Introducción a Inteligencia Artificial para ciencias e ingenierías

Entrega Final

Estudiantes:

Omar Alberto Torres CC 91220873
Carlos Alfredo Pinto Hernández CC 1100953378

Profesor:
Raúl Ramos Pollan

Faculta de ingeniería
Departamento de Ingeniería de Sistemas
Ude@
2023

I. Introducción

1. Problema predictivo

Dadas unas características físicas, enfermedades y resultados de laboratorios de una persona se predecirá la probabilidad de presentar enfermedad cerebro vascular o stroke.

2. Dataset elegido

Se utilizará el dataset de kaggle de la competición denominada “Playground Series - Season 3, Episode 2” disponible en el siguiente enlace: https://www.kaggle.com/competitions/playground-series-s3e2/ el cual contiene más de 15.000 muestras y divididas en 12 columna incluyendo: edad, género, índice de masa corporal, promedio de nivel de glucosa, tipo de residencia, tipo de trabajo, si ha estado casado, si presenta hipertensión o enfermedad cardíaca.

Se obtienen dos archivos uno denominado “train.csv - the training dataset”: donde se especifica el target “stroke” en formato binario y el otro documento “test.csv - the test dataset” con el cual se probará el modelo para predecir la probabilidad de presentar un stroke.

3. Métrica de desempeño

Como métrica de machine learning vamos a utilizar el área bajo la curva ROC entre la probabilidad del modelo vs el resultado (stroke); esta métrica fue definida en la competencia. Como métrica de negocio se podría plantear el incremento del número de personas que ingresan a controles de riesgo cardiovascular detectados por el modelo.

4. Criterio sobre el cual sería el desempeño deseable en producción

Se espera tener un recall (sensibilidad) mayor al 90% dado que son los valores de los modelos de prevención y tamizaje en el ámbito de salud.

II. Exploración descriptiva del Dataset

1. Variable objetivo (y)

La variable stroke es la variable objetivo del dataset y se representa de forma binaria, 0 significa que la persona no presentó stroke y 1 que si lo presentó. En el grafico 1 se muestra la frecuencia de las categorías de stroke en las muestras del dataset.

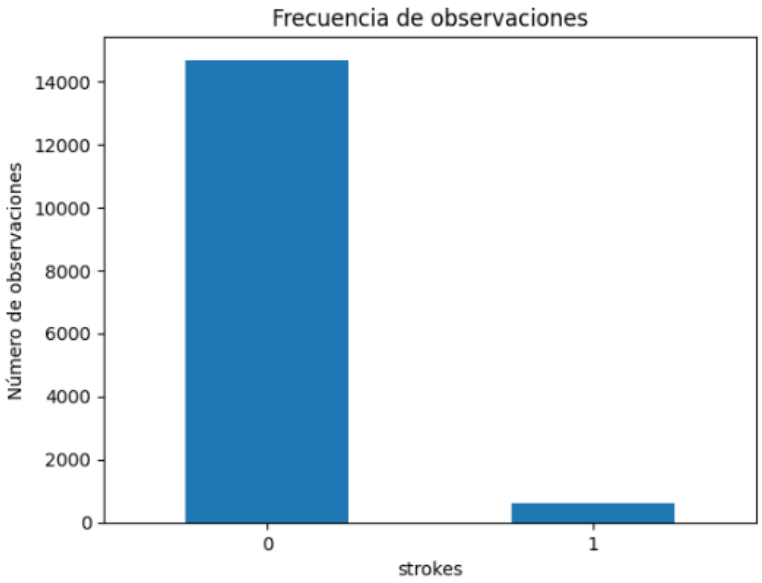


Gráfico 1*: Frecuencia de la variable stroke*

Se evidencia un dataset muy desbalanceado, dado el número mayor de la categoría 0, pero se considera que es algo habitual dado que el dominio del problema está relacionado con el área de la salud.

2. Características

Usando el método head() de pandas, se obtienen la tabla con las primeras 5 muestras con todas las características y variable de salida u objetivo. Con el método shape() se evidencia la cantidad de muestras o filas y las características/salida como columnas. En este caso el database contiene 15304 columnas y 12 columnas.

index	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	0	Male	28.0	0	0	Yes	Private	Urban	79.53	31.1	never smoked	0
1	1	Male	33.0	0	0	Yes	Private	Rural	78.44	23.9	formerly smoked	0
2	2	Female	42.0	0	0	Yes	Private	Rural	103.0	40.3	Unknown	0
3	3	Male	56.0	0	0	Yes	Private	Urban	64.87	28.8	never smoked	0
4	4	Female	24.0	0	0	No	Private	Rural	73.36	28.8	never smoked	0

Tabla 1: Visualización de database.head()

Luego al utilizar el método info() podemos visualizar el tipo de variable de cada columna y la cantidad de nulos que tiene el dataset. Para efectos académicos a esta BD se insertaron valores nulos del 5% para poder realizar manejo de estos, dado que originalmente no traían datos nulos. Se realizó la inclusión de valores nulos a las variables age y avg_glucose_level.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15304 entries, 0 to 15303
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 15304 non-null  int64
 1   gender             15304 non-null  object
 2   age                15304 non-null  float64
 3   hypertension       15304 non-null  int64
 4   heart_disease      15304 non-null  int64
 5   ever_married       15304 non-null  object
 6   work_type          15304 non-null  object
 7   Residence_type     15304 non-null  object
 8   avg_glucose_level  15304 non-null  float64
 9   bmi                15304 non-null  float64
 10  smoking_status     15304 non-null  object
 11  stroke             15304 non-null  int64
dtypes: float64(3), int64(4), object(5)
```

Gráfico 2*: Tipo de variables y datos nulos en BD original*

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15304 entries, 0 to 15303
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 15304 non-null  int64
 1   gender             15304 non-null  object
 2   age                14556 non-null  float64
 3   hypertension       15304 non-null  int64
 4   heart_disease      15304 non-null  int64
 5   ever_married       15304 non-null  object
 6   work_type          15304 non-null  object
 7   Residence_type     15304 non-null  object
 8   avg_glucose_level  14552 non-null  float64
 9   bmi                15304 non-null  float64
 10  smoking_status     15304 non-null  object
 11  stroke             15304 non-null  int64
dtypes: float64(3), int64(4), object(5)
```

Gráfico 3*: Tipo de variables y datos nulos en BD con inclusión de valores nulos*

2.1 Descripción estadística de variables numéricas

Se tomaron las variable numéricas y se realizó análisis con método describe, con el fin de obtener valores como número total, media, desviación estándar, valor mínimo, valor máximo y los percentiles 25, 50 y 75. En la tabla 2, se evidencian estos resultados para las variables age, avg_glucose_level y bmi.

index	age	avg_glucose_level	bmi
count	14556.0	14552.0	15304.0
mean	41.41943391041495	88.97298584387025	28.112720857292214
std	21.45553096417527	25.48028637456789	6.722315422227762
min	0.08	55.22	10.3
25%	25.0	74.8175	23.5
50%	43.0	85.07	27.6
75%	57.0	96.92	32.0
max	82.0	267.6	80.1

Tabla 2: Descripción estadística de variables numéricas

A través de boxplot se pueden visualizar distribución de los datos, simetría y datos atípicos de variables numéricas. En la siguiente figura se presenta un boxplot de la variable avg_glucose_level.

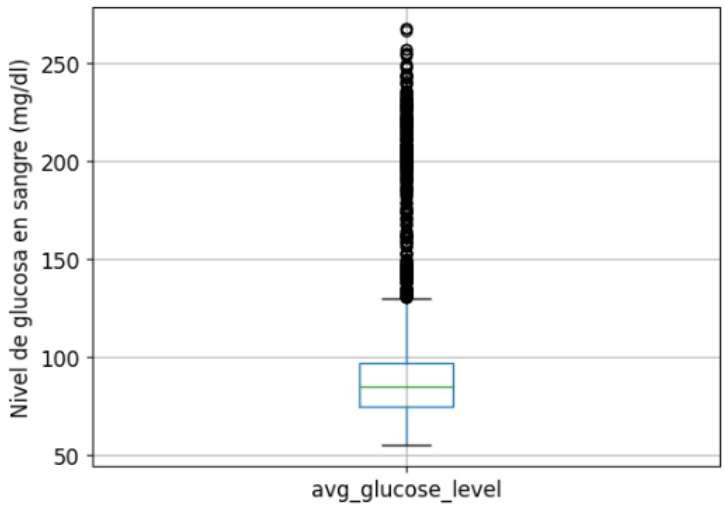


Gráfico 4*: Boxplot de distribución de la variable avg_glucose_level*

2.2 Variables categóricas

Para cuantificar las variables categóricas utilizamos el método value_counts() y con esto se pudieron realizar algunas gráficas de distribución. En el caso de la variable work_type se encontraron las frecuencias de sus 5 categorías las cuales fueron graficadas en plot tipo pie.

```
Private          9752
children         2038
Self-employed    1939
Govt_job         1533
Never_worked       42
Name: work_type, dtype: int64
```

Gráfico 5*: Frecuencias de las categorías de la variable work_type*

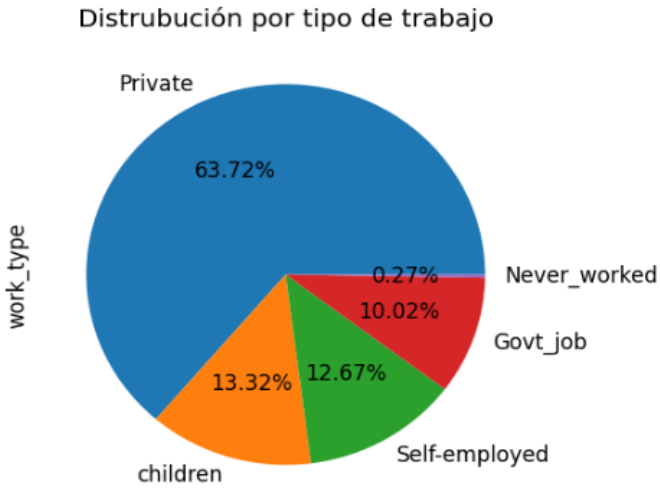


Gráfico 6*: Distribución por tipo de trabajo*

III. Desarrollo

Durante el desarrollo del proyecto se presentaron varios retos a la hora de poner en práctica lo estudiado durante los diferentes módulos de la materia y es así como la implementación de los modelos y la exploración de diferente hiperparámetros con el fin de encontrar una solución más eficiente. Se trabajaron con varios valores con el fin de entender un poco el funcionamiento del algoritmo y a su vez buscando un mejor rendimiento. Pero antes de todo este trabajo se realizó un trabajo de manejo de datos nulos que se ingresaron manualmente con el fin de realizar el manejo de estos valores para posterior hacer el proceso de modelamiento.

1. Preprocesamiento

Como parte del proceso de formación generamos 5% de nulos en dos variables (age y avg_glucose_level) en el dataset original para realizar el proceso de manejo de este tipo de valores. Los valores nulos fueron llenados con el promedio de cada una de las clases. Además, realizamos la discretización de 3 variables: age, avg_glucose_level y bmi, generando 4 grupos para age y bmi y 3 para avg_glucose_level.

```
[ ]  #Se transforma campo numerico age a un atributo con valores categoricos, se discretisa
     #=============================================================================================================================================

     df = datos.copy()

     edades = pd.cut(df['age'], bins = [0,10,20,40,np.inf], labels =['niño','joven','adulto','mayor'], include_lowest=True, retbins = True)
     df['new_age'] = list(edades[0])

     glucosa = pd.cut(df['avg_glucose_level'], bins = [0,100,125,np.inf], labels = ['normal', 'prediabetico','diabetico'], include_lowest=T
     df["avg_glucose"] = list(glucosa[0])

     im = pd.cut(df['bmi'], bins = [18.5,24.9,25,29.9,np.inf], labels = ['low_weight', 'Healthy_weight','overweight','obesity'], include_lo
     df['bmi_range'] = list(im[0])
     df.shape

     (15304, 15)
```

Finalmente se hizo la transformación de variables categóricas a variables numéricas, generando un total de 33 características incluyendo la variable de salida (Gráfico 8); al final de exportó el dataset sin variable de salida como ‘Dataset.csv’ y la variable de salida en el archivo ‘y.csv’ (Gráfico 7).

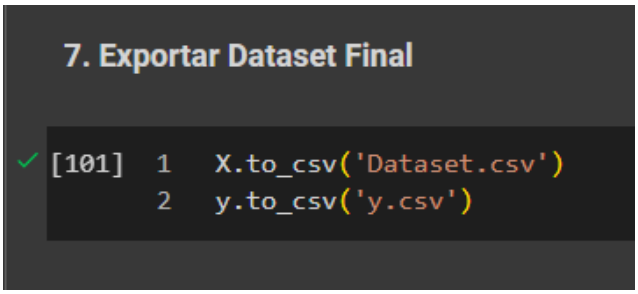


Gráfico 8*: Documentos exportados*

```
Data columns (total 34 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   new_age_adulto                  15304 non-null  uint8
 1   new_age_joven                   15304 non-null  uint8
 2   new_age_mayor                   15304 non-null  uint8
 3   new_age_niño                    15304 non-null  uint8
 4   avg_glucose_diabetico           15304 non-null  uint8
 5   avg_glucose_normal              15304 non-null  uint8
 6   avg_glucose_prediabetico        15304 non-null  uint8
 7   bmi_range_Healthy_weight        15304 non-null  uint8
 8   bmi_range_low_weight            15304 non-null  uint8
 9   bmi_range_obesity               15304 non-null  uint8
 10  bmi_range_overweight            15304 non-null  uint8
 11  gender_Female                   15304 non-null  uint8
 12  gender_Male                     15304 non-null  uint8
 13  gender_Other                    15304 non-null  uint8
 14  ever_married_No                 15304 non-null  uint8
 15  ever_married_Yes                15304 non-null  uint8
 16  work_type_Govt_job              15304 non-null  uint8
 17  work_type_Never_worked          15304 non-null  uint8
 18  work_type_Private               15304 non-null  uint8
 19  work_type_Self-employed         15304 non-null  uint8
 20  work_type_children              15304 non-null  uint8
 21  Residence_type_Rural            15304 non-null  uint8
 22  Residence_type_Urban            15304 non-null  uint8
 23  smoking_status_Unknown          15304 non-null  uint8
 24  smoking_status_formerly smoked  15304 non-null  uint8
 25  smoking_status_never smoked     15304 non-null  uint8
 26  smoking_status_smokes           15304 non-null  uint8
 27  bmi_range_Healthy_weight        15304 non-null  uint8
 28  bmi_range_low_weight            15304 non-null  uint8
 29  bmi_range_obesity               15304 non-null  uint8
 30  bmi_range_overweight            15304 non-null  uint8
 31  hypertension                    15304 non-null  int64
 32  heart_disease                   15304 non-null  int64
 33  stroke                          15304 non-null  int64
dtypes: int64(3), uint8(31)
```

Gráfico 7*: Variables del Dataset Final*

2. Modelos

Para cada uno de los modelos implementados cargamos el dataset exportado de la etapa de preprocesamiento de datos incluye el archivo de variable de salida, por lo tanto, los modelos evaluados son modelos supervisados (máquina de soporte vectorial, random forest y árbol de decisión).

Previo a cada uno de estos algoritmos realizamos la aplicación del método de reducción de dimensionalidad (Principal Component Analysis PCA) el cual permite simplificar la complejidad de espacios con múltiples dimensiones a la vez que conserva la información. Para esto utilizamos el método PCA importado de sklearn.decomposition; Dada su naturaleza, decidimos generar valores sintéticos para intentar corregir el problema de desbalance.

2. Generador de muestras sintéticas

```
[14]  1  pca=PCA(n_components=30)
      2  pca.fit(X) #
      3  X_pca=pca.transform(X)
```

Gráfico 9*: Generador de muestras sintéticas*

Además, realizamos un random over sample previo a la distribución de los datos en train y test (90/10).

```
ros = RandomOverSampler(random_state=42, sampling_strategy=1.0)
Xres, yres = ros.fit_resample(X_pca, y)
Xtrain, Xtst, ytrain, ytst = train_test_split(Xres,yres, test_size=0.1)
print (Xtrain.shape, ytrain.shape, Xtst.shape, ytst.shape)
X=Xtrain
y=ytrain
```

```
(26409, 30) (26409, 1) (2935, 30) (2935, 1)
```

Gráfico 10*: RandomOverSampler método*

Se realizó asignación de pesos a las clases de modelo dado que Los modelos con clases desbalanceadas afectan a los algoritmos en su tarea de generalización, en perjuicio de la clase minoritaria, esto provoca que el modelo tenga un bajo performance, para solventar este problema, se calculó los pesos de tal forma que se le da una mayor importancia a la clase minoritaria.

3. Calculo pesos de la clase Parametro que se usa para minimizar el efecto del desbalance del dataset

```
[19]
def dar_peso_clase():
  Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size = 0.2, train_size = 0.8, random_state = 17)
  Number_Muestras_por_Clase = ytrain.value_counts()
  Number_Muestras_por_Clase = Number_Muestras_por_Clase.sort_index(axis = 0, ascending = True)
  Numero_de_Clases = len(Number_Muestras_por_Clase.index)
  Peso_por_Clase = len(ytrain)/(Numero_de_Clases*Number_Muestras_por_Clase)
  return Peso_por_Clase
```

Gráfico 11: Función para dar pesos a las clases

Para elegir el modelo, seleccionamos tres algoritmos, cuyas bondades son bastante reconocidas para el desarrollo de un modelo de predicción, como son los algoritmos de árboles de decisión, Máquina de soporte vectorial, y el algoritmo de Random Forest, nuestro propósito es entrenar los modelos y luego de tener los resultados, poder comparar las métricas que hemos establecido (Matriz de confusión y curva ROC) de tal forma que podamos valorar que, algoritmo es el que mejores resultados nos ofrece para la solución del problema.

Como métricas usaremos es la matriz de confusión y la curva ROC, dado que son herramientas que ayuda a medir la calidad del modelo: a través de esta información nos permite conocer la precisión, exactitud del modelo, saber si el modelo hace predicciones ingenuas por que se ha aprendido los datos, entre otras bondades.

La matriz de confusión junto con la curva ROC son herramientas estadísticas que nos ayudan en el proceso de valorar los resultados de un modelo. Estas se obtienen haciendo uso de las clases roc_auc_score, roc_curve, auc, f1_score y confusion_matrix de la librería sklearn.metrics.

2.1 Máquina de soporte vectorial

Se plantearon diferente hiperparámetros para mejorar el performance del algoritmo a continuación se presenta el último código de este modelo.

Código SVM

```python
kf = KFold(n_splits=5, random_state = True,shuffle=True)
c=1
weight= dar_peso_clase()
X_test =0
y_test =0
#degree =3 los resultados fueron regulares con el kernel ="poly" , no calcula area AUC

svclassifier = SVC(C=c,kernel="rbf", degree=3, gamma='auto',shrinking=True,tol=0.00010,
                   class_weight = {0: weight[0], 1: weight[1]},
                   decision_function_shape='ovr')

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    modelo = svclassifier.fit(X_train, y_train)
    y_pred = modelo.predict(X_test)
    print()

    print(confusion_matrix(y_test, y_pred))
    print(classification_report(y_test, y_pred, digits =6, labels=[0,1],zero_division=1))
    s = roc_auc_score(y_test, modelo.predict(X_test))
    print(f'Best val auc: {s: .4f}')
    print()
```

Siendo para este modelo los mejores valores una accuracy 0.728461 con la siguiente matriz de confusión y variables asociadas:

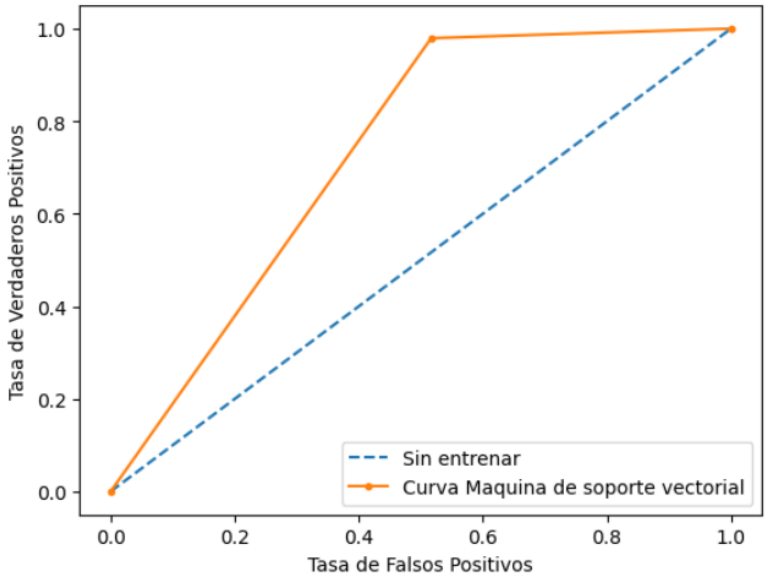
```
[[1291 1380]
 [  54 2556]]
              precision    recall  f1-score   support

           0   0.959851  0.483340  0.642928      2671
           1   0.649390  0.979310  0.780935      2610

    accuracy                       0.728461      5281
   macro avg   0.804621  0.731325  0.711932      5281
weighted avg   0.806414  0.728461  0.711135      5281

Best val auc:  0.7313
```

Y su curva ROC AUC = 0.731 con un Average_presicion_score 0.49.



2.2 Random Forest

Al igual que en el modelo anterior se realizó modelamiento con diferente hiperparámetros, el código final es el siguiente:

Código RF

```
weight = dar_peso_clase()
kf = KFokf = KFold(n_splits=10, random_state = True,shuffle=True)

X_test =0
Y_test =0

#warm_start mejoro la velocidad de ejecucion, mejoró la metrica
classifier = RandomForestClassifier(min_samples_split=10,
                                    ccp_alpha=0.000001,
                                    criterion='gini',
                                    class_weight = {0: weight[0], 1: weight[1]},
                                    random_state=True,max_features ="sqrt",
                                    warm_start=True,min_impurity_decrease=0.000005)

for train_index, test_index in kf.split(X):

    Xtrain, Xtest = X[train_index], X[test_index]
    ytrain, ytest = y.iloc[train_index], y.iloc[test_index]
    modelo = classifier.fit(Xtrain, ytrain)
    ypred = modelo.predict(Xtest)

    print()
    print(confusion_matrix(ytest, ypred))
    print(classification_report(ytest, ypred, digits =4, labels=[0,1]))
    s = roc_auc_score(ytest, modelo.predict(Xtest))
    print(f'Best val auc: {s: .4f}')
```

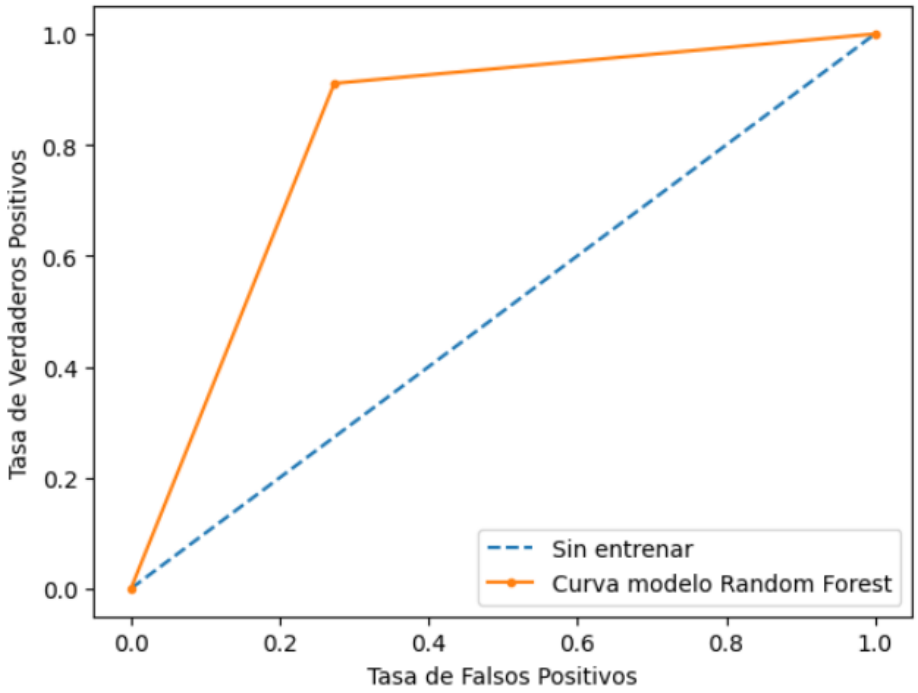
El mejor valor en el modelo de random fores fue una accuracy 0.8254 con la siguiente matriz de confusión y variables asociadas y la curva ROC-AUC:

```
[[ 985  351]
 [ 110 1195]]
              precision    recall  f1-score   support

           0     0.8995    0.7373    0.8104      1336
           1     0.7730    0.9157    0.8383      1305

    accuracy                         0.8254      2641
   macro avg     0.8363    0.8265    0.8243      2641
weighted avg     0.8370    0.8254    0.8242      2641

Best val auc:  0.8265
```



La curva ROC AUC fue del 0.819.

2.3 Árbol de decisión

Código DT

```
kf = KFold(n_splits=20, random_state = True,shuffle=True)

preds = []
scores = []

weight= dar_peso_clase()

classifier = DecisionTreeClassifier(criterion='gini',
                                    ccp_alpha=0.0000001,
                                    min_samples_split=10,
                                    class_weight= {0: weight[0], 1: weight[1]},
                                    random_state=42,
                                    max_features="sqrt",min_impurity_decrease=0.0
                                    )  #"log2"

X_test =0
y_test =0
for train_index, test_index in kf.split(X):

    X_train, X_test = X[train_index], X[test_index]  #linea de código para pca
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    #modelo = LassoCV(**model_params)
    modelo = classifier.fit(X_train, y_train)
    y_pred = modelo.predict(X_test)

    print()
    print(confusion_matrix(y_test, y_pred))
    print(classification_report(y_test, y_pred, digits =4, labels=[0,1]))
    preds.append(modelo.predict(X_test))
    s = roc_auc_score(y_test, modelo.predict(X_test))
    print(f'Best val auc: {s: .4f}')
    scores.append(s)
    print()
```

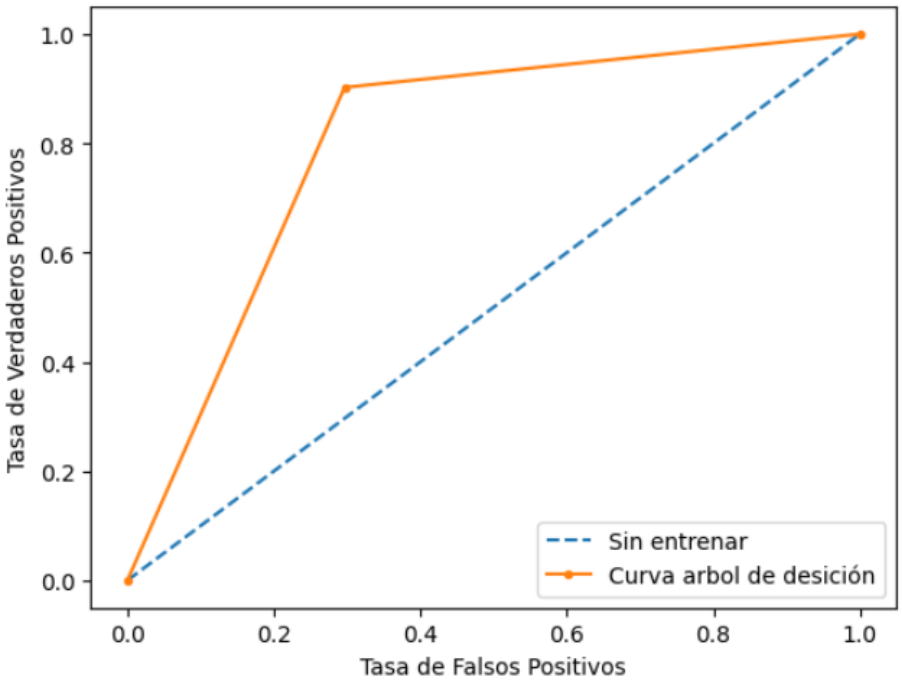
En cuanto a la matriz de confusión el mejor modelo con accuracy fue de 0.8256 y su ROC AUC fue de 0.803.

```
[[500 165]
 [ 66 589]]
              precision    recall  f1-score   support

           0     0.8834    0.7519    0.8123       665
           1     0.7812    0.8992    0.8361       655

    accuracy                         0.8250      1320
   macro avg     0.8323    0.8256    0.8242      1320
weighted avg     0.8327    0.8250    0.8241      1320

Best val auc:  0.8256
```



Es así como al final se demostró un mejor rendimiento por parte de los algoritmos de random forest y árbol de decisión sobre el modelo de máquina de soporte vectorial con una diferencia de más del 10%.

IV. Retos y consideraciones de despliegue

Teniendo en cuenta que se pretendía diseñar una herramienta de apoyo diagnóstico, para la detección temprana de un accidente cerebrovascular o stroke, el modelo tiene un rendimiento no muy eficiente dado que en el contexto de prevención es ideal tener valores de detección o f1 score por encima de 0.95; por lo tanto consideramos que el modelo debe ser mejorado, entrenándolo con una mayor información clínica de los pacientes, pudiéndose aumentar el número de muestras o incluir otras variables que han sido relacionadas con alto impacto en enfermedad cerebrovasculares como lo son presencia o no de: diabetes, colesterol total, colesterol lipoproteínas de baja densidad (LDL), niveles de colesterol HDL, nivel de triglicéridos, colesterol no HDL, Proteína C, lipo proteína (a) entre otros.

Si el modelo se puede entrenar con mejor información y obtener resultados superiores al 95% en el rendimiento, permitirá predecir correctamente si una persona puede sufrir un ataque cerebrovascular o no, de tal forma que el modelo pueda ser llevado a producción para su despliegue.

Para los profesionales de la salud podría servirse de esta herramienta, instalada en su computadora personal, en su teléfono móvil. La aplicación debería permitir digitar la información médica del paciente y en tiempo real entregar los resultados.

V. Conclusiones

1. A través de este ejercicio académico pudimos integrar la teoría con la práctica y así entender la forma en que funcionan los algoritmos de predicción en modelos supervisados.

2. Durante el entrenamiento del modelo tuvimos problemas técnicos que pudimos superar parcialmente, como fue el hecho de que el modelo se comportaba tontamente, al predecir la clase con mayor presencia en el dataset “0”, pero era pésimo para predecir la clase “1”, esto debido a que el dataset era extremadamente desbalanceado. Para esto tuvimos que plantear varias opciones y utilizar otros métodos para contrarrestar este desbalanceo. Se lograron resultados aceptables, otorgándoles un mayor peso a la clase minoritaria, y creando muestras sintéticas.

3. Trabajamos en encontrar los mejores hiperparámetros para lograr un modelo más confiable, pero esto requiere de un conocimiento más profundo de la forma en que funcionan los algoritmos predictivos, hecho que pudiera ser motivo de estudio en próximos cursos o aprendizaje autónomo; Aunque en el proyecto hicimos varios ejemplos de ensayo-error para definir los mejores valores de los hiperparámetros, usando como guía la información técnica de la librería de Sklearn. De esta manera, modelos que nos entregaban una línea recta sin área para la gráfica ROC, tuvieron un comportamiento aceptable para el ejercicio académico.

4. Se puede aprovechar el potencial de las librerías de Sklearn para encontrar los mejores hiperparámetros, de hecho, en este trabajo, se calculó mediante un algoritmo sencillo los pesos de las clases, y se calculó algunos hiperparámetros del modelo SVC (kernel, gamma, C, tol)

5. Esta área de conocimiento nos presenta múltiples herramientas que podemos utilizar para procesar información de diversas fuentes y que en la vida profesional harán parte de nuestro desarrollo profesional, por lo tanto, es indispensable aprender a usarla y poder sacar el mejor uso a la información que se tiene; recordando que el poder que tiene la data en la actualidad.