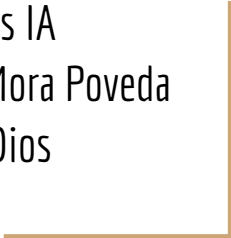




# Especialización en Inteligencia Artificial

## Trabajo Semana 2 Dataset Diabetes

Asignatura: Fundamentos IA  
Realizado por: Michael Andrés Mora Poveda  
Universidad Minuto de Dios  
Marzo 2023



# Propósito:

El objetivo de este trabajo es realizar el análisis exploratorio de datos e implementación del modelo de regresión logística para el Diabetes dataset, el cual se puede encontrar en la siguiente página web (y dentro de este folder también) y el cual se encuentra referenciado al final de este notebook:

- <https://www.kaggle.com/datasets/mathchi/diabetes-data-set>

Además, es importante tener en cuenta que dentro de este análisis se aplicarán varias librerías de Python para mayor practicidad.

Las siguientes descripciones serán útiles para conocer la estructura del dataset:

## Descripción general:

Según el repositorio oficial, esta información ha sido dispuesta por el National Institute of Diabetes and Digestive and Kidney Diseases cuyo objetivo es predecir si una paciente femenina tiene diabetes o no basado en distintos features:

1. Pregnancies: Número de embarazos
2. Glucose: Concentración de glucosa en 2 horas de acuerdo a test de tolerancia
3. BloodPressure: Presión sanguínea en milímetros de mercurio (mm Hg)
4. SkinThickness: Grosor del pliegue cutáneo del tríceps (mm)
5. Insulin: Insulina en 2 horas ( $\mu$  U/ml)
6. BMI: Índice de masa corporal (Peso en kg/(altura en m)<sup>2</sup>)
7. DiabetesPedigreeFunction: Diabetes pedigree function
8. Age: Edad (years)
9. Outcome: Variable objetivo de clasificación (0 para test negativo ó 1 para test positivo)

**Nota:** Es importante resaltar, que se seleccionará solamente una de las variables con el fin de realizar el entrenamiento del modelo mencionado.

# Notas:

Esta presentación contiene partes del script generado en Jupyter Notebook de acuerdo a la actividad de la semana 2 de la Especialización en IA de la Universidad Minuto de Dios. El script completo se encuentra en el siguiente repositorio de Github:

[https://github.com/micmorap/Machine\\_Learning\\_Portfolio/blob/main/AI%20Specialization%20-%20U%20of%20Minuto%20de%20Dios/Fundamentos%20de%20IA/Semana\\_2/Fundamentos\\_IA\\_Semana\\_2\\_Diabetes\\_Regresi%C3%B3nLog%C3%ADstica.ipynb](https://github.com/micmorap/Machine_Learning_Portfolio/blob/main/AI%20Specialization%20-%20U%20of%20Minuto%20de%20Dios/Fundamentos%20de%20IA/Semana_2/Fundamentos_IA_Semana_2_Diabetes_Regresi%C3%B3nLog%C3%ADstica.ipynb)

# Análisis exploratorio:

Vamos a revisar la estructura general de los datos:

```
#Se importan las librerías clásicas:
```

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

```
#Importamos los archivos contenidos en el folder y chequeamos los datasets:
```

```
dfDiabetes = pd.read_csv('diabetes.csv', sep=',')
dfDiabetes.head(5)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
# Revisamos las dimensiones del dataset:
```

```
print('Total filas y columnas: {}'.format(dfDiabetes.shape))
```

Total filas y columnas: (768, 9)

# Validación de datos:

```
# Ciclo para ver el número de registros por feature
for i in dfDiabetes.columns:
    print('Columna (feature) {} y número de registros: {}'.format(i, dfDiabetes[i].value_counts().sum()))
```

```
Columna (feature) Pregnancies y número de registros: 768
Columna (feature) Glucose y número de registros: 768
Columna (feature) BloodPressure y número de registros: 768
Columna (feature) SkinThickness y número de registros: 768
Columna (feature) Insulin y número de registros: 768
Columna (feature) BMI y número de registros: 768
Columna (feature) DiabetesPedigreeFunction y número de registros: 768
Columna (feature) Age y número de registros: 768
Columna (feature) Outcome y número de registros: 768
```

```
# Número de valores NA
for i in dfDiabetes.columns:
    print('Columna (feature) {} y número de registros NA: {}'.format(i, dfDiabetes[i].isna().sum()))
```

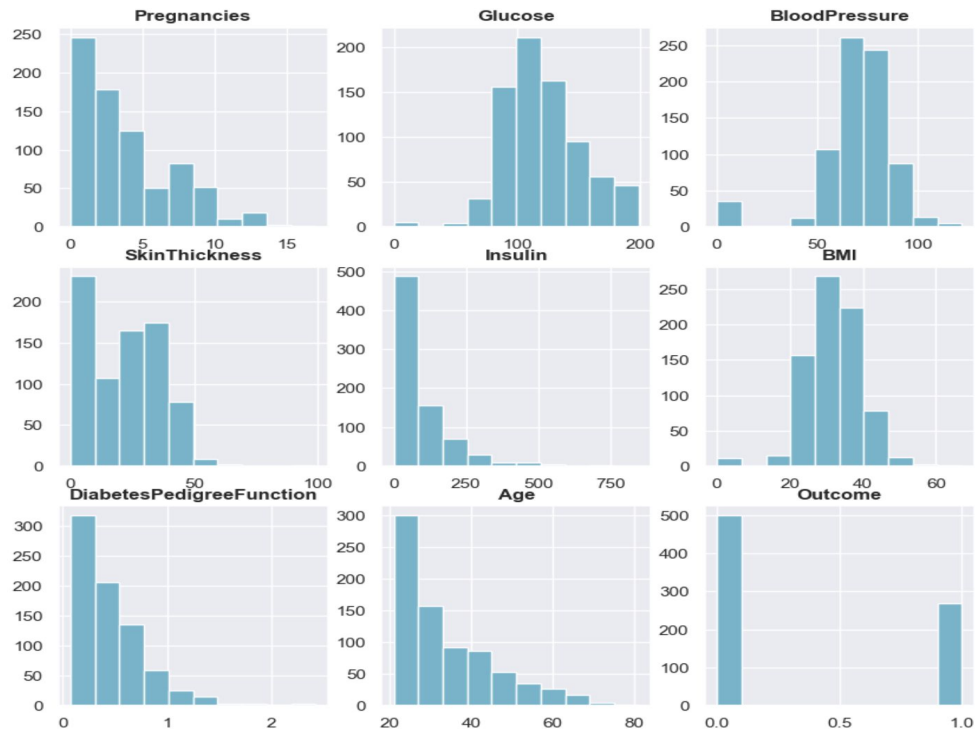
```
Columna (feature) Pregnancies y número de registros NA: 0
Columna (feature) Glucose y número de registros NA: 0
Columna (feature) BloodPressure y número de registros NA: 0
Columna (feature) SkinThickness y número de registros NA: 0
Columna (feature) Insulin y número de registros NA: 0
Columna (feature) BMI y número de registros NA: 0
Columna (feature) DiabetesPedigreeFunction y número de registros NA: 0
Columna (feature) Age y número de registros NA: 0
Columna (feature) Outcome y número de registros NA: 0
```

```
# Conteo de valores nulos por feature:
dfDiabetes.isnull().sum()
```

```
Pregnancies      0
Glucose           0
BloodPressure     0
SkinThickness     0
Insulin           0
BMI               0
DiabetesPedigreeFunction  0
Age               0
Outcome           0
dtype: int64
```

# Distribuciones:

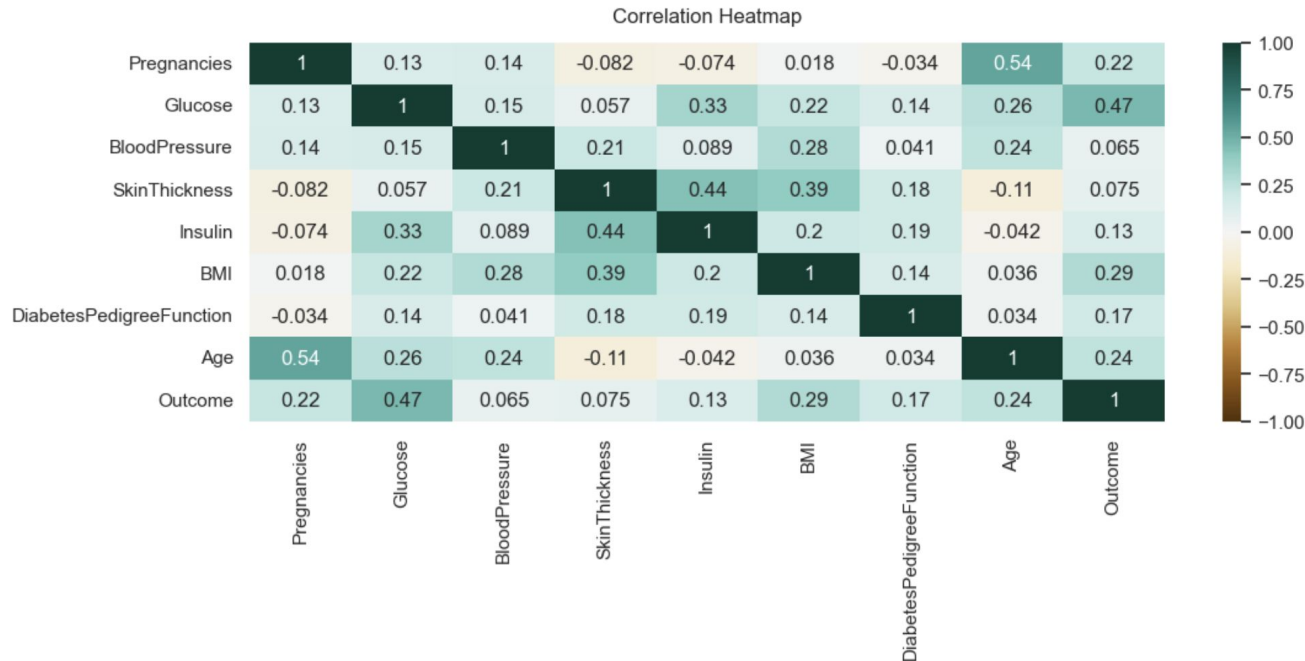
En términos generales, vemos que la data no tiene inconsistencias frente a valores nulos y sus descripciones son claras para el entendimiento del ejercicio. Ahora, revisaremos algunas gráficas y concentraciones:



*Imagen y código de autoría propia*

# Correlación de variables:

```
# Creamos el mapa de calor de correlaciones:  
plt.figure(figsize=(12, 4))  
heatmap = sns.heatmap(dfDiabetes.corr(), vmin=-1, vmax=1, annot=True, cmap='BrBG')  
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12);
```



# Observaciones de acuerdo a distribuciones y correlaciones:

De acuerdo a los resultados anteriores, se tienen las siguientes observaciones:

- \* Los histogramas a nivel general no muestran concentraciones de outliers, a excepción de la insulina. La variable de pregnancies muestra cierta cantidad de mujeres con un alto número de embarazos.
- \* La distribución del target variable está balanceada de los pacientes positivos vs. negativos.
- \* De acuerdo al mapa de calor de correlaciones, hay una fuerte relación entre las siguientes variables independientes:
  - \* Age – Pregnancies
  - \* BMI – SkinThickness
  - \* BMI – Insuline

Ahora, para el modelo de regresión se seleccionará la variable **Glucosa**, recordemos que es el azúcar producido en nuestra sangre gracias a todo el proceso del organismo. Es ampliamente utilizado para test de diabetes.



# Modelo de Regresión logística:

A continuación, se muestran algunos detalles del modelo implementado:

## Regresión Logística

```
In [192]: # Importamos la librerías de sklearn:
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from scipy.special import expit
```

```
In [125]: # Generamos un dataset con la columna Glucose y outcome, redefinimos las dimensiones del predictor:
X = dfDiabetes[['Glucose']]
y = dfDiabetes[['Outcome']]
#y = y.values.reshape(-1, 1)
```

```
In [126]: # Confirmamos las dimensiones de ambas columnas de datos:
print(X.shape, y.shape)

(768, 1) (768, 1)
```

```
In [127]: # Dividimos la data en training y testing set:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
```

```
In [128]: # Instanciamos un objeto de la clase LogisticRegression:
logistic_model = LogisticRegression()
```

```
In [129]: # Hacemos el entrenamiento del modelo de acuerdo al objeto instanciado:
logistic_model.fit(X_train, y_train)
```

```
/Users/michaelandr/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py:1111: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
Out[129]: ▾ LogisticRegression
LogisticRegression()
```

# Score del training y test sets:

```
In [135]: Revisamos ahora el score del modelo de acuerdo al entrenamiento realizado:
print("Score from the logistic regression model (training): {}".format(round(logistic_model.score(X_train, y_train),
Score from the logistic regression model (training): 76.0%

In [136]: Revisamos de manera análoga el score del modelo de acuerdo a la data de testing:
print("Score from the logistic regression model (testing): {}".format(round(logistic_model.score(X_test, y_test), 2)
Score from the logistic regression model (testing): 71.0%

In [137]: # Traemos el coeficiente asociado a la variable Glucosa y por ende a la ecuación de la regresión logística:
print(logistic_model.coef_)
[[0.04031273]]

In [150]: # Vemos el resultado de las predicciones:
predictions = logistic_model.predict(X_test)
predictions

Out[150]: array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,
1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0,
1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0])

In [155]: # Número de casos cuya predicción es negativa y positiva respectivamente:
print("Total casos negativos predcidos: {}".format((predictions == 0).sum()))
print("Total casos positivos predcidos: {}".format((predictions == 1).sum()))

Total casos negativos predcidos: 175
Total casos positivos predcidos: 56
```

# Matriz de confusión y performance:

```
In [166]: #Matriz de confusión
cm = confusion_matrix(y_test, predictions)
cm
```

```
Out[166]: array([[133, 26],
                [ 42, 30]])
```

```
In [176]: # Revisamos las métricas de rendimiento:
print(classification_report(y_test, predictions))
```

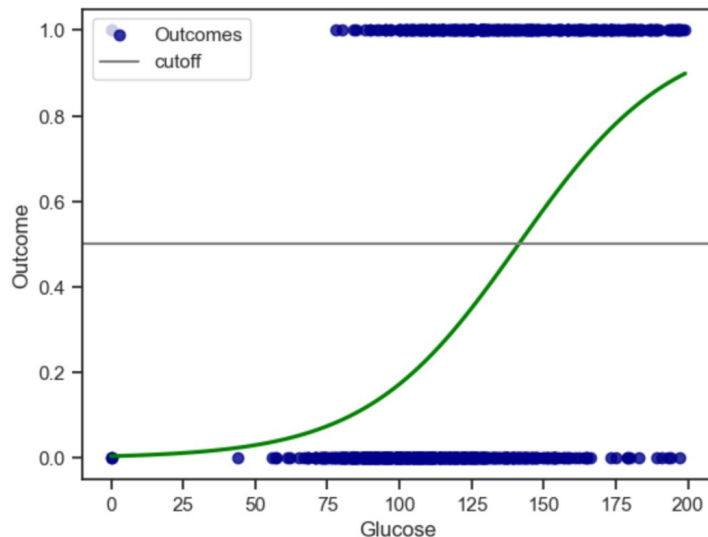
	precision	recall	f1-score	support
0	0.76	0.84	0.80	159
1	0.54	0.42	0.47	72
accuracy			0.71	231
macro avg	0.65	0.63	0.63	231
weighted avg	0.69	0.71	0.69	231

# Gráfica de la regresión logística y threshold:

```
In [230]: # graficamos el target variable, el threshold y la función sigmoidea asociada a las probabilidades:
X = dfDiabetes['Glucose']
y = dfDiabetes['Outcome']

#plot logistic regression curve
sns.regplot(x=X, y=y, data=dfDiabetes, logistic=True, ci=None,
            scatter_kws={'color': 'darkblue'}, line_kws={'color': 'green'}, label = 'Outcomes')
plt.legend(loc="lower right")
plt.axhline(.5, color="gray", label="cutoff")
#plt.axvline(5.4, color="purple", label="")
plt.legend(loc="upper left")
```

Out[230]: <matplotlib.legend.Legend at 0x7ff522386e80>



## Conclusiones

- Al entrenar este modelo vemos que la calidad es relativamente baja, dado que tiene solamente 76% y 71% respectivamente de accuracy para el training y testing test. Se debe considerar reentrenar el modelo ampliando la data del predictor ó agregando más predictores por ejemplo.
- El F1-Score (ponderado entre el recall y precision) puede mejorar, es una buena primera aproximación.
- Nuestro objetivo en general es detectar principalmente que pacientes sí tienen cáncer, es decir, reducir al máximo dentro de lo permitido los falsos positivos (casos que si se diagnostican como diabetes pero que están errados). Por lo cual debemos priorizar la métrica de Precision y evaluar si el threshold debe ser aumentado.
- Vemos la facilidad de las librerías en Python para optimizar tiempos en construcción y análisis de modelos.

## Referencias:

- [1]. Diabetes dataset (2022). Kaggle. Taken from: <https://www.kaggle.com/datasets/mathchi/diabetes-data-set>
- [2]. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
- [3]. J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.
- [4]. La función subplots. Interactive chaos. Taken from: <https://interactivechaos.com/es/manual/tutorial-de-matplotlib/la-funcion-subplots>
- [5]. Ajay Tech. Logistic Regression .Taken from: <https://ajaytech.co/python-logistic-regression/>