UNIVERSIDAD COMPLUTENSE MADRID

Facultad de Estudios Estadísticos

Machine Learning

Predicción de accidentes cerebro-vasculares

Autor:
Sumit Kumar Jethani Jethani

26/04/2023

Índice

Índice Figuras

Índice Tablas

1 Introducción

El accidente cerebro-vascular es una preocupación importante en el ámbito de la salud pública y una de las principales causas de mortalidad a nivel mundial, responsable de alrededor del 11% de las muertes, según informa la Organización Mundial de la Salud (OMS). En este contexto, predecir la probabilidad de que un paciente experimente un accidente cerebro-vascular puede contribuir significativamente a las medidas preventivas y a mejores resultados de atención médica.

Tabla 1: Variables iniciales

Variable	**Descripción**
Id (*id*)	Identificador único del paciente
Género (*gender*)	Género del paciente: masculino, femenino u otro
Edad (*age*)	Edad del paciente en años
Hipertensión (*hypertension*)	0 si el paciente no tiene hipertensión, 1 si el paciente tiene hipertensión
Enfermedad cardíaca (*heart_disease*)	0 si el paciente no tiene enfermedades cardíacas, 1 si el paciente tiene una enfermedad cardíaca
Casado (*ever_married*)	Indica si el paciente está casado o no
Tipo de trabajo (*work_type*)	Tipo de trabajo del paciente: niño, trabajo del gobierno, nunca trabajó, privado o autónomo
Tipo de residencia (*residence_type*)	Tipo de residencia del paciente: rural o urbano
Nivel de glucemia (*avg_glucose_level*)	Nivel promedio de glucemia en sangre del paciente (en mg/dl)
IMC (*bmi*)	Índice de masa corporal del paciente
Estatus de fumador (*smoking_status*)	Antes fumaba, nunca fumó, fuma o desconocido (información no disponible para el paciente)
Accidente cerebro-vascular (*stroke*)	1 si el paciente tuvo un accidente cerebro-vascular o 0 si no lo tuvo

Fuente: Elaboración propia del autor

Para llevar a cabo dicha predicción, se ha recopilado un conjunto de datos publicado en la plataforma **Kaggle**. Dicho conjunto contiene la información necesaria para la predicción de accidentes cerebro-vasculares y se encuentra compuesto por diferentes variables clínicas, las cuales se detallan en la Tabla 1.

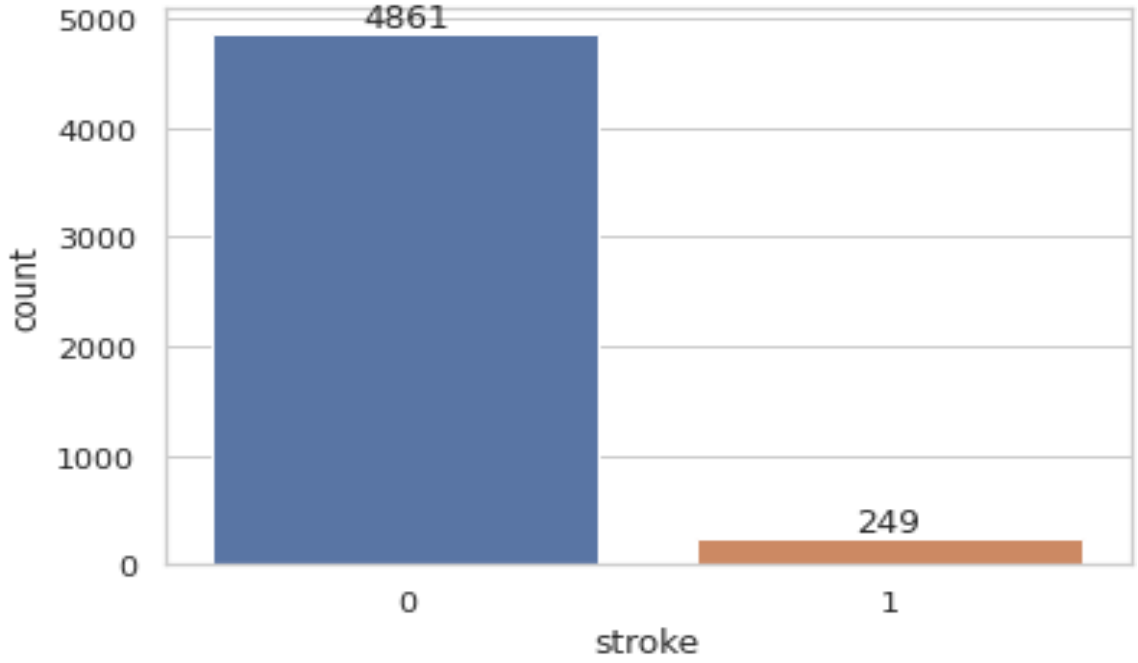
Finalmente, los principales objetivos que se persiguen en este informe son:

- La exploración y análisis del conjunto de datos clínico con el fin de obtener un conocimiento profundo de los mismos, así como sus características.
- El ensayo de diversos algoritmos de clasificación de aprendizaje automático que permitan lograr una predicción precisa de los accidentes cerebro-vasculares y obtener un modelo final óptimo.

2 Análisis exploratorio de los datos

El conjunto de datos inicial consta de **once posibles variables explicativas** y **una variable objetivo denominada *stroke***. Esta última variable indica si el paciente ha experimentado un accidente cerebro-vascular y se compone de **5110 observaciones**, de las cuales **4861 (95.13%)** no han sufrido dicho evento, mientras que las restantes **249 (4.87%)** sí lo han hecho.

Figura 1: Distribución de la variable objetivo



Fuente: Elaboración propia del autor

Como se puede apreciar en la Figura 1, el conjunto de datos presentado muestra un **desbalanceo significativo** en la variable objetivo **stroke**, donde solo el 4.87% de las observaciones representan casos positivos de accidente cerebro-vascular (etiqueta 1). Dicho desbalanceo puede dificultar el entrenamiento de un modelo de aprendizaje automático que pueda predecir con precisión la presencia o ausencia de esta afección. Para abordar este problema existen diferentes técnicas tales como el **sub-muestreo aleatorio**, el **sobre-muestreo de la clase minoritaria** (la utilizada en este informe), la **combinación de ambas técnicas** (sub-muestreo y sobre-muestreo), y **la utilización de algoritmos de aprendizaje automático** específicos para el desbalanceo de los datos, como los algoritmos de ensamble y los algoritmos basados en coste.

Figura 2: Tipo de las variables iniciales

```
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   category
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   category
 4   heart_disease      5110 non-null   category
 5   ever_married       5110 non-null   category
 6   work_type          5110 non-null   category
 7   Residence_type     5110 non-null   category
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   category
 11  stroke             5110 non-null   category
dtypes: category(8), float64(3), int64(1)
```

Fuente: Elaboración propia del autor

Figura 3: Descriptivo de las variables continuas

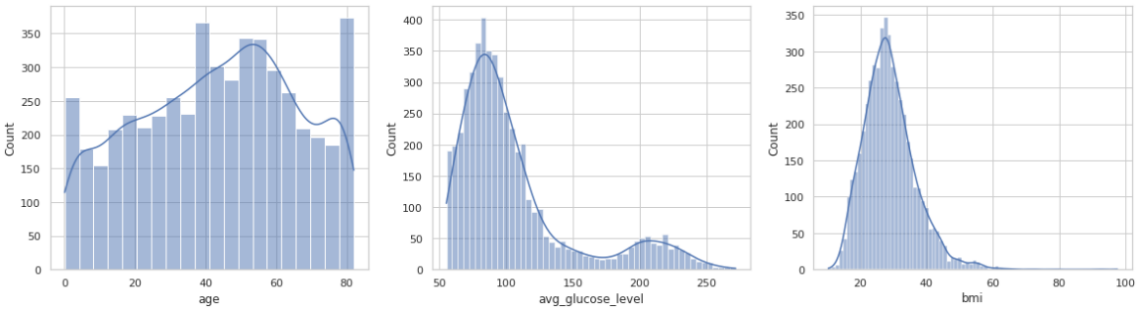
	age	avg_glucose_level	bmi
count	5110.000000	5110.000000	4909.000000
mean	43.226614	106.147677	28.893237
std	22.612647	45.283560	7.854067
min	0.080000	55.120000	10.300000
25%	25.000000	77.245000	23.500000
50%	45.000000	91.885000	28.100000
75%	61.000000	114.090000	33.100000
max	82.000000	271.740000	97.600000

Fuente: Elaboración propia del autor

Como siguiente paso en el análisis del conjunto de datos, se procede a la descripción de los tipos de variables presentes en el mismo. De acuerdo con la Figura 2, se puede observar que de las doce variables iniciales, **cinco son variables categóricas nominales**: *gender*, *ever_married*, *work_type*, *Residence_type* y *smoking_status*; **tres son variables categóricas binarias**: *hypertension*, *ever_married* y *stroke*; y las restantes son **variables de naturaleza continua**: *id*, *age*, *avg_glucose_level* y *bmi*, presentando está última 201 valores perdidos. Asimismo, para los siguientes pasos del análisis, se procede a eliminar la variable que identifica de manera única a cada paciente (*id*), puesto que no aportará información relevante para el modelo de predicción.

Considerando lo anteriormente expuesto, se procede a llevar a cabo un análisis descriptivo inicial exclusivamente de las variables continuas (ver Figura 3). Se observa que la variable *age* presenta un mínimo de 0.08 años, lo cual indica la presencia de pacientes en la muestra que son bebés. Además, se destaca que tanto la media (43.22) como la mediana (45) se encuentran en valores cercanos, lo que sugiere la inexistencia o presencia de pocos valores atípicos en dicha variable. Con respecto a la variable *bmi*, esta presenta valores máximos y mínimos extremadamente disímiles a lo normal, ya que el índice de masa corporal suele variar entre 18.5 y 30 [2]. Finalmente, la variable *avg_glucose_level* que indica el nivel promedio de glucemia en sangre del paciente, considerándose normal cuando los valores de azúcar en la sangre son menores a 140 mg/dl, los valores de 140 a 199 mg/dl indican la presencia de prediabetes y los de 200 mg/dl o mayores indican la existencia de diabetes [1].

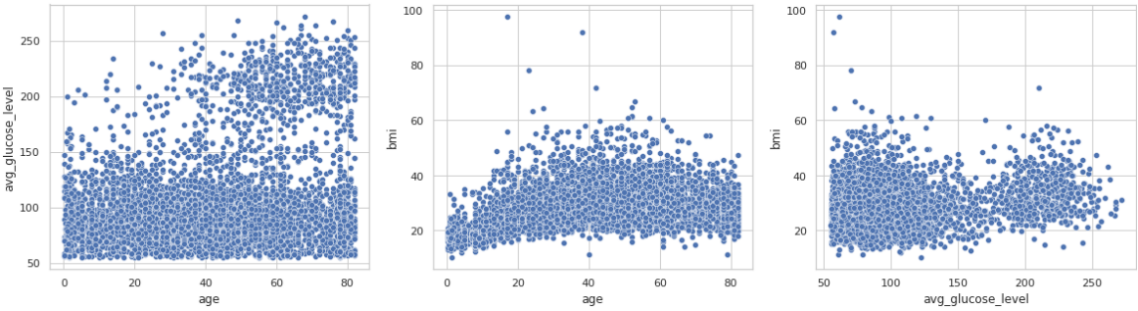
Figura 4: Distribución de las variables continuas



Fuente: Elaboración propia del autor

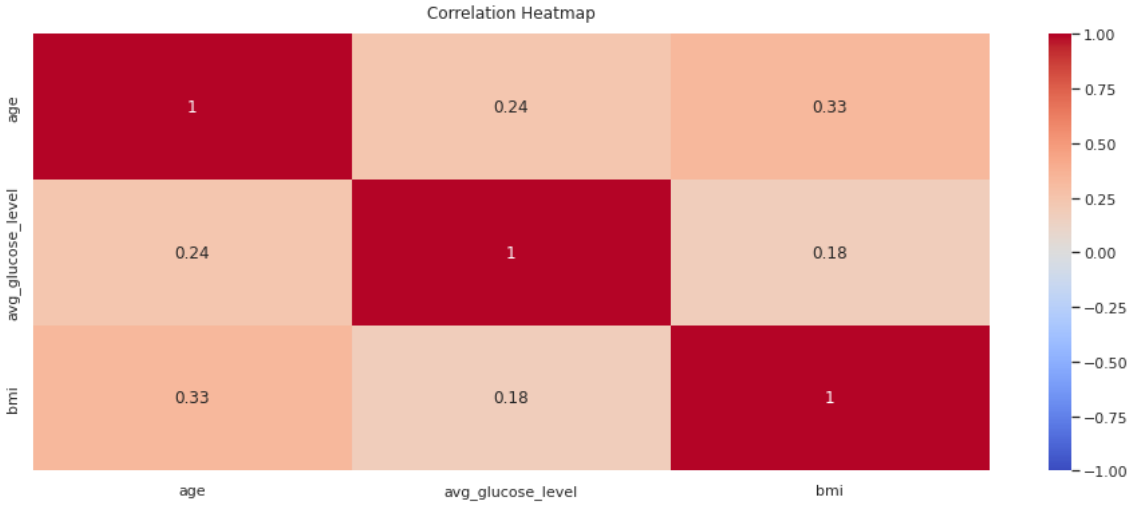
Analizando la distribución de las variables anteriores (ver Figura 4), se puede deducir que ninguna de las tres sigue una distribución normal. En consecuencia, la variable *age* está representada adecuadamente en todos sus rangos, lo que sugiere la existencia de pocos o ningún valor atípico. En contraposición, las variables *avg_glucose_level* y *bmi* parecen estar sesgadas hacia la izquierda, lo cual indica la presencia de colas en la derecha y, por consiguiente, probablemente valores atípicos. Además, cabe destacar que dicha cola hacia la derecha es más pronunciada en la primera variable que en la segunda.

Figura 5: Variables continuas cruzadas



Fuente: Elaboración propia del autor

Figura 6: Correlación entre las variables continuas

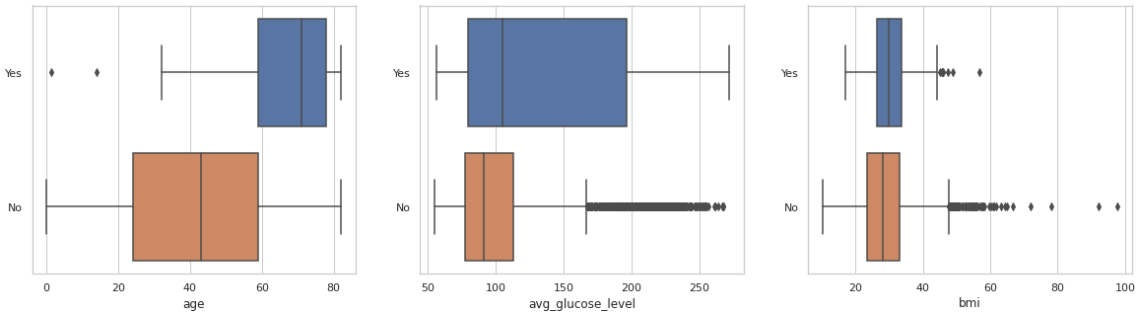


Fuente: Elaboración propia del autor

Al tratarse de un conjunto limitado de variables cuantitativas, resulta viable efectuar un análisis cruzado entre ellas con el fin de detectar si existe alguna relación, independientemente de si ésta es lineal o no. Tal y como se muestra en la Figura 5, no se evidencia ningún tipo de relación entre las variables *age-avg_glucose_level* y *avg_glucose_level-bmi*. No obstante, parece ser que **el índice de**

masa corporal (*bmi*) tiende a incrementarse a medida que avanza la edad del paciente, lo cual resulta coherente. Esto también se puede confirmar en la matriz de correlación (ver Figura 6), donde la mayor **correlación lineal** se da entre las variables *age* y *bmi*, tal y como se mencionó anteriormente.

Figura 7: Variables continuas cruzadas con la objetivo



Fuente: Elaboración propia del autor

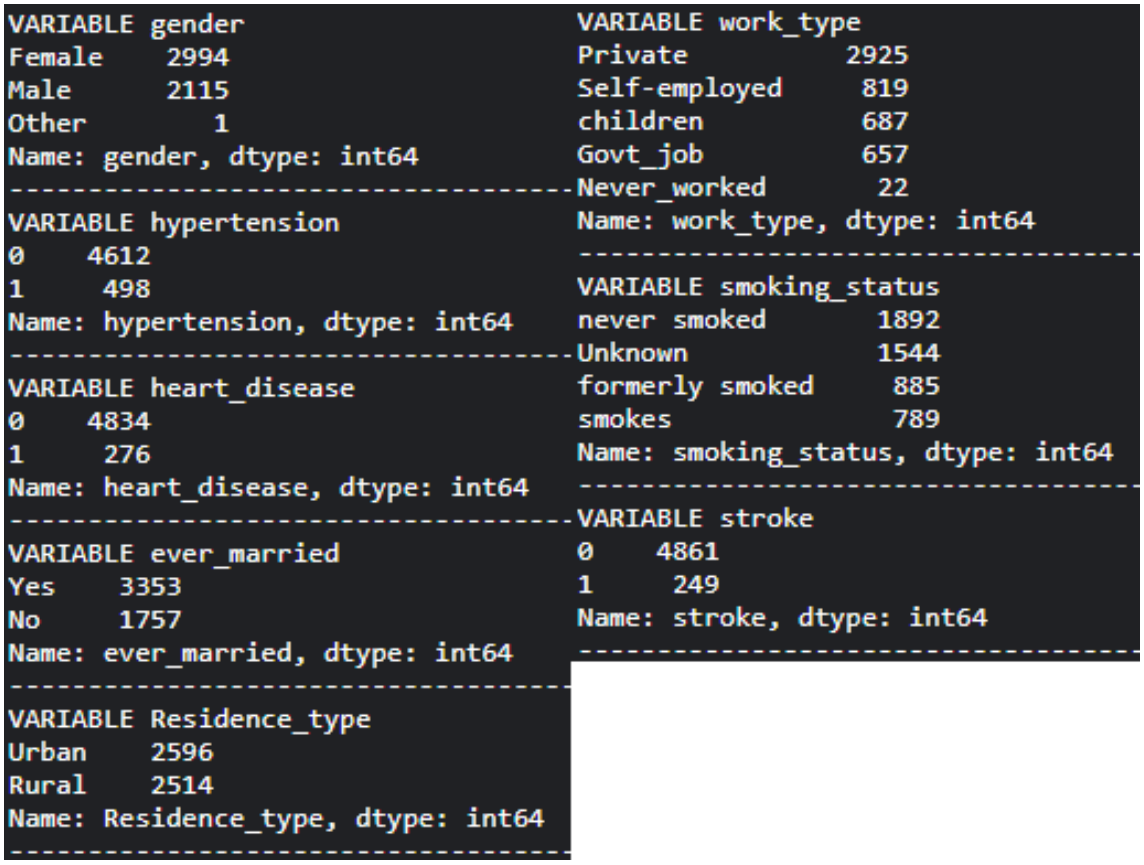
Además, en un problema de clasificación resulta de gran interés determinar si las variables continuas son capaces de discriminar las distintas categorías a predecir de la variable objetivo, ya que si no lo hacen, es probable que dichas variables tengan una importancia limitada en los modelos de aprendizaje automático. En este sentido, en la Figura 7 se presentan los cruces de las distintas variables continuas con la variable binaria a predecir, donde se observa que en general **los pacientes que sufrieron un accidente cerebro-vascular presentaban una mayor edad y nivel de glucemia en sangre que los que no lo sufrieron**. Sin embargo, **no se encuentran diferencias significativas en la variable *bmi***, lo que sugiere que dicha variable no tenga un impacto discriminatorio en la variable a predecir.

Figura 8: Descriptivo de las variables categóricas

	gender	hypertension	heart_disease	ever_married	work_type	Residence_type	smoking_status	stroke
count	5110	5110	5110	5110	5110	5110	5110	5110
unique	3	2	2	2	5	2	4	2
top	Female	0	0	Yes	Private	Urban	never smoked	0
freq	2994	4612	4834	3353	2925	2596	1892	4861

Fuente: Elaboración propia del autor

Figura 9: Número de observaciones por categoría de las variables cualitativas

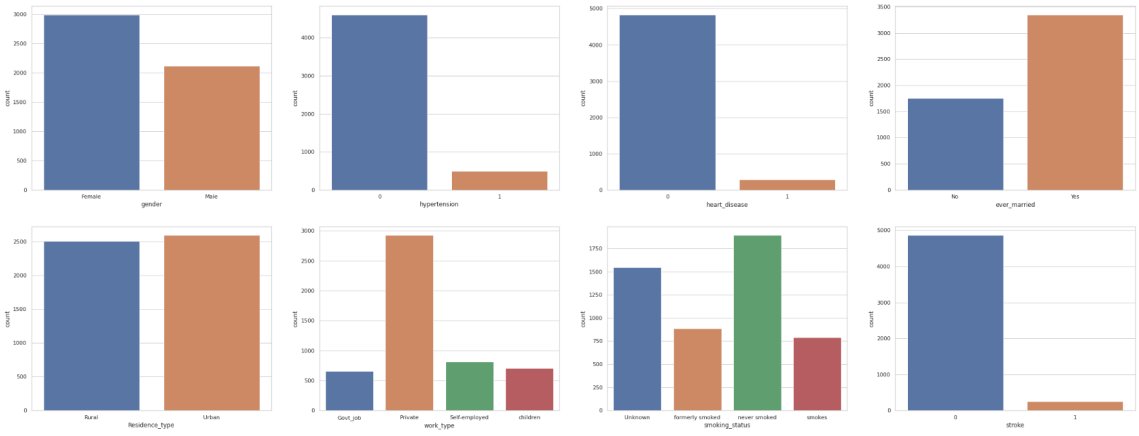


Fuente: Elaboración propia del autor

Continuando con el análisis exploratorio de los datos correspondientes a las variables categóricas, se presenta en la Figura 8 un conjunto de estadísticas descriptivas, que incluyen la moda, la frecuencia y el número de valores únicos para cada variable. Adicionalmente, al examinar el número de observaciones en cada categoría de la variable cualitativa (ver Figura 9), se observa que en la variable *gender*, la categoría *Other* contiene solamente una observación, lo cual sugiere su eliminación. Asimismo, se observa que la categoría *Never_worked* de la variable *work_type* no cuenta con un número significativo de observaciones en comparación con las otras categorías.

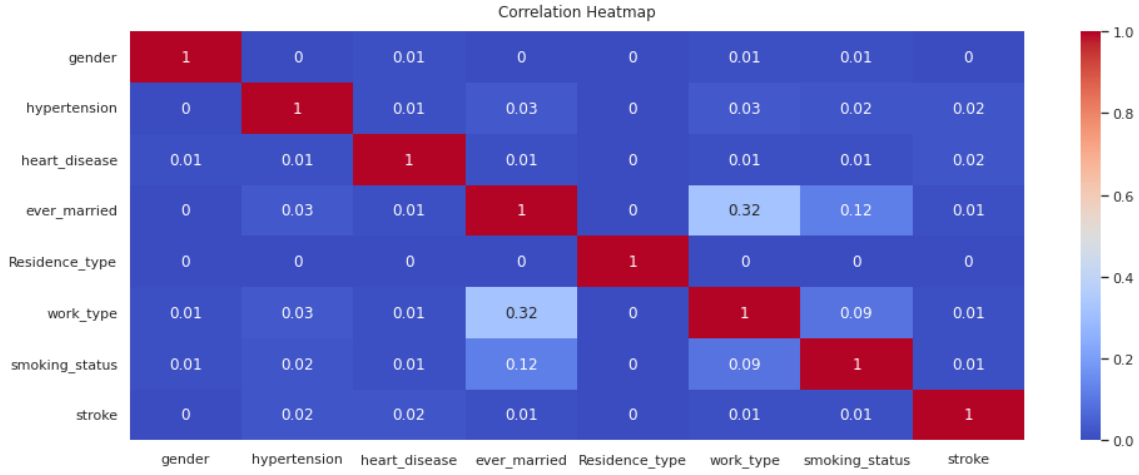
Al analizar las 22 observaciones correspondientes a la categoría *Never_worked* de la variable *work_type*, se descubre que no se registró ningún paciente que haya sufrido un accidente cerebrovascular. Además, la mayoría de las observaciones (20) corresponden a pacientes con edades comprendidas entre 13 y 18 años. Por tanto, se agrupan estas observaciones en la categoría *children* de la misma variable.

Figura 10: Distribución de las categorías de las variables cualitativas



Fuente: Elaboración propia del autor

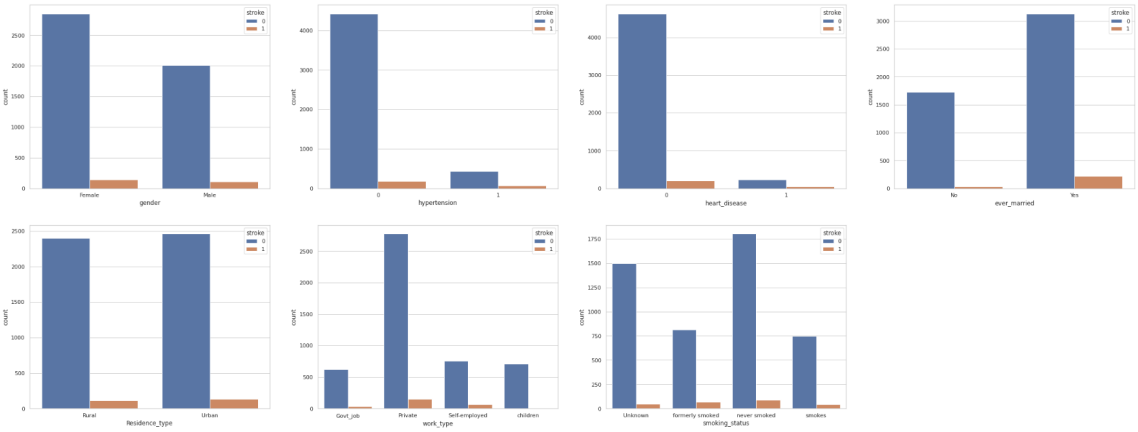
Figura 11: Matriz V de Cramer de las variables categóricas



Fuente: Elaboración propia del autor

Tras la eliminación de la observación correspondiente al género *Other* y la reorganización de la categoría *Never_worked* como *children* en la variable *work_type*, se presenta en la Figura 10 la distribución de las categorías para cada variable cualitativa. Se puede apreciar que todas las categorías están representadas adecuadamente en cada variable, excepto en los casos de la variable *hypertension* y *heart_disease*, lo cual resulta coherente con el contexto del problema analizado. Asimismo, se realiza el cálculo de la **matriz de V de Cramer** entre todas las variables categóricas (ver Figura 11), donde se aprecia una **asociación moderada** entre las variables *work_type* y *ever_married*, y falta de asociación con la variable objetivo que se pretende predecir.

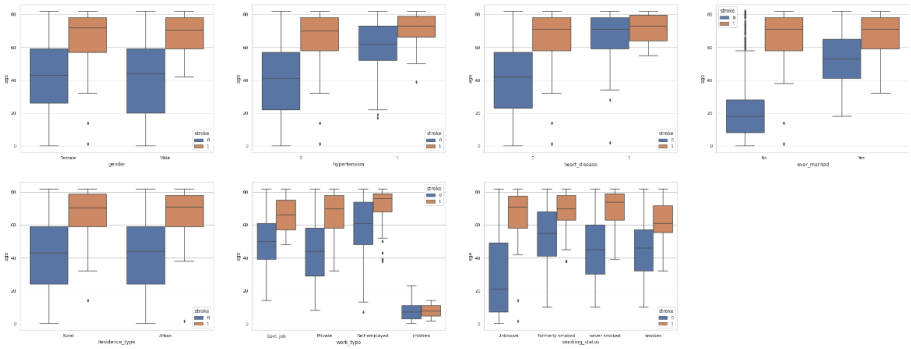
Figura 12: Variables categóricas cruzadas con la objetivo



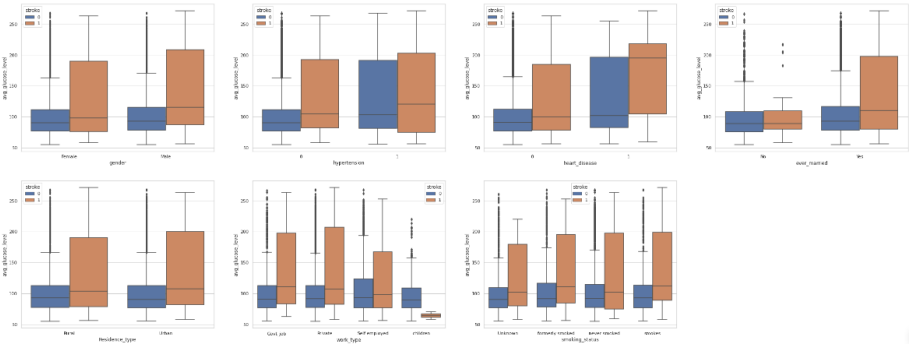
Fuente: Elaboración propia del autor

Figura 13: Variables continuas y categóricas cruzadas con la objetivo

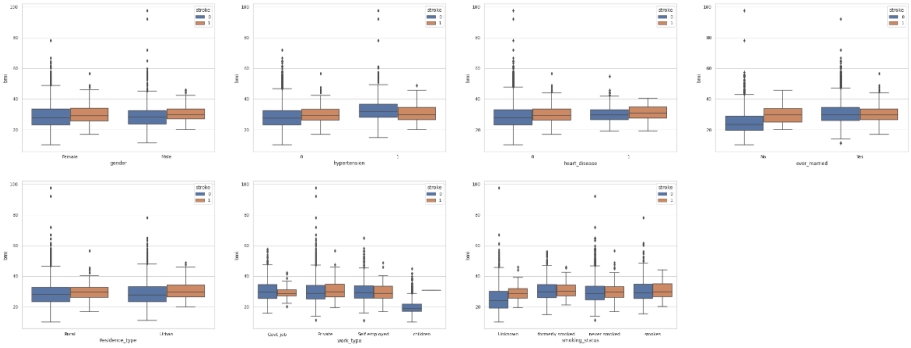
(a) age



(b) avg_glucose_level



(c) bmi



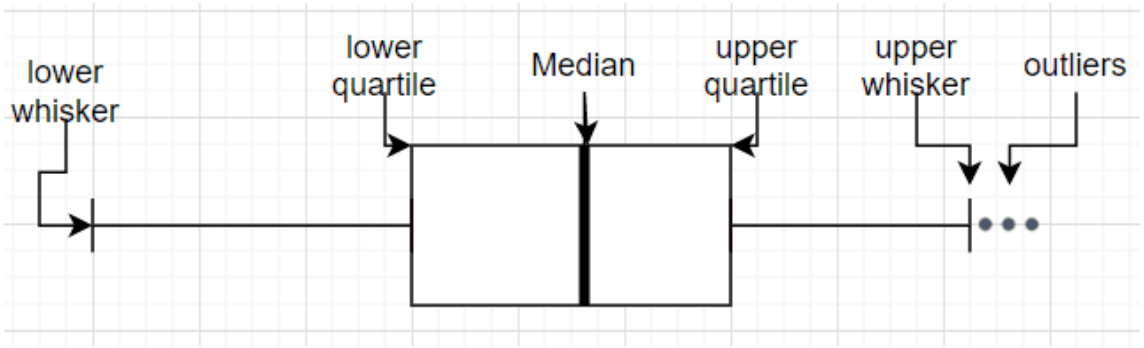
Fuente: Elaboración propia del autor

Al igual que con las variables continuas, se realiza el cruce entre las variables categóricas y la variable objetivo a predecir (ver Figura 12). Se observa que, en general, **las personas casadas o aquellas que trabajan en el sector privado** son las que presentan una mayor incidencia de accidentes cerebro-vasculares. Además, es importante destacar que aquellos **pacientes que no padecen de hipertensión o enfermedades cardíacas** son los que presentan mayor número de accidentes cerebro-vasculares en la muestra analizada. Por último, se destaca que, al menos de manera visual, no parece haber una relación clara entre **el género y el tipo de residencia** con el número de pacientes que presentan accidentes cerebro-vasculares.

Como paso final en la exploración inicial de los datos, se realiza un análisis de todas las variables continuas en conjunto con todas las variables categóricas, y a su vez con la variable objetivo a predecir (ver Figura 13). A nivel global, se observa que las variables *age* y *avg_glucose_level* son consistentemente mayores en pacientes que han sufrido un accidente cerebro-vascular en comparación con aquellos que no lo han sufrido, independientemente de la categoría de la variable analizada (ver Figuras 13a y 13b). Sin embargo, en pacientes que han sufrido un accidente cerebro-vascular, la variable *bmi* no parece mostrar diferencias significativas en ninguna de las categorías analizadas en comparación con aquellos que no lo han sufrido (ver Figura 13c).

3 Análisis y tratamiento de los datos atípicos

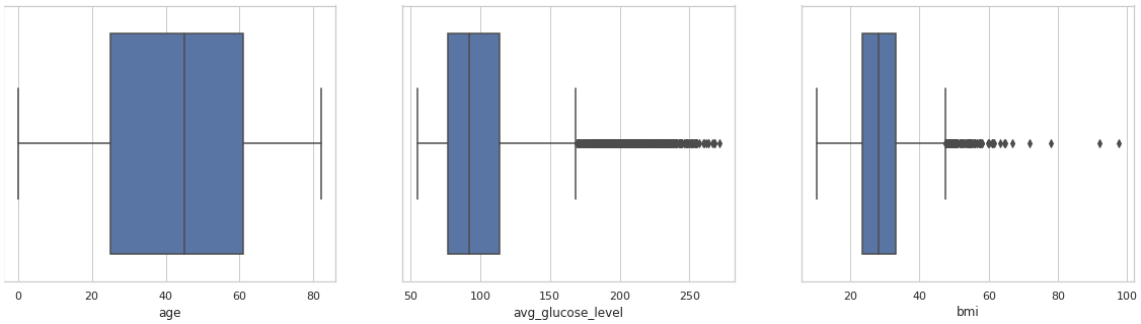
Figura 14: Detección de valores atípicos con IQR



Fuente: Elaboración propia del autor

En la literatura se han propuesto diversas técnicas para la detección de valores atípicos, como los métodos basados en estadística descriptiva, los métodos de clustering y los métodos de aprendizaje automático. No obstante, dado que se ha comprobado que ninguna de las variables cuantitativas sigue una distribución normal (ver Figura 4), en el presente informe se ha optado por utilizar los diagramas de caja y bigotes en conjunción con el rango intercuartílico (IQR), que se define como la diferencia entre el tercer y el primer cuartil. En consecuencia, cualquier observación que se encuentre por debajo del valor de **Q1-3*IQR** o por encima de **Q3+3*IQR** se considerará como un valor atípico (ver Figura 14).

Figura 15: Valores atípicos en las variables continuas

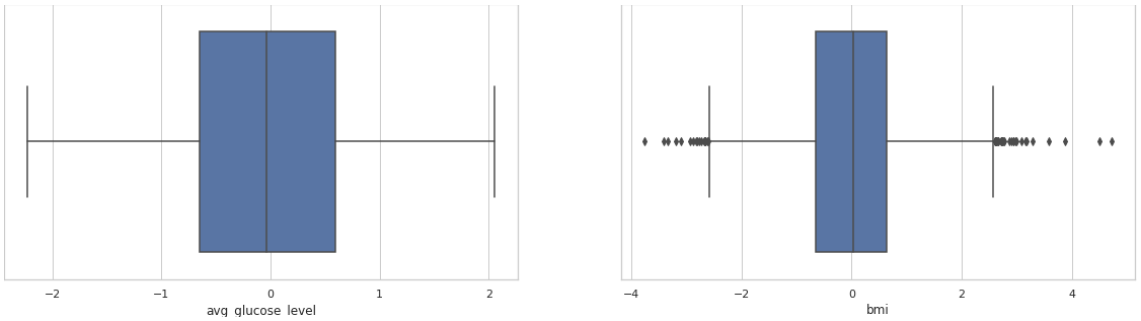


Fuente: Elaboración propia del autor

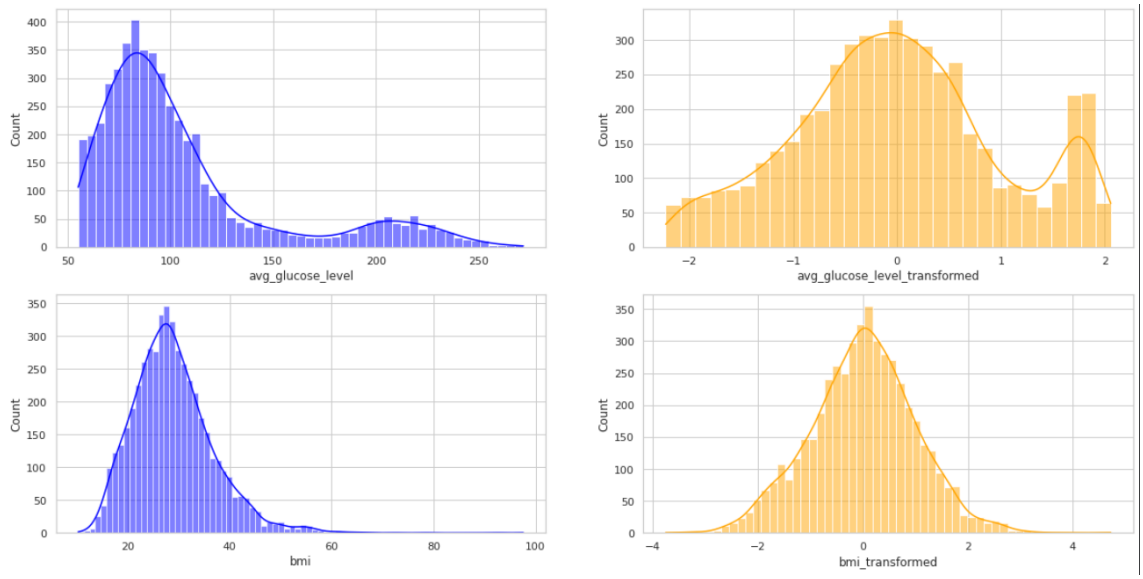
De acuerdo con el criterio seleccionado, en la Figura 15 se puede observar que la variable *age* no presenta valores atípicos, lo cual era esperable. No obstante, debido a su sesgo hacia la izquierda, se esperaba que las variables *avg_glucose_level* y *bmi* presentaran valores atípicos en su distribución. En concreto, se detectaron **165** (aproximadamente el 3,2%) y **8** (aproximadamente el 0,16%) valores atípicos en las variables *avg_glucose_level* y *bmi*, respectivamente.

Figura 16: Variables continuas tras las transformaciones Box-Cox

(a) Diagrama caja y bigotes



(b) Histogramas



Fuente: Elaboración propia del autor

Una de las estrategias más habituales para el tratamiento de los valores atípicos consiste en la transformación de la variable mediante alguna de las funciones **Box-Cox**. Dichas funciones permiten ajustar la distribución de la variable a una distribución normal y, por ende, reducir la presencia de valores atípicos. No obstante, antes de proceder con el tratamiento de los valores atípicos, es recomendable realizar una prueba para verificar si se logra reducir el número de valores atípicos mediante alguna de las transformaciones Box-Cox. Es por ello que, en la Figura 16a se muestra la distribución de las variables *avg_glucose_level* y *bmi* después de aplicar la mejor transformación Box-Cox posible, cuyos valores de λ son **-0.00085** y **-1.05**, respectivamente. Como se puede apreciar, **según el criterio seleccionado**, la variable *avg_glucose_level* ya no presenta valores atípicos, mientras que la variable *bmi* solo cuenta con un dato atípico. Además, se puede comprobar que las transformaciones Box-Cox aplicadas han logrado centralizar la distribución de los datos, los cuales presentaban un sesgo hacia la izquierda previamente (ver Figura 16b).

Código 1: Conjuntos de entrenamiento y test

```
sss = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=99)
X_train, X_test, y_train, y_test = None, None, None, None

for train_index, test_index in sss.split(stroke_df.drop(columns=['stroke']),
    stroke_df['stroke']):
    train_df = stroke_df.iloc[train_index]
    test_df = stroke_df.iloc[test_index]
    X_train, y_train = train_df.drop(columns=['stroke']), train_df.stroke
    X_test, y_test = test_df.drop(columns=['stroke']), test_df.stroke
```

Fuente: Elaboración propia del autor

No obstante, es importante destacar que no se recomienda realizar el tratamiento de los valores atípicos sobre todo el conjunto de datos, ya que esto podría provocar una **fuga de datos (*data leakage*)**. Es por ello, que se procederá a dividir inicialmente el conjunto de datos en subconjuntos de entrenamiento y test, para posteriormente llevar a cabo el tratamiento de los datos atípicos y perdidos. Dado que el conjunto de datos presenta un desequilibrio significativo en cuanto a la distribución de clases, se utilizará el objeto ***StratifiedShuffleSplit()*** de la librería ***sklearn*** para la división, el cual permite mantener las proporciones de cada una de las clases en los subconjuntos correspondientes. En consecuencia, se conserva el **20%** del conjunto de datos como muestra de test, mientras que el **80%** restante se utilizará como conjunto de entrenamiento para los distintos algoritmos de aprendizaje automático (ver Código 1).

Código 2: Transformaciones Box-Cox en train-test

```
# Variable avg_glucose_level
power_transformer = PowerTransformer(method='box-cox')
avg_glucose_level_transformed = power_transformer.fit_transform(X_train[['
    avg_glucose_level']])
X_train['avg_glucose_level'] = avg_glucose_level_transformed.flatten()
X_test['avg_glucose_level'] = power_transformer.transform(X_test[['
    avg_glucose_level']])

# Variable bmi
power_transformer = PowerTransformer(method='box-cox')
bmi_transformed = power_transformer.fit_transform(X_train[['bmi']])
X_train['bmi'] = bmi_transformed.flatten()
X_test['bmi'] = power_transformer.transform(X_test[['bmi']])
```

Fuente: Elaboración propia del autor

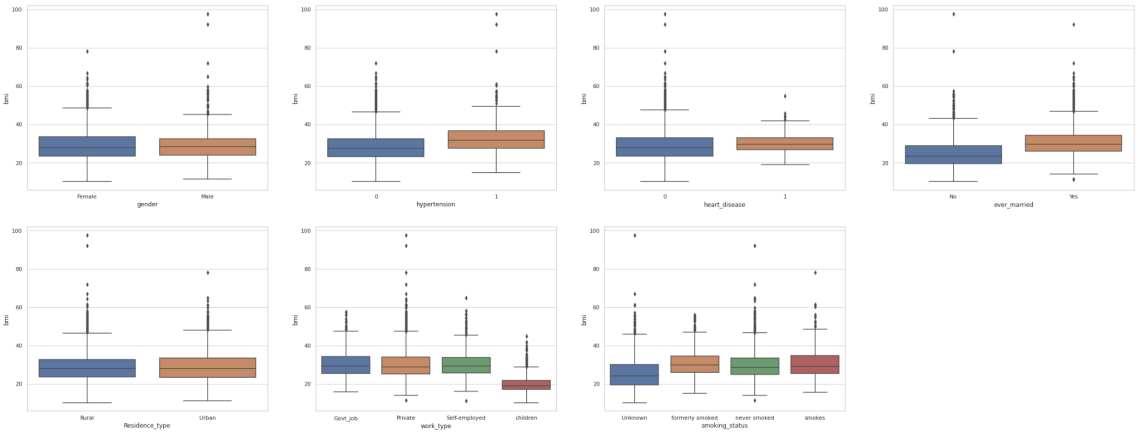
Una vez dividido el conjunto de datos en los subconjuntos de entrenamiento y test, se emplea el objeto ***PowerTransformer()*** de la librería ***sklearn*** para buscar la mejor transformación Box-Cox para las variables *avg_glucose_level* y *bmi* en el conjunto de entrenamiento mediante la función *fit_transform()*. Posteriormente, se aplica la misma transformación al conjunto de test mediante la función *transform()* (ver Código 2).

Por lo tanto, tras esta etapa del análisis, se han eliminado los valores atípicos del conjunto de datos. No obstante, aún existen valores perdidos que requerirán un tratamiento en la próxima sección.

4 Análisis y tratamiento de los datos perdidos

En la Figura 2 se identificó que la única variable que presentaba valores perdidos era *bmi*. Concretamente, se encontraron **201** valores perdidos, lo que representan aproximadamente el **4%** de la muestra. Al analizar más detalladamente, se observa que de esos 201 valores perdidos, **161 pertenecen a la clase mayoritaria (etiqueta 0)**, mientras que **40 a la clase minoritaria (etiqueta 1)**. Si se eliminasen dichas observaciones, la proporción de pacientes que han padecido accidente cerebro-vascular se reduciría en torno al **16%**, mientras que la reducción para las observaciones que no lo han sufrido sería de alrededor del **3%**. Por dicha razón, no se ha considerado la eliminación de estas observaciones como la opción más adecuada para el tratamiento de los valores perdidos. En lugar de ello, se ha decidido aplicar la técnica del **vecino más cercano (KNN)** para imputar los valores perdidos en la variable *bmi*. Para llevar a cabo esta técnica, se utilizarán las variables del conjunto de datos que presentan una mayor influencia en el valor de *bmi* como base para el algoritmo.

Figura 17: Variables categóricas cruzadas con bmi



Fuente: Elaboración propia del autor

Así, en primer lugar, se observa que, entre las variables categóricas (ver Figura 17), *hypertension* y *ever_married* presentan diferencias significativas en su distribución, por lo que serán consideradas en la imputación. Asimismo, las variables *work_type* y *smoking_status* también presentan diferencias notables en sus categorías *children* y *unknown*, respectivamente, en comparación con las demás categorías. Por lo tanto, se considerarán en la imputación, pero se realizará una reagrupación de sus categorías. En la variable *work_type* se agruparán todas las categorías diferentes a *children* en una sola categoría, mientras que en la variable *smoking_status* se agruparán todas las categorías diferentes a *unknown* en una sola categoría. Por otra parte, en relación a la variable continua *age* esta será incluida en la imputación debido a que varios estudios han demostrado su influencia en el *bmi*, a pesar de que no se haya detectado una correlación significativa entre ambas variables en nuestro conjunto de datos (ver Figura 5).

Código 3: Transformaciones variables categóricas-continuas para imputación bmi

```
# Variables categoricas
categorical_vars = ['hypertension', 'ever_married', 'work_type', 'smoking_status']
label_encoder = LabelEncoder()

for var in categorical_vars:
  X_train_imputation[var] = label_encoder.fit_transform(X_train_imputation[var])
  X_test_imputation[var] = label_encoder.transform(X_test_imputation[var])

# Variables continuas
min_max_scaler = MinMaxScaler()
X_train_imputation[['age', 'bmi']] = min_max_scaler.fit_transform(
    X_train_imputation[['age', 'bmi']])
X_test_imputation[['age', 'bmi']] = min_max_scaler.transform(X_test_imputation[['
    age', 'bmi']])
```

Fuente: Elaboración propia del autor

Código 4: Imputación variable bmi por KNN

```
knn_imputer = KNNImputer(n_neighbors=5)
X_train_imputation[list(X_train_imputation.columns)] = knn_imputer.fit_transform(
    X_train_imputation)
X_test_imputation[list(X_test_imputation.columns)] = knn_imputer.transform(
    X_test_imputation)
```

Fuente: Elaboración propia del autor

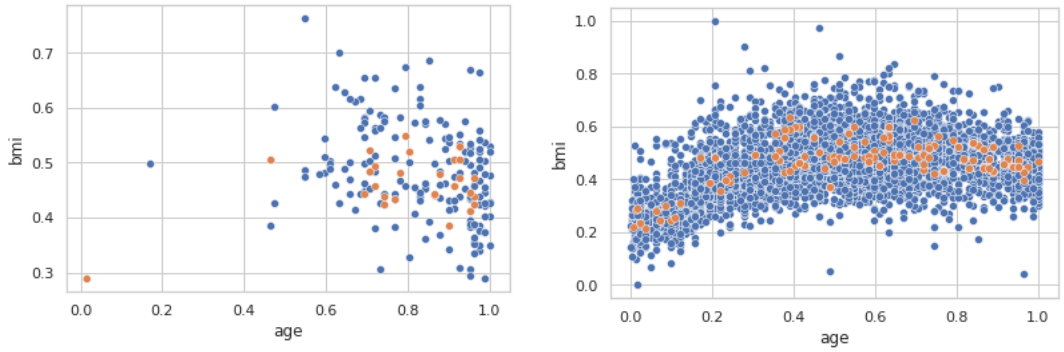
Una vez seleccionadas las variables que se emplearán para la imputación, se procede a codificar las variables categóricas utilizando el objeto ***LabelEncoder()*** de la librería ***sklearn***. En este caso, todas las variables categóricas son binarias, por lo que los posibles valores que pueden tomar son 0 o 1 (ver Código 3). Con respecto a las variables continuas, es importante mencionar que el algoritmo del vecino más cercano es un método de imputación basado en la distancia y requiere que se normalicen los datos. De lo contrario, las diferentes escalas de los datos pueden llevar al imputador KNN a generar reemplazos sesgados para los valores perdidos. Es por ello que se utiliza

el objeto ***MinMaxScaler()*** de la librería ***sklearn***, que escala las variables continuas para que tengan valores entre 0 y 1. En última instancia, se utiliza el objeto ***KNNImputer()*** de la misma librería (ver Código 4) para imputar los valores perdidos en el conjunto de entrenamiento mediante la función ***fit_transform()***, y únicamente transformar el conjunto de prueba mediante la función ***transform()***.

Figura 18: Resultado imputaciones KNN en el conjunto de entrenamiento

(a) Pacientes con accidente cerebro-vascular

(b) Pacientes sin accidente cerebro-vascular



Los puntos de color azul representan observaciones del conjunto de entrenamiento que no presentan valores perdidos para la variable bmi, mientras que los de color naranja corresponden a observaciones que tienen valores perdidos en dicha variable y que han sido imputados mediante la técnica del vecino más cercano (KNN).

Fuente: Elaboración propia del autor

Para verificar la adecuada realización de las imputaciones, se presentan en la Figura 18 las distribuciones de la variable bmi con respecto a age, para pacientes con (ver Figura 18a) y sin (ver Figura 18b) accidente cerebro-vascular en el conjunto de entrenamiento. De este modo, se puede observar que los valores imputados (puntos naranjas) parecen ajustarse a la distribución de ambas muestras.

5 Preparación final del conjunto de datos

Una vez tratados los valores atípicos y perdidos, en esta sección se aborda la correcta transformación de las diferentes variables del conjunto de datos con el objetivo de entrenar adecuadamente los modelos de aprendizaje automático.

Código 5: Transformaciones finales variables categóricas-continuas

```
# Variables categoricas binarias
categorical_vars = ['gender', 'hypertension', 'heart_disease', 'ever_married', '
    Residence_type']
label_encoder = LabelEncoder()

for var in categorical_vars:
  X_train_final[var] = label_encoder.fit_transform(X_train_final[var])
  X_test_final[var] = label_encoder.transform(X_test_final[var])

# Variables categoricas no binarias
X_train_final = pd.get_dummies(X_train_final, columns=['work_type', 'smoking_status
    '], drop_first=True)
X_test_final = pd.get_dummies(X_test_final, columns=['work_type', 'smoking_status'
    ], drop_first=True)

# Variables continuas
min_max_scaler = MinMaxScaler()
X_train_final[['avg_glucose_level']] = min_max_scaler.fit_transform(X_train_final[[
    'avg_glucose_level']])
X_test_final[['avg_glucose_level']] = min_max_scaler.transform(X_test_final[['
    avg_glucose_level']])
```

Fuente: Elaboración propia del autor

En este punto, es importante destacar que las variables continuas *age* y *bmi* ya han sido escaladas al rango de 0 y 1 debido a su uso en la imputación de valores perdidos. Por consiguiente, no se les aplicará de nuevo la transformación, sino que simplemente se agregarán al conjunto de datos final. En cuanto a la variable *avg_glucose_level*, se le aplica la misma transformación que al resto de las variables continuas del conjunto de datos (ver Código 5).

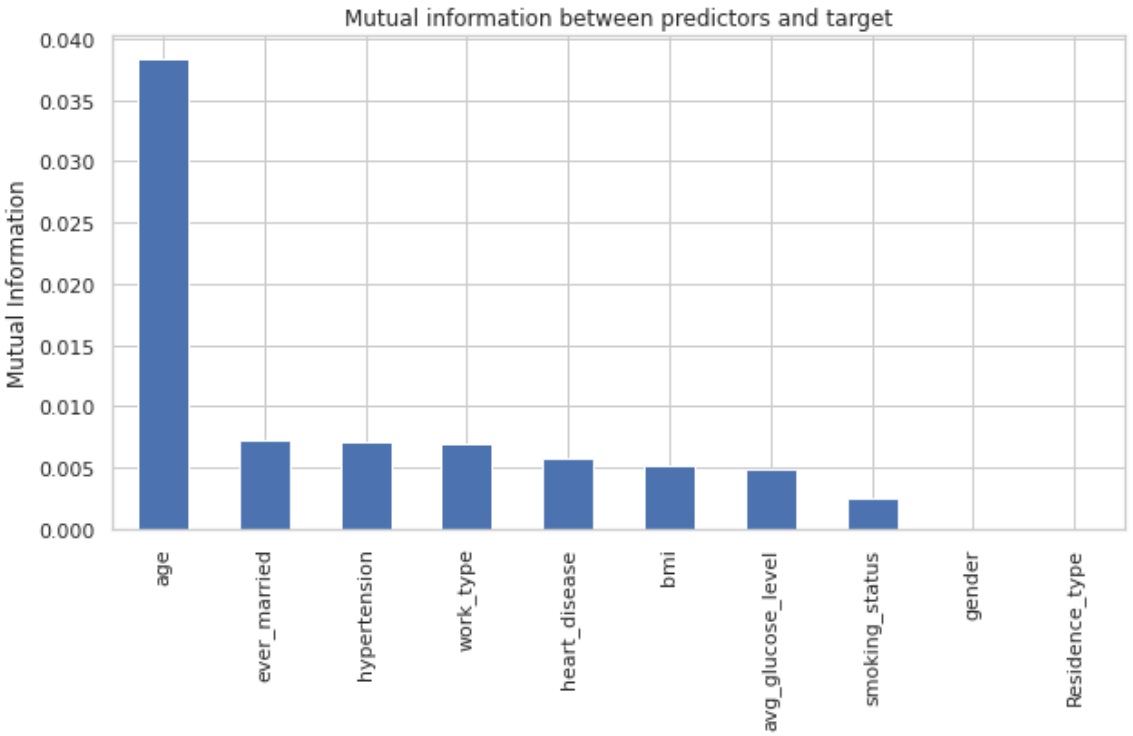
En relación a las variables categóricas binarias: *gender*, *hypertension*, *heart_disease*, *ever_married* y *residence_type*, se utiliza nuevamente el objeto ***LabelEncoder()*** de la librería ***sklearn*** para transformar sus valores a 0 o 1 (ver Código 5). Mientras que, para las variables categóricas no binarias: *work_type* y *smoking_status*, se aplica la técnica de ***OneHotEncoding*** (ver Código 5).

De esta manera, los conjuntos de entrenamiento y test finales se componen de un total de **14 variables explicativas**, sobre las cuales se llevará a cabo la selección de variables con el fin de reducir la dimensionalidad de los datos.

6 Selección de variables

En esta sección, se lleva a cabo la selección de variables utilizando el **criterio de información mutua (MI)** con el fin de reducir la dimensionalidad del conjunto de datos. La información mutua es un valor no negativo que mide la dependencia mutua entre dos variables aleatorias. Se utiliza para medir la cantidad de información que se puede obtener de una variable observando los valores de la otra variable. Este criterio es una alternativa al coeficiente de correlación de Pearson y es capaz de medir cualquier tipo de relación entre variables, no solo asociaciones lineales. Además, es adecuado tanto para variables continuas como discretas, a diferencia del coeficiente de correlación de Pearson.

Figura 19: Información mutua entre variables explicativas y objetivo



Fuente: Elaboración propia del autor

La librería de ***sklearn*** proporciona una implementación del criterio de información mutua para problemas de clasificación mediante la función ***mutual_info_classif()***. Los valores de información mutua correspondientes a cada variable explicativa con respecto a la variable objetivo en el conjunto de entrenamiento se presentan en la Figura 19. Así, se observa que las variables categóricas *gender* y *residence_type* presentan un valor de información mutua de cero, lo que significa que no aportan

información alguna sobre la variable objetivo y, por lo tanto, pueden ser eliminadas de los conjuntos finales de entrenamiento y test. Además, se puede apreciar que la variable que mayor cantidad de información aporta sobre la variable objetivo es *age*, lo cual concuerda con la Figura 7, donde se muestra que las personas de mayor edad tienen una mayor incidencia de accidentes cerebrovasculares.

Por lo tanto, tras la eliminación de las variables mencionadas, los conjuntos finales de entrenamiento y test constan de un total de 12 variables explicativas, sobre las cuales se trabajará en la resolución del problema de desequilibrio en la variable objetivo a predecir.

7 Sobre-muestreo de la clase minoritaria

En el análisis exploratorio de datos se evidenció un desequilibrio significativo en el conjunto de datos inicial. Por lo tanto, el objetivo de esta sección es abordar dicho problema mediante el **sobre-muestreo de la clase minoritaria**. Cabe destacar que este sobre-muestreo se aplicará únicamente en el conjunto de entrenamiento, con el fin de mantener la muestra de test sin manipular y utilizarla únicamente para evaluar la efectividad de los algoritmos de aprendizaje automático.

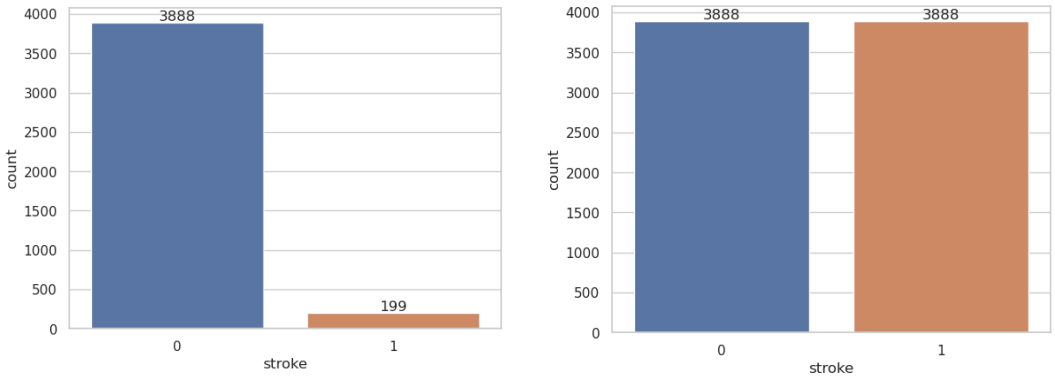
Código 6: Sobre-muestreo de la clase minoritaria

```
sm = BorderlineSMOTE(random_state=99, sampling_strategy='minority', kind='
    borderline-1')
X_train_smote, y_train_smote = sm.fit_resample(train.drop(['stroke'], axis=1),
    train.stroke)
```

Fuente: Elaboración propia del autor

Figura 20: Distribución de la variable objetivo en el conjunto de entrenamiento

(a) Previamente al sobre-muestreo (b) Posteriormente al sobre-muestreo

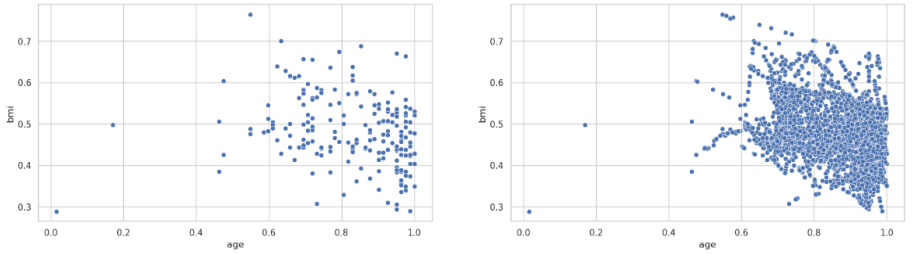


Fuente: Elaboración propia del autor

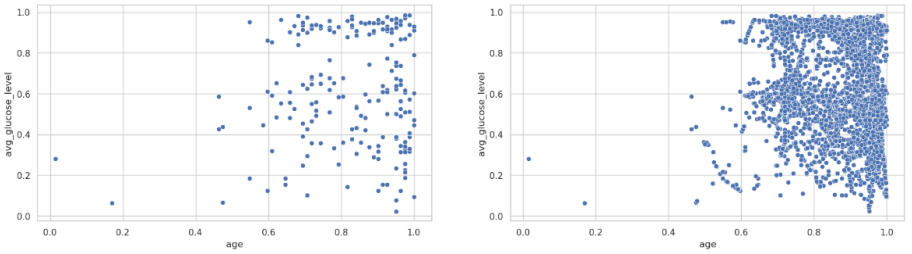
Para realizar el sobre-muestreo de la clase minoritaria se emplea el objeto ***BorderlineSMOTE()*** de la librería ***imblearn*** con la opción ***kind=borderline-1*** (ver Código 6). Cuando se especifica ***borderline-1***, cada muestra x_i se clasifica en una de tres categorías: (i) **ruido** (es decir, todos los vecinos más cercanos pertenecen a una clase diferente a la de x_i), (ii) **en peligro** (es decir, al menos la mitad de los vecinos más cercanos son de la misma clase que x_i) y (iii) **seguro** (es decir, todos los vecinos más cercanos son de la misma clase que x_i). ***Borderline-1*** **usará las muestras en peligro para generar nuevas muestras que pertenecerán a la misma clase que la muestra** x_i. Con lo anterior en mente, se presenta en la Figura 20 la distribución de la variable objetivo en el conjunto de entrenamiento antes (ver Figura 20a) y después (ver Figura 20b) del aumento de la clase minoritaria.

Figura 21: Variable continuas-categóricas antes y después del sobre-muestreo

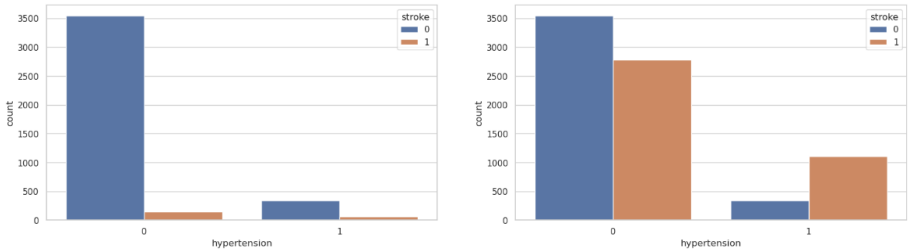
(a) age-bmi



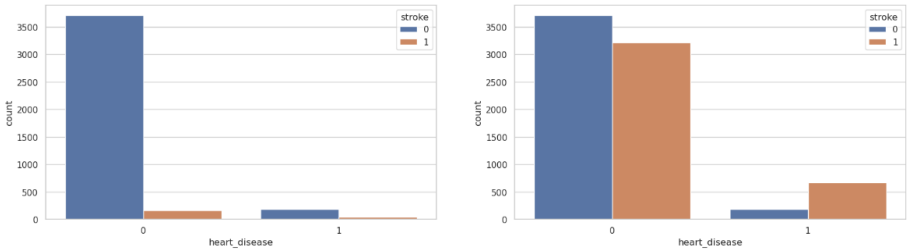
(b) age-avg_glucose_level



(c) hypertension



(d) heart_disease



Las figuras ubicadas a la izquierda indican la distribución de la variable antes del sobre-muestreo, mientras que las figuras ubicadas a la derecha muestran la distribución después del sobre-muestreo.

Fuente: Elaboración propia del autor

En conclusión, tal y como se aprecia en la Figura 21, el proceso de sobre-muestreo aplicado a la clase minoritaria ha generado muestras sintéticas con diferentes categorías, manteniendo la distribución en el caso de las variables continuas. Además, es importante mencionar que los conjuntos de entrenamiento y test constan de **7776** y **1022** observaciones, respectivamente, sobre las que se llevará a cabo el entrenamiento y evaluación de los distintos modelos de aprendizaje automático.

8 Modelado

En esta sección se procederá a evaluar diversos algoritmos de aprendizaje automático para la clasificación de pacientes con o sin accidente cerebro-vascular (variable objetivo *stroke*). Dado que se trata de un problema de clasificación binaria, se utilizará la métrica **área bajo la curva ROC (*ROC-AUC* en *sklearn*)** para evaluar el rendimiento de los distintos modelos.

Código 7: Objeto para la validación cruzada

```
StratifiedShuffleSplit(n_splits=5, random_state=99, test_size=0.2, train_size=0.8)
```

Fuente: Elaboración propia del autor

Además, todos los modelos que se evalúen en las siguientes sub-secciones utilizarán el mismo objeto ***StratifiedShuffleSplit()*** de la librería ***sklearn*** para la realización de la validación cruzada (ver Código 7). Dicha validación se llevará a cabo únicamente sobre el conjunto de entrenamiento, ya que el conjunto de prueba se reservará para la evaluación de los modelos ganadores. Por último, para cada modelo también se realizará una búsqueda exhaustiva de sus hiper-parámetros con el fin de seleccionar aquel que presente la mejor métrica posible.

8.1 Regresión logística

Tabla 2: Hiper-parámetros regresión logística

Parámetro	**Valores**						
penalty	l1			l2			
solver	liblinear			saga			
C	0.001	0.01	0.1	0.5	1	10	100

penalty: técnica de regularización; ***solver***: algoritmo a utilizar en el problema de optimización; ***C***: inverso de la intensidad de regularización; Las celdas en color verde indican los hiper-parámetros del modelo ganador

Fuente: Elaboración propia del autor

El primer algoritmo evaluado para la tarea de clasificación es la **regresión logística**. En la Tabla 2 se presentan los diferentes hiper-parámetros y sus respectivos valores evaluados durante el proceso de ajuste. En dicha tabla, el parámetro ***penalty*** se utiliza para evitar el sobre-ajuste (*overfitting*) del modelo y puede ser, entre otros, de tipo ***l1 (Lasso)***, el cual agrega el valor absoluto de los coeficientes como término de penalización, o de tipo ***l2 (Ridge)***, el cual agrega el cuadrado de los coeficientes como término de penalización. Por su parte, el parámetro ***C*** se refiere a la inversa de la fuerza de regularización. En este sentido, un valor mayor de C indica una fuerza de regularización más débil, lo que implica que se permiten coeficientes más grandes en la regresión y se ajusta mejor a los datos de entrenamiento. Por el contrario, un valor menor de C indica una fuerza de regularización más fuerte, lo que significa que se permiten coeficientes más pequeños en la regresión y se ajusta menos a los datos de entrenamiento.

Código 8: Modelo ganador de la regresión logística

```
log_reg_model = LogisticRegression(C=0.5, penalty='l2',
                                   solver='saga', random_state=99)
```

Fuente: Elaboración propia del autor

Una vez realizada la búsqueda de hiper-parámetros para la regresión logística, se concluye que el modelo seleccionado como ganador es el que se muestra en el Código 8, el cual logra obtener un área bajo la curva ROC promedio de **0.896** en la validación cruzada llevada a cabo.

8.2 Árbol de decisión

Tabla 3: Hiper-parámetros árbol de decisión

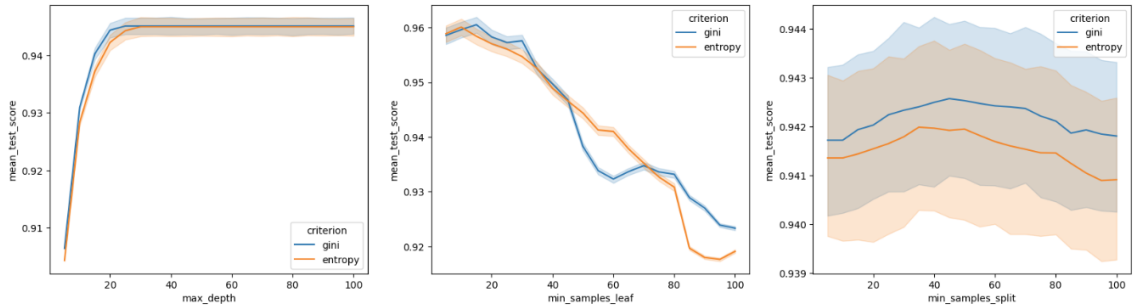
Parámetro	**Valores**																				
criterion	gini										entropy										
max_depth	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100	
min_samples_leaf	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100	
min_samples_split	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	100	200	

criterion: función utilizada para medir la calidad de la división; **max_depth**: profundidad máxima del árbol; ***min_samples_leaf***: número mínimo de muestras que se requieren en un nodo hoja; ***min_samples_split***: número mínimo de muestras requeridas para dividir un nodo interno; Las celdas en color verde indican los hiper-parámetros del modelo ganador

Fuente: Elaboración propia del autor

El objetivo de esta sub-sección es la búsqueda de un árbol de decisión con **baja complejidad** y **alta eficacia** en la clasificación de los pacientes, que sirva como base para los diferentes algoritmos de ***bagging*** y ***boosting*** en secciones posteriores. Para lograr esto, en la Tabla 3 se definen diferentes hiper-parámetros, tales como el criterio de división, la profundidad máxima del árbol, etc, junto con los valores que se probarán durante la optimización de dichos hiper-parámetros.

Figura 22: Hiper-parámetros árbol de decisión



Fuente: Elaboración propia del autor

Código 9: Modelo ganador del árbol de decisión

```
decision_tree_model = DecisionTreeClassifier(
                            criterion='gini', max_depth=10,
                            min_samples_leaf=45, min_samples_split=200,
                            random_state=99)
```

Fuente: Elaboración propia del autor

Tras realizar la búsqueda exhaustiva de los hiper-parámetros del árbol de decisión (ver Figura 22), se concluye que el criterio de división ***gini*** es el que ofrece los mejores resultados en términos de AUC-ROC. Además, se observa que la longitud máxima del árbol parece ser de aproximadamente **10**, ya que a partir de dicho punto la mejora en la métrica seleccionada no supera el 1% y solo genera un árbol más complejo. También se concluye que el número mínimo de muestras por nodo hoja es de **20**; sin embargo, se selecciona el valor **45** debido a que la disminución en la métrica es solo del 1% y se obtiene un árbol menos complejo. Finalmente, se selecciona el valor **200** como el número mínimo de muestras requeridas para dividir un nodo interno después de realizar pruebas por error, y ver que la métrica apenas varía. El modelo seleccionado en esta sub-sección se encuentra definido en el Código 9.

8.3 Bagging con árbol de decisión

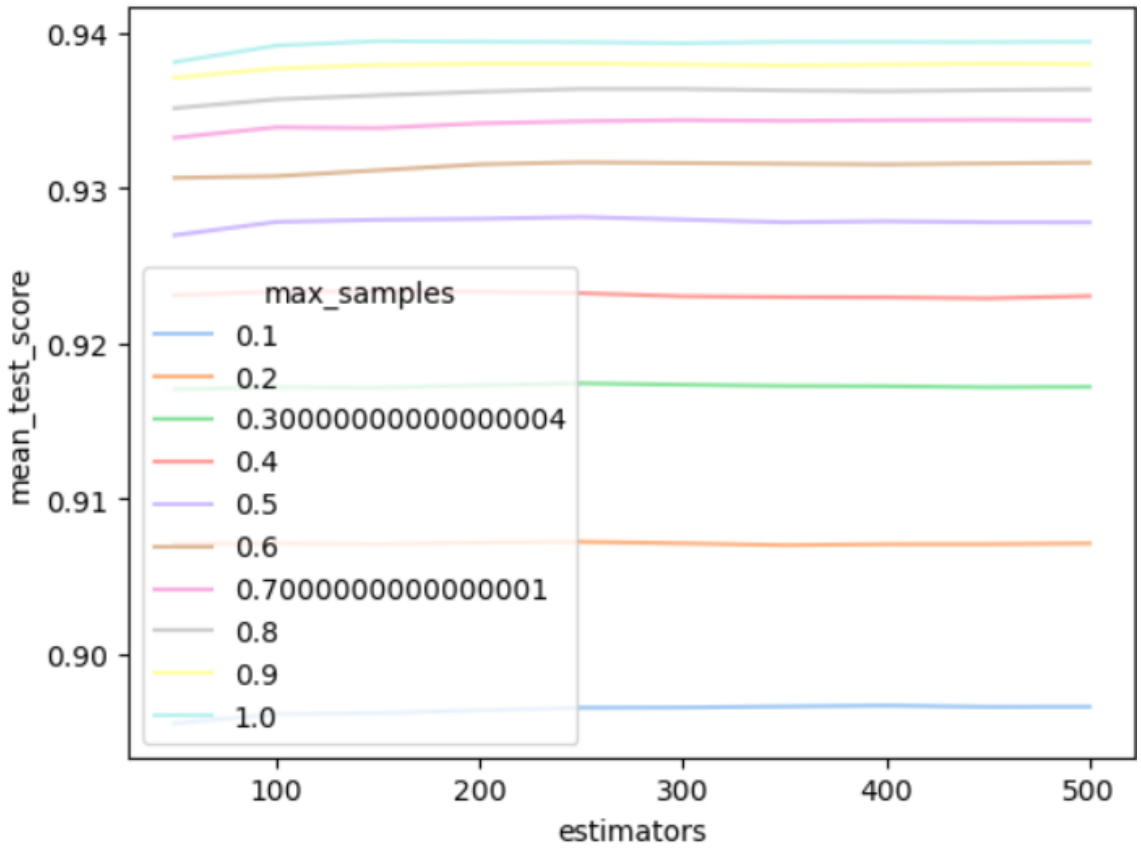
Tabla 4: Hiper-parámetros bagging con árbol de decisión

Parámetro	**Valores**									
n_estimators	50	100	150	200	250	300	350	400	450	500
max_samples	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1

n_estimators: número de estimadores base que se utilizarán en el ensamblado; ***max_samples***: porcentaje de muestras para el entrenamiento del estimador base (con reemplazo por defecto); Las celdas en color verde indican los hiper-parámetros del modelo ganador

Fuente: Elaboración propia del autor

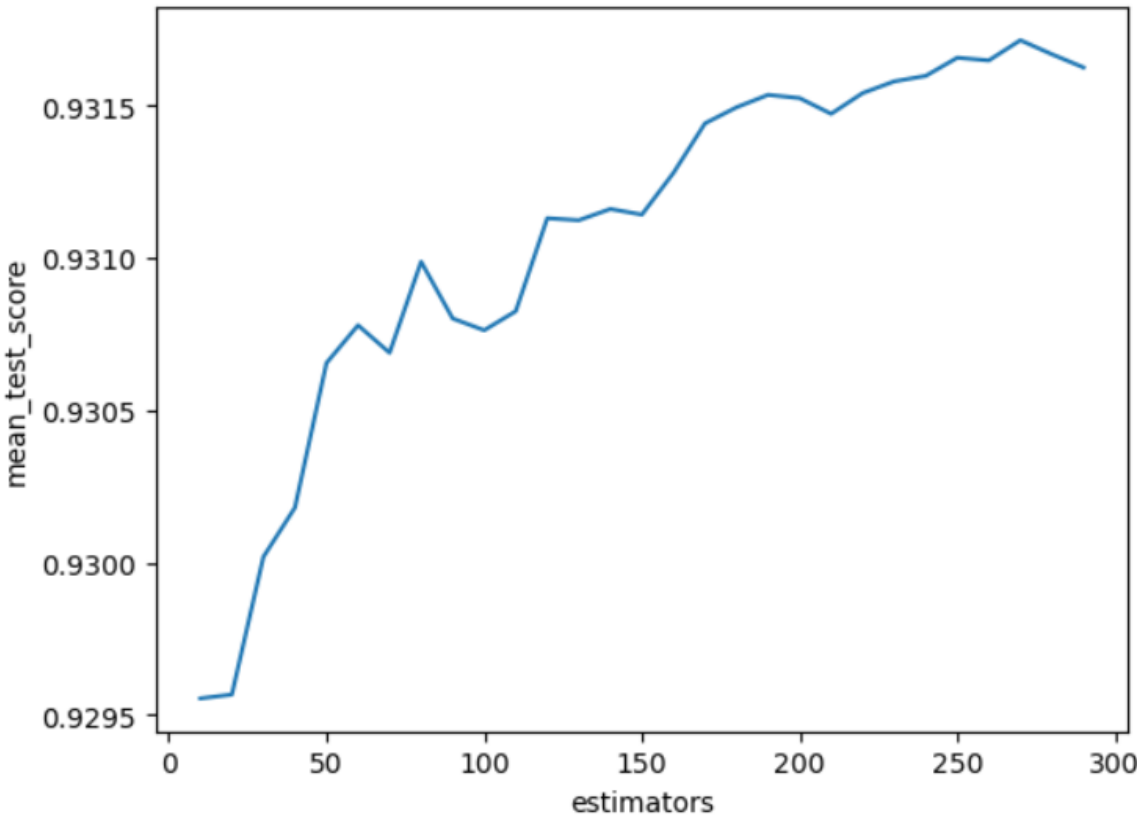
Figura 23: Hiper-parámetros bagging con árbol de decisión



Fuente: Elaboración propia del autor

En este apartado se lleva a cabo la búsqueda del mejor modelo de *bagging*, utilizando como estimador base el árbol de decisión definido en la Sección 8.2 y como hiper-parámetros los especificados en la Tabla 4. Los resultados obtenidos indican que la mejora en la métrica seleccionada deja de ser significativa cuando se toma más del **60%** de la muestra para el entrenamiento de los estimadores base, ya que los últimos cuatro porcentajes se encuentran bastante cercanos entre sí (ver Figura 23).

Figura 24: Búsqueda del parámetro n_estimators para bagging con árbol de decisión



Fuente: Elaboración propia del autor

Código 10: Modelo ganador del bagging con árbol de decisión

```
bagging_tree_model = BaggingClassifier(
  base_estimator=DecisionTreeClassifier(
    criterion=decision_tree_model.criterion,
    max_depth=decision_tree_model.max_depth,
    min_samples_leaf=decision_tree_model.min_samples_leaf,
    min_samples_split=decision_tree_model.min_samples_split,
    random_state=99
  ),
  n_estimators=10, max_samples=0.6, random_state=99
)
```

Fuente: Elaboración propia del autor

En la Figura 23 también se puede apreciar que el aumento en el número de estimadores no mejora el área bajo la curva ROC, independientemente del porcentaje de muestra utilizado. Por lo tanto, se realiza una búsqueda más detallada del parámetro ***n_estimators*** en un rango de 10 a 300. Se observa que no existe una mejora significativa al variar el número de estimadores base entre 10 y 290. Por consiguiente, se selecciona el valor **10** como número de estimadores base para el modelo final de *bagging* con árbol de decisión (ver Código 10).

8.4 Bagging con regresión logística

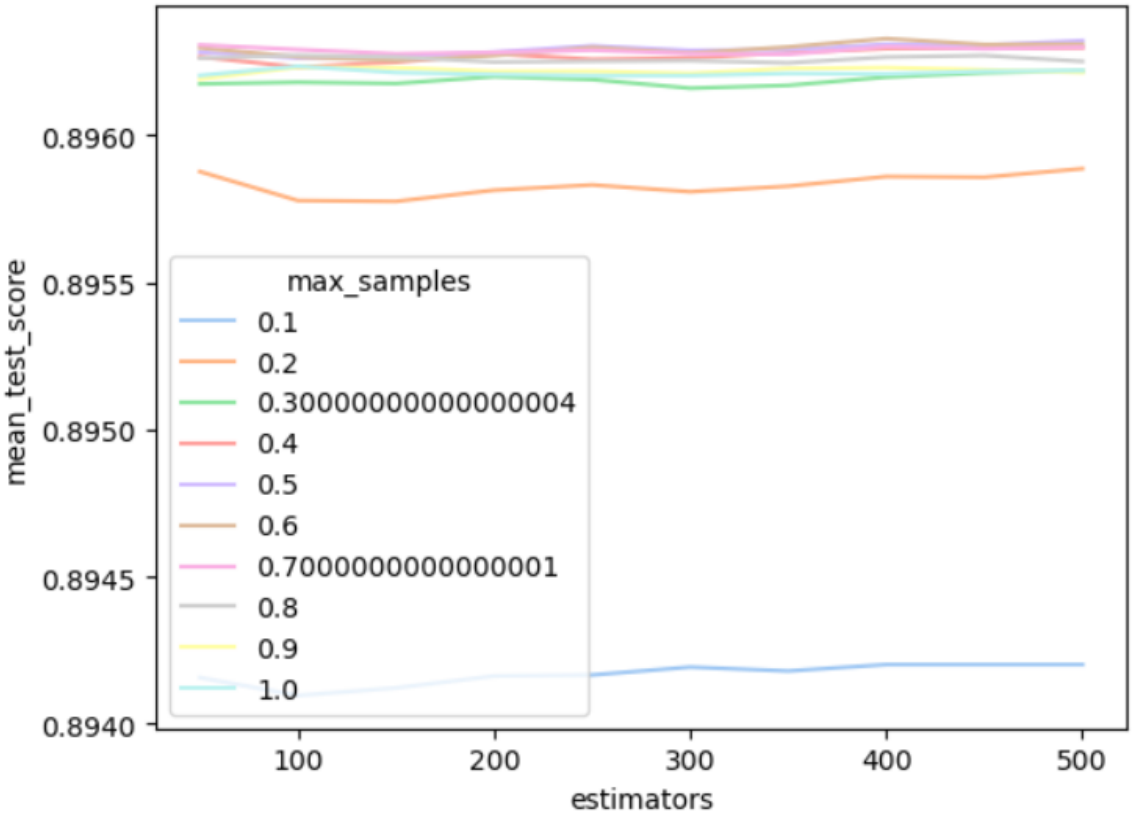
Tabla 5: Hiper-parámetros bagging con regresión logística

Parámetro	**Valores**									
n_estimators	50	100	150	200	250	300	350	400	450	500
max_samples	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1

n_estimators: número de estimadores base que se utilizarán en el ensamblado; ***max_samples***: porcentaje de muestras para el entrenamiento del estimador base (con reemplazo por defecto); Las celdas en color verde indican los hiper-parámetros del modelo ganador

Fuente: Elaboración propia del autor

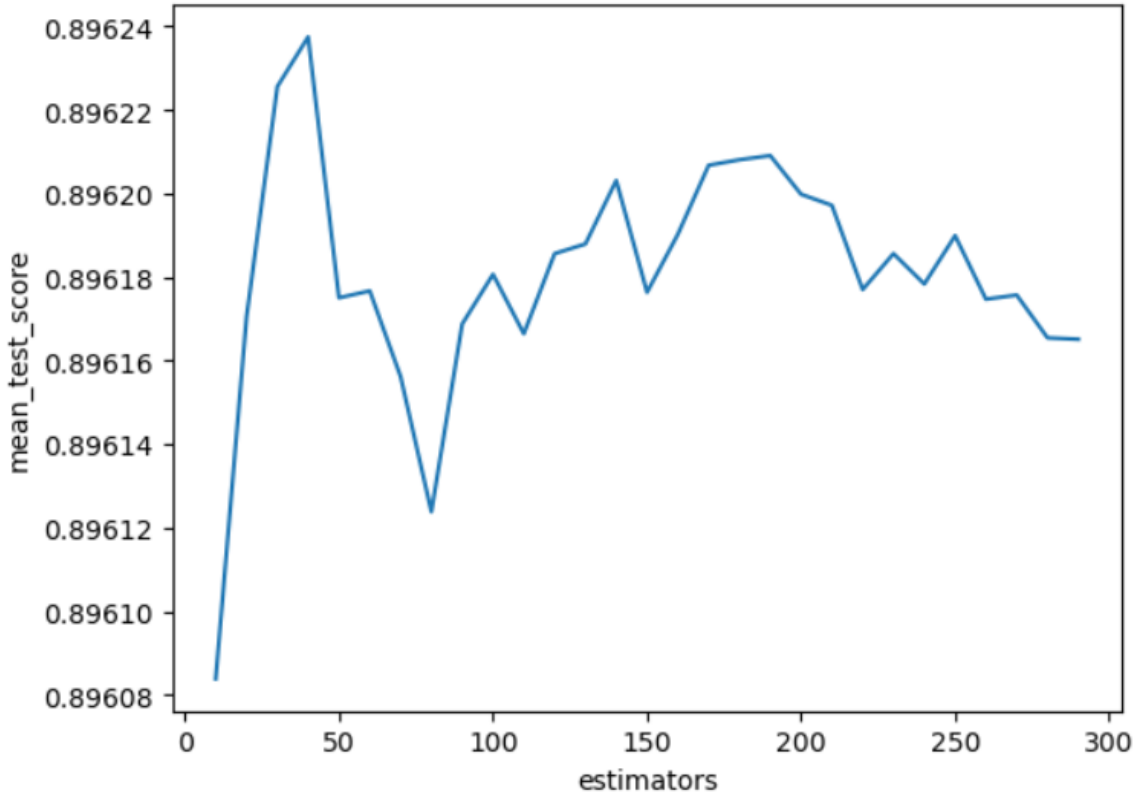
Figura 25: Hiper-parámetros bagging con regresión logística



Fuente: Elaboración propia del autor

Se procede a realizar de nuevo la búsqueda del mejor modelo de ***bagging***, utilizando como estimador base la regresión logística definida en la Sección 8.1 y como hiper-parámetros los especificados en la Tabla 5. Los resultados obtenidos muestran que, a partir del **30%** de la muestra, el área bajo la curva ROC deja de aumentar significativamente, ya que los últimos porcentajes se encuentran cercanos entre sí (ver Figura 25).

Figura 26: Búsqueda del parámetro n_estimators para bagging con regresión logística



Fuente: Elaboración propia del autor

Código 11: Modelo ganador del bagging con regresión logística

```
bagging_log_reg_model = BaggingClassifier(
  base_estimator=LogisticRegression(
    C=log_reg_model.C, solver=log_reg_model.solver,
    random_state=log_reg_model.random_state
  ),
  n_estimators=10, max_samples=0.3, random_state=99
)
```

Fuente: Elaboración propia del autor

Se presenta nuevamente la situación en la que, a medida que se aumenta el número de estimadores, el área bajo la curva ROC no mejora significativamente, independientemente del porcentaje de muestra utilizado. Debido a esto, se realiza una búsqueda detallada del hiper-parámetro ***n_estimators*** dentro del rango de 10 a 300, y se observa que no hay una mejora significativa en el desempeño del modelo (ver Figura 26). Por lo tanto, se selecciona el valor **10** como número de estimadores base para el modelo final de *bagging* con regresión logística (ver Código 11)

8.5 Random Forest

Tabla 6: Hiper-parámetros random forest

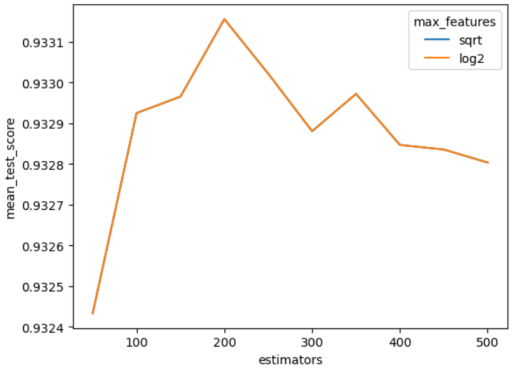
Parámetro	**Valores**									
n_estimators	10	20	30	40	50	60	70	80	...	300
max_samples	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
max_features	sqrt			log2			None			

n_estimators: número de estimadores base (árboles de decisión) que se utilizarán en el ensamblado; ***max_samples***: porcentaje de muestras para el entrenamiento del estimador base (con reemplazo por defecto); ***max_features***: número máximo de características consideradas para la mejor división; Las celdas en color verde indican los hiper-parámetros del modelo ganador

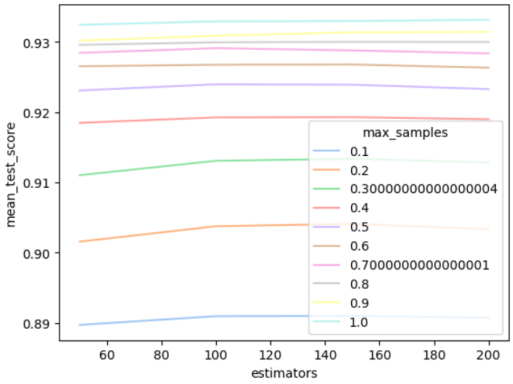
Fuente: Elaboración propia del autor

Figura 27: Hiper-parámetros random forest

(a) max_features



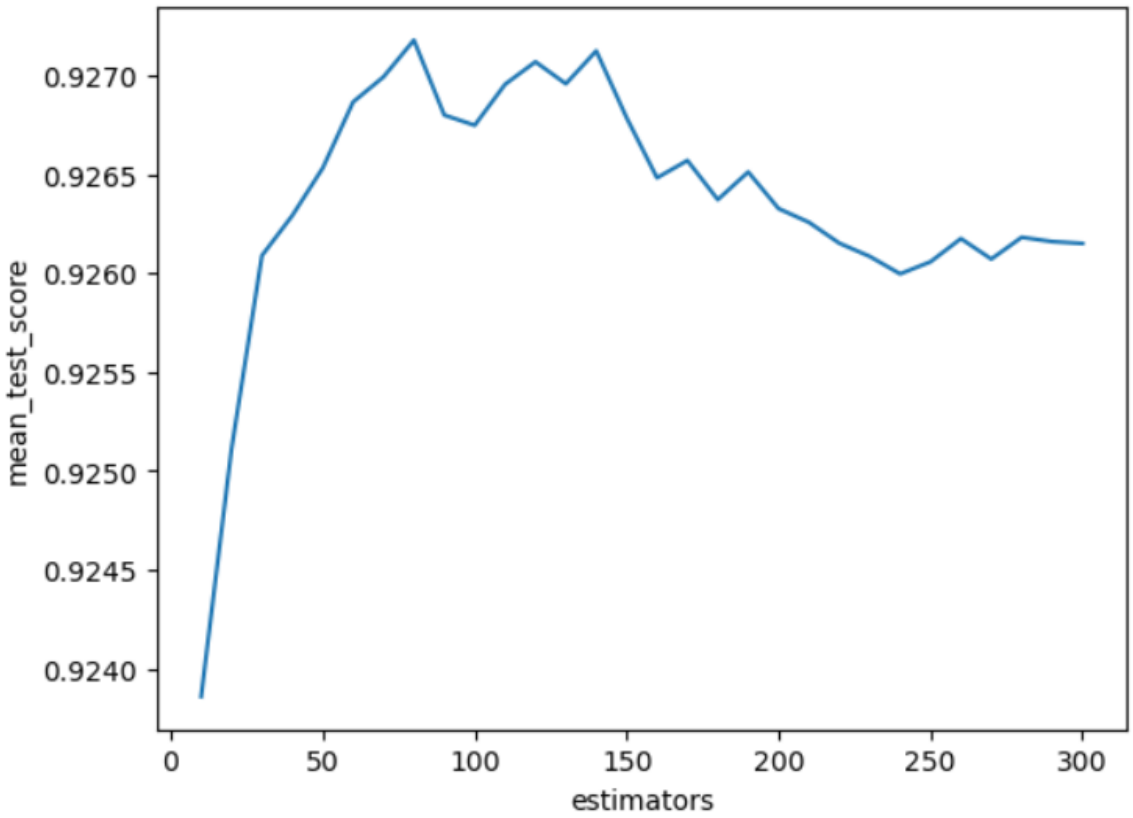
(b) max_samples



Fuente: Elaboración propia del autor

Se procede con la búsqueda del mejor modelo de *random forest*, utilizando como estimador base el árbol de decisión definido en la Sección 8.2 y llevando a cabo la exploración de los hiper-parámetros especificados en la Tabla 6. Así, la Figura 27 muestra que el área bajo la curva ROC es idéntica para los dos posibles valores del parámetro *max_features*: *sqrt* (raíz cuadrada) y *log2* (logaritmo en base 2). Esta igualdad se debe a que en el conjunto de entrenamiento existen doce variables explicativas y al aplicar la raíz cuadrada o el logaritmo en base 2, ambos valores resultan en el mismo número, por lo que la elección de uno u otro es irrelevante en este caso. No obstante, se observa que a partir del **60%** de la muestra, la métrica seleccionada deja de crecer de manera significativa, como se puede apreciar en la Figura 27b.

Figura 28: Búsqueda del parámetro n_estimators para random forest



Fuente: Elaboración propia del autor

Código 12: Modelo ganador del random forest

```
random_forest_model = RandomForestClassifier(
  criterion=decision_tree_model.criterion,
  max_depth=decision_tree_model.max_depth,
  max_features='sqrt', max_samples=0.6, n_estimators=10,
  min_samples_leaf=decision_tree_model.min_samples_leaf,
  min_samples_split=decision_tree_model.min_samples_split,
  random_state=99
)
```

Fuente: Elaboración propia del autor

Nuevamente, sucede la situación en la que, a medida que se aumenta el número de estimadores, el área bajo la curva ROC no mejora significativamente, independientemente del porcentaje de muestra utilizado. Debido a esto, se realiza una búsqueda detallada del hiper-parámetro *n_estimators* dentro del rango de 10 a 300, y se observa que no hay una mejora significativa en el desempeño del modelo (ver Figura 28). Por lo tanto, se selecciona el valor **10** como número de estimadores base para el modelo final de random forest (ver Código 12)

8.6 Gradient Boosting

Tabla 7: Hiper-parámetros gradient boosting

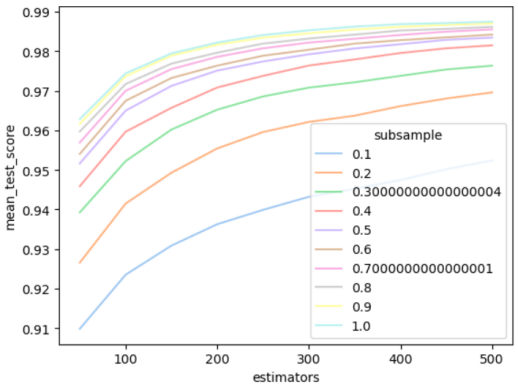
Parámetro	**Valores**									
n_estimators	10	20	30	40	50	60	70	80	...	1000
subsample	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
learning_rate	0.1		0.05		0.03		0.01		0.001	

n_estimators: número de etapas del boosting; ***subsample***: porcentaje de muestras utilizada para ajustar cada árbol; ***learning_rate***: contribución de cada árbol a la predicción final del modelo; Las celdas en color verde indican los hiper-parámetros del modelo ganador

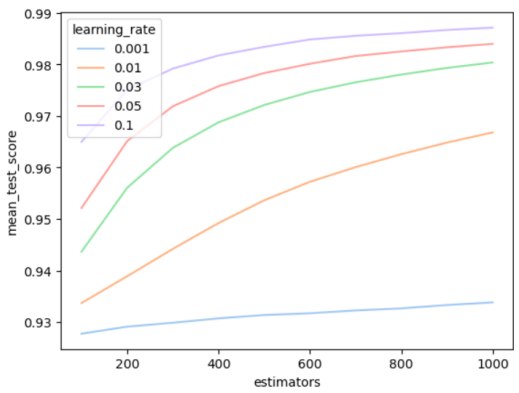
Fuente: Elaboración propia del autor

Figura 29: Hiper-parámetros gradient boosting

(a) subsample



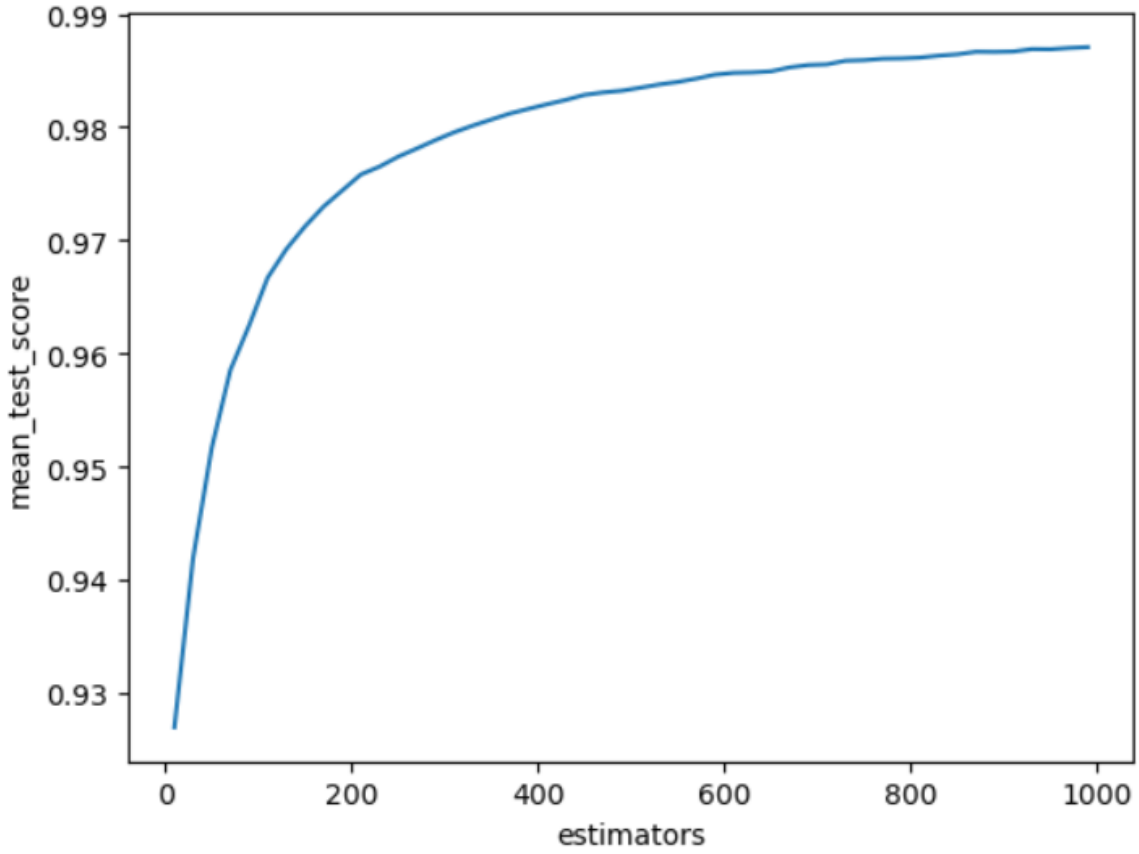
(b) learning_rate



Fuente: Elaboración propia del autor

En la presente sub-sección se realizará el proceso de entrenamiento y búsqueda de hiper-parámetros del modelo de *Gradient Boosting*, haciendo uso del árbol de decisión descrito en la Sección 8.2 como ***weak learner***. En particular, en este informe se empleará la versión estocástica del algoritmo mencionado al incorporar el parámetro ***subsample*** (ver Tabla 7). La versión estocástica del *Gradient Boosting* se basa en la selección aleatoria de subconjuntos de los datos de entrenamiento y de las características para cada árbol en el proceso de construcción del modelo, lo que puede reducir el sobre-ajuste y mejorar la eficiencia computacional.

Figura 30: Búsqueda del parámetro n_estimators para el gradient boosting



Fuente: Elaboración propia del autor

Código 13: Modelo ganador del gradient boosting

```
gradient_boosting_model = GradientBoostingClassifier(
  max_features='sqrt', learning_rate=0.1, subsample=0.5,
  max_depth=decision_tree_model.max_depth,
  min_samples_leaf=decision_tree_model.min_samples_leaf,
  n_estimators=20,
  min_samples_split=decision_tree_model.min_samples_split,
  random_state=99
)
```

Fuente: Elaboración propia del autor

En función de lo expuesto con anterioridad, en la Figura 29a se puede constatar que, a partir de valores superiores al **50%** del parámetro *subsample*, el incremento en el área bajo la curva ROC carece de significancia. Además, se puede constatar que el *learning_rate* que proporciona los mejores resultados es de **0.1**. Por último, se procede a realizar una búsqueda más exhaustiva del hiper-parámetro *n_estimators* en un rango de 10 a 1000, concluyendo que el mejor modelo (ver Código 13) en términos de la métrica seleccionada es aquel que utiliza 20 *weak learners*.

8.7 XGBoosting

Tabla 8: Hiper-parámetros xgboosting

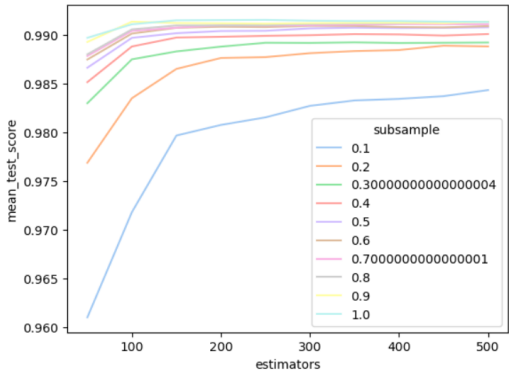
Parámetro	**Valores**									
n_estimators	10	20	30	40	50	60	70	80	...	1000
subsample	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
eta	0.001		0.01		0.03		0.05		0.1	
lambda	0.001		0.01		0.1		0.5		1	

n_estimators: número de árboles de decisión que se construirán durante el entrenamiento del modelo; ***subsample***: porcentaje de muestras utilizada para ajustar cada árbol; ***eta***: contribución de cada árbol a la predicción final del modelo; ***lambda***: término de regularización L2 (regresión *Ridge*) que se aplica los pesos del modelo durante el entrenamiento; Las celdas en color verde indican los hiper-parámetros del modelo ganador

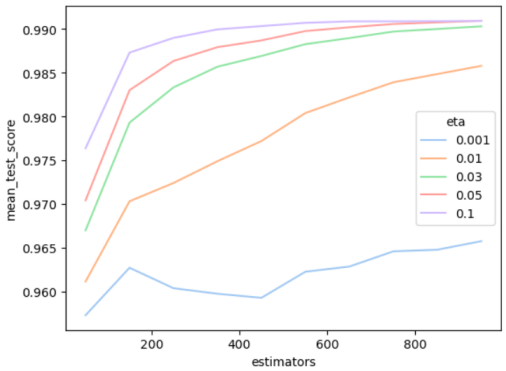
Fuente: Elaboración propia del autor

Figura 31: Hiper-parámetros xgboosting

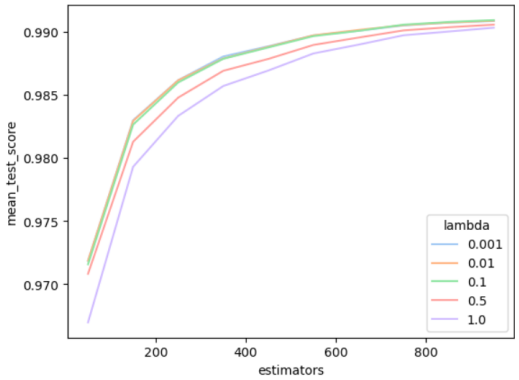
(a) subsample



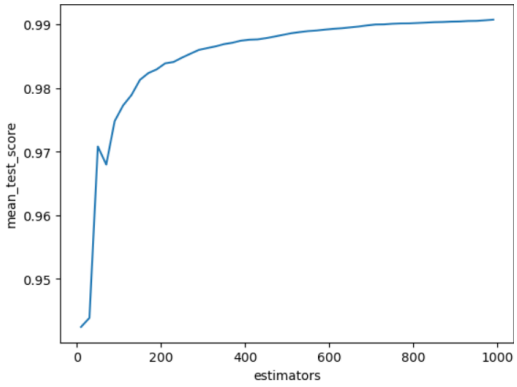
(b) eta



(c) lambda



(d) n_estimators



Fuente: Elaboración propia del autor

En esta sub-sección, se realizará el proceso de entrenamiento y búsqueda de hiper-parámetros para el modelo *XGBoosting* (ver Tabla 8). Para ello se utilizará el árbol de decisión descrito en la Sección 8.2 como *weak learner*. De este modo, en la Figura 31a, se observa que a partir de un valor de **0.4**, el parámetro *subsample* no mejora significativamente la métrica seleccionada. Además, se encuentra que una **tasa de aprendizaje (*eta*)** baja de 0.03 es capaz de lograr niveles cercanos a los obtenidos con valores de eta más altos. En cuanto a la **regularización (*lambda*)**, se opta por una penalización L2 conocida como *Ridge*, y se aprecia que el valor óptimo para el parámetro *lambda* es de **0.5**. Finalmente, por prueba y error, se concluye que el número de estimadores óptimo (parámetro *n_estimators*) es de **20**.

Código 14: Modelo ganador del xgboosting

```
xgboosting_model = XGBClassifier(
  max_depth=decision_tree_model.max_depth,
  colsample_bytree=0.3, n_estimators=20,
  reg_lambda=0.5, subsample=0.4, eta=0.03,
  random_state=99
)
```

Fuente: Elaboración propia del autor

En conclusión, en el Código 15 se presenta el modelo ganador que se considerará en la elección del modelo final, junto con el resto de modelos ganadores.

8.8 Support Vector Machine

Tabla 9: Hiper-parámetros support vector machine

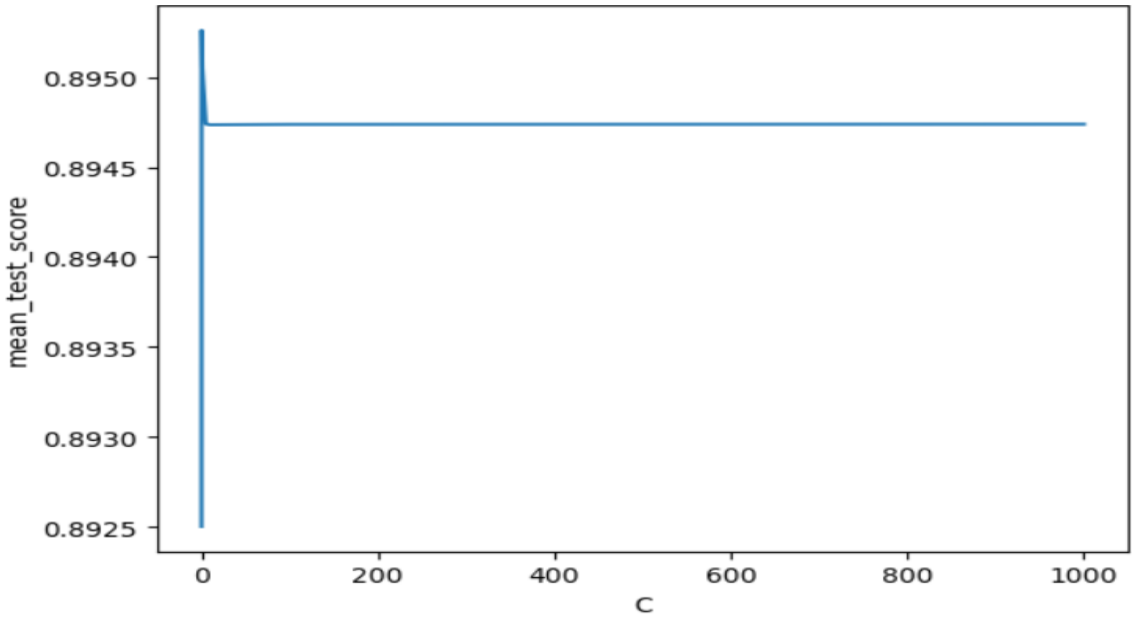
Parámetro	**Valores**										
Kernel Lineal											
C	0.01	0.05	0.1	0.2	0.5	1	2	5	10	100	1000
Kernel Poly											
C	0.01			0.1			1			2	
degree	2						3				
gamma	0.1			0.5			1			2	
Kernel RBF											
C	0.01			0.1		1		2		5	
gamma	0.1			0.5			1			2	

C: parámetro de regularización; ***degree***: grado del polinomio para la función de kernel polinómico; ***gamma***: influencia de un solo punto de datos en el modelo; Las celdas en color verde indican los hiper-parámetros del modelo ganador

Fuente: Elaboración propia del autor

En esta sección, se lleva a cabo la búsqueda del mejor clasificador mediante el uso de máquinas de vector soporte (SVM) con tres diferentes funciones de *kernel*: **lineal**, **polinómico (*poly*)** y **radial (*rbf*)**. Además, se realiza la búsqueda de hiper-parámetros para cada uno de estos *kernels*, partiendo de la Tabla 9.

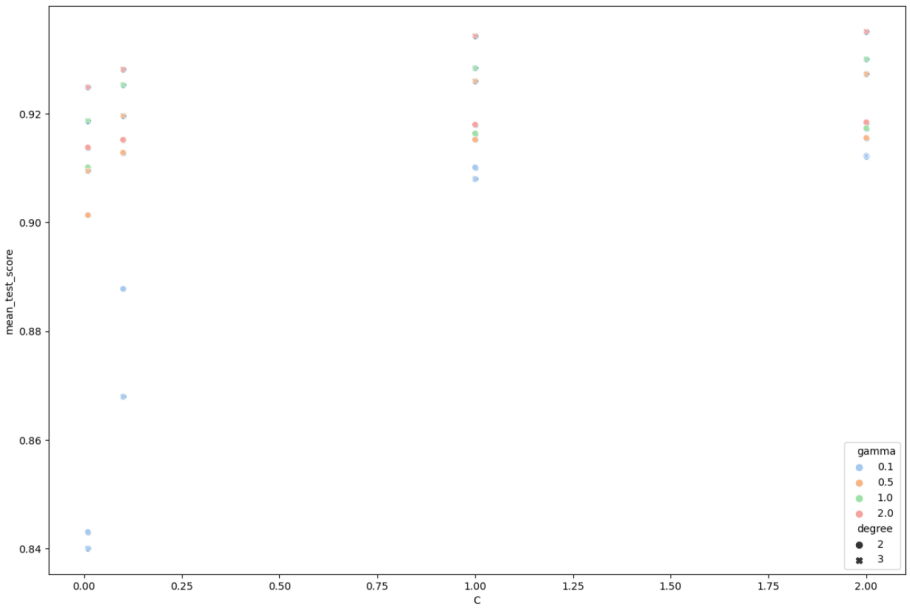
Figura 32: Búsqueda del hiper-parámetro C para el kernel lineal



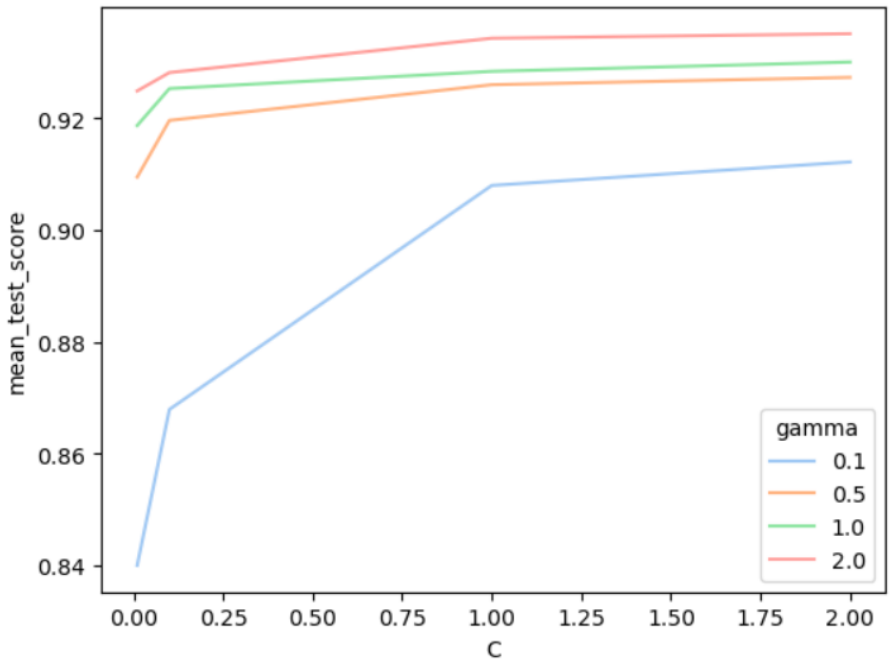
Fuente: Elaboración propia del autor

Figura 33: Búsqueda de los hiper-parámetros para el kernel polinómico

(a) C, degree y gamma



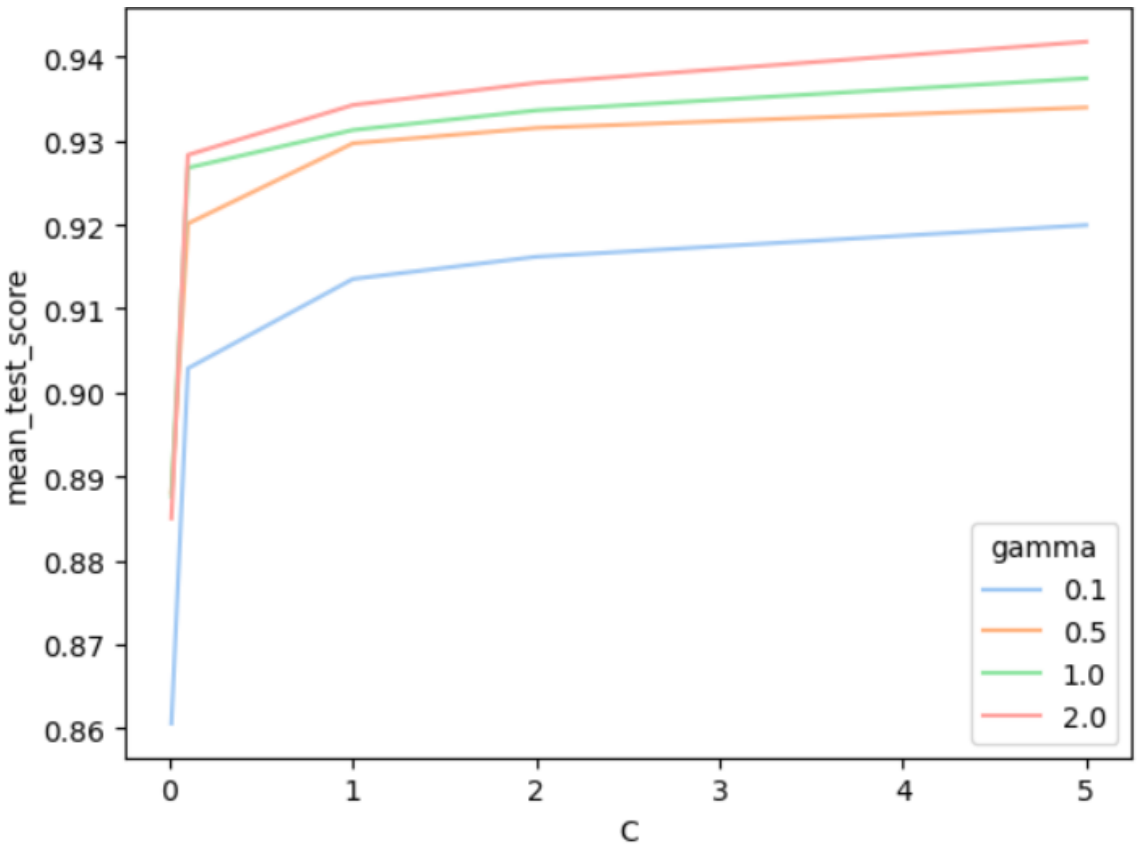
(b) C y gamma



Fuente: Elaboración propia del autor

En vista de lo expuesto anteriormente, se ha llevado a cabo la búsqueda del **hiper-parámetro de regularización (C)** para el *kernel* lineal, llegando a la conclusión de que el valor de **0.01** ofrece el mejor rendimiento en función de la métrica utilizada (ver Figura 32). Además, se ha realizado la búsqueda de hiper-parámetros para el ***kernel* polinómico**, encontrando que un **orden de tercer grado** es el que arroja mejores resultados en términos del área bajo la curva ROC (ver Figura 33a). Por lo tanto, a partir de dicho orden, se ha concluido que los valores óptimos para los hiper-parámetros *gamma* y *C* son **0.5** y **0.01**, respectivamente (ver Figura 33b).

Figura 34: Búsqueda del hiper-parámetro C y gamma para el kernel radial



Fuente: Elaboración propia del autor

Continuando con el mismo enfoque metodológico, se lleva a cabo la búsqueda de los hiper-parámetros *C* y *gamma* para el ***kernel* radial**, y se determina que los valores óptimos son de 1 y 0.5, respectivamente.

Código 15: Modelos ganadores del support vector machine

```
# kernel lineal
svc_lineal_model = SVC(kernel='linear', probability=True, C=0.01, random_state=99)

# kernel poly
svc_poly_model = SVC(kernel='poly', degree=3, gamma=0.5, C=0.01, random_state=99)

# kernel rbf
svc_rbf_model = SVC(kernel='rbf', gamma=0.5, C=1, random_state=99)
```

Fuente: Elaboración propia del autor

Finalmente, se presenta en el Código 15 los modelos con mejores rendimientos para los distintos *kernels* evaluados en esta sección, los cuales serán utilizados en la comparativa final para la selección del modelo ganador para la predicción de accidentes cerebro-vasculares.

8.9 Red neuronal

Tabla 10: Hiper-parámetros red neuronal

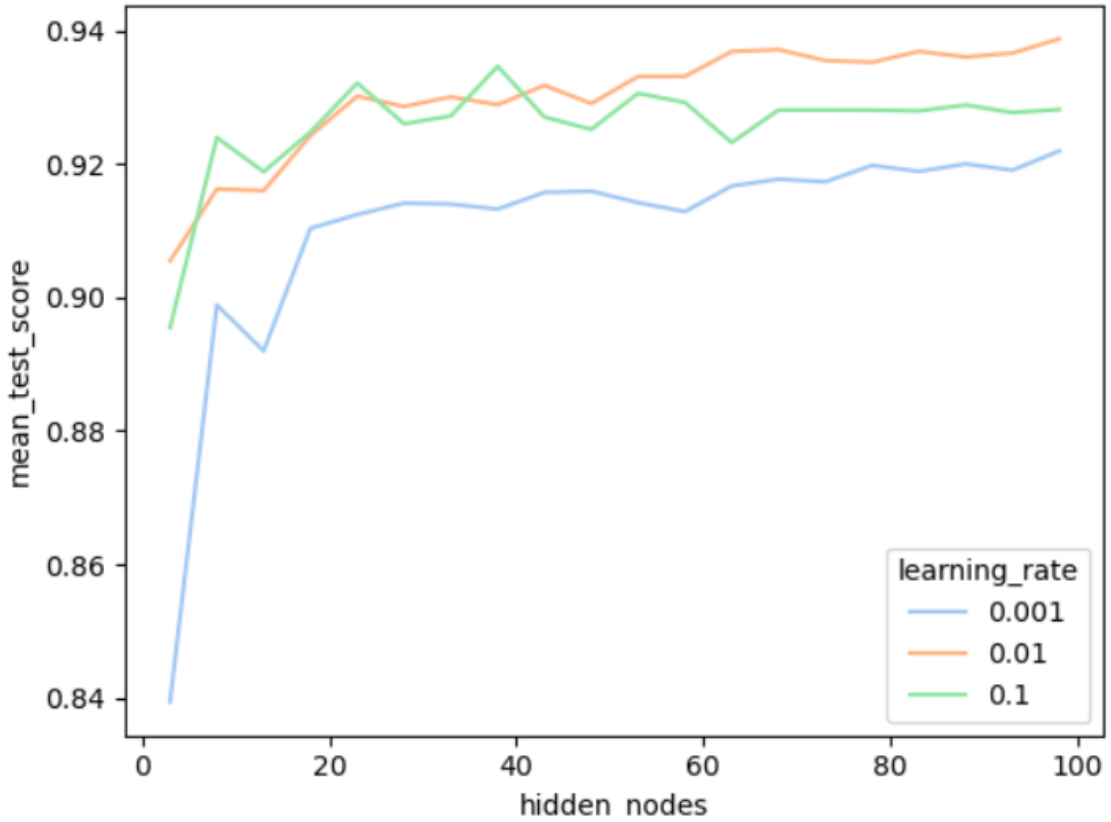
Parámetro	**Valores**									
hidden_layer_sizes	3	8	13	18	23	28	33	40	...	98
learning_rate_init	0.001				0.01			0.1		

hidden_layer_sizes: número de neuronas en la capa oculta; ***learning_rate_init***: ritmo de aprendizaje para la actualización de los pesos de la red neuronal; Las celdas en color verde indican los hiper-parámetros del modelo ganador

Fuente: Elaboración propia del autor

El propósito de esta sub-sección es la búsqueda de una red neuronal que permita la predicción de accidentes cerebro-vasculares con un alto rendimiento. Es importante destacar que los tipos de redes que se evaluarán en esta sección son redes simples, compuestas únicamente por tres capas: entrada, oculta y salida. En consecuencia, en la Tabla 10 se presentan los distintos hiper-parámetros que serán ajustados durante la búsqueda de la red mencionada, junto con su respectivo significado.

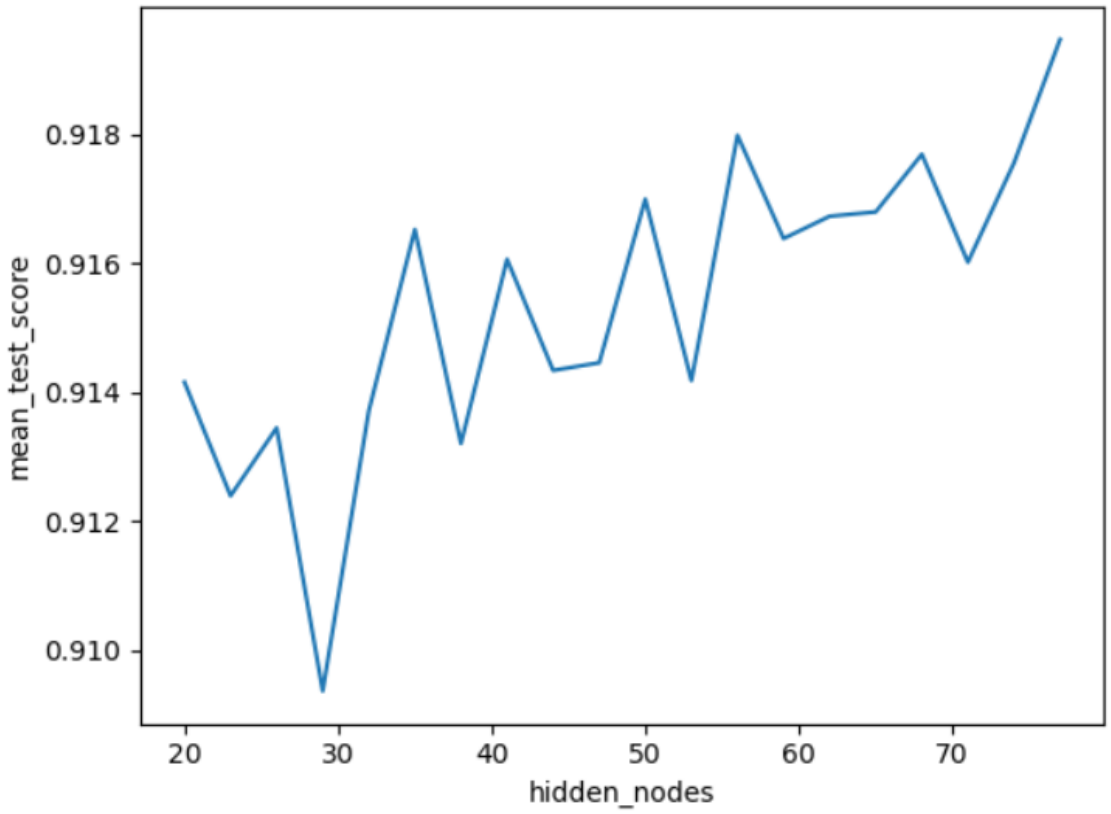
Figura 35: Búsqueda de los hiper-parámetros red neuronal



Fuente: Elaboración propia del autor

Realizando la búsqueda de los hiper-parámetros, se ha observado que a pesar de que el *learning_rate_init* de **0.001** ofrece el peor resultado en términos del área bajo la curva ROC, permite la obtención de un modelo menos sobre-ajustado y no presenta una gran diferencia en comparación con los otros valores (ver Figura 35). Por otro lado, se ha observado que el **número óptimo de neuronas** en la capa oculta parece estar en el rango de **20 a 80** (ver Figura 35).

Figura 36: Búsqueda de los hiper-parámetro hidden_layer_sizes



Fuente: Elaboración propia del autor

Código 16: Modelo ganador de la red neuronal

```
nn_model = MLPClassifier(
  activation='relu', solver='adam', hidden_layer_sizes=(40,),
  learning_rate_init=0.001, max_iter=1000, tol=0.005, random_state=99
)
```

Fuente: Elaboración propia del autor

Por lo tanto, fijando el valor de *learning_rate_init* en 0.001, se llevó a cabo una búsqueda más limitada para determinar el número óptimo de neuronas en la capa oculta. De esta forma, mediante un proceso de prueba y error, se determinó que el número óptimo de neuronas en la capa oculta es **40**, tal como se muestra en la Figura 36. Finalmente, en el Código 16 se presenta la red neuronal seleccionada para la fase final donde se compararán los distintos modelos ganadores.

8.10 Stacking

El ***stacking* de modelos** es una técnica de aprendizaje automático que tiene como objetivo mejorar el rendimiento predictivo de un modelo mediante la combinación de varios modelos más simples en una estructura jerárquica. En esencia, el proceso de *stacking* combina las predicciones de varios **modelos base** para crear un modelo de nivel superior, conocido como **meta-modelo**, que utiliza las predicciones de los modelos base como características para hacer una predicción final.

En esta sub-sección se han seleccionado como **modelos base** el **árbol de decisión** (ver Sección 8.2), la **máquina de vector soporte con el *kernel* lineal** (ver Sección 8.8) y la **red neuronal** (ver Sección 8.9) para llevar a cabo la técnica de *stacking* de modelos. Por otro lado, el **meta-modelo**, o modelo que combina las predicciones de los anteriores modelos, ha sido escogido como la **regresión logística** por su menor complejidad. Asimismo, es importante destacar que la elección de dichos modelos no ha sido aleatoria, sino que se basa en los resultados que han demostrado estos en el conjunto de test que se introducirá en la siguiente sección.

Código 17: Modelo stacking

```
# modelos base
level0 = list()
level0.append(('decision_tree', decision_tree_model))
level0.append(('svc_lineal', svc_lineal_model))
level0.append(('nn', nn_model))

# meta-modelo
level1 = LogisticRegression(random_state=99)

# modelo stacking
stacking_model = StackingClassifier(
    estimators=level0, final_estimator=level1,
    cv=5, stack_method='predict_proba', verbose=0,
    n_jobs=-1
)
```

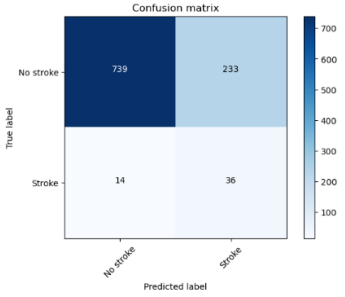
Fuente: Elaboración propia del autor

Por último, se presenta en el Código 17 el modelo de stacking que será evaluado en las próximas secciones junto con los demás modelos candidatos para la predicción de accidentes cerebro-vasculares.

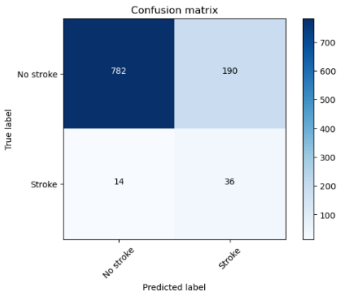
9 Evaluación de los modelos ganadores

Figura 37: Matrices de confusión

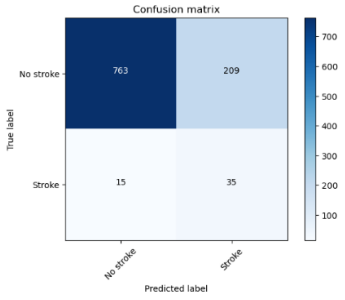
(a) Regresión logística



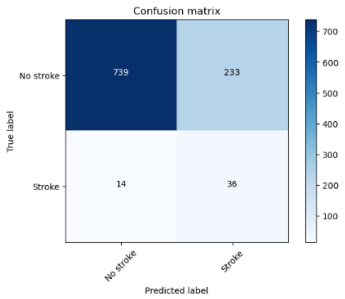
(b) Árbol de decisión



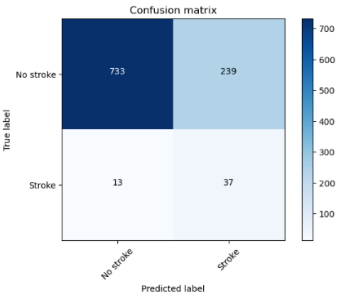
(c) Bagging árbol de decisión



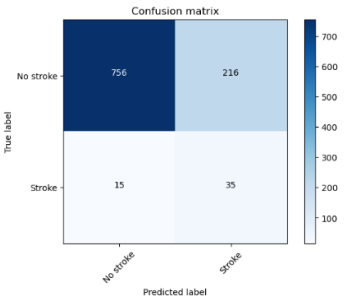
(d) Bagging regresión logística



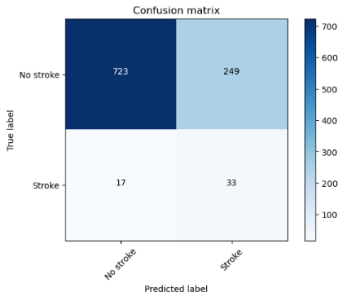
(e) Random forest



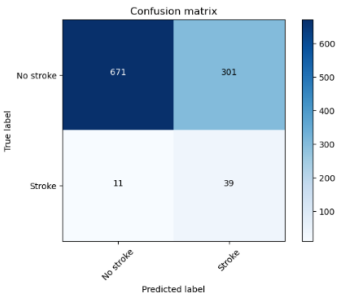
(f) Gradient boosting



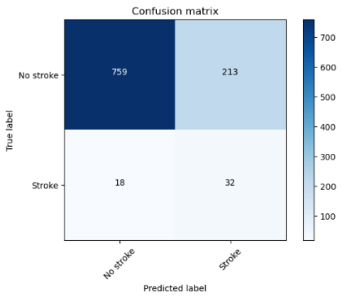
(g) XGBoosting



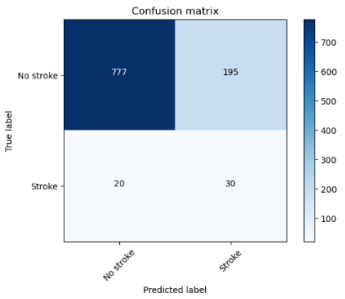
(h) SVM lineal



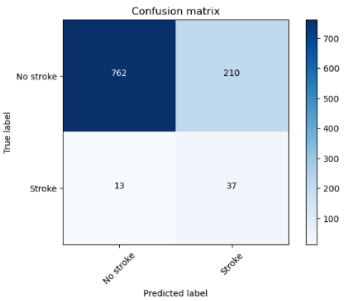
(i) SVM polinómico



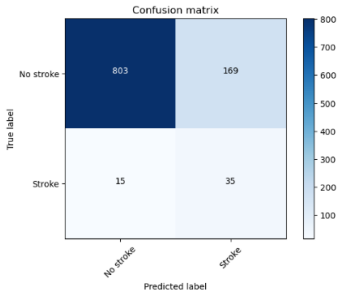
(j) SVM rbf



(k) Red neuronal



(l) Stacking



Fuente: Elaboración propia del autor

Tabla 11: Métricas modelos

Modelo	**Accuracy**	**Precisión**		**Recall**		**F1-score**		**AUC**
		0	**1**	**0**	**1**	**0**	**1**	
Regresión logística	0.76	0.98	0.13	0.76	0.72	0.86	0.23	0.741
Árbol de decisión	0.80	0.98	0.16	0.80	0.72	0.88	0.26	0.762
Bagging árbol de decisión	0.78	0.98	0.14	0.78	0.70	0.87	0.24	0.742
Bagging regresión logística	0.76	0.98	0.13	0.76	0.72	0.86	0.23	0.740
Random forest	0.75	0.98	0.13	0.75	0.74	0.85	0.23	0.747
Gradient boosting	0.77	0.98	0.14	0.78	0.70	0.87	0.23	0.738
XGBoosting	0.74	0.98	0.12	0.74	0.66	0.84	0.20	0.702
SVM lineal	0.69	0.98	0.11	0.69	0.78	0.81	0.20	0.735
SVM polinómico	0.77	0.98	0.13	0.78	0.64	0.87	0.22	0.710
SVM rbf	0.79	0.97	0.13	0.80	0.60	0.88	0.22	0.700
Red neuronal	0.78	0.98	0.15	0.78	0.74	0.87	0.25	0.762
Stacking	0.82	0.98	0.17	0.83	0.70	0.90	0.28	0.763

Las celdas en color verde indican las mejores métricas; Las celdas en color naranja indican las segunda y tercera mejores métricas, respectivamente

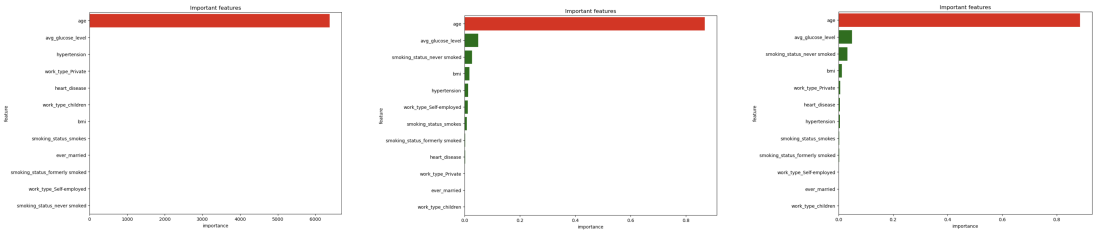
Fuente: Elaboración propia del autor

En esta sección, se presentará el rendimiento de los diferentes modelos ganadores en el conjunto de prueba. Para ello, se utilizarán las matrices de confusión correspondientes (ver Figura 37), y se calcularán diversas métricas, tales como la exactitud (*accuracy*), la precisión y el área bajo la curva ROC (ver Tabla 11). Cabe destacar que dichos indicadores resultan útiles para evaluar el desempeño de los modelos y compararlos entre sí, lo que permitirá tomar decisiones informadas acerca de cuál de ellos es el más adecuado para los fines buscados.

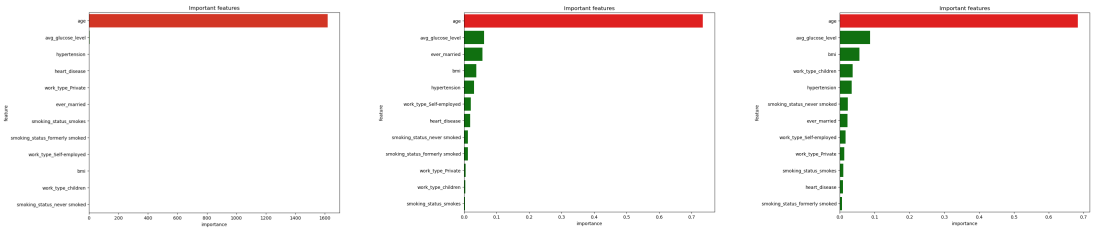
Considerando lo anteriormente expuesto, en la Tabla 11 se puede apreciar que el modelo que ofrece, en general, las mejores métricas sobre el conjunto de prueba es el ***Stacking*** (ver Sección 8.10). Sin embargo, a pesar de contar con las mejores métricas, se puede observar que la **sensibilidad (*recall* clase 1)** de dicho modelo no es la más alta, sino que existen otros modelos, tales como el **SVM lineal**, ***Random Forest*** o la **Red Neuronal**, que presentan valores más elevados para dicha métrica.

Figura 38: Importancia de las variables

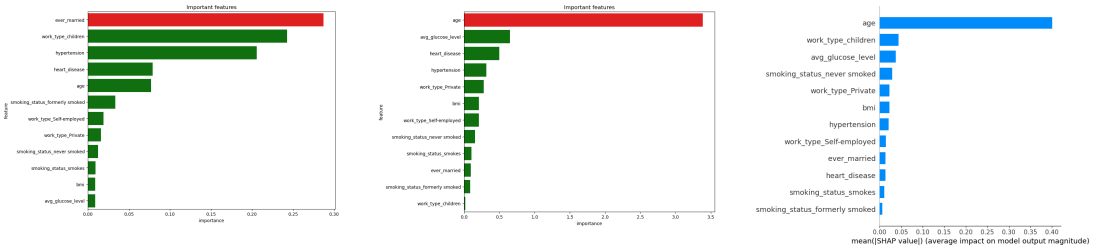
(a) Regresión logística (b) Árbol de decisión (c) Bagging árbol de decisión



(d) Bagging regresión logística (e) Random forest (f) Gradient boosting



(g) XGBoosting (h) SVM lineal (i) Red neuronal



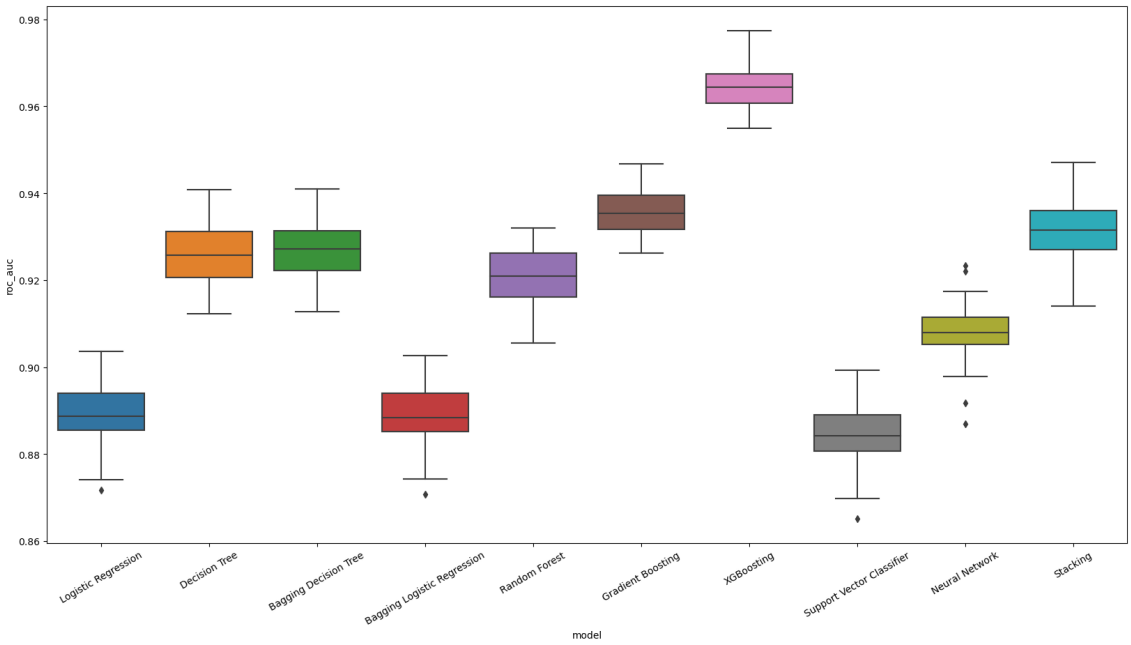
Fuente: Elaboración propia del autor

Para finalizar, al analizar la importancia de las variables en cada uno de los algoritmos testeados (ver Figura 38), se puede observar claramente que las variables *age* y *avg_glucose_level* son las que predominan en la gran mayoría de los modelos, a excepción de *XGBoosting*.

10 Selección del modelo final

Para seleccionar el modelo final, se lleva a cabo una **validación cruzada repetida**, utilizando **5 *folds*** y **10 repeticiones**, sobre **la unión del conjunto de entrenamiento y prueba**. Dicho proceso se realiza sobre todo el conjunto, ya que el objetivo principal de esta sección consiste en la elección del modelo final y no en su evaluación, algo que ya se realizó en la sección previa.

Figura 39: Validación cruzada repetida



Fuente: Elaboración propia del autor

Por lo tanto, al observar la Figura 39, se pueden descartar a simple vista los modelos de **regresión logística con o sin *bagging***, **máquina de vectores de soporte con *kernel* lineal** y la **red neuronal**, debido a que presentan valores inferiores para el área bajo la curva ROC. Además, el **árbol de decisión** parece mostrar una alta variabilidad frente al resto de modelos, por lo que también se descarta. Finalmente, entre los modelos restantes, se procede a seleccionar el ***Gradient Boosting*** como modelo final, debido a su **menor variabilidad** y por presentar una **alta área bajo la curva ROC**, lo que indica un buen rendimiento en la clasificación.

11 Anexo

Todo el código del presente informe se puede encontrar en el siguiente repositorio de **Github**

Bibliografía

[1] *Diabetes en español*. Dec. 2022. URL: https://www.cdc.gov/diabetes/spanish/basics/getting-tested.html.

[2] *Peso, nutrición y Actividad Física Saludables*. Aug. 2021. URL: https://www.cdc.gov/healthyweight/spanish/index.html.