conocimiento y entra en coma (Dritsas & Trigka, 2022).

En la gran mayoría de los casos, las primeras 24 horas son críticas. El diagnóstico determinará el tratamiento, que generalmente implica el uso de medicamentos y, en ocasiones limitadas, procedimientos quirúrgicos. La intubación y el uso de ventilación mecánica en la unidad de cuidados intensivos son requeridos cuando el paciente entra en estado de coma (Dritsas & Trigka, 2022).

Aunque algunos pacientes se recuperan después de un episodio de accidente cerebrovascular, la mayoría experimenta dificultades que varían según la gravedad del episodio, como problemas de memoria, concentración y atención, dificultades para hablar o comprender el lenguaje, alteraciones emocionales como depresión, desequilibrio o pérdida de capacidad para caminar, entumecimiento en un lado del cuerpo y dificultades para tragar alimentos (Delpont et al., 2018).

El proceso de recuperación tiene como objetivo restablecer las funciones perdidas después de un accidente cerebrovascular. Se desarrolla un plan de rehabilitación personalizado para facilitar la reintegración psicológica y social del paciente, con la participación de fisioterapeutas, logopedas y neurólogos. Para minimizar el riesgo de sufrir un accidente cerebrovascular, es importante controlar regularmente la presión arterial, hacer ejercicio de manera constante, mantener un peso saludable, dejar de fumar y reducir el consumo de alcohol, además de seguir una dieta equilibrada baja en grasas y sodio (Dritsas & Trigka et al., 2022).

Los métodos tradicionales de identificación del ictus suelen ser la resonancia magnética (RM) y la tomografía computarizada (TC), que son caras e invasivas [10]. Sin embargo, dado que el ictus es un problema que requiere mucho tiempo, es muy

importante tratarlo a tiempo y con eficacia, ya que en la mayoría de los casos puede evitarse la muerte o un daño permanente si el diagnóstico se realiza a tiempo. Por lo tanto, es esencial desarrollar herramientas y dispositivos médicos que permitan a los médicos diagnosticar un ictus sin ser invasivos o incómodos, basándose por ejemplo en biomarcadores o estudiando los factores de riesgo. El conocimiento sobre los factores de riesgo de ACV ha evolucionado considerablemente en las últimas décadas (Powers et al., 2018). El Machine Learning se presenta como la herramienta perfecta para predecir si puede producirse o no un ictus en función de diferentes factores (Alageel et al., 2023).

El Machine Learning es una rama de la inteligencia artificial que se basa en algoritmos y modelos estadísticos para aprender patrones y hacer predicciones a partir de datos. Es capaz de desarrollar modelos que integran múltiples variables clínicas, genéticas y de estilo de vida para estimar el riesgo personalizado de sufrir un ACV en poblaciones específicas. Esta rama ha emergido como una herramienta prometedora en la predicción del riesgo de ACV y en la optimización de su tratamiento. Los modelos de Machine Learning son capaces de identificar patrones complejos y no lineales en los datos que pueden pasar desapercibidos para los métodos tradicionales de análisis estadístico. Además, puede ayudar a mejorar la precisión de los modelos predictivos al ajustar constantemente sus parámetros en función de nuevos datos y hallazgos clínicos (Jain et al., 2020). Estos modelos pueden ser utilizados para desarrollar sistemas de apoyo a la toma de decisiones clínicas, ayudando a los profesionales de la salud a identificar y priorizar intervenciones preventivas y terapéuticas para pacientes con mayor riesgo de ACV (Saposnik et al., 2020).

La comunidad científica ha demostrado un notable interés en la utilización de herramientas como el Machine Learning para predecir la probabilidad de sufrir un episodio de accidente cerebrovascular. En (Dritsas & Trigka, 2022) han utilizado varios modelos para evaluar los accidentes cerebrovasculares, como el Naive Bayes, Random Forest, Logistic Regression, K-nearest Neighbors, Stochastic Gradient Descent, Decision Tree, Multilayer Perception, Majority Voting y Stacking. En este artículo destacamos la precisión de los modelos de Random Forest y Stacking, obteniendo un 97% y 98%, respectivamente.

En (Shoily et al., 2019) se aplicaron cuatro modelos de Machine Learning: Naive Bayes, J48, K-nearest Neighbors y Random Forest. La precisión de Naive Bayes fue de 85,6%, mientras que la del resto de los modelos fue de 99,8%.

En (Chen et al., 2023) compararon y evaluaron once modelos, de los cuales, los más recomendables por su alta precisión obtenida fueron Random Forest (99,85%) y Support Vector Machine (99,99%).

En (Wiryaseputra, 2017) para predecir el ictus utilizaron modelos como Decision Tree, Random Forest, XGBoost y Logistic Regression Algorithm. Los dos que más precisión obtuvieron, y con bastante ventaja de los otros dos, fueron Random Forest (99,27%) y Decision Tree (97,51%).

Después de exponer investigaciones más recientes que emplean métodos de inteligencia artificial para predecir la probabilidad de sufrir un episodio de accidente cerebrovascular, destacando su precisión, el TFM a realizar constará del manejo clínico de esta enfermedad con ciertos datos en los que evaluaremos múltiples variables utilizando herramientas de Machine Learning.

OBJETIVOS

- Desarrollar modelos predictivos sobre el riesgo de sufrir ictus utilizando técnicas de Machine Learning, utilizando datos clínicos y otros factores relevantes para estimar la probabilidad individual de experimentar un ictus en una población específica.
- Validar y evaluar el rendimiento de los diferentes modelos predictivos de riesgo de ictus utilizando conjuntos de datos independientes y comparando sus métricas entre ellos.
- Validar y evaluar el rendimiento de los diferentes modelos predictivos comparando las métricos obtenidas con la de otros autores que hayan utilizado la misma base de datos o similar.

MATERIALES Y MÉTODOS

Los materiales utilizados son los datos que han sido sacados de la base de datos Kaggle. Son una gran cantidad de datos de pacientes. El dataset utilizado tiene 5110 observaciones de pacientes diferentes, en los que cada uno tiene 11 variables. Todas son variables que, como hemos visto en la introducción, pueden causar un accidente cerebrovascular. Estas variables se muestran en la tabla 1.

Número de variable	Nombre de la variable	Descripción
1	ID	Número identificador de cada paciente
2	Género	Masculino o Femenino
3	Edad	Edad en años del paciente
4	Hipertensión	0 si NO tiene hipertensión; 1 SI tiene hipertensión
5	Cardiopatías	0 si NO tiene cardiopatías; 1 SI tiene cardiopatías
6	Casado alguna vez	No o Sí
7	Tipo de trabajo	MENOR, FUNCIONARIO, NUNCA ha trabajado, trabajo PRIVADO, AUTÓNOMO
8	Tipo de residencia	Rural o Urbano
9	Nivel medio de glucosa	Nivel de glucosa (mg/dL) medio en sangre
10	IMC (IBM)	Índice de masa corporal (Kg/m)
11	Hábito de fumar	ANTES fumaba, NUNCA ha fumado, FUMA, DESCONOCIDO
12	Ictus	0 si NO ha sufrido Ictus; 1 SI ha sufrido ictus

Tabla 1: Descripción de las variables utilizadas en el estudio.

Las variables en la tabla1 están traducidas al español, ya que este trabajo se realiza en castellano. Las variables, tanto en el Excel del que obtenemos los datos como el código, están en inglés.

La mayoría de las variables del estudio son nominales, excepto la edad, el nivel medio de glucosa y el IMC, que son numéricos. El ID no se utilizará ya que no aporta datos significativos y tan solo sirve para separar los datos de un paciente de otro.

Lo primero que realizamos en un pre-procesamiento de los datos son los análisis visuales mediantes gráficos. En las variables categóricas hemos utilizado un diagrama de barras; mientras que la opción del boxplot ha sido la elegida en las variables numéricas.

Se ha observado que hay muchos casos donde tenemos ruído, es decir, hay outliers, hay datos nulos o desconocidos, por lo que podría afectar a nuestro estudio. Sobre los datos N/A vemos que hay 201, solamente, en la variable BMI. 201 filas representan aproximadamente el 4% de los datos. No es un valor excesivamente alto como para poder interferir en los resultados finales, pero si eliminamos las 201 filas puede que alguna variable sí que pueda tener interferencias en los resultados finales. Por lo que he decidido que, al ser la variable BMI, que no tiene valores muy altos ni muy bajos, es decir, que la población normalmente está en valores “cercanos”, es realizar la media y sustituir dicho resultado por los valores N/A. Por lo tanto, se decide eliminar los outliers y los nulos, sustituyéndolos por la media de los datos válidos.

En la imagen 1 podemos destacar como, sin conocer la relación con las otras variables, las mujeres sufren un poco menos este tipo de accidente cerebrovasculares. El 4,7% de las mujeres sufren ictus; mientras que en los hombres, sube ligeramente, el 5,1% lo sufren.

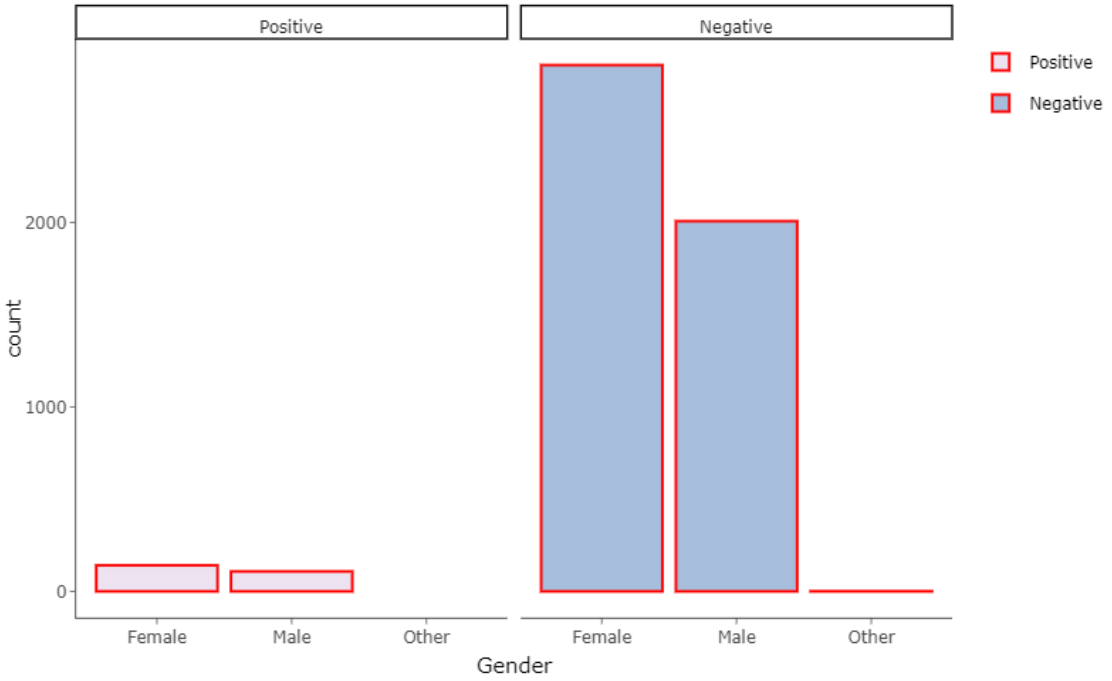

Ilustración 1: Comparación de géneros VS a sufrir un ictus.

En la imagen 2 se compara si el haber estado casado influye con la posibilidad de sufrir un ictus. El estar o haber estado en un matrimonio tiene una posibilidad de 6,56 % de sufrir un ictus; mientras las personas del estudio que no se casaron, fueron un 1,65% las que sufrieron el ictus. Parece que esta variable influye significativamente, ya que existe una diferencia de x4 en la posibilidad de sufrir ictus.

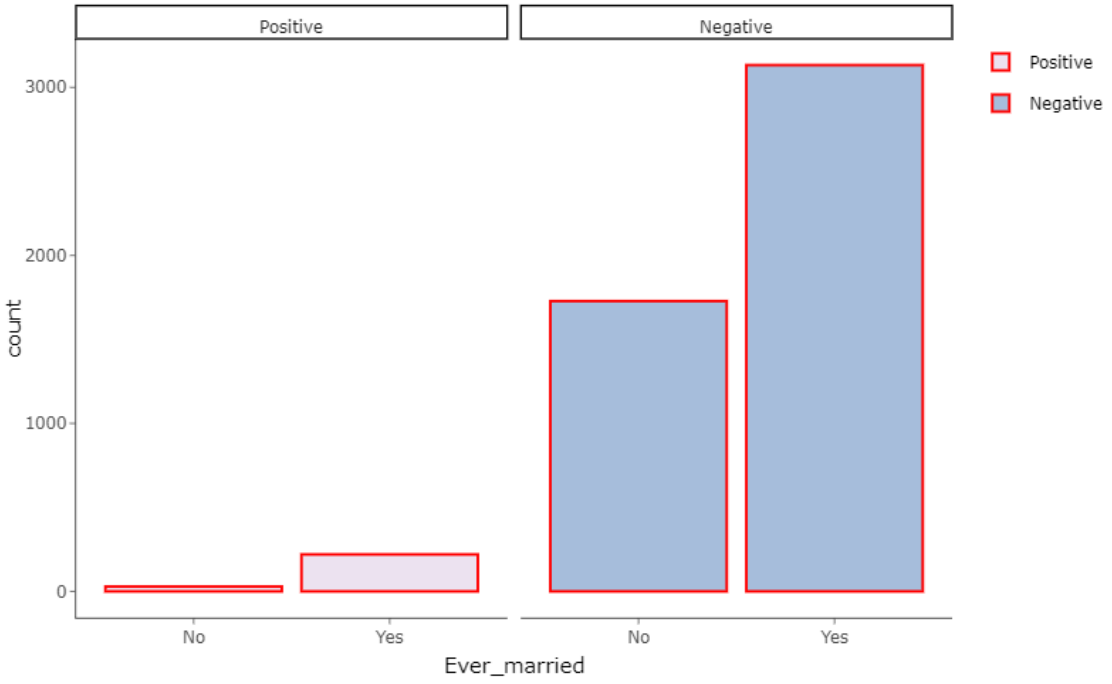

Ilustración 2: Comparación de si se han casado VS a sufrir un ictus.

En la imagen 3 se compara si el haber sufrido un ataque al corazón puede estar relacionado con sufrir un ictus. Las personas que sufrieron un infarto y posteriormente un ictus fueron del 17%; mientras las que sí que sufrieron el ataque al corazón pero no el ictus fueron de 83%. Un porcentaje significativo que parece que influye a la hora de sufrir un ictus.

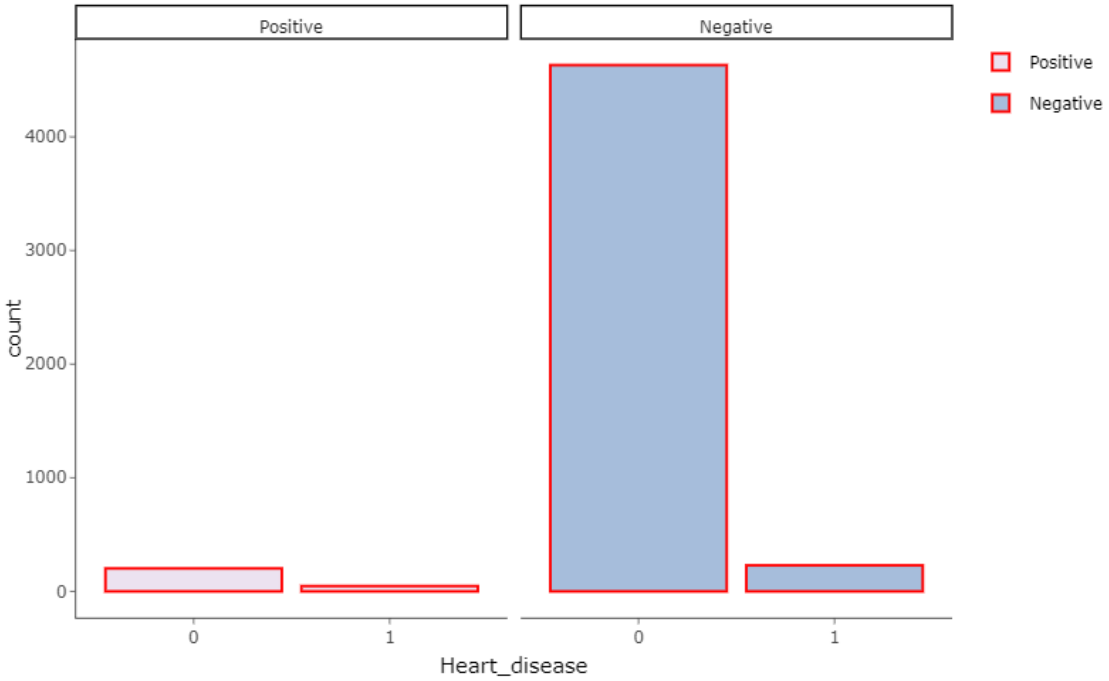

Ilustración 3: Comparación de si han sufrido infarto previamente VS a sufrir un ictus.

En la imagen 4 se compara si el tener hipertensión puede tener influencia en tener un ictus. Las personas que tienen hipertensión y sufrieron un ictus fueron el 13,25% del estudio. Otro dato que muestra un número significativo.

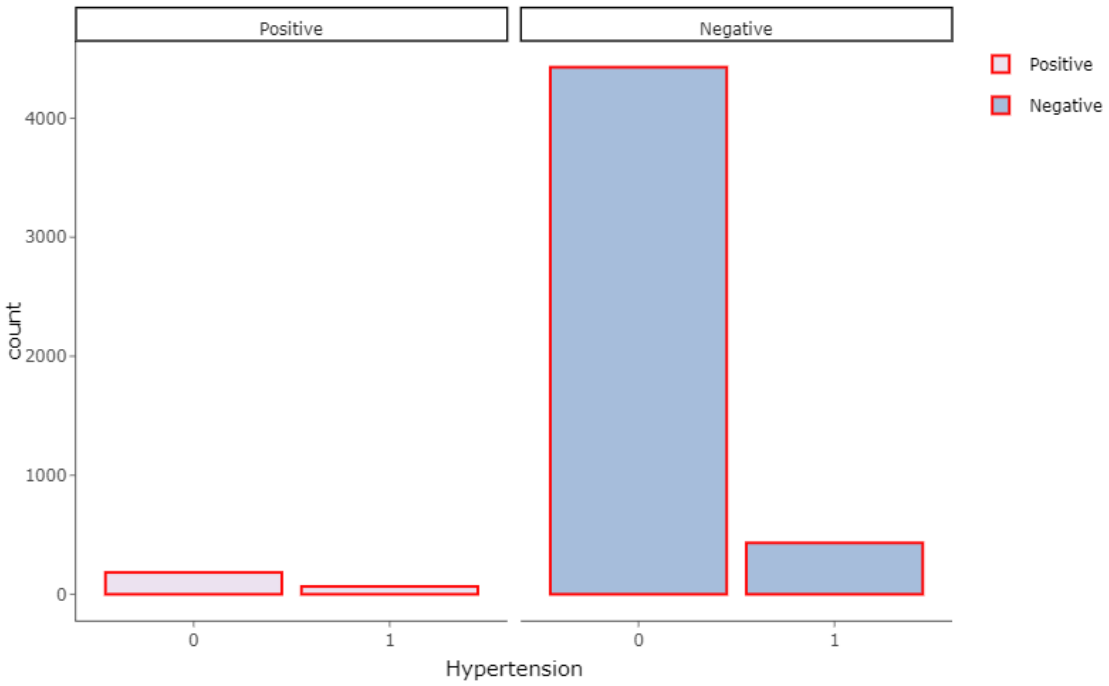

Ilustración 4: Comparación de si tienen hipertensión VS a sufrir un ictus.

En la imagen 5 se compara si el tipo de residencia donde se vive puede afectar a la hora de tener un ictus. Se observa como el 4,53% de las personas que viven en un medio rural sufren un ictus; mientras que el 5,2% de las personas que viven en la ciudad sufren un ictus. Por lo que se observa, parece que hay una tendencia ligeramente mayor a sufrir ictus en las personas que viven en la ciudad.

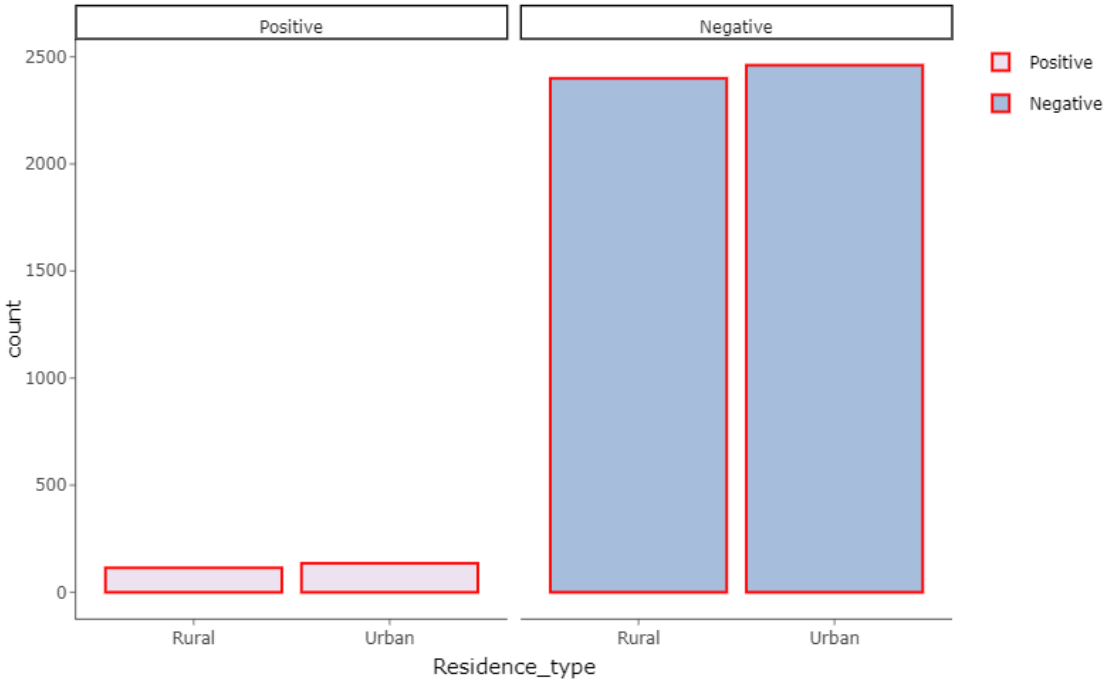

Ilustración 5: Comparación del tipo de residencia VS a sufrir un ictus.

En la imagen 6 se compara la relación que tienen los pacientes con el tabaco y su posibilidad de sufrir un ictus. Se estudian 4 grupos: ANTES fumaba, NUNCA ha fumado, FUMA y DESCONOCIDO. Las personas que antes tenían el hábito de fumar y sufren el ictus son de 7,9%. Las personas que nunca han fumado y han sufrido un ictus son el 4,75%. Y las personas que fuman tienen una posibilidad de 5,32% de sufrir ictus. Lo que queda claro es que el no fumar reduce las posibilidades de sufrir un ictus.

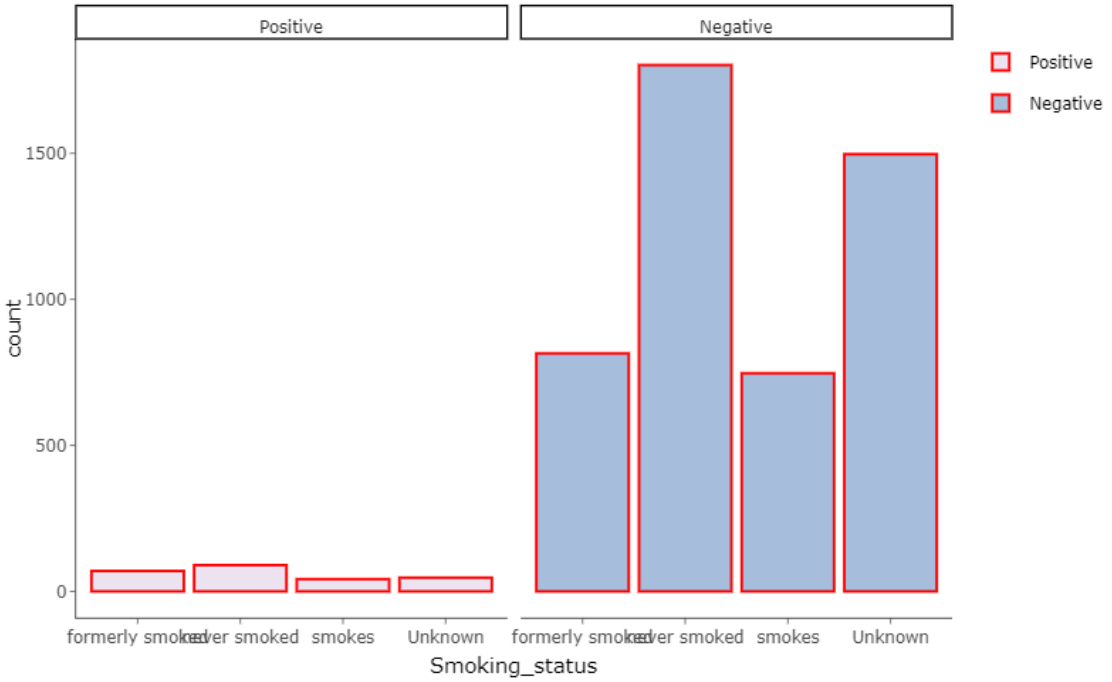

Ilustración 6: Comparación de su relación con el tabaco VS a sufrir un ictus.

En la imagen 7 se observa si el tipo de trabajo de las personas del estudio tiene relación con la posibilidad de sufrir un ictus. En esta variable se estudian 5 casos diferentes: MENOR, FUNCIONARIO, NUNCA ha trabajado, trabajo PRIVADO y AUTÓNOMO. Los menores y los que nunca han trabajado no los vamos a comentar puesto que se entiende que no tienen vida laboral; trabajar como funcionario tiene un 5,02% de sufrir un ictus; las personas que trabajan en el sector privado y sufren un ictus son el 5,09%; y los autónomos que sufren un ictus son un 7,93%. Se observa que los que mayor posibilidad de sufrir un ictus son los autónomos.

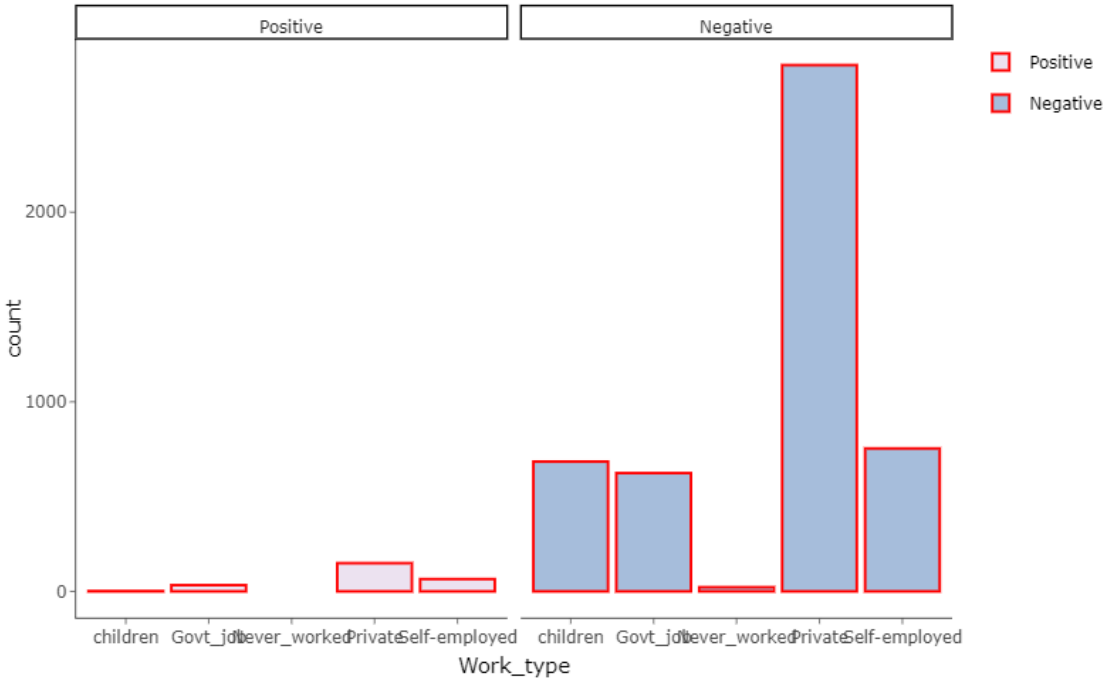

Ilustración 7: Comparación del tipo de trabajo VS a sufrir un ictus.

En la imágenes 8 y 9 podemos observar el diagrama de cajas de la edad de los pacientes que participan en el estudio. En el resumen vemos que hay pacientes menores a 1 año y pacientes que llegan a los 82 años. La mayoría de los que participan en el estudio están entre los 25 y 61 años.

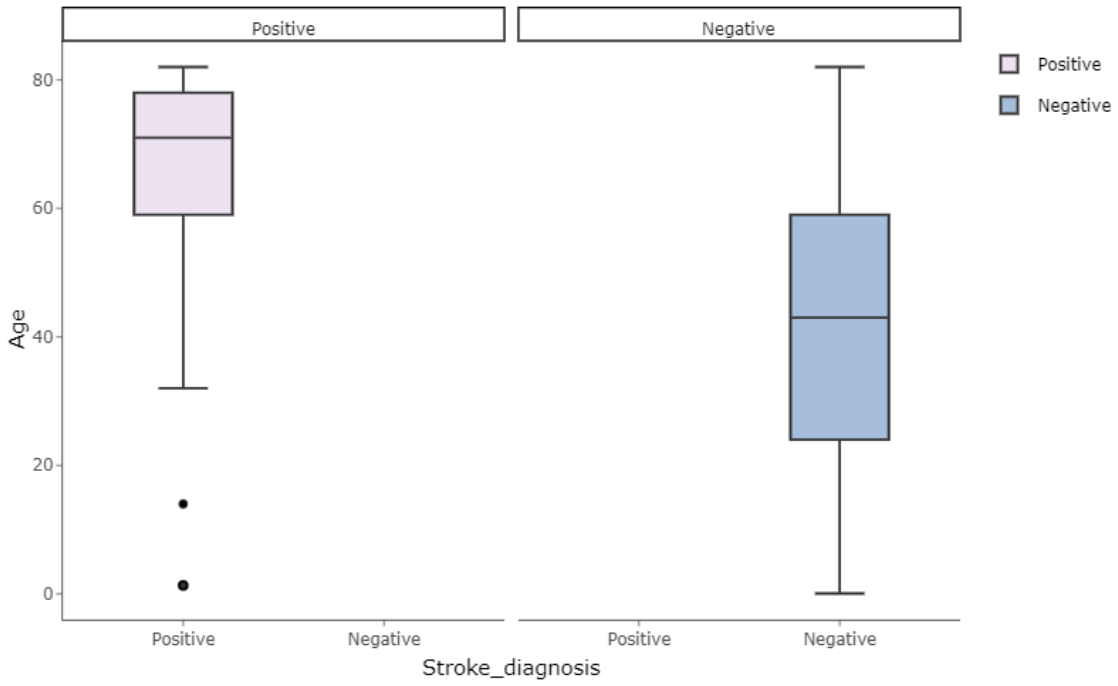

Ilustración 8: Boxplot de la edad de los pacientes que sufren o no ictus.

```r
summary(data$Age)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.08   25.00   45.00   43.23   61.00   82.00
```

Ilustración 9: Resumen del boxplot de la edad de los pacientes.

En las imágenes 10 y 11 observamos el boxplot y resumen del BMI (IMC) de los pacientes que participan en el estudio. Como mencionamos arriba, el BMI (IMC) es un dato que no suele tener diferencias abismales. Se puede comprobar que la mayoría de los datos están entre el 23.5 y 33.10. Cierto es que contamos con varios outliers que llegan a 97.6. También vemos en el resumen que hay 201 pacientes que no tenemos datos sobre su BMI (IMC), por lo que, como se ha comentado anteriormente, se sustituyen por la media del resto de datos que sí tenemos.

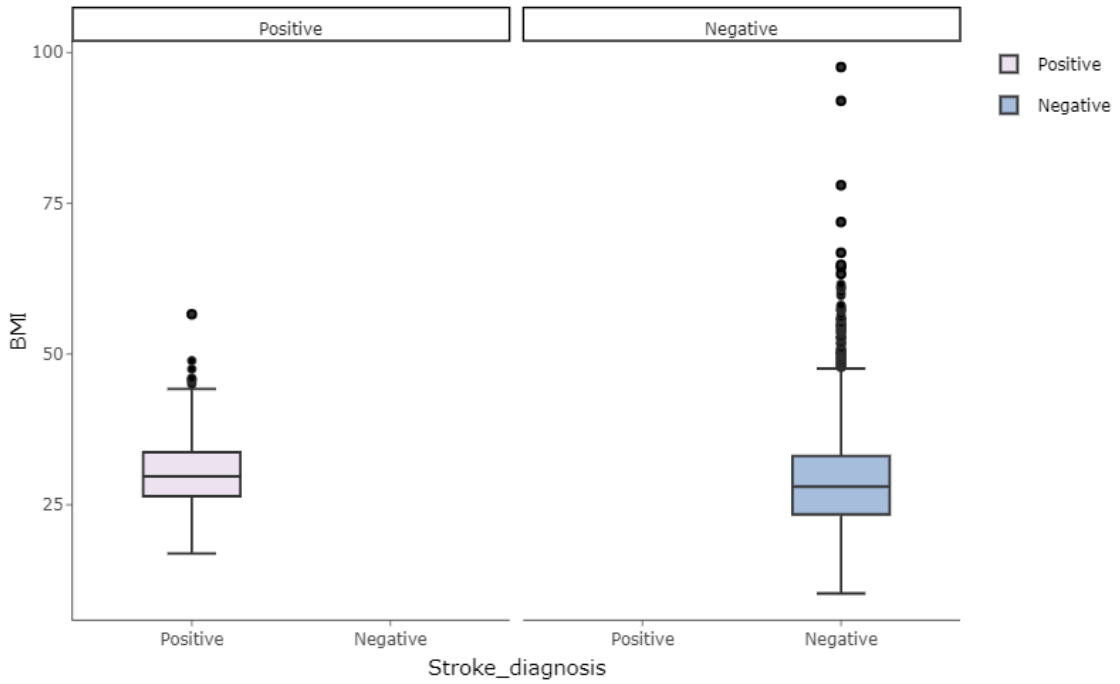

Ilustración 10: Boxplot del BMI (IMC) de los pacientes del estudio.

```r
summary(data$BMI)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  10.30   23.50   28.10   28.89   33.10   97.60     201
```

Ilustración 11: Resumen del BMI (IMC) de los pacientes del estudio.

En las imágenes 12 y 13 observamos el diagrama de cajas y el resumen del nivel medio de glucosa en sangre que presentan los participantes del estudio. En esta variable si que vemos que vemos una gran cantidad de outliers. Los datos que más predominan están entre 77.25 y 114.09.

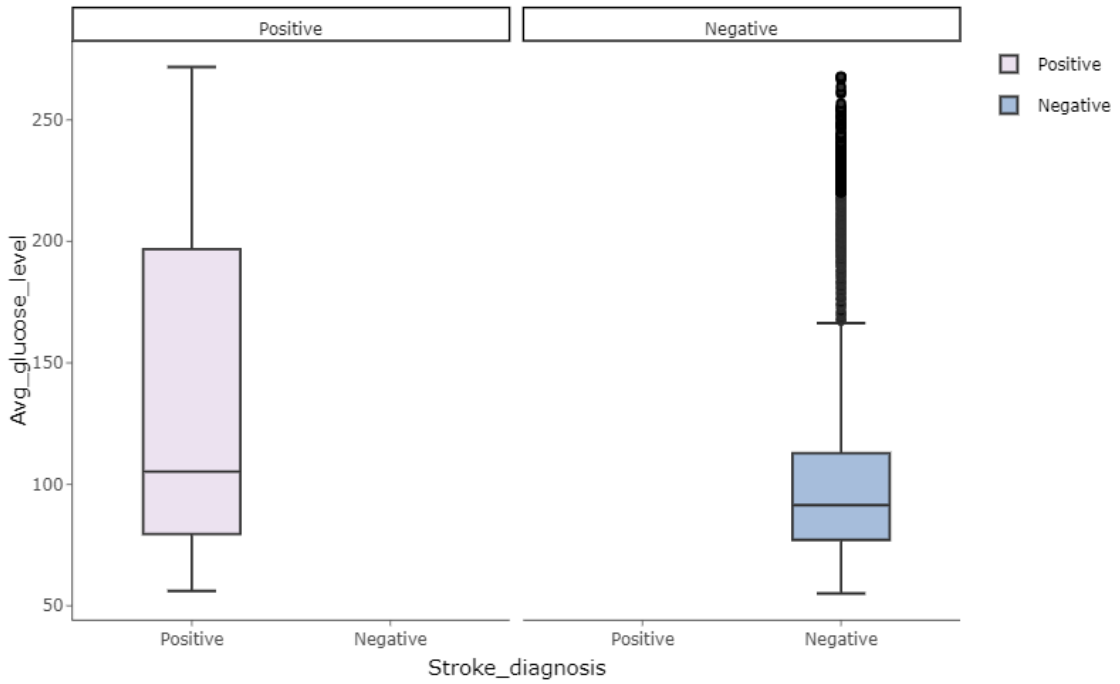

Ilustración 12: Boxplot del nivel de glucosa medio de los pacientes.

```{r}
summary(data$Avg_glucose_level)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  55.12   77.25   91.89  106.15  114.09  271.74 
```

Ilustración 13: Resumen del nivel de glucosa medio de los pacientes.

Después del pre-procesamiento, los datos se dividen en dos partes: los datos para entrenamiento y los datos para testear. A continuación, los datos de entrenamiento se introducen en los distintos algoritmos de Machine Learning para predecir el resultado del ictus.

Random Forest

Hay múltiples árboles de decisión en un clasificador Random Forest (RF) que se entrenan individualmente en una muestra aleatoria de datos. Se utiliza un proceso denominado votación para determinar el pronóstico final realizado por este algoritmo. Cada árbol de decision debe votar por una de las dos clases de salida (en este caso, ictus o no ictus). La predicción final la determina el método RF, que elige la clase con más votos (Tazin et al., 2021).

Support Vector Machine

Support Vector Machine es un algoritmo popular de aprendizaje supervisado que se utiliza en el campo médico desde hace muchos años para predecir el resultado y se utiliza para problemas de clasificación y regresión. Esta clasificación realiza el hiperplano más adecuado distinguiendo el conjunto de datos entre dos clases (Chen et al., 2023).

Naive Bayes

Naive Bayes es un clasificador sencillo y eficaz que puede predecir rápidamente el resultado en muchos problemas complicados. Funciona basándose en el teorema de Bayes y asume una determinada característica de forma independiente (Chen et al., 2023).

RESULTADOS Y DISCUSIÓN

Una vez analizados los datos que tenemos, realizamos comparaciones entre las diferentes variables, categóricas y numéricas. Para poder realizar las comparaciones en las variables categóricas utilizamos un Chi-cuadrado. Es una prueba no paramétrica y compara grupos (variables) independientes entre ellas. Aprovechando la prueba del Chi-cuadrado, realizamos también la V de Cramer, con lo que medimos la forma en la que están asociados dos campos categóricos.

	p.valor	Coeficiente_V_de_Cramer
Gender	0.7895	0.7895
Hypertension	0.0000	0.0000
Heart_disease	0.0000	0.0000
Ever_married	0.0000	0.0000
Work_type	0.0000	0.0000
Residence_type	0.2983	0.2983
Smoking_status	0.0000	0.0000

Ilustración 14: p-valor de la prueba de Chi-cuadrado & V de Cramer.

Los resultados de las pruebas se encuentran en la imagen 14. Según los resultados, las variables de Género y del tipo de Residencia tienen un p-valor mayor a 0.05, por lo que no podemos concluir que existe una diferencia significativa; por el contrario, el resto de variables categóricas tienen un p-valor menor a 0.05, por lo que se rechaza la hipótesis nula y concluimos que sí existe una diferencia significativa.

En cuanto a la V de Cramer, recordarmos que si los resultados son <0.1 significan que tienen una pequeña asociación entre las variables; 0.1-0.3 significa que tienen una asociación mediana; y si son >0.3 tienen una asociación grande.

La única variable con una asociación grande es el Género, ya que según los gráficos analizados anteriormente, se puede observar una ligera predominancia a que las mujeres sufran un ictus. La variable del tipo de Residencia tiene una asociación mediana, ya que no supera el 0.3. Y el resto de variables categóricas tienen una asociación baja.

También se realiza la comparación entre las variables numéricas. Utilizamos la función pb2gen, porque los datos de las variables numéricas son robustos, ya que al comprobar los gráficos que anteriormente hemos analizado, se observan outiliers en todas las variables.

En este test se comparan las medianas y cómo podemos ver en la imagen 15, todas las variables numéricas tienen un p-valor menos a 0.05 por lo que concluimos que sí que hay diferencias significativas y se rechaza la hipótesis nula. Se puede comprobar en los gráficos realizados anteriormente, donde se observa la diferencia entre las medianas.

	p-valor
Age	0.0000000
Avg_glucose_level	0.0000000
BMI	0.0050083

Ilustración 15: p-valor de las comparaciones de las variables numéricas.

Para calcular los estadísticos de las variables numéricas se ha realizado una tabla descriptiva donde se reúnen los estadísticos a estudiar: Mediana y Rango intercuartílico (IQR). También se le añade el p-valor que se calculó anteriormente. Queda todo recogido en la imagen 16.

Numerical variables

	Negative		Positive		
Variable	Median	IQR	Median	IQR	p-value
Age	43.00	35.00	71.00	19.00	0.0000000
Avg_glucose_level	91.47	35.71	105.22	116.92	0.0000000
BMI	28.00	9.70	29.70	7.30	0.0001026

Note:
p-value < 0.05 significant

Ilustración 16: Tabla descriptiva de los estadísticos de las variables numéricas.

Una vez realizado los estadísticos descriptivos y los p-valores obtenidos de los tests realizados, pasamos a la realización de 3 modelos predictivos. Para este estudio se han elegido Random Forest, Support Vector Machine y Naive Bayes. En cada modelo se ha realizado 10 iteraciones.

Se ha graficado los valores de sensibilidad, especificidad y AUC de cada una de la iteraciones realizadas. En la imagen 17 podemos observar la Sensibilidad, la Especificidad y el Área debajo de la curva (AUC) de los 3 modelos realizados. Estos datos son la mediana obtenida de las 10 iteraciones.

	Sensibilidad	Especificidad	AUC
Naive Bayes	0.9795918	0.3251029	0.6538906
Random forest	0.7653061	0.7479424	0.7530707
SVM	0.8163265	0.7109053	0.7618155

Ilustración 17: mediana de las 10 iteraciones realizadas en cada uno de los modelos utilizados.

Claramente, se observa en los resultados que sí que hay diferencia. También al realizar una comparación de las medianas con med1way, en la imagen 18 vemos que el p-valor de la prueba es menor a 0.05, por lo que sí que existe una diferencia significativa entre las medianas.

	p-valor
sensitivity	0
specificity	0
AUC	0

Ilustración 18: p-valor de la prueba med1way que compara las medianas de los diferentes modelos.

Para evaluar los resultados obtenidos de los diferentes modelos (imagen 17), nos basaremos en la sensibilidad y la especificidad. Recordemos que la sensibilidad es la proporción de individuos enfermos que poseen un diagnóstico positivo; mientras que la especificidad es la proporción de individuos enfermos que poseen un diagnóstico negativo.

A mayor sensibilidad, menor cantidad de datos de falsos negativos, lo que supone un éxito en el modelo.

A mayor especificidad, menor cantidad de datos de falsos positivos, lo que supone un éxito en el modelo.

AUC proporciona valores entre 0 y 1. A medida que se acerca a 1, mayor será la capacidad de diferenciar individuos sanos frente individuos enfermos.

Estas métricas son suficientemente válidas para obtener una visión general del rendimiento de los modelos, ya que al realizar 10 iteraciones no podríamos hacer la media de cada una de las matrices de confusión que se generan y obtener una matriz general.

Como hemos dicho, la sensibilidad indica la proporción de ejemplos positivos que están identificados correctamente por el modelo entre todos los positivos reales. Lo ideal es maximizar la sensibilidad, pero esta métrica por sí sola no asegura que tengamos el mejor modelo. Pero también debemos maximizar los parámetros de la especificidad y de AUC.

Si solamente nos basáramos en la sensibilidad, el modelo de Naive Bayes sería el ideal, con 0.98, pero su baja especificidad y AUC hace que en conjunto de las métricas, no sea tan ideal como parece y considerar a los demás modelos. El segundo mejor modelo en sensibilidad sería el SVM con 0.82, seguido muy de cerca con el Random Forest con 0.77.

Para el parámetro de la especificidad, el mejor modelo sería el Random Forest con 0.75, seguido muy de cerca por SVM con 0.71. Dos resultados muy parejos que se distancian mucho de Naive Bayes que solamente obtiene un 0.33.

Y en el último parámetro a evaluar, AUC, el mejor modelo sería el SVM con 0.76, prácticamente igual que el Random Forest con 0.75. Por debajo encontramos a Naive Bayes con 0.65.

Después del análisis de estos resultados, se concluye por números, que el modelo más óptimo de los 3 sea el Support Vector Machine, seguido muy de cerca por Random Forest. Una diferencia mínima, sin apenas significancia, que hace que ambos modelos sean los mejores de los evaluados.

Cuando comparamos con otros estudios, la primera “gran” diferencia es que normalmente ellos realizan tan solo una validación cruzada o una repetición del

entrenamiento y del test, por lo que se genera una matriz de confusión, a partir de la cual, los autores evalúan la precisión de cada modelo predictivo. En el caso del trabajo de Dritsas & Trigka, 2022, que utilizaron una gran variedad de modelos para predecir la probabilidad de sufrir ictus, entre los que destacamos Naive Bayes y Random Forest, uno de los que mejor resultado de precisión obtuvo fue Random Forest con un 97% de precisión. Este resultado coincide con nuestros resultados: Random Forest es más óptimo que Naive Bayes.

En el estudio de Tazin et al., 2021, Random Forest también fue el mejor modelo de los estudiados. En este estudio se evaluaron los modelos en función del F1 score. Se vuelve a observar que el Random Forest es un modelo predictivo muy óptimo, como nos dicen nuestros resultados.

En el estudio de Chen et al., 2023, se observó cómo tanto el SVM como el Random Forest fueron los que mejor precisión obtuvieron de los once diferentes modelos que se evaluaron. 99.99% y 99.87% respectivamente. Coincide otra vez con nuestros resultados, siendo ligeramente superior el SVM al Random forest. Una diferencia, que a mi parecer, no se podría descartar Random Forest por obtener dichas métricas. Por lo que volvemos a destacar, la gran validez de ambos modelos.

En el estudio de Alageel et al., 2023, se evaluaron 7 diferentes modelos predictivos entre los que se encuentran los 3 que se han realizado en este estudio. Otra vez se evaluó por su parte la precisión y en la nuestra la sensibilidad. Pero en ambos trabajos, los datos coinciden que el modelo más bajo, o menos óptimo, es el Naive Bayes y que tanto SVM como Random Forest tienen mejores resultados y están a la par entre ellos.

CONCLUSIONES

El ictus se encuentra entre los accidentes médicos más comunes que conducen a la muerte, pero incluso en caso de supervivencia, el ictus deja graves secuelas en la vida de sus pacientes. Dado que existen varios factores de riesgo que aumentan las probabilidades de sufrir un ictus, es posible predecirlo de antemano. Para ello se han empleado algoritmos de Machine Learning que prometen resultados de predicción rápidos y eficaces.

En este estudio se investiga la eficacia de varios algoritmos de ML para predecir adecuadamente el ictus basándose en una serie de variables fisiológicas recopiladas de una muestra de pacientes. Se emplean 3 diferentes modelos predictivos: Random Forest, Support Vector Machine y Naive Bayes. Se elige el modelo más óptimo en función de los resultados obtenidos. Para ello, se evalúan mediante la sensibilidad, especificidad y la AUC obtenida.

Destacamos que, a pesar de la gran sensibilidad que ofrece el modelo Naive Bayes (0,98), en conjunto con las demás métricas, hace que no sea tan favorable como por cifras de sensibilidad puede parecer. Sus métricas de especificidad y AUC son las más bajas de los 3 modelos, en especial la especificidad (proporción de verdaderos negativos) que tan solo obtiene un 0,33. En cambio, los modelos Random Forest y SVM son más planos en sus resultados de las métricas. Son datos altos que rondan siempre el 0,8, dando suficiente veracidad a las 3 métricas, y por lo tanto, al modelo en sí.

El objetivo futuro es seguir mejorando los algoritmos de Machine Learning. Por último, una dirección desafiante pero prometedora es recopilar datos de imágenes de tomografías computarizadas cerebrales y evaluar la capacidad predictiva de los modelos de aprendizaje profundo en la ocurrencia de accidentes cerebrovasculares.

BIBLIOGRAFIA

- Alageel, N., Alharbi, R., Alharbi, R., Alsayil, M., & Alharbi, L. (2023). Using Machine Learning Algorithm as a Methos for Improving Stroke Prediction. International Journal of Advanced Computer Science and Applicants, 14 (4). http://dx.doi.org/10.14569/IJACSA.2023.0140481
- Benjamin, E.J., Muntner, P., Alonso, A., et al. (2019). Heart Disease and Stroke Statistics—2019 Update: A Report From the American Heart Association. Circulation, 139(10), e56–e528.
- Chen, M., Tan, X., & Padman, R. (2023). A Machine Learning Approach to Support Urgent Stroke Triage Using Administrative Data and Social Determinants of Health at Hospital Presentation: Retrospective Study. Journal of medical Internet research, 25, e36477. https://doi.org/10.2196/36477
- Delpont, B., Blanc, C., Osseby, G. V., Hervieu-Bègue, M., Giroud, M., & Béjot, Y. (2018). Pain after stroke: A review. Revue neurologique, 174(10), 671–674. https://doi.org/10.1016/j.neurol.2017.11.011
- Dritsas, E., & Trigka, M. (2022). Stroke Risk Prediction with Machine Learning Techniques. Sensors (Basel, Switzerland), 22(13), 4670. https://doi.org/10.3390/s22134670
- Feigin, V.L., Krishnamurthi, R.V., Parmar, P., et al. (2015). Update on the Global Burden of Ischemic and Hemorrhagic Stroke in 1990-2013: The GBD 2013 Study. Neuroepidemiology, 45(3), 161-176.
- Jain, S., Levy, M., Lin, R., et al. (2020). Machine Learning Predictive Model for the Risk of Ischemic Stroke. Frontiers in Neurology, 11, 582511.
- O'Donnell, M.J., Chin, S.L., Rangarajan, S., et al. (2016). Global and regional effects of potentially modifiable risk factors associated with acute stroke in 32 countries (INTERSTROKE): a case-control study. The Lancet, 388(10046), 761-775.
- Powers, W.J., Rabinstein, A.A., Ackerson, T., et al. (2018). 2018 Guidelines for the Early Management of Patients With Acute Ischemic Stroke: A Guideline for Healthcare Professionals From the American Heart Association/American Stroke Association. Stroke, 49(3), e46-e110.
- Saposnik, G., Redelmeier, D., Ruff, C.C., et al. (2020). Cognitive biases associated with medical decisions: A systematic review. BMC Medical Informatics and Decision Making, 20(1), 1-11.

- T. I. Shoily, T. Islam, S. Jannat, S. A. Tanna, T. M. Alif and R. R. Ema, "Detection of Stroke Disease using Machine Learning Algorithms," 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 2019, pp. 1-6, doi: 10.1109/ICCCNT45670.2019.8944689.
- Tazin, T., Alam, M. N., Dola, N. N., Bari, M. S., Bourouis, S., & Monirujjaman Khan, M. (2021). Stroke Disease Detection and Prediction Using Robust Learning Approaches. Journal of healthcare engineering, 2021, 7633381. https://doi.org/10.1155/2021/7633381
- Wiryaseputra, M. (2017). Stroke Prediction Using Machine Learning Classification Algorithm. International Journal of Scientific & Engineering Research, 8 (1).

ANEXOS

CÓDIGO TFM

````
---
title: "TFM"
author: "Pablo Vercet"
date: "2024-06-17"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

###Cargar librerias

```{r}
library(caret) # models, createDataPartition
library(ConfusionTableR)
library(DataExplorer)
library(dplyr)
library(ggplot2)
library(ggthemes)
library(kableExtra)
library(ModelMetrics)
library(openxlsx)
library(plotly)
library(probably) # for balancing performance
library(pROC) # AUC
library(psych)
````
````

library(purrr) # map

library(randomForest)

library(reshape2)

library(skimr) # descriptive stats

library(stringr)

library(tidymodels)

library(tidyverse) # %>%

library(univariateML)

library(vip) # for variable importance

library(xgboost)

library(WRS2)

library(corrplot)

library(vcd)

library(gtsummary)

library(ROCR)

library(kernlab)

```

### Cargar el dataset

```{r}

data = read.csv("stroke.csv")

data %>% head() %>% kable %>% kable_styling()

```

Cambio de nombre las variables para que todas empiecen por letra mayuscula
```

```{r}

names(data) = c("Id", "Gender", "Age", "Hypertension", "Heart_disease", "Ever_married", "Work_type", "Residence_type", "Avg_glucose_level", "BMI", "Smoking_status", "Stroke")

data %>% head() %>% kable %>% kable_styling("striped", "bordered", "hover", full_width = TRUE, position = "center", font_size = 15) %>% row_spec(0, align = "center", bold = TRUE)

```

```{r}

str(data$Stroke)

```

###Cambiar mi variable predictora de numérica a factor

```{r}

data = data %>%

  mutate(Stroke_diagnosis = if_else(str_detect(Stroke, "0"), "Negative", "Positive")) %>%

  mutate(Stroke_diagnosis = factor(Stroke_diagnosis, levels = c("Positive", "Negative"), labels = c("Positive", "Negative"))) %>%

  relocate(Stroke_diagnosis, .before = Stroke) %>%

  select(-Stroke)

DT::datatable(data, rownames = FALSE, option = list(pageLength = 10, scrollx = TRUE), class = "white_space:nowrap")

```

###Filas y columnas con las que estamos trabajando

```{r}
dim(data)
```

```{r}
str(data)
```

###Establecer las variables que desee como factor

```{r}
columnas = c(2, 4:8, 11, 12)

data[,columnas] = lapply(data[,columnas], as.factor)

data$BMI = as.numeric(data$BMI)
```

```{r}
str(data)
```

###Comprobar si existen valores ausentes

```{r}
```

na_values = function(x){sum(is.na(x))}

sapply(data, na_values)

```

###Vista general y rápida de los datos de BMI

```{r}

data$BMI

```

###Elimino la variable ID que no nos aporta ningún tipo de valor relevante

```{r}

data = data[ ,-1]

data %>% head() %>% kable %>% kable_styling("striped", "bordered", "hover", full_width = TRUE, position = "center", font_size = 15) %>% row_spec(0, align = "center", bold = TRUE)

```

###GRAFICOS ####Variables Factor/Categóricas

```{r}

p1 = ggplot(data, aes(Gender)) + geom_bar(aes(fill = Stroke_diagnosis), color = "red", show.legend = FALSE) + facet_wrap(~Stroke_diagnosis) + scale_fill_brewer(palette = 10) + theme_classic()

ggplotly(p1)

```
p3 = ggplot(data, aes(Hypertension)) + geom_bar(aes(fill = Stroke_diagnosis), color = "red",
show.legend = FALSE) + facet_wrap(~Stroke_diagnosis) + scale_fill_brewer(palette = 10) +
theme_classic()

ggplotly(p3)

p4 = ggplot(data, aes(Heart_disease)) + geom_bar(aes(fill = Stroke_diagnosis), color = "red",
show.legend = FALSE) + facet_wrap(~Stroke_diagnosis) + scale_fill_brewer(palette = 10) +
theme_classic()

ggplotly(p4)

p5 = ggplot(data, aes(Ever_married)) + geom_bar(aes(fill = Stroke_diagnosis), color = "red",
show.legend = FALSE) + facet_wrap(~Stroke_diagnosis) + scale_fill_brewer(palette = 10) +
theme_classic()

ggplotly(p5)

p6 = ggplot(data, aes(Work_type)) + geom_bar(aes(fill = Stroke_diagnosis), color = "red",
show.legend = FALSE) + facet_wrap(~Stroke_diagnosis) + scale_fill_brewer(palette = 10) +
theme_classic()

ggplotly(p6)

p7 = ggplot(data, aes(Residence_type)) + geom_bar(aes(fill = Stroke_diagnosis), color = "red",
show.legend = FALSE) + facet_wrap(~Stroke_diagnosis) + scale_fill_brewer(palette = 10) +
theme_classic()

ggplotly(p7)

p10 = ggplot(data, aes(Smoking_status)) + geom_bar(aes(fill = Stroke_diagnosis), color = "red",
show.legend = FALSE) + facet_wrap(~Stroke_diagnosis) + scale_fill_brewer(palette = 10) +
theme_classic()

ggplotly(p10)
```

```

####Variables numéricas

```{r}

p2 = ggplot(data, aes(Stroke_diagnosis, Age)) + geom_boxplot(aes(fill = Stroke_diagnosis), show.legend = FALSE) + facet_wrap(~Stroke_diagnosis) + scale_fill_brewer(palette = 10) + theme_classic()

ggplotly(p2)

p8 = ggplot(data, aes(Stroke_diagnosis, Avg_glucose_level)) + geom_boxplot(aes(fill = Stroke_diagnosis), show.legend = FALSE) + facet_wrap(~Stroke_diagnosis) + scale_fill_brewer(palette = 10) + theme_classic()

ggplotly(p8)

p9 = ggplot(data, aes(Stroke_diagnosis, BMI)) + geom_boxplot(aes(fill = Stroke_diagnosis), show.legend = FALSE) + facet_wrap(~Stroke_diagnosis) + scale_fill_brewer(palette = 10) + theme_classic()

ggplotly(p9)

```

```{r}

summary(data$Age)

```

```{r}

summary(data$Avg_glucose_level)
```
```

```

```{r}

summary(data$BMI)

```

###Comparaciones ####Variables Factor/categóricas

```{r}

m = matrix(nrow = 7, ncol = 2, dimnames = list(colnames(data[,c(1,3:7, 10)]), c("p-valor","Coeficiente_V_de_Cramer")))

for (i in c(1,3:7, 10)) {
  tabla = table(data$Stroke_diagnosis,data[[i]]
      )
test = chisq.test(tabla)
cramer = function(x){
  unname(sqrt(chisq.test(x)$statistics/(sum(x)*(min(dim(x))-1))))

  }

m[colnames(data)[i], ]= c(round(test$p.value,4), round(cramer(tabla),4))

}

m = data.frame(m)
```
```

```
formato = c("striped", "hover", "condensed")


m %>% kable() %>% kable_styling(bootstrap_options = formato,
                                position = "center",
                                full_width = FALSE) %>%
  column_spec(2, background = ifelse(m$p.valor > (0.05), "#ffda9e", "#fdf9c4"))
```

```
####Variables numéricas ###Variables ROBUSTAS
```

```{r}
tabla_p_valor_1 = matrix(nrow = 3, ncol = 1, dimnames = list(colnames(data[,c(2,8,9)]), c("p-valor")))


for (i in c(2,8,9)) {
  f= formula(paste(colnames(data)[i], "~Stroke_diagnosis"))
  test = pb2gen(f, data = data, est = "median")
  tabla_p_valor_1[colnames(data)[i], ] = c(test$p.value)
}


tabla_p_valor_1 %>% kable()%>% kable_styling("striped", "hover", "condensed", position = "center", full_width = FALSE)
```

```
###Cálculo estadísticos de las variables numéricas #Stroke = 0
```

```{r}
```

```
stroke_0 = data |> filter(Stroke_diagnosis%in%c("Negative"))

names_num = colnames(select_if(stroke_0, is.numeric))

data_num_stroke_0 = stroke_0 |> dplyr::select(all_of(names_num))

results_data_num_stroke_0 = NULL

for (i in colnames(data_num_stroke_0)) {
  median = median(data_num_stroke_0[[i]], na.rm = TRUE)
  iqr = IQR(data_num_stroke_0[[i]], na.rm = TRUE)
  row = data.frame("Variable" = i, "Median_Stroke_0" = median, "IQR_Stroke_0" = iqr)
  results_data_num_stroke_0 = rbind(results_data_num_stroke_0, row)
}
rownames(results_data_num_stroke_0)=NULL
results_data_num_stroke_0 %>% kable()%>% kable_styling(position = "center", full_width = FALSE)
```

```
#Stroke = 1

```{r}
stroke_1 = data |> filter(Stroke_diagnosis%in%c("Positive"))

names_num = colnames(select_if(stroke_1, is.numeric))

data_num_stroke_1 = stroke_1|> dplyr::select(all_of(names_num))
```
```

```
results_data_num_stroke_1 = NULL


for (i in colnames(data_num_stroke_1)) {
  median = median(data_num_stroke_1[[i]], na.rm = TRUE)
  iqr = IQR(data_num_stroke_1[[i]], na.rm = TRUE)
  row = data.frame("Variable" = i, "Median_Stroke_1" = median, "IQR_Stroke_1" = iqr)
  results_data_num_stroke_1 = rbind(results_data_num_stroke_1, row)
}
rownames(results_data_num_stroke_1)=NULL
results_data_num_stroke_1 %>% kable()%>% kable_styling(position = "center", full_width = FALSE)
```

```
#tablas descriptivas


```{r}
descriptive_table = data.frame(
  variable = c("Age", "Avg_glucose_level", "BMI"),
  median_stroke_0 = results_data_num_stroke_0$Median_Stroke_0,
  IQR_stroke_0 = results_data_num_stroke_0$IQR_Stroke_0,
  median_stroke_1 = results_data_num_stroke_1$Median_Stroke_1,
  IQR_stroke_1 = results_data_num_stroke_1$IQR_Stroke_1,
  p_value = tabla_p_valor)


descriptive_table
```
```

```
names(descriptive_table) = c("Variable", "Median", "IQR", "Median", "IQR", "p-value")


rownames(descriptive_table) = NULL


descriptive_table



kbl(descriptive_table,
    caption = "Numerical variables") %>%
  kable_paper("striped", full_width = FALSE) %>%
  add_header_above(c(" " = 1, "Negative" = 2, "Positive" = 2, " " = 1),
                   italic = TRUE,
                   align = "c",
                   color = "white",
                   background = "lightsalmon") %>%
  column_spec(6, background = ifelse(tabla_p_valor > (0.05), "#ffda9e", "#fdf9c4")) %>%
  footnote(general = "p-value < 0.05 significant")
```

```

###Sustitución de los N/A por la MEDIANA del resto de datos de BMI

```{r}
data$BMI[is.na(data$BMI)] = mean(data$BMI, na.rm = TRUE)

data %>% head() %>% kable() %>% kable_styling("striped", "bordered", "hover", full_width = TRUE, position = "center", font_size = 15) %>% row_spec(0, align = "center", bold = TRUE)
```
```

#Modelo de Random Forest

```{r}
library(tidymodels)
library(caret)
library(ROCR)

modelo_rf = NULL


for (i in (1:10)){
  set.seed(i)
  train_row_numbers = createDataPartition(data$Stroke_diagnosis, p = 0.8, list = FALSE)
  data_train = data[train_row_numbers, ]
  data_test = data[-train_row_numbers, ]

  transformer = recipe(formula = Stroke_diagnosis ~ .,
                       data = data_train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_center(where(is.numeric)) %>%
  step_scale(where(is.numeric))

  prep_data = prep(transformer)
  data_train = bake(prep_data, new_data = data_train)
  data_test = bake(prep_data, new_data = data_test)
```

```
ctrl = trainControl(method = "cv",
                    number = 10,
                    returnResamp = "final",
                    verboseIter = FALSE,
                    summaryFunction = twoClassSummary,
                    classProbs = TRUE,
                    savePredictions = T,
                    allowParallel = TRUE,
                    sampling = "up")



tuneGrid = expand.grid(mtry = 1:10)


set.seed(i)
rf_fit = train(Stroke_diagnosis ~ .,
               data = data_train,
               method = "rf",
               metric = "ROC",
               trControl = ctrl,
               tuneGrid = tuneGrid,
               )



probs = seq(0.1, 0.9, by = 0.1)
set.seed(i)
ths_rf_fit = thresholder(rf_fit,
                         threshold = probs,
```

```
                    final = TRUE,
                    statistics = "all")



  ths_rf_fit %>%
  mutate(prob = probs) %>%
  filter(J == max(J)) %>%
  pull(prob) -> thresh_prob_rf_fit
  ths_rf_fit %>%
  mutate(prob = probs) %>%
  filter(J == max(J)) %>%
  pull(J) -> max_J_train
  preds = as.factor(ifelse(predict(rf_fit, data_test, type = "prob")[,"Positive"] >=
thresh_prob_rf_fit,"Positive","Negative"))
  real = factor(data_test$Stroke_diagnosis)
  cm = ConfusionTableR::binary_class_cm(preds,
                          real,
                          mode = 'everything',
                          positive = 'Positive')



  sensitivity = cm$confusion_matrix$byClass[1]
  specificity = cm$confusion_matrix$byClass[2]
  df = data.frame(preds = preds, real = real)
  df$preds = as.numeric(ifelse(df$preds == "Positive", 1, 0))
  df$real = as.numeric(ifelse(df$real == "Positive", 1, 0))
  prediction = prediction(df$preds, df$real)
  AUC = as.numeric(performance(prediction,"auc")@y.values)
```

```
  row = data.frame(model = "Random forest",
                   seed = i,
                   probab = thresh_prob_rf_fit,
                   max_J_train = max_J_train,
                   sensitivity = sensitivity,
                   specificity = specificity,
                   AUC = AUC)
  modelo_rf = rbind(modelo_rf, row)

}
```

```
modelo_rf %>% kable() %>% kable_styling()
```

```
###guardar modelo
```

```{r}
write.xlsx(modelo_rf, "RF_resultados_TFM.xlsx")
```

```
#Modelo Support Vector Machine linear
```

```{r}
modelo_svm = NULL



for (i in (1:10)){
  set.seed(i)
  train_row_numbers = createDataPartition(data$Stroke_diagnosis, p = 0.8, list = FALSE)
  data_train = data[train_row_numbers, ]
  data_test = data[-train_row_numbers, ]

  transformer = recipe(formula = Stroke_diagnosis ~ .,
                       data = data_train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_center(where(is.numeric)) %>%
  step_scale(where(is.numeric))



  prep_data = prep(transformer)
  data_train = bake(prep_data, new_data = data_train)
  data_test = bake(prep_data, new_data = data_test)

  ctrl = trainControl(method = "cv",
                      number = 10,
                      returnResamp = "final",
                      verboseIter = FALSE,
                      summaryFunction = twoClassSummary,
                      classProbs = TRUE,
```

```
                savePredictions = T,
                allowParallel = TRUE,
                sampling = "up")



tuneGrid = expand.grid(C = seq(0, 2, length = 20))


set.seed(i)
svm_fit = train(Stroke_diagnosis ~ .,
          data = data_train,
          method = "svmLinear",
          metric = "ROC",
          trControl = ctrl,
          tuneGrid = tuneGrid,
          )



probs = seq(0.1, 0.9, by = 0.1)
set.seed(i)
ths_svm_fit = thresholder(svm_fit,
                          threshold = probs,
                          final = TRUE,
                          statistics = "all")



ths_svm_fit %>%
mutate(prob = probs) %>%
```

```
  filter(J == max(J)) %>%
  pull(prob) -> thresh_prob_svm_fit
  ths_svm_fit %>%
  mutate(prob = probs) %>%
  filter(J == max(J)) %>%
  pull(J) -> max_J_train
  preds = as.factor(ifelse(predict(svm_fit, data_test, type = "prob")[,"Positive"] >=
thresh_prob_svm_fit,"Positive","Negative"))
  real = factor(data_test$Stroke_diagnosis)
  cm = ConfusionTableR::binary_class_cm(preds,
                                        real,
                                        mode = 'everything',
                                        positive = 'Positive')


  sensitivity = cm$confusion_matrix$byClass[1]
  specificity = cm$confusion_matrix$byClass[2]
  df = data.frame(preds = preds, real = real)
  df$preds = as.numeric(ifelse(df$preds == "Positive", 1, 0))
  df$real = as.numeric(ifelse(df$real == "Positive", 1, 0))
  prediction = prediction(df$preds, df$real)
  AUC = as.numeric(performance(prediction,"auc")@y.values)
  row = data.frame(model = "SVM",
                   seed = i,
                   probab = thresh_prob_svm_fit,
                   max_J_train = max_J_train,
                   sensitivity = sensitivity,
                   specificity = specificity,
```

```
                AUC = AUC)

  modelo_svm = rbind(modelo_svm, row)

}
```

```
modelo_svm %>% kable() %>% kable_styling()
```

```
###Guardar en excel
```

```{r}
write.xlsx(modelo_svm, "SVM_resultados_TFM.xlsx")
```

```
#Modelo Naives Bayes
```

```{r}
library(caret)
library(caretEnsemble)

modelo_NB = NULL
```

```
for (i in (1:10)){
  set.seed(i)
  train_row_numbers = createDataPartition(data$Stroke_diagnosis, p = 0.8, list = FALSE)
  data_train = data[train_row_numbers, ]
  data_test = data[-train_row_numbers, ]

  transformer = recipe(formula = Stroke_diagnosis ~ .,
                       data = data_train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_center(where(is.numeric)) %>%
  step_scale(where(is.numeric))


  prep_data = prep(transformer)
  data_train = bake(prep_data, new_data = data_train)
  data_test = bake(prep_data, new_data = data_test)

  ctrl = trainControl(method = "cv",
                      number = 10,
                      returnResamp = "final",
                      verboseIter = FALSE,
                      summaryFunction = twoClassSummary,
                      classProbs = TRUE,
                      savePredictions = T,
                      allowParallel = TRUE,
                      sampling = "up")
```

```
tuneGrid = expand.grid(usekernel = c(TRUE, FALSE), laplace = 0:5, adjust = seq(0,5,0.5))


set.seed(i)
NB_fit = train(Stroke_diagnosis ~ .,
          data = data_train,
          method = "naive_bayes",
          metric = "ROC",
          trControl = ctrl,
          tuneGrid = tuneGrid,
          )



probs = seq(0.1, 0.9, by = 0.1)
set.seed(i)
ths_NB_fit = thresholder(NB_fit,
                         threshold = probs,
                         final = TRUE,
                         statistics = "all")



ths_NB_fit %>%
mutate(prob = probs) %>%
filter(J == max(J)) %>%
pull(prob) -> thresh_prob_NB_fit
ths_NB_fit %>%
mutate(prob = probs) %>%
```

```
  filter(J == max(J)) %>%
  pull(J) -> max_J_train
  preds = as.factor(ifelse(predict(NB_fit, data_test, type = "prob")[,"Positive"] >=
thresh_prob_NB_fit,"Positive","Negative"))
  real = factor(data_test$Stroke_diagnosis)
  cm = ConfusionTableR::binary_class_cm(preds,
                                        real,
                                        mode = 'everything',
                                        positive = 'Positive')



  sensitivity = cm$confusion_matrix$byClass[1]
  specificity = cm$confusion_matrix$byClass[2]
  df = data.frame(preds = preds, real = real)
  df$preds = as.numeric(ifelse(df$preds == "Positive", 1, 0))
  df$real = as.numeric(ifelse(df$real == "Positive", 1, 0))
  prediction = prediction(df$preds, df$real)
  AUC = as.numeric(performance(prediction,"auc")@y.values)
  row = data.frame(model = "Naive Bayes",
                   seed = i,
                   probab = thresh_prob_NB_fit,
                   max_J_train = max_J_train,
                   sensitivity = sensitivity,
                   specificity = specificity,
                   AUC = AUC)


  modelo_NB = rbind(modelo_NB, row)
```

```
}
```

```
modelo_NB %>% kable() %>% kable_styling()
```

```
###Guardar en excel
```

```{r}
write.xlsx(modelo_NB, "NB_resultados_TFM_1.xlsx")
```

```
###Juntar los resultados
```

```{r}
resultados_juntos = rbind(modelo_rf, modelo_svm, modelo_NB)
resultados_juntos %>% kable() %>% kable_styling()
```

```
###Eliminar columnas que no interesan
```

```{r}
resultados_juntos["X"] = NULL
resultados_juntos["seed"] = NULL
resultados_juntos["max_J_train"] = NULL
resultados_juntos["probab"] = NULL
```

```
resultados_juntos$model = as.factor(as.character(resultados_juntos$model))

formato = c("striped", "hover", "responsive")

resultados_juntos %>% kable() %>% kable_styling(bootstrap_options = formato, full_width = FALSE, position = "center", font_size = 12) %>% row_spec(0, bold = TRUE, color = "black", align = "center") %>% row_spec(1:nrow(resultados_juntos), bold = TRUE, color = "grey", align = "center")

write.csv(resultados_juntos, "resultados_modelos_TFM.csv")
```

```
###Sensibilidad

```{r}
grafico_sensibilidad = ggplot(resultados_juntos, aes(x=model, y=resultados_juntos$sensitivity, fill=model)) + geom_boxplot() + scale_fill_brewer(palette = 10) + theme_classic()

ggplotly(grafico_sensibilidad)
```

```{r}
sensibilidad = tapply(resultados_juntos$sensitivity, resultados_juntos$model, median)

sensibilidad = data.frame(sensibilidad)

names(sensibilidad) = c("Sensibilidad")
```

```
###Especificidad

```{r}
```

```
grafico_especificidad = ggplot(resultados_juntos, aes(x=model,
y=resultados_juntos$specificity, fill=model)) + geom_boxplot() + scale_fill_brewer(palette =
10) + theme_classic()

ggplotly(grafico_especificidad)
```

```{r}
especificidad = tapply(resultados_juntos$specificity, resultados_juntos$model, median)

especificidad = data.frame(especificidad)

names(especificidad) = c("Especificidad")
```

###AUC

```{r}
grafico_AUC = ggplot(resultados_juntos, aes(x=model, y=resultados_juntos$AUC, fill=model)) +
geom_boxplot() + scale_fill_brewer(palette = 10) + theme_classic()

ggplotly(grafico_AUC)
```

```{r}
AUC = tapply(resultados_juntos$AUC, resultados_juntos$model, median)

AUC = data.frame(AUC)

names(AUC) = c("AUC")
```

###Resultado final

```{r}

final = cbind(sensibilidad, especificidad, AUC)

formato = c("striped", "bordered", "hover", "responsive")

final %>% kable() %>% kable_styling(bootstrap_options = formato, full_width = FALSE, position = "center", font_size = 12)

```

###Comparar métricas

```{r}

comparación = melt(resultados_juntos, id.vars = "model", measure.vars = 2:4)

ggplot(comparación, aes(x=model, y=value)) + geom_boxplot() + facet_wrap(~variable, nrow = 3, ncol = 1, scales = "free")

```

###Comparar medianas

```{r}

medianas = matrix(nrow = 3, ncol = 1, dimnames = list(colnames(resultados_juntos)[-1], c("p-valor")))

medianas

for (i in 2:4) {

  f = formula(paste(colnames(resultados_juntos)[i], "~model"))

  test = med1way(f, data = resultados_juntos)

  medianas[colnames(resultados_juntos)[i], ] = c(round(test$p.value,4))

}
```

formato = c("striped", "bordered", "hover", "responsive")

medianas %>% kable() %>% kable_styling(bootstrap_options = formato, full_width = FALSE, position = "center", font_size = 12)

```

## CÓDIGO DASHBOARD

---

title: "Dashboard Stroke TFM"

author: "Pablo Vercet"

date: "2024-06-11"

output:

flexdashboard::flex_dashboard:

orientation: columns

vertical_layout: fill

theme: space_lab

runtime: shiny

---

```{r setup, include=FALSE}

knitr::opts_chunk$set(echo = TRUE)

```

This R Markdown document is made interactive using Shiny. Unlike the more traditional workflow of creating static reports, you can now create documents that allow your readers to change the assumptions underlying your analysis and see the results immediately.
```

To learn more, see [Interactive Documents](http://rmarkdown.rstudio.com/authoring_shiny.html).

Note the use of the `height` parameter to determine how much vertical space the embedded application should occupy.

You can also use the `shinyApp` function to define an application inline rather then in an external directory.

In all of R code chunks above the `echo = FALSE` attribute is used. This is to prevent the R code within the chunk from rendering in the document alongside the Shiny components.

```{r}
getOption("repos")
```

###Cargar librerias

```{r}
library(caret) # models, createDataPartition
library(ConfusionTableR)
library(DataExplorer)
library(dplyr)
library(ggplot2)
```

```
library(ggthemes)

library(kableExtra)

library(ModelMetrics)

library(openxlsx)

library(plotly)

library(probably) # for balancing performance

library(pROC) # AUC

library(psych)

library(purrr) # map

library(randomForest)

library(reshape2)

library(skimr) # descriptive stats

library(stringr)

library(tidymodels)

library(tidyverse) # %>%

library(univariateML)

library(vip) # for variable importance

library(xgboost)

library(WRS2)

library(corrplot)

library(vcd)

library(gtsummary)

library(ROCR)
```

library(kernlab)

library(shiny)

library(selectr)

```

### Cargar el dataset

```{r}

data = read.csv("stroke.csv")

data %>% head() %>% kable %>% kable_styling()

```

Cambio de nombre las variables para que todas empiecen por letra mayuscula

```{r}

names(data) = c("Id", "Gender", "Age", "Hypertension", "Heart_disease", "Ever_married", "Work_type", "Residence_type", "Avg_glucose_level", "BMI", "Smoking_status", "Stroke")

data %>% head() %>% kable %>% kable_styling("striped", "bordered", "hover", full_width = TRUE, position = "center", font_size = 15) %>% row_spec(0, align = "center", bold = TRUE)

```

###Cambiar mi variable predictora de numérica a factor

```{r}

```
data = data %>%
  mutate(Stroke_diagnosis = if_else(str_detect(Stroke, "0"), "Negative", "Positive")) %>%
  mutate(Stroke_diagnosis = factor(Stroke_diagnosis, levels = c("Positive", "Negative"), labels = c("Positive", "Negative"))) %>%
  relocate(Stroke_diagnosis, .before = Stroke) %>% dplyr::select(-Stroke)

DT::datatable(data, rownames = FALSE, option = list(pageLength = 10, scrollx = TRUE), class = "white_space:nowrap")
```

```
###Cargar resultados

```{r}
data_res = read.csv("resultados_modelos_TFM.csv")
```

Mostrar datos

```{r}
DT::datatable(data, rownames = FALSE, option = list(pageLength = 10, scrollx = TRUE), class = "white_space:nowrap")
```

Variables categóricas

```{r}
data1 = data %>% dplyr::select(-Stroke_diagnosis)
```
```

```
choices1 = names(data[sapply(data1, is.factor)])

selectInput('x', # saved as

        label = 'Select a variable',

        choices = choices1,

        selected = choices1)
```

```

# Gráficos

```{r}

renderPlot({

 ggplot(data, aes_string(input$x)) +

   geom_bar(aes(fill = Stroke_diagnosis), show.legend =TRUE) +

   facet_wrap(~Stroke_diagnosis) +

   scale_fill_brewer(palette = 10) +

   theme_classic()


})

```

Variables numéricas

```{r}

```
data2 = data %>% dplyr::select(-Stroke_diagnosis)

choices2 = names(data2[sapply(data2, is.numeric)])

selectInput('x', label = 'Select a variable', choices = choices2, selected = choices2[[1]])
```

```
# Gráficos

```{r}

renderPlot({

 ggplot(data, aes_string(data$Stroke_diagnosis, input$y)) +

   geom_bar(aes(fill = Stroke_diagnosis), show.legend =TRUE) +

   facet_wrap(~Stroke_diagnosis) +

   scale_fill_brewer(palette = 10) +

   labs(x = "Diagnosis", y = "Age") +

    theme_classic()


})
```

```
# Estadísticas variables categóricas
```
```

```{r}

m = matrix(nrow=7, ncol=1, dimnames=list(colnames(data[,c(2,4:8,11)]), c("p-valor")))

for (i in c(2,4:8,11)) {

  tabla = table(data$Stroke, data[[i]])

  test = chisq.test(tabla)

  m[colnames(data)[i],] = c(round(test$p.value,4))

}

m

```

Estadísticas variables numéricas

```{r}

data$Age = as.character(data$Age)
```

```
data$Age = as.numeric(data$Age)


data$Avg_glucose_level = as.character(data$Avg_glucose_level)

data$Avg_glucose_level = as.numeric(data$Avg_glucose_level)


data$BMI = as.character(data$BMI)

data$BMI = as.numeric(data$BMI)




m = matrix(nrow=3, ncol=1, dimnames=list(colnames(data[,c(3,9,10)]), c("p-valor")))

for (i in c(3,9,10)){

 f = formula(paste(colnames(data)[i], "~Stroke_diagnosis"))

 test = pb2gen(f, data = data, est = "median")

 m[colnames(data)[i],] = c(round(test$p.value,4))


}


m


```


Sensibilidad {data-navmenu: "Métricas"}

```{r}
```

```
data_res["X"] = NULL

data_res["seed"] = NULL

data_res["max_J_train"] = NULL

data_res["probab"] = NULL

data_res$model = as.factor(as.character(data_res$model))


gsen =

  ggplot(data_res, aes(x=model, y=data_res$sensitivity, fill=model)) + geom_boxplot() +
scale_fill_brewer(palette=10) + labs(title = "SENSIBILIDAD", x= "Modelo", y = "Sensibilidad") +
theme_classic()

ggplotly(gsen)
```

```

# Especificidad {data-navmenu: "Métricas"}

```{r}

data_res["X"] = NULL

data_res["seed"] = NULL

data_res["max_J_train"] = NULL

data_res["probab"] = NULL

data_res$model = as.factor(as.character(data_res$model))


gsen =
```
```

```
 ggplot(data_res, aes(x=model, y=data_res$specificity, fill=model)) + geom_boxplot() +
scale_fill_brewer(palette=10) + labs(title = "ESPECIFICIDAD", x= "Modelo", y = "Especificidad")
+ theme_classic()

ggplotly(gsen)
```

```
# Sensibilidad {data-navmenu: "Métricas"}

```{r}

data_res["X"] = NULL

data_res["seed"] = NULL

data_res["max_J_train"] = NULL

data_res["probab"] = NULL

data_res$model = as.factor(as.character(data_res$model))


gsen =

 ggplot(data_res, aes(x=model, y=data_res$AUC, fill=model)) + geom_boxplot() +
scale_fill_brewer(palette=10) + labs(title = "AUC", x= "Modelo", y = "AUC") + theme_classic()

ggplotly(gsen)
```

```
# Tabla de resultados

```{r}
```

```
sensibilidad = tapply(data_res$sensitivity, data_res$model, median)

sensibilidad = data.frame(sensibilidad)

names(sensibilidad) = c("Sensbilidad")


especificidad = tapply(data_res$specificity, data_res$model, median)

especificidad = data.frame(especificidad)

names(especificidad) = c("Especificidad")


AUC = tapply(data_res$AUC, data_res$model, median)

AUC = data.frame(AUC)

names(AUC) = c("AUC")


final = cbind(sensibilidad, especificidad, AUC)


final
```

```
###Comparación de métricas

```{r}

m = matrix(nrow = 3,

        ncol = 1,

        dimnames = list(colnames(data_res)[-1],
```
```

```
        c("p-valor")))



for (i in 2:4) {

  f = formula(paste(colnames(data_res)[i], "~model"))

  test = med1way(f, data = data_res)

  m[colnames(data_res)[i], ] = c(round(test$p.value,4))


}


formato = c("striped", "hover", "condensed")


m %>% kable() %>% kable_styling(bootstrap_options = formato,
                                position = "center",
                                full_width = FALSE,
                                font_size = 12)
```