ANÁLISIS Y PREDICCIÓN DE DIABETES MEDIANTE MODELOS DE MACHINE LEARNING

RASSHID ORTIZ RODRÍGUEZ [a], BULMARO JUÁREZ HERNÁNDEZ [a]

[a]Facultad de Ciencias F ısico Matematicas Benemérita Universidad Autónoma de Puebla Avenida San Claudio y 18 Sur, Colonia San Manuel, Ciudad Universitaria, C.P. 72570, Puebla, Pue.
e-mail: rasshid.ortiz@alumno.buap.mx,bjuarez@fcfm.buap.mx

Abstract. This study presents a predictive analysis based on the BRFSS 2015 dataset, using machine learning techniques to detect the presence of diabetes. A balanced set of clinical data is used to reduce bias and improve the model's generalization. The process includes several key stages, such as data loading and cleaning, where missing values are handled and variables are normalized to ensure better training quality. Subsequently, a visual exploration and feature selection is performed to identify the most influential variables in the prediction. For modeling, algorithms such as logistic regression and random forest are used to analyze the relationship between risk factors and the disease. These models are evaluated using performance metrics, such as accuracy, to determine which offers the best predictive capability. Finally, the benefits of analysis are highlighted, and possible improvements are suggested to enhance the effectiveness and robustness of the predictive system for future applications.

Resumen. Este trabajo presenta un análisis de predicción basado en el conjunto de datos BRFSS 2015, utilizando técnicas de machine learning para detectar la presencia de diabetes. Se emplea un conjunto balanceado de datos clínicos con el objetivo de reducir sesgos y mejorar la generalización del modelo. El proceso incluye varias etapas fundamentales, como la carga y limpieza de datos, donde se manejan valores faltantes y se normalizan variables para garantizar una mejor calidad del entrenamiento. Posteriormente, se realiza una exploración visual y selección de características, identificando aquellas variables con mayor influencia en la predicción. Para el modelado, se utilizan algoritmos como regresión logística y bosques aleatorios, que permiten analizar la relación entre factores de riesgo y la enfermedad. Estos modelos son evaluados mediante métricas de desempeño, como la exactitud, con el fin de determinar cuál ofrece la mejor capacidad predictiva. Finalmente, se destacan los beneficios del análisis y se sugieren posibles mejoras para incrementar la efectividad y robustez del sistema predictivo en aplicaciones futuras.

Keywords: Predicción de diabetes, Machine learning, BRFSS 2015, Regresión logística, Bosques aleatorios.

Palabras clave: Diabetes prediction, Machine learning, BRFSS 2015, Logistic regression, Random forest.

1. Introducción

La **diabetes** es una de las enfermedades crónicas más prevalentes y preocupantes en los Estados Unidos y en el mundo, afectando a millones de personas y representando una carga significativa tanto para el sistema de salud como para la economía del país. Esta condición se caracteriza por la incapacidad del organismo para regular eficazmente los niveles de glucosa en sangre, lo que puede derivar en complicaciones graves como enfermedades cardiovasculares, daño renal, neuropatías, pérdida de visión e incluso amputaciones de extremidades. Su impacto no solo se limita a la salud del individuo, sino que también genera costos elevados en tratamientos médicos y reduce la calidad de vida de quienes la padecen.

Según los datos proporcionados por los **Centros para el Control y la Prevención de Enfermedades** (CDC), en 2018, aproximadamente **34.2 millones de estadounidenses tenían diabetes**, mientras que cerca de **88 millones** presentaban prediabetes, una condición en la que los niveles de glucosa son elevados, pero aún no alcanzan el umbral diagnóstico de la enfermedad. Lo más alarmante es que un **20 %** de las personas con diabetes y hasta un **80 %** de los prediabéticos desconocían su estado de salud, lo que retrasa el acceso a tratamientos adecuados y aumenta el riesgo de desarrollar complicaciones a largo plazo.

Ante este panorama, la detección temprana de

la diabetes y la identificación de individuos con alto riesgo de desarrollarla son aspectos fundamentales para mejorar los resultados clínicos y reducir el impacto de la enfermedad. En este contexto, los modelos predictivos basados en *machine learning* han emergido como una herramienta poderosa para fortalecer el diagnóstico y la prevención. Algoritmos como la **regresión logística** y los **bosques aleatorios** han demostrado su eficacia en la clasificación de pacientes en riesgo, permitiendo la implementación de intervenciones oportunas. Estos enfoques no solo optimizan la detección temprana de la enfermedad, sino que también contribuyen a la toma de decisiones médicas informadas, promoviendo estrategias preventivas que pueden mejorar la calidad de vida de los pacientes y reducir la carga económica sobre los sistemas de salud.

Uno de los principales desafíos al abordar esta tarea es la correcta implementación de técnicas de *machine learning*, que incluyen el procesamiento, la limpieza y la visualización de los datos. En este trabajo, nos enfocaremos en los aspectos esenciales de estas etapas, sin perder de vista nuestro objetivo principal: analizar y evaluar dos de los modelos más populares para realizar una predicción eficaz de la enfermedad usando nuestros conocimientos en el área.

2. Sobre el conjunto de datos

Para este proyecto se utilizó el conjunto de datos **BRFSS** (Behavioral Risk Factor Surveillance System) correspondiente al año 2015, disponible en la plataforma **Kaggle**. El BRFSS es una encuesta telefónica anual llevada a cabo por los CDC desde 1984, que recopila información sobre factores de riesgo relacionados con la salud, condiciones crónicas y uso de servicios preventivos.

El conjunto de datos original contiene:

- **441,455** individuos encuestados.
- **330** características que incluyen preguntas directas a los participantes y variables calculadas a partir de sus respuestas.

Aquí vamos a destacar algunos puntos, en primera el conjunto de datos ha sido previamente limpiado (tarea que más adelante verificaremos); en segundo, el archivo final contiene **70,692** datos con **21** características. Esto porque el conjunto de datos original tiene demasiado sesgo en la variable predicha teniendo demasiadas muestras de personas no diabéticas, lo que generaría impresición a la hora de comparar los modelos, dicho esto el conjunto de datos que utilizaremos está balanceado, esto es, 50 % de los datos son de un clase y 50 % de la otra clase; también nos aseguramos de tener las características más indispensables para el modelo las cuales son las siguientes:

- **Blood pressure (high):** 0 = Presión arterial No alta; 1 = Presión arterial alta.
- **Cholesterol (high):** 0 = Colesterol no alto; 1 = Colesterol alto.
- **Cholesterol check:** 0 = No se ha checado el colesterol en 5 años; 1 = sí lo ha hecho.
- **Smoking:** Responde a la pregunta "¿has fumado 100 cigarrillos en toda tu vida?"0 = No; 1 = Sí.
- **Diabetes:** Variable a predecir. 0 = No diabetes; 1 = diabetes o prediabetes.
- **Stroke:** La pregunta es si ha sufrido un derrame cerebral. 0 = No; 1 = Sí.
- **Hearth disease or attack:** 0 = No; 1 = Sí.
- **Age:** Categoría de edad de 13 niveles (intervalos).
- **Sex:** 0 = Femenino; 1 = Masculino.
- **Fruits:** Consumo de fruta 1 o más veces por día. 0 = No; 1 = Sí.
- **Veggies:** Consumo de vegetales 1 o más veces por día. 0 = No; 1 = Sí.
- **Exercise:** Actividad física en los últimos 30 días. 0 = No; 1 = Sí.
- **Alcohol consumption:** En hombres adultos más de 14 bebidas a la semana y en mujeres adultas más de 7 por semana. 0 = No; 1 = Sí.
- **BMI:** Índice de masa corporal.
- **Income:** Escala de ingresos (escala 1-8) 1 = menos de $10,000; 5 = menos de $35,000; 8 = $75,000 o más.
- **Education:** Nivel de educación, (escala 1-6). 1 = Nunca asistió a la escuela o solo al jardín de infantes; 2 = Primaria, etc.
- **Health care coverage:** Tiene algún tipo de cobertura de atención médica, incluido seguro médico, planes prepagos como HMO, etc. 0 = No; 1 = Sí.
- **Med Cost:** ¿Hubo alguna ocasión en los últimos 12 meses en la que necesitó ver a un médico pero no pudo debido al costo? 0 = No; 1 = Sí.
- **Health:** ¿Diría usted que en general su salud es:(escala 1-5) 1 = excelente; 2 = muy buena; 3 = buena; 4 = regular; 5 = mala?
- **Mental Health:** Escala de días de mala salud mental de 1 a 30 días.

- **Physical illness:** Días de enfermedad o lesión física en los últimos 30 días escala 1-30.
- **Diff walk:** ¿Tiene usted serias dificultades para caminar o subir escaleras? 0 = No; 1 = Sí.

Esta riqueza en la información permite analizar diversos factores de riesgo asociados con la diabetes y desarrollar modelos robustos que predigan la presencia de esta condición.

3. Sobre los modelos

3.1. Regresión Logística. La regresión logística es un modelo lineal ampliamente utilizado para resolver problemas de clasificación binaria. Aunque el nombre sugiere regresión", en realidad se basa en la probabilidad de que una observación pertenezca a una de las dos clases. La fórmula subyacente es una transformación logística o función sigmoide aplicada a una combinación lineal de las características de entrada:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0+\beta_1 X_1+...+\beta_n X_n)}}$$

Donde

- $P(Y = 1)$ es la probabilidad de pertenecer a la clase positiva.
- β_0 es el término de sesgo (intercepto), y $\beta_1, \ldots, \beta_n$ son los coeficientes asociados a cada característica $X_1, \ldots, X_n$.

Este modelo es eficiente, interpretable y funciona bien cuando las características son linealmente separables. Sin embargo, puede verse limitado en presencia de relaciones no lineales en los datos.

3.2. Bosques aleatorios. El modelo de bosques aleatorios está basado en un conjunto de árboles de decisión (ensemble). Utiliza el método de bagging (Bootstrap Aggregating) para crear múltiples árboles de decisión a partir de subconjuntos aleatorios del conjunto de datos y de sus características. Posteriormente, combina las predicciones individuales de cada árbol para realizar una clasificación más robusta y precisa.

Ventajas clave de los bosques aleatorios:

- Capacidad no lineal: Puede capturar relaciones complejas entre las características y la variable objetivo.
- Reducción del sobreajuste: Al combinar múltiples árboles, se reduce el riesgo de sobreajustar el modelo a los datos de entrenamiento.
- Importancia de características: Los bosques aleatorios permiten medir la importancia de cada característica en la predicción, lo cual es útil para la interpretación del modelo. En general, este modelo es una excelente opción para conjuntos de datos grandes y complejos, especialmente cuando se desconoce la relación exacta entre las variables predictoras y la variable objetivo.

Para asegurar una implementación neutral y reproducible, utilizaremos las versiones estándar de regresión logística y bosques aleatorios disponibles en la biblioteca **Scikit-learn** de Python. Esta biblioteca es una herramienta robusta, optimizada y ampliamente utilizada en la comunidad científica y profesional para la construcción de modelos de **Machine Learning**.

4. Análisis y visualización de los datos

Cargar las librerías y el conjunto de datos.

```
# Libreras
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Cargar conjunto de datos
diabetes = pd.read_csv('/content/drive/MyDrive/
    ColabNotebooks/
    diabetes_binary_5050split_health_indicators_BRFSS2015
    .csv')
```

Listing 1. Mostrar primeras filas del dataset

```
diabetes.head()
```

Diabetes	HighBP	CholCheck	BMI	Smoker	Stroke	...
0.0	1.0	0.0	26.0	0.0	0.0	...
0.0	1.0	1.0	26.0	1.0	1.0	...
0.0	0.0	1.0	26.0	0.0	0.0	...
0.0	1.0	1.0	28.0	1.0	0.0	...
0.0	0.0	1.0	29.0	1.0	0.0	...

Listing 2. Nombre de las columnas (features)

```
diabetes.columns
```

```
Index(['Diabetes_binary','HighBP','HighChol',
'CholCheck', 'BMI','Smoker','Stroke',
'HeartDiseaseorAttack','PhysActivity','Fruits',
'Veggies','HvyAlcoholConsump','AnyHealthcare',
'NoDocbcCost','GenHlth','MentHlth','PhysHlth',
'DiffWalk','Sex', 'Age','Education','Income'],
dtype='object')
```

4.1. Análisis exploratorio de los datos.

Listing 3. Información del dataset

```
diabetes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70692 entries, 0 to 70691
Data columns (total 22 columns):
 #   Column                Non-Null Count
---  ------                --------------
 0   Diabetes_binary       70692 non-null
 1   HighBP                70692 non-null
 2   HighChol              70692 non-null
 3   CholCheck             70692 non-null
 4   BMI                   70692 non-null
 5   Smoker                70692 non-null
 6   Stroke                70692 non-null
 7   HeartDiseaseorAttack  70692 non-null
 8   PhysActivity          70692 non-null
 9   Fruits                70692 non-null
 10  Veggies               70692 non-null
 11  HvyAlcoholConsump     70692 non-null
 12  AnyHealthcare         70692 non-null
 13  NoDocbcCost           70692 non-null
 14  GenHlth               70692 non-null
 15  MentHlth              70692 non-null
 16  PhysHlth              70692 non-null
 17  DiffWalk              70692 non-null
 18  Sex                   70692 non-null
 19  Age                   70692 non-null
 20  Education             70692 non-null
 21  Income                70692 non-null
dtypes: float64(22)
memory usage: 11.9 MB
```

Listing 4. Descripción estadística del dataset

```
diabetes.describe()
```

	Diabetes	HighBP	HighChol	CholCheck	BMI	...
count	70692	70692	70692	70692	70692	...
mean	0.500	0.56	0.52	0.97	29	...
std	0.500	0.49	0.49	0.15	7	...
min	0.000	0.00	0.00	0.00	12	...
25%	0.000	0.00	0.00	1.00	25	...
50%	0.500	1.00	1.00	1.00	29	...
75%	1.000	1.00	1.00	1.00	33	...
max	1.000	1.00	1.00	1.00	98	...

Con esto se observa que no hay datos nulos en el conjunto de datos para ninguna característica, de igual forma no contiene errores de formato.

Pairplot

Listing 5. Selección de columnas y creación de pairplot

```
# Seleccionar las primeras 10 caractersticas y la
    variable objetivo
selected_columns = diabetes.columns[:10].tolist()
diabetes_subset = diabetes[selected_columns]

# Crear el pairplot solo con las primeras 10
    caractersticas
sns.pairplot(diabetes_subset, hue='Diabetes_binary',
    diag_kind='kde', height=3)
plt.show()
```

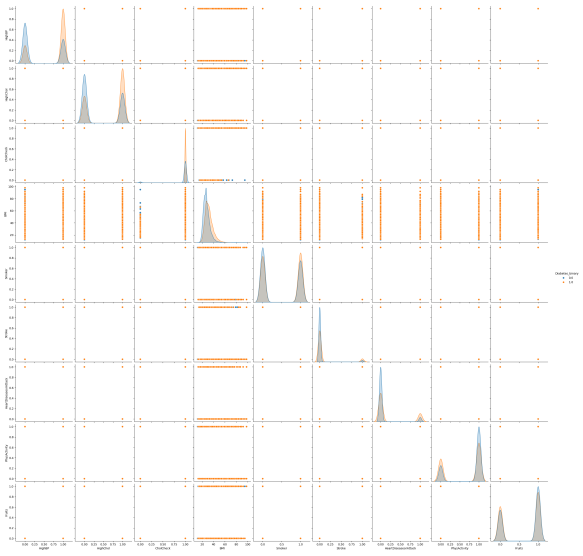

Fig. 1. Pairplot.

Histogramas

Listing 6. Histograma de las variables del dataset

```
diabetes.hist(figsize=(20, 14))
plt.show()
```

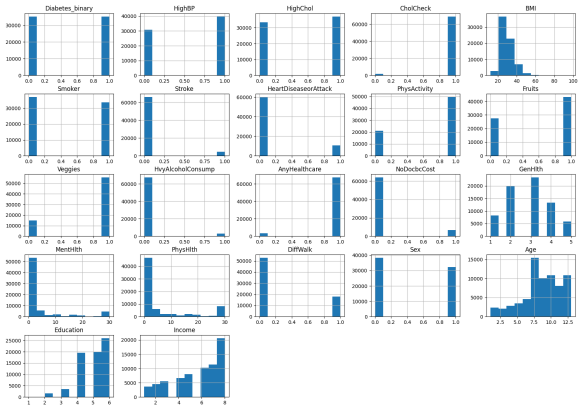

Fig. 2. Histogramas.

4.2. Detección de outliers. Comprobaremos si contiene o no valores atípicos nuestro conjunto de datos.

Identificación con el Rango Intercuartil (IQR)

Listing 7. Detección de valores atípicos usando IQR

```
# Detectar valores atpicos usando IQR
Q1 = diabetes.quantile(0.25)
Q3 = diabetes.quantile(0.75)
IQR = Q3 - Q1

# Detectar valores fuera de los lmites
outliers = ((diabetes < (Q1 - 1.5 * IQR)) | (diabetes > (
    Q3 + 1.5 * IQR)))
print(outliers.sum())  # Total de valores atpicos por
    columna
```

```
Diabetes_binary             0
HighBP                      0
HighChol                    0
CholCheck                1749
BMI                      2181
Smoker                      0
Stroke                   4395
HeartDiseaseorAttack    10449
PhysActivity                0
Fruits                      0
Veggies                 14932
HvyAlcoholConsump        3020
AnyHealthcare            3184
NoDocbcCost              6639
GenHlth                     0
MentHlth                11816
PhysHlth                10624
DiffWalk                    0
Sex                         0
Age                         0
Education                   0
Income                      0
dtype: int64
```

Veamos que, por ejemplo, con este criterio del IQR obtuvimos 14932 outliers para la característica *veggies*, sin embargo este criterio no es suficiente para determinar si hay que eliminar estos valores, pues al ser *veggies* una característica binaria, si no está balanceada puede ocurrir que uno de sus dos valores binarios los reconozca todos como outliers. El código siguiente muestra a lo que me refiero.

Listing 8. Conteo de valores en la columna "Veggies"

```
diabetes["Veggies"].value_counts()
```

```
Veggies
1.0    55760
0.0    14932
Name: count, dtype: int64
```

No hay outliers en este caso. Las gráficas de cajas nos ayudarán a visualizar mejor características que puedan tener o no valores atípicos.

Creación de una boxplot para cada característica

Listing 9. Boxplots de todas las características

```
# Crear boxplots para todas las caractersticas
plt.figure(figsize=(15, 10))
sns.boxplot(data=diabetes)
plt.xticks(rotation=90)
plt.show()
```

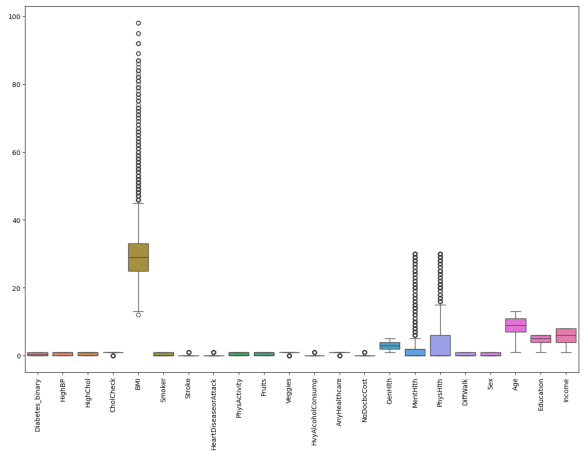

Fig. 3. Boxplot.

Ahora parecen ser el IMC, Mental Health y Physic Health las características con más ruido. Usaremos un último criterio para finalizar el análisis.

Z-score: Un valor es atípico si su Z-score es mayor a un umbral (típicamente 3).

Listing 10. Cálculo de Z-scores y detección de valores atípicos

```
from scipy.stats import zscore

# Calcular Z-scores
z_scores = diabetes.apply(zscore)
outliers = (z_scores.abs() > 3).sum()
print(outliers)  # Total de valores atpicos por columna
```

```
Diabetes_binary             0
HighBP                      0
HighChol                    0
CholCheck                1749
BMI                       801
Smoker                      0
Stroke                   4395
HeartDiseaseorAttack        0
PhysActivity                0
Fruits                      0
Veggies                     0
HvyAlcoholConsump        3020
AnyHealthcare            3184
NoDocbcCost              6639
GenHlth                     0
MentHlth                 4373
PhysHlth                    0
DiffWalk                    0
Sex                         0
Age                         0
Education                  75
Income                      0
dtype: int64
```

Visualización de valores atípicos para BMI

Listing 11. Cálculo de Z-scores para BMI y visualización de valores atípicos

```
# Calcular Z-scores para la caracterstica 'BMI'
z_scores_bmi = zscore(diabetes['BMI'])

# Filtrar los valores atpicos (Z-score mayor a 3 o menor a -3)
outliers_bmi = diabetes.loc[(z_scores_bmi > 3) | (z_scores_bmi < -3), 'BMI']

# Visualizar histogramas de los valores atpicos
plt.figure(figsize=(10, 5))
sns.histplot(outliers_bmi, kde=True)
plt.title('Distribucin de valores atpicos en BMI')
plt.show()
```

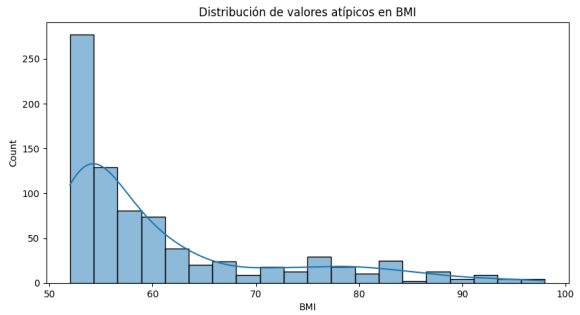

Fig. 4. Distribución de valores atípicos.

Listing 12. Muestra de los primeros 10 valores atípicos en BMI

```
outliers_bmi.head(10)
```

```
13     58.0
192    52.0
199    69.0
284    56.0
446    52.0
553    92.0
587    53.0
612    98.0
783    79.0
791    58.0
Name: BMI, dtype: float64
```

Un IMC de 40 o más representa obesidad grado 3, que si bien es cierto que en USA hay demasiados casos de obesidad grado 3, un IMC de 58, por ejemplo, está muy por encima del límite de obesidad mórbida (obesidad grado 3). Uno de 98 es impensable.

Listing 13. Estadísticas descriptivas de los valores atípicos en BMI

```
outliers_bmi.describe()
```

```
count    801.000000
mean      60.901373
std       10.629394
min       52.000000
25%       53.000000
50%       56.000000
75%       64.000000
max       98.000000
Name: BMI, dtype: float64
```

Finalmente podemos decir que tanto el IMC como Mental Health son características que es probable que contengan valores atípicos (outliers), debido a que no podemos asegurar que son variables que nos interesen, aún no borraremos instancias, sin embargo de requerirse usar la variable "BMI"por ejemplo, haremos limpieza de algunas instancias con el siguiente código.

Listing 14. Filtrar el dataset eliminando los valores atípicos de BMI

```
# Filtrar el dataset eliminando los valores atpicos de BMI
diabetes_cleaned = diabetes[(z_scores_bmi <= 3) & (z_scores_bmi >= -3)]
```

Listing 15. Información del dataset limpio

```
diabetes_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 69891 entries, 0 to 70691
Data columns (total 22 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Diabetes_binary       69891 non-null  float64
 1   HighBP                69891 non-null  float64
 2   HighChol              69891 non-null  float64
 3   CholCheck             69891 non-null  float64
 4   BMI                   69891 non-null  float64
 5   Smoker                69891 non-null  float64
 6   Stroke                69891 non-null  float64
 7   HeartDiseaseorAttack  69891 non-null  float64
 8   PhysActivity          69891 non-null  float64
 9   Fruits                69891 non-null  float64
 10  Veggies               69891 non-null  float64
 11  HvyAlcoholConsump     69891 non-null  float64
 12  AnyHealthcare         69891 non-null  float64
 13  NoDocbcCost           69891 non-null  float64
 14  GenHlth               69891 non-null  float64
 15  MentHlth              69891 non-null  float64
 16  PhysHlth              69891 non-null  float64
 17  DiffWalk              69891 non-null  float64
 18  Sex                   69891 non-null  float64
 19  Age                   69891 non-null  float64
 20  Education             69891 non-null  float64
 21  Income                69891 non-null  float64
dtypes: float64(22)
memory usage: 12.3 MB
```

5. Selección de variables

Generación de un mapa de calor para analizar la correlación con la variable objetivo

Listing 16. Heatmap de la matriz de correlación

```
# Heatmap de la matriz de correlacin
plt.figure(figsize=(15, 12))
sns.heatmap(diabetes_cleaned.corr(), annot=True, cmap='
    coolwarm', fmt=".2f")
plt.title('Matriz de correlacin')
plt.show()
```

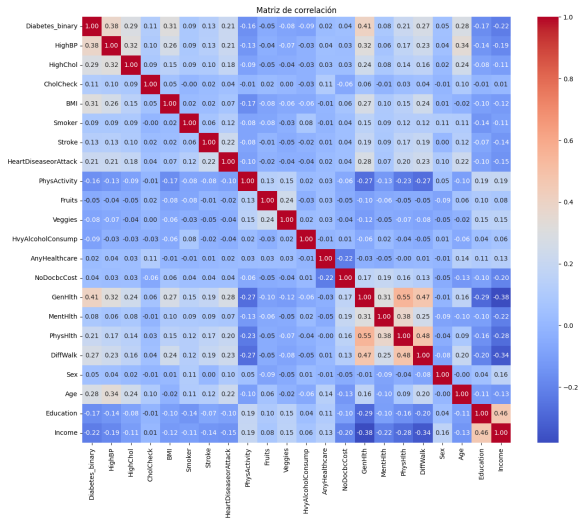

Fig. 5. Mapa de correlación.

Observamos que no existen correlaciones fuertes directas con la variable objetivo, por lo que procedemos a usar otros métodos para la selección de variables

Métodos de selección de características basados en pruebas estadísticas

Los *p-valores* indican la significancia estadística de cada característica en relación con la variable objetivo.

Listing 17. Regresión logística para la predicción de diabetes

```
# Separamos las caractersticas (X) y la variable objetivo
    (y)
X = diabetes_cleaned.drop('Diabetes_binary', axis=1)
y = diabetes_cleaned['Diabetes_binary']

# Ajustamos el modelo con statsmodels
import statsmodels.api as sm
X_with_const = sm.add_constant(X)
logit_model = sm.Logit(y, X_with_const)
result = logit_model.fit()

# Ver los p-valores de las caracteristicas
print(result.summary())
```

```
                 coef    std err          z      P>|z|
-------------------------------------------------------
const         -7.2282      0.127    -57.099      0.000
HighBP         0.7128      0.020     35.832      0.000
HighChol       0.5818      0.019     30.589      0.000
CholCheck      1.3310      0.082     16.305      0.000
BMI            0.0888      0.002     52.134      0.000
Smoker         0.0004      0.019      0.022      0.982
Stroke         0.1777      0.041      4.312      0.000
HeartDis.      0.2540      0.029      8.873      0.000
PhysAct       -0.0232      0.021     -1.080      0.280
Fruits        -0.0334      0.020     -1.690      0.091
Veggies       -0.0576      0.024     -2.451      0.014
HvyAlcohol    -0.7408      0.049    -15.112      0.000
AnyHealth      0.0490      0.048      1.028      0.304
NoDocCost      0.0170      0.034      0.494      0.621
GenHlth        0.5810      0.012     50.353      0.000
MentHlth      -0.0045      0.001     -3.466      0.001
PhysHlth      -0.0079      0.001     -6.592      0.000
DiffWalk       0.1006      0.026      3.852      0.000
Sex            0.2676      0.019     13.867      0.000
Age            0.1566      0.004     39.658      0.000
Education     -0.0345      0.010     -3.345      0.001
Income        -0.0606      0.005    -11.561      0.000
```

Regularización (Lasso)

Valores iguales o muy cercanos a 0 pueden ser descartados

Listing 18. Ajuste del modelo de regresión logística con L1

```
from sklearn.linear_model import LogisticRegressionCV

# Ajusta un modelo de regresin logstica con
    regularizacin L1
model_lasso = LogisticRegressionCV(penalty='l1', solver=
    'liblinear', max_iter=1000)
model_lasso.fit(X, y)
```

```
LogisticRegressionCV(max_iter=1000, penalty='l1',
solver='liblinear')
```

Listing 19. Coeficientes del modelo con regularización L1

```
df = pd.DataFrame(model_lasso.coef_, columns=X.columns)
df.value_counts()
```

```
 HighBP    HighChol  CholCheck      BMI    Smoker
 0.711849  0.576437  1.060543   0.086801   0.0

 Stroke   HeartDiseaseorAttack  PhysActivity   Fruits
 0.147009      0.249335         -0.023482   -0.029748

 Veggies HvyAlcoholConsump  AnyHealthcare  NoDocbcCost
 -0.056004    -0.692189           0.0           0.0

 GenHlth   MentHlth  PhysHlth  DiffWalk     Sex
 0.570769  -0.004673  -0.0073   0.095373  0.255479

 Age    Education    Income
0.153843  -0.042935  -0.061675
```

Procedemos a crear un data frame con las características seleccionadas basado en lo anteriormente mencionado.

Listing 20. Selección de características

```
# Seleccionar caracteristicas
```

```
diabetes_new = diabetes_cleaned.drop(columns=["NoDocbcCost", "Smoker", "AnyHealthcare", "PhysActivity", "Fruits", "MentHlth", "Veggies"])
```

Listing 21. Extracción de valores para X e y

```
# Extraer los valores para X e y
X = diabetes_new.drop(columns=['Diabetes_binary']).values
y = diabetes_new['Diabetes_binary'].values
```

5.1. Preparación de los datos.

Listing 22. Mostrar los valores de X e y

```
X, y
```

```
(array([[ 1.,  0.,  1., ...,  4.,  6.,  8.],
        [ 1.,  1.,  1., ..., 12.,  6.,  8.],
        [ 0.,  0.,  1., ..., 13.,  6.,  8.],
        ...,
        [ 1.,  1.,  1., ..., 13.,  6.,  4.],
        [ 1.,  1.,  1., ..., 11.,  2.,  4.],
        [ 1.,  1.,  1., ...,  9.,  6.,  2.]]),
 array([0., 0., 0., ..., 1., 1., 1.]))
```

Listing 23. Forma de X e y

```
X.shape, y.shape
```

```
((69891, 14), (69891,))
```

Listing 24. Estandarización de X

```
# Estandarizar X
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Listing 25. Mostrar X escalado

```
X_scaled
```

```
array([[ 0.8842701 , -1.05183461,  0.15936162,
..., -1.61103496, 1.04610815,  1.05474772],
[ 0.8842701 ,  0.95071981,  0.15936162, ...,
1.19252298, 1.04610815,  1.05474772],
[-1.13087619, -1.05183461,  0.15936162, ...,
1.54296772, 1.04610815,  1.05474772], ...,
[ 0.8842701 ,  0.95071981,  0.15936162, ...,
1.54296772, 1.04610815, -0.78865897],
[ 0.8842701 ,  0.95071981,  0.15936162, ...,
0.84207823,  -2.84342741, -0.78865897],
[ 0.8842701 ,  0.95071981,  0.15936162, ...,
0.14118875, 1.04610815, -1.71036232]] )
```

Separamos el conjunto de datos en prueba y entrenamiento.

Listing 26. Separar conjunto de datos

```
#separar conjunto de datos
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

6. Modelo de regresión logística

Listing 27. Crear y entrenar el modelo de regresión logística

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Crear el modelo de regresin logstica
model = LogisticRegression(random_state=42, max_iter=1000)
# Entrenar el modelo
model.fit(X_train, y_train)

# Hacer predicciones en el conjunto de prueba
y_pred = model.predict(X_test)
```

Calcular accuracy para el conjunto de prueba.

Listing 28. Calcular la precisión del modelo

```
# Calcular el accuracy
accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy del modelo de regresin logstica : {accuracy:.2f}")
```

```
Accuracy del modelo de regresión logística: 0.75
```

Matriz de confusión para el modelo.

Listing 29. Matriz de confusión

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Crear la matriz de confusin
cm = confusion_matrix(y_test, y_pred)

# Visualizar la matriz de confusin
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model.classes_)
disp.plot(cmap="Blues")
```

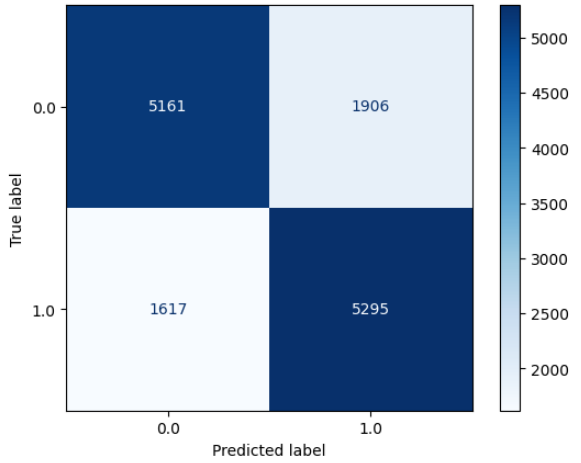

Fig. 6. Matriz de confusión para el modelo de regresión logística.

6.1. Reporte de la clasificación.

Listing 30. Generar el reporte de clasificación

```
from sklearn.metrics import classification_report

# Generar el reporte de clasificacin
report = classification_report(y_test, y_pred,
    target_names=["No Diabetes", "Diabetes"])
print(report)
```

```
              precision    recall  f1-score   support

 No Diabetes       0.76      0.73      0.75      7067
    Diabetes       0.74      0.77      0.75      6912

    accuracy                           0.75     13979
   macro avg       0.75      0.75      0.75     13979
weighted avg       0.75      0.75      0.75     13979
```

Curva de ROC.

Listing 31. Calcular y graficar la curva ROC

```
from sklearn.metrics import roc_curve, roc_auc_score
# Calcular las probabilidades predichas
y_proba = model.predict_proba(X_test)[:, 1]

# Calcular la curva ROC
fpr, tpr, thresholds = roc_curve(y_test, y_proba)

# Calcular el AUC
auc = roc_auc_score(y_test, y_proba)

# Graficar la curva ROC
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"AUC = {auc:.2f}")
plt.plot([0, 1], [0, 1], linestyle="--", color="gray")
    # Lnea de no discriminacin
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Curva ROC")
plt.legend(loc="lower right")
plt.grid()
plt.show()
```

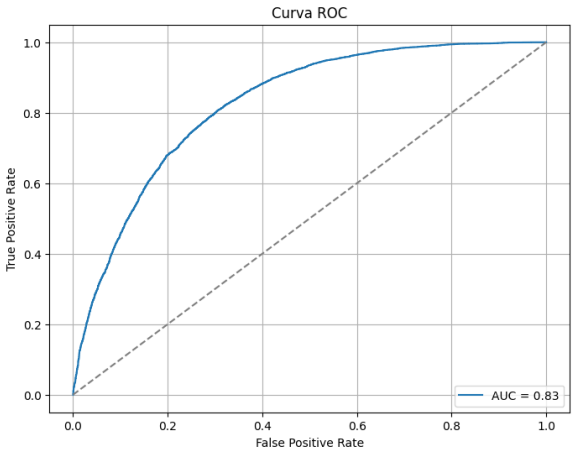

Fig. 7. Curva de ROC.

Listing 32. Comparación de valores reales y predichos

```
# Crear un DataFrame para comparar
results = pd.DataFrame({"Actual": y_test, "Predicted":
    y_pred})

#  Visualizacin
plt.figure(figsize=(6, 4))
sns.countplot(x="Actual", hue="Predicted", data=results,
    palette="viridis")
plt.title("Comparacin de valores reales y predichos")
plt.xlabel("Clase Real")
plt.ylabel("Conteo")
plt.legend(title="Clase Predicha")
plt.show()
```

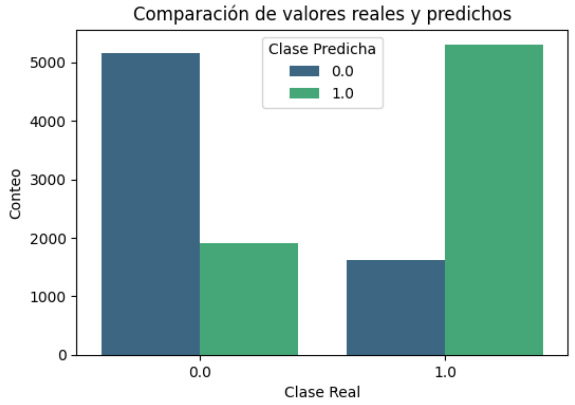

Fig. 8. Valores reales vs predichos.

6.2. Validación cruzada.

Listing 33. Validación cruzada del modelo de regresión logística

```
from sklearn.model_selection import cross_val_score, StratifiedKFold

# Crear el modelo de regresin logstica
logistic_model = LogisticRegression(max_iter=1000, random_state=42)

# Definir la validacin cruzada
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42) # 5 particiones estratificadas

# Calcular el puntaje de validacin cruzada
scores = cross_val_score(logistic_model, X_scaled, y, cv=cv, scoring='accuracy')

# Resultados
print(f"Puntajes de validacin cruzada: {scores}")
print(f"Precisin promedio: {np.mean(scores):.2f}")
print(f"Desviacin estndar: {np.std(scores):.2f}")
```

```
Puntajes de validación cruzada: [0.74826526
0.74645872 0.74688797 0.74731721 0.75075118]
Precisión promedio: 0.75
Desviación estándar: 0.00
```

7. Modelo de Bosques aleatorios

Listing 34. Creación y entrenamiento del modelo de bosque aleatorio

```
from sklearn.ensemble import RandomForestClassifier

# Crear el modelo Random Forest
random_forest = RandomForestClassifier(n_estimators=200, random_state=102)

# Entrenar el modelo
random_forest.fit(X_train, y_train)

# Realizar predicciones en el conjunto de prueba
y_pred = random_forest.predict(X_test)
```

Listing 35. Cálculo de la precisión del modelo Random Forest

```
# Calcular el accuracy del modelo
accuracy = accuracy_score(y_test, y_pred)

# Mostrar el resultado
print(f"Accuracy del modelo Random Forest: {accuracy:.2f}")
```

```
Accuracy del modelo Random Forest: 0.72
```

Listing 36. Visualización del primer árbol de decisión del bosque aleatorio

```
from sklearn.tree import plot_tree

# Seleccionar uno de los rboles del bosque aleatorio
arbol_individual = random_forest.estimators_[0] # Seleccionamos el primer rbol

class_names = ['No Diabetes', 'Diabetes']
# Limitar la profundidad del rbol al graficar
plt.figure(figsize=(20, 10))
plot_tree(arbol_individual,
          feature_names=diabetes_new.drop(columns=['Diabetes_binary']).columns,
          class_names=class_names,
          max_depth=4, # Limitar la profundidad del rbol
          filled=True,
          rounded=True,
          fontsize=10)
plt.show()
```

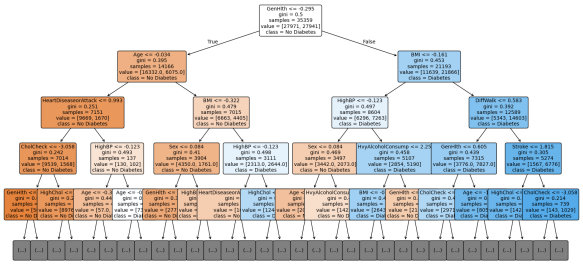

Fig. 9. Árbol de decisión aleatorio.

Matriz de confusión para el modelo.

Listing 37. Creación y visualización de la matriz de confusión

```
# Crear la matriz de confusin
cm = confusion_matrix(y_test, y_pred)

# Visualizar la matriz de confusin
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model.classes_)
disp.plot(cmap="Blues")
```

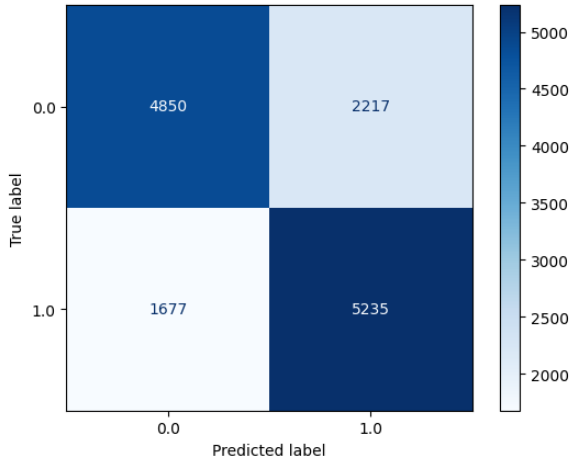

Fig. 10. Matriz de confusión del modelo de bosques aleatorios.

7.1. Validación cruzada.

Listing 38. Cálculo de la validación cruzada

```
# Definir la validacin cruzada
cv = StratifiedKFold ( n_splits =5, shuffle =True,
    random_state=42)

# Calcular el puntaje de validacin cruzada
scores = cross_val_score (random_forest, X_scaled, y, cv=
    cv, scoring='accuracy')

# Resultados
print(f"Puntajes de validacin cruzada: {scores}")
print(f" Precisin promedio: {np.mean(scores):.2 f}")
print(f" Desviacin estndar : {np.std( scores ):.2 f}")
```

```
Puntajes de validación cruzada: [0.72122469 0.7238517
0.72420947 0.72313636 0.72299328]
Precisión promedio: 0.72
Desviación estándar: 0.00
```

8. Conlusión y trabajo futuro

En base a las pruebas realizadas a ambos modelos, podemos afirmar que la regresión logística ofrece una clasificación ligeramente superior a la de los bosques aleatorios, alcanzando de forma estable un accuracy del 75 %, frente al 72 % obtenido por el segundo modelo. En este caso, consideramos válida la comparación mediante esta métrica, ya que fue calculada a través de validación cruzada y el conjunto de datos estaba balanceado, lo que vuelve al accuracy una medida razonablemente representativa del rendimiento general.

No obstante, existen diversas sugerencias, consideraciones y oportunidades de mejora en el desarrollo de este trabajo. Una de las más importantes consiste en **ajustar** el tipo de error que el modelo puede cometer, dado que en un escenario clínico, no es igual de grave un falso positivo que un falso negativo. En el caso del diagnóstico de diabetes, un falso negativo podría tener consecuencias más severas al no alertar de una condición existente. Aquí es donde las métricas de **Recall** y **Precision** cobran mayor relevancia, ya que permiten evaluar y controlar este tipo de errores. Ajustando el umbral de decisión del modelo, es posible priorizar la detección de verdaderos positivos, aunque esto puede reducir ligeramente el accuracy, lo que introduce un dilema común en problemas sensibles: **balancear métricas**. Así, la primera recomendación clave es encontrar un equilibrio adecuado entre accuracy, recall y precision, considerando siempre el contexto clínico en el que se aplica el modelo.

Otra recomendación es ampliar el número de modelos evaluados. Aunque en este trabajo nos enfocamos en regresión logística y bosques aleatorios, existen **otros algoritmos** que podrían ofrecer mejores resultados en este conjunto de datos. Entre ellos, se podrían considerar: SVM (con kernel lineal y no lineal), K-Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Naive Bayes, o incluso redes neuronales multicapa (MLP). Todos estos modelos deben evaluarse bajo esquemas de validación cruzada para asegurar resultados comparables y robustos.

Además, puede implementarse el uso de **técnicas de ensamblado** *(ensemble learning)*, como bagging, boosting o stacking, las cuales combinan múltiples modelos para mejorar el rendimiento general. Aunque suelen aumentar el tiempo de entrenamiento, suelen ofrecer mejoras en métricas como accuracy y recall.

Una mejora importante adicional es la **optimización de hiperparámetros**. Tanto para regresión logística como para bosques aleatorios (y los modelos mencionados anteriormente), es recomendable utilizar técnicas como *GridSearchCV()* o *RandomizedSearchCV()* de scikit-learn, que permiten buscar de manera sistemática los parámetros óptimos, como la cantidad de árboles, la profundidad máxima, o el parámetro de regularización.

También es aconsejable considerar técnicas de **regularización**, como $L2$ (Ridge) en la regresión logística, para evitar el sobreajuste, así como limitar la profundidad de los árboles en los bosques aleatorios como medida de control de complejidad.

Por último, pero no menos importante, se debe prestar mayor atención al **procesamiento** de los datos. Algunos criterios aplicados en esta etapa fueron arbitrarios y podrían mejorarse. Es recomendable profundizar en el análisis estadístico exploratorio, mejorar la limpieza del conjunto de datos, aplicar estrategias más sistemáticas de detección y tratamiento de outliers, así como evaluar otras transformaciones que puedan beneficiar el aprendizaje del modelo.

Referencias

[1] Centers for Disease Control and Prevention (CDC). (2015). *Behavioral Risk Factor Surveillance System Survey Data*. U.S. Department of Health and Human Services. Recuperado de: https://www.cdc.gov/brfss/annual_data/annual_2015.html

[2] Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10), 78–87. https://doi.org/10.1145/2347736.2347755

[3] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2ª ed.). O'Reilly Media.

[4] Kaggle. (n.d.). *Diabetes dataset*. Recuperado en enero de 2025, de https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset

[5] Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

[6] Raschka, S., Mirjalili, V. (2019). *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2* (2ª ed.). Packt Publishing.

[7] World Health Organization (WHO). (2023). *Diabetes: Fact Sheet*. Recuperado de: https://www.who.int/news-room/fact-sheets/detail/diabetes