Predicción de accidente cerebrovascular en personas usando Machine Learning

Pablo Gonzalez Baron
Ciencias de la computación
Universidad Nacional De Colombia
Bogotá, Colombia
pgonzalezb@unal.edu.co

Pablo Andres Dorado
Ciencias de la computación
Universidad Nacional De Colombia
Bogotá, Colombia
padorados@unal.edu.co

Daniel Ricardo Rodriguez
Ciencias de la computación
Universidad Nacional De Colombia
Bogotá, Colombia
drodriguezol@unal.edu.co

Profesor. Francisco Gomez
Departamento de Matemáticas
Universidad Nacional De Colombia
Bogotá, Colombia
fagomezj@unal.edu.co

Keywords—Accidente cerebrovascular, Machine learning, Problema de clasificación, Aprendizaje supervisado, Arboles de decisión, Modelo, Predicción.

I. INTRODUCCIÓN

Los accidentes cerebrovasculares son comunes y prevenibles. 1 de cada 20 muertes de adultos se debe a un accidente cerebrovascular. Muchas de estas muertes son prevenibles. Cuando se produce un accidente cerebrovascular, partes del cerebro se dañan y pueden comenzar a morir en minutos, es por eso que buscamos implementar un algoritmo para predecir la probabilidad de que estos accidentes ocurran con técnicas y conceptos de inteligencia artificial y aprendizaje de máquina, en complemento a esto se utilizaran conceptos matemáticos que nos serán de ayuda para resolver el problema planteado y mostrar resultados.

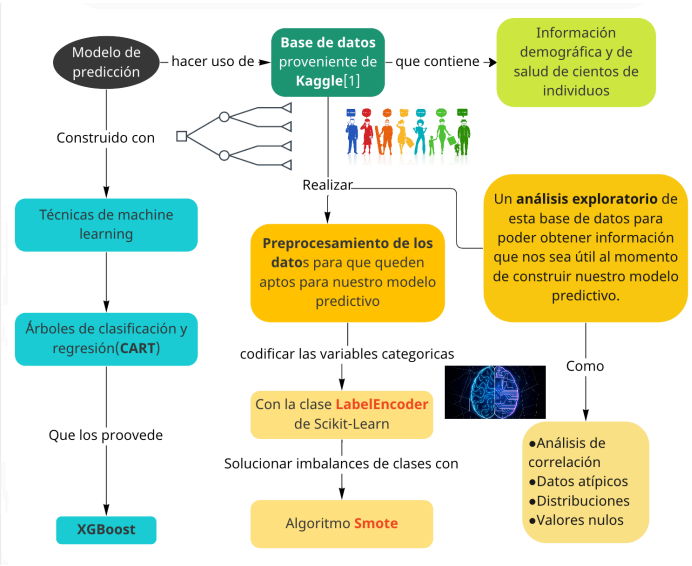
Problemas específicos:

- Realizar un análisis exploratorio de la base de datos con la que vamos a trabajar
- Analizar cada variable de nuestra base de datos por separado para ganar visión de la distribución de cada una
- Aplicar técnicas de preprocesamiento sobre los datos para volverlos aptos para el modelo de Machine Learning
- Ajustar un modelo base llamado XGBoost que se compone de arboles de decisión como predicadores débiles para combinarlos y formar un predictor mas robusto
- Realizar tuning de hiperparametros para obtener un modelo con mejor rendimiento

II. MATERIALES Y MÉTODOS

A. *Solución computacional propuesta*

El siguiente diagrama de bloques muestra la solución general propuesta para resolver el problema de modelo de predicción de accidentes cerebrovasculares.



B. *Atributos de la base de datos:*

1. Id. Identificador único
2. gender (Género): Male(Masculino), Female(Femenino) y Other(Otro)
3. age (Edad): edad del individuo.
4. hypertension (Hipertensión): 0 si el individuo no tiene hipertensión, 1 si el individuo tiene hipertensión.
5. heart-disease (enfermedad del corazón): 0 si el individuo no tiene ninguna enfermedad del corazón, 1 si el individuo tiene una enfermedad del corazón
6. ever-married (Casado): No (No) ó Yes (Si)
7. work-type(Tipo de trabajo):children(Niño), Govt-jov(Gobierno),Never-worked (Nunca ha trabajado), Private (Privado) ó Self-employed (Independiente).
8. Residence-type (Tipo de residencia): Rural o Urban(Urbana).
9. avg-glucose-level: nivel promedio de glucosa en sangre.
10. BMI: índice de masa corporal.
11. smoking-status(Fumador):formerly-smoked(Antes fumaba),never smoked(Nunca fumó), smokes(Fuma actualmente) o Unknown(NA).
12. stroke: 1 si el individuo ha tenido un accidente cerebrovascular ó 0 en caso contrario (esta será nuestra variable objetivo)

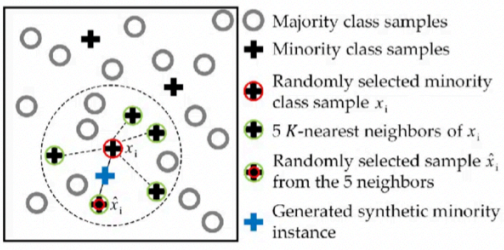
C. *Análisis exploratorio de datos (análisis de correlación, datos atípicos, distribuciones, valores nulos):*

Al momento de realizar el análisis exploratorio de datos, lo primero que hacemos es un análisis univariado de cada atributo, para esto nuestra principal herramienta matemática son las estadísticas de resumen de los atributos, en particular, la media, mediana, desviación estándar y los percentiles. Estas nos dan una noción de la distribución de las variables. Sin embargo, pueden ser difíciles de interpretar, por esta razón también usamos distintos tipos de visualizaciones, como gráficos de barras para visualizar las variables categóricas, histogramas para interpretar la distribución de variables continuas y matriz de dispersión para visualizar las posibles relaciones pares de variables.

Otro aspecto importante para nuestro análisis es la identificación de valores atípicos, para esto usamos el rango intercuartil y también el diagrama de caja y bigotes.
Luego realizamos un análisis de valores nulos, en el cual primero contamos el número de valores nulos por atributo y después lo graficamos en lo que se conoce como una matriz de valores nulos. Después de identificar los atributos con valores nulos, presentamos algunas posibles razones para este fenómeno y sugerimos una estrategia para imputar estos datos.
Mas adelante graficamos múltiples diagramas de dispersión por par de variables, esto para poder visualizar alguna relación entre variables y del cual no podemos apreciar mucha relación entre estas.
Por último, realizamos un mapa de calor de la matriz de correlación para encontrar alguna relación entre las variables y encontramos que todos valores de la matriz se encuentran aproximadamente entre 0 y 0,4, a partir de esto podemos concluir que no existe ninguna correlación apreciable entre las variables

D. *Preprocesamiento de variables*

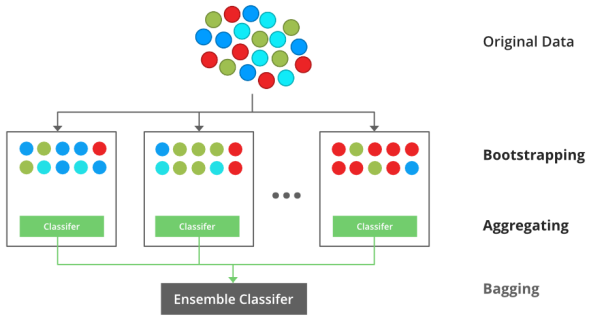
El primer paso en el preprocesamiento fue codificar las variables categóricas utilizando la clase LabelEncoder de Scikit-Learn, esto se realiza ya que él modelo que vamos a usar solo acepta como entrada valores numéricos. Luego, como en el análisis exploratorio de datos notamos un problema de imbalance de clases, donde la clase 0 (pacientes que no han tenido un accidente cerebrovascular) representaba un 95 % de los datos y la clase 1 (pacientes que han tenido un accidente cerebrovascular) el 5 % restante, lo solucionamos con un método llamado oversampling utilizando un algoritmo llamado SMOTE (pacientes que han tenido un accidente cerebrovascular), en el cual se generan datos sintéticos de la clase minoría.



Representación gráfica del algoritmo Smote.

E. *Entrenamiento y validación de modelo de clasificación*

Para el modelo de clasificación se usa un modelo ensamblado por árboles de decisión llamado XGBoost.



A diferencia de otros modelos similares como por ejemplo un RandomForest, XGBoost es un modelo que durante el entrenamiento implementa un algoritmo llamado Boosting que combina un conjunto de predictores débiles (en este caso arboles de decision) para generar un predictor mas fuerte y minimizar los errores en el entrenamiento, reduciendo así la varianza y el sesgo del modelo. Además el modelo XGBoost es util para datos con imbalance de clases como en este caso.

Los árboles de decisión individuales son modelos de bajo sesgo y alta varianza. Son increíblemente buenos a la hora de encontrar las relaciones en cualquier tipo de datos de entrenamiento, pero les cuesta generalizar bien en datos no vistos.

Para hallar los mejores hiperparametros para nuestro clasificador XGBoost utilizaremos la clase GridSearchCV de Scikit-learn. El *grid search* en Machine Learning es un método de búsqueda que toma en cuenta diferentes combinaciones de hiperparámetros y elige la combinación que arroja un margen de error más bajo. Los hiperparámetros a modificar son:

1. **learning_rate**: También llamado eta, especifica la rapidez con la que el modelo se ajusta a los errores residuales utilizando aprendices de base adicionales.
   - Valores típicos: 0.01-0.2
2. **gamma, reg_alpha, reg_lambda**: Estos 3 parámetros especifican los valores para 3 tipos de regularización realizados por XGBoost - reducción de pérdidas mínima para crear una nueva división, L1 reg en pesos de hoja, L2 reg pesos de hoja respectivamente.
   - Valores típicos de gamma: 0 - 0,5, pero dependen en gran medida de los datos.
   - Valores típicos para reg_alpha y reg_lambda: 0 - 1 es un buen punto de partida pero, de nuevo, depende de los datos.
3. **max_depth**: La profundidad de los nodos de decisión del árbol. Debe ser un número entero positivo.
   - Valores típicos: 1-10.

4. **subsample**: Fracción del conjunto de entrenamiento que puede utilizarse para entrenar cada árbol. Si este valor es bajo, puede llevar a un infra-ajuste o si es demasiado alto, puede llevar a un sobreajuste.
   - Valores típicos: 0.5-0.9
5. **colsample_bytree**: Fracción de las características que pueden utilizarse para entrenar cada árbol. Un valor grande significa que se pueden utilizar casi todas las características para construir el árbol de decisión
   - Valores típicos: 0.5-0.9

III. Resultados

A continuación mostramos los resultados parciales que van desde el análisis exploratorio hasta el desarrollo del modelo y la generación de predicciones, estos resultados se presentan de forma interactiva en el formato de Jupyter Notebook, por esto hemos decidido subir el notebook a la plataforma Deepnote, en la cual puede reproducir el proyecto desde el navegador. En el siguiente link puede acceder al notebook: Notebook

En adición se creo una aplicación web que toma el input del usuario y realiza una predicción utilizando el modelo ya entrenado: https://predict-stroke-agent-un2027631.netlify.app/.

IV. Discusión y conclusiones

Los modelos ensamblados de Machine Learning como XGBoost o GradientBoosting han demostrado ser muy buenos y robustos para tareas tanto de clasificación o regresión. En este caso nos enfrentamos a un problema de clasificación, y después de obtener una precisión del 95%, realizamos tuning de hiperparámetros lo cual nos permitió aumentar nuestra precisión a un 97.7%. Si hubiésemos atacado el problema con un modelo de Machine Learning mas sencillo como una regresión logística hubiésemos tenido que tener en cuenta problemas como estandarización de variables, multicolinealidad o incluso no hubiésemos podido atacar el problema de imbalance de clases, pues este modelo no es tan robusto como XGBoost ya que no aplica ningún algoritmo de boosting y es muy sensible a la distribución de las variables numéricas.

V. Referencias

1. Kaggle.com, Stroke Prediction Dataset, https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset
2. Towardsdatascience.com, Beginner's Guide to XGBoost, https://towardsdatascience.com/beginners-guide-to-xgboost-for-classification-problems-50f75aac5390
3. Xgboost.readthedocs.io, Decision Tree Ensembles, https://xgboost.readthedocs.io/en/stable/tutorials/model.html#decision-tree-ensembles.
4. Machinelearningmastery.com, Extreme Gradient Boosting (XGBoost) Ensemble in Python, https://machinelearningmastery.com/extreme-gradient-boosting-ensemble-in-python/
5. Edureka.co, How To Implement Classification In Machine Learning?, https://www.edureka.co/blog/classification-in-machine-learning/#:~:text=In%20machine%20learning%2C%20classification%20is,recognition%2C%20document%20classification%2C%20etc.
6. Keepcoding.io, Grid search en Python, https://keepcoding.io/blog/grid-search-en-python/