MODELO DE REGRESIÓN LOGISTICA

La regresión logística es un método estadístico que trata de modelar la probabilidad de una variable cualitativa binaria (dos posibles valores) en función de una o más variables independientes. La principal aplicación de la regresión logística es la creación de modelos de clasificación binaria.

Se llama regresión logística simple cuando solo hay una variable independiente y regresión logística múltiple cuando hay más de una. Dependiendo del contexto, a la variable modelada se le conoce como variable dependiente o variable respuesta, y a las variables independientes como regresores, predictores o features.

Un estudio quiere establecer un modelo que permita calcular la probabilidad de averiguar si Mujeres indias pime de 21 años o más que viven en Phoenix, la quinta ciudad más grande del estado de Arizona en EE.UU padecen o no diabetes, los datos usados para la investigacion se conservan en los Institutos Nacionales de Diabetes, Enfermedades Digestivas y Renales de EE.UU.. La variable objetivo se especifica como "resultado"; está codificada como: 1 si el resultado de la prueba de diabetes indica positivo y 0 si el resultado de la prueba de diabetes indica negativo.

Variables

Embarazos: Número de embarazos

Glucosa: Glucosa.

PresiónArterial: Presión arterial. BloodPressure

SkinThickness: Grosor de la piel

Insulina: Insulina.

IMC: Índice de masa corporal.

DiabetesPedigreeFunction: una función que calcula la probabilidad de tener diabetes según la edad de las personas y sus antecedentes familiares diabéticos.

Edad: Edad (años)

Resultado: Información sobre si la persona tiene diabetes o no. Tener la enfermedad (1) o no (0)

Para esta investigacion de llevaran a acabo los siguientes procesos:

1. Exploratory Data Analysis
2. Data Preprocessing
3. Model & Prediction

4. Model Evaluation
5. Model Validation: Holdout
6. Model Validation: 10-Fold Cross Validation
7. Prediction for A New Observation

In [ ]:
```
# Importar Librerias
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
pd.options.display.max_seq_items = 2000
import seaborn as sns
import os # directorios de trabajo

from sklearn.preprocessing import RobustScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, roc_auc_score, confusion_matrix, classi
from sklearn.model_selection import train_test_split, cross_validate
from sklearn.metrics import roc_curve, auc

pd.set_option('display.max_columns', None)
pd.set_option('display.float_format', lambda x: '%.3f' % x)
pd.set_option('display.width', 500)
```

In [ ]:
```
# Definir funciones

def outlier_thresholds(dataframe, col_name, q1=0.05, q3=0.95):
    quartile1 = dataframe[col_name].quantile(q1)
    quartile3 = dataframe[col_name].quantile(q3)
    interquantile_range = quartile3 - quartile1
    low_limit = quartile1 - 1.5 * interquantile_range
    up_limit = quartile3 + 1.5 * interquantile_range
    return low_limit, up_limit

def check_outlier(dataframe, col_name):
    low_limit, up_limit = outlier_thresholds(dataframe, col_name)
    if dataframe[(dataframe[col_name] < low_limit) | (dataframe[col_name] > up_limi
        return True
    else:
        return False

def replace_with_thresholds(dataframe, variable):
    low_limit, up_limit = outlier_thresholds(dataframe, variable)
    dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
    dataframe.loc[(dataframe[variable] > up_limit), variable] = up_limit
```

In [ ]:
```
# Establecer directorio de trabajo (working directory) e importar el Dataframe forb

os.getcwd()
```

Out[ ]:
```
'c:\\Users\\USUARIO\\OneDrive\\Documentos\\Uibero Ing de Software\\8 SEMESTRE\\Int
eligencia_artificial-main'
```

In [ ]:
```
# Mostrar los Archivos que hay en el ditrectorio de trabajo
```

```
os.listdir()
```

Out[ ]:
```
['.git',
 '1. Introduccion.ipynb',
 '2. Numpy.ipynb',
 '3. PandasForbes.ipynb',
 'Aactividad 1.docx',
 'caracteristicas de vinos.csv',
 'cluster_k-means.html',
 'cluster_k-means.ipynb',
 'cluster_k-significa.pdf',
 'Diabetes',
 'Explicacion del modelo kmeans.ipynb',
 'Forbes',
 'forbes23.ipynb',
 'forbes23AarbolDecisiones.ipynb',
 'image-1.png',
 'image.png',
 'inteligencia-artificial-metodos-tecnicas-y-aplicaciones.pdf',
 'ipynb',
 'Kmeansforbes2023',
 'Mall_Customers.csv',
 'modelo Kmeans forbes 2023.docx',
 'ModelodeRegresionLogistica.ipynb',
 'Project kmeans customer.ipynb',
 'pruebaforbes.ipynb']
```

Exploratory Data Analysis - Análisis exploratorio de datos

In [ ]:
```
# Importar la base de datos
diabetes = pd.read_csv(r"C:\Users\USUARIO\OneDrive\Documentos\Uibero Ing de Softwar
diabetes.head()
```

Out[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
0	6	148	72	35	0	33.600	
1	1	85	66	29	0	26.600	
2	8	183	64	0	0	23.300	
3	1	89	66	23	94	28.100	
4	0	137	40	35	168	43.100	

In [ ]:
```
# muestra las columnas del DataFrame.
print(diabetes.columns)
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI',
'DiabetesPedigreeFunction', 'Age', 'Outcome'], dtype='object')
```

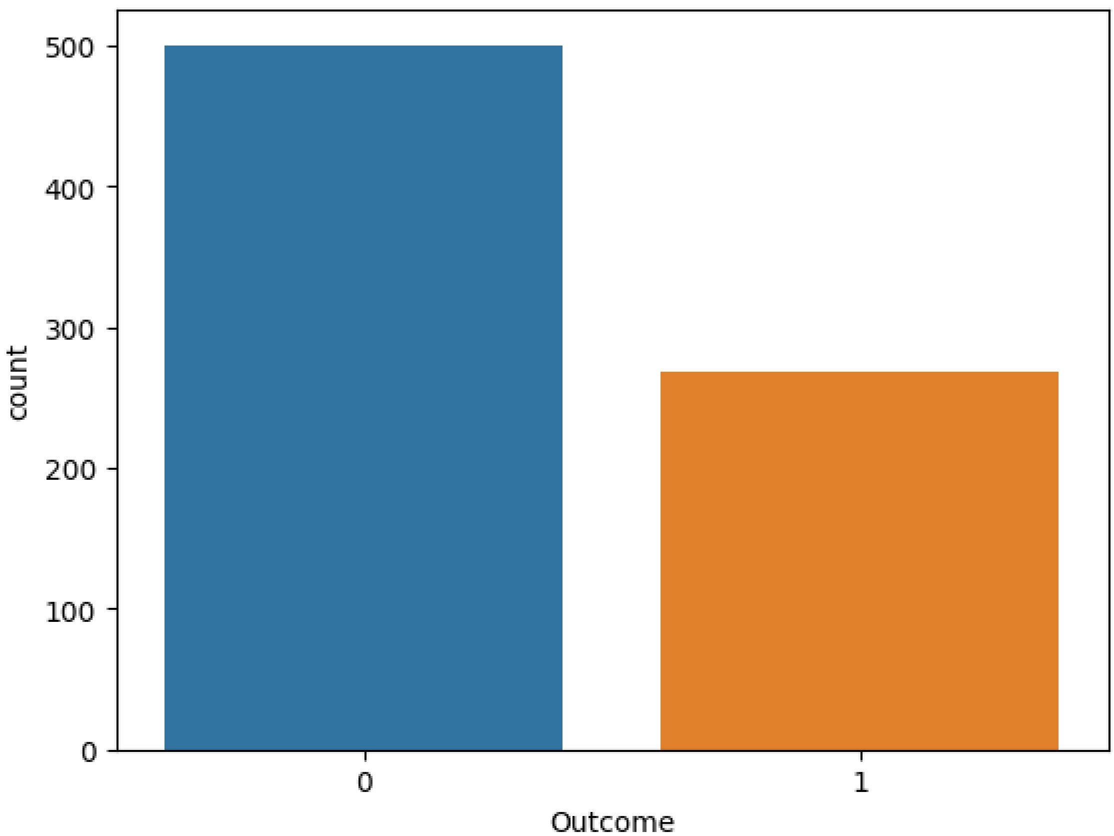
Target Analysis - Análisis del objetivo de la investigacion

In [ ]:
```
diabetes["Outcome"].value_counts()  # Outcome = resultado de si tiene o no la enfer
```

```
print(100 * diabetes["Outcome"].value_counts() / len(diabetes)) # formula para sabe

sns.countplot(x="Outcome", data=diabetes) # se grafica
plt.show()
```

```
Outcome
0   65.104
1   34.896
Name: count, dtype: float64
```

```
c:\Users\USUARIO\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_
oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be remov
ed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\USUARIO\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_
oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be remov
ed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\USUARIO\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_
oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be remov
ed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

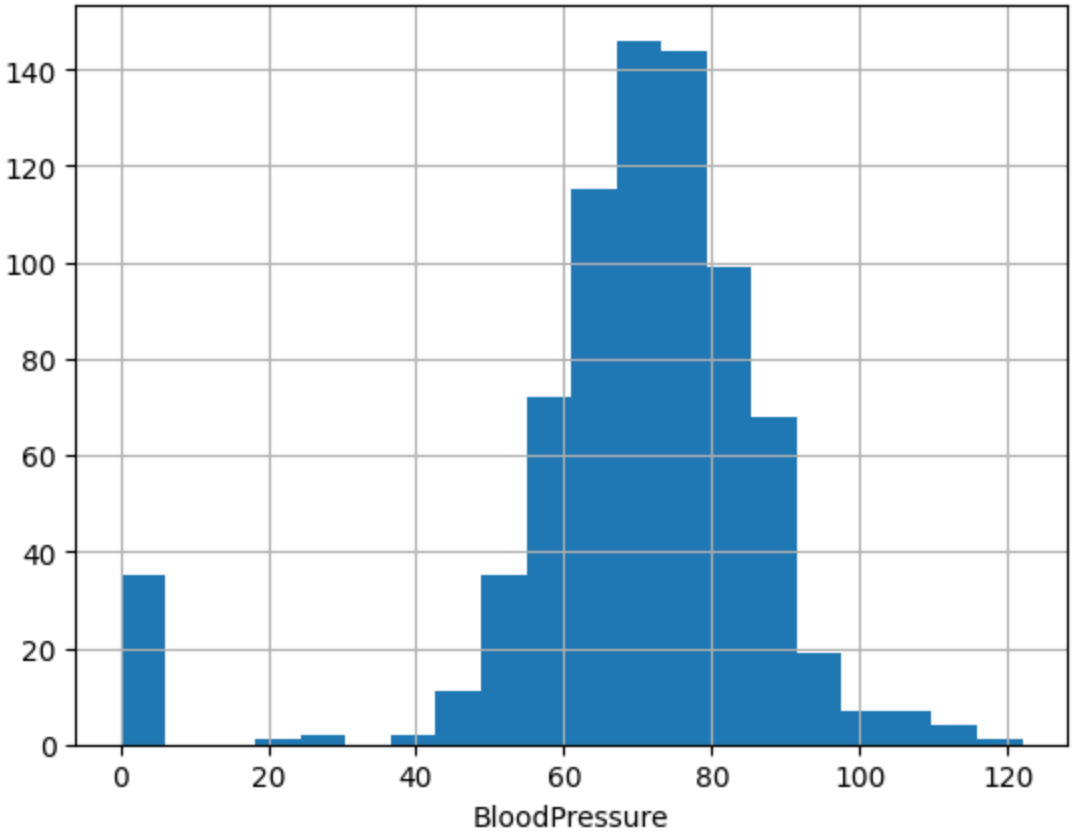


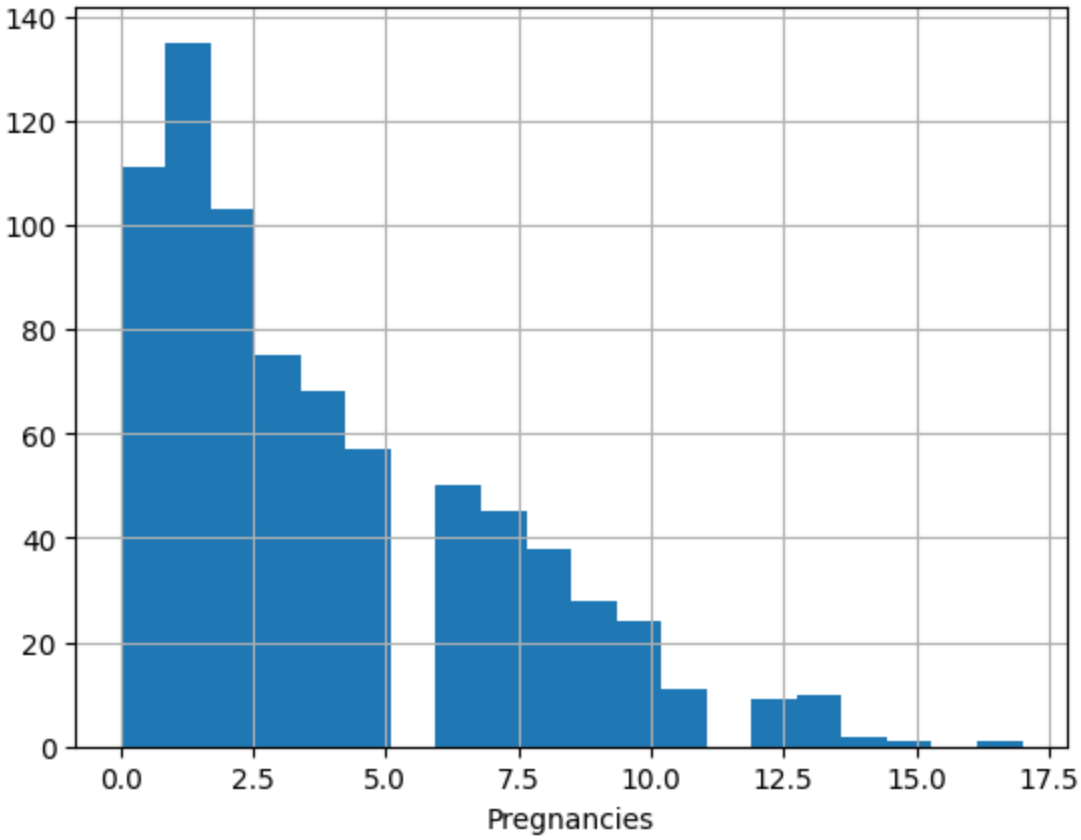
segun la grafica hay en la poblacion de indias pime 65.104 = 0 que no son diabeticas y 34.896 indias pime que padecen la enfermedad = 1

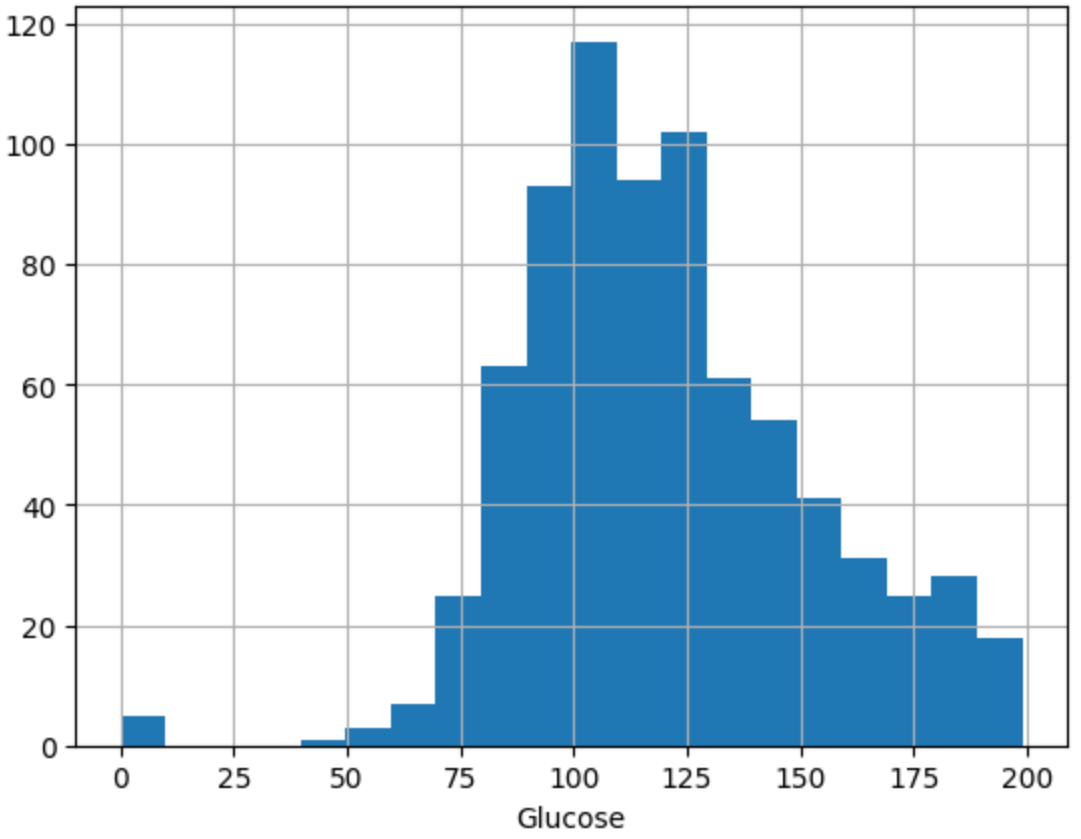
Feature'ların Analizi - Análisis de características

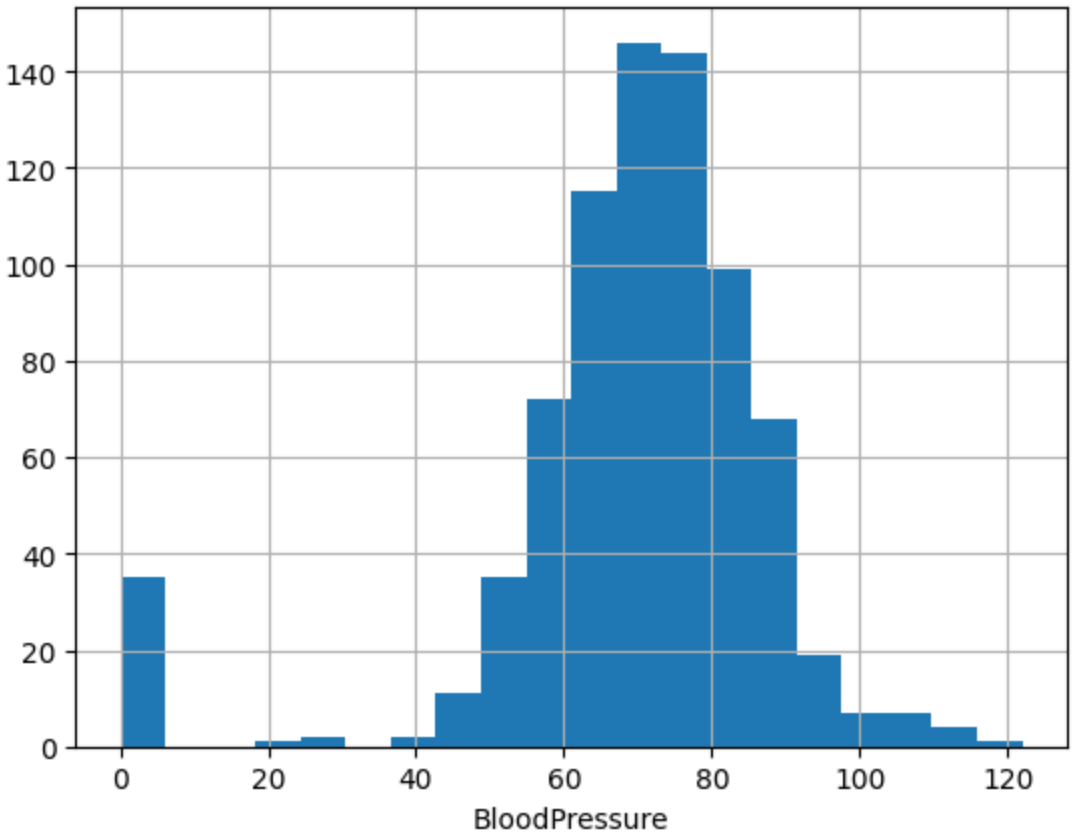
In [ ]:
```
diabetes["BloodPressure"].hist(bins=20)  # caracteristica BloodPressure = presión a
plt.xlabel("BloodPressure")
```

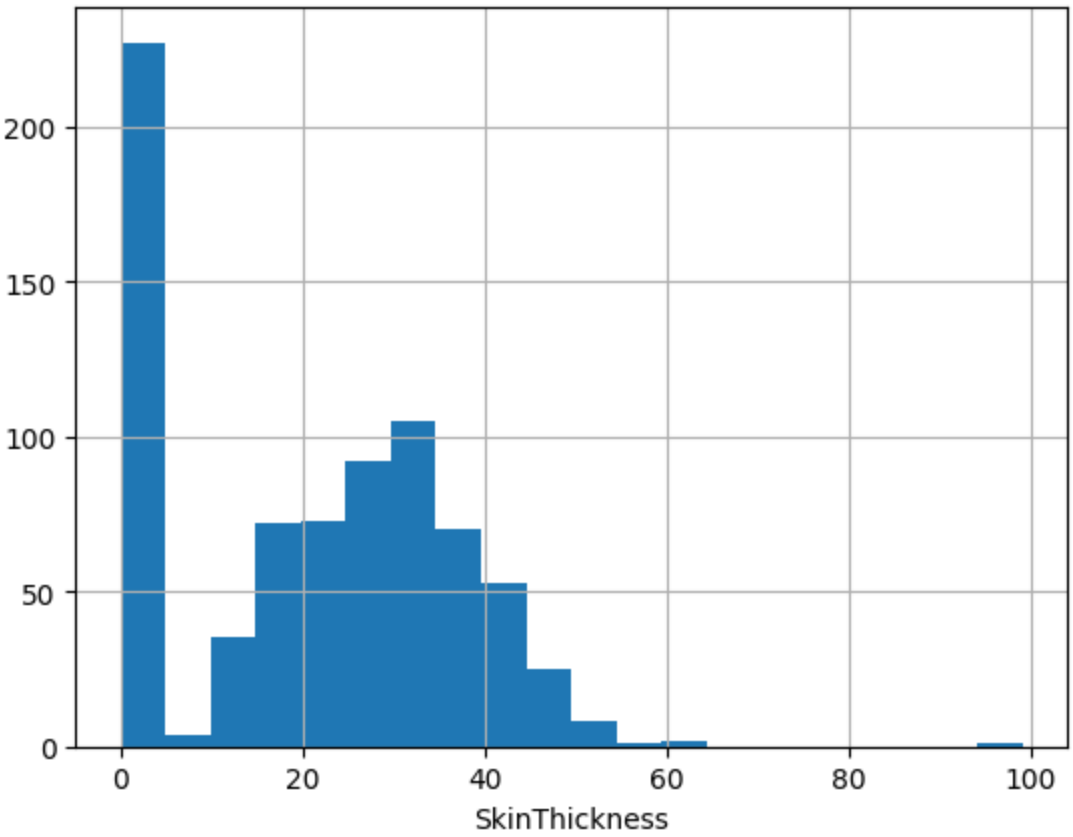
```
plt.show()

def plot_numerical_col(dataframe, numerical_col):  # se hace histogramacon todas la
    dataframe[numerical_col].hist(bins=20)
    plt.xlabel(numerical_col)
    plt.show()

for col in diabetes.columns:                  # se grafica
    plot_numerical_col(diabetes, col)
```

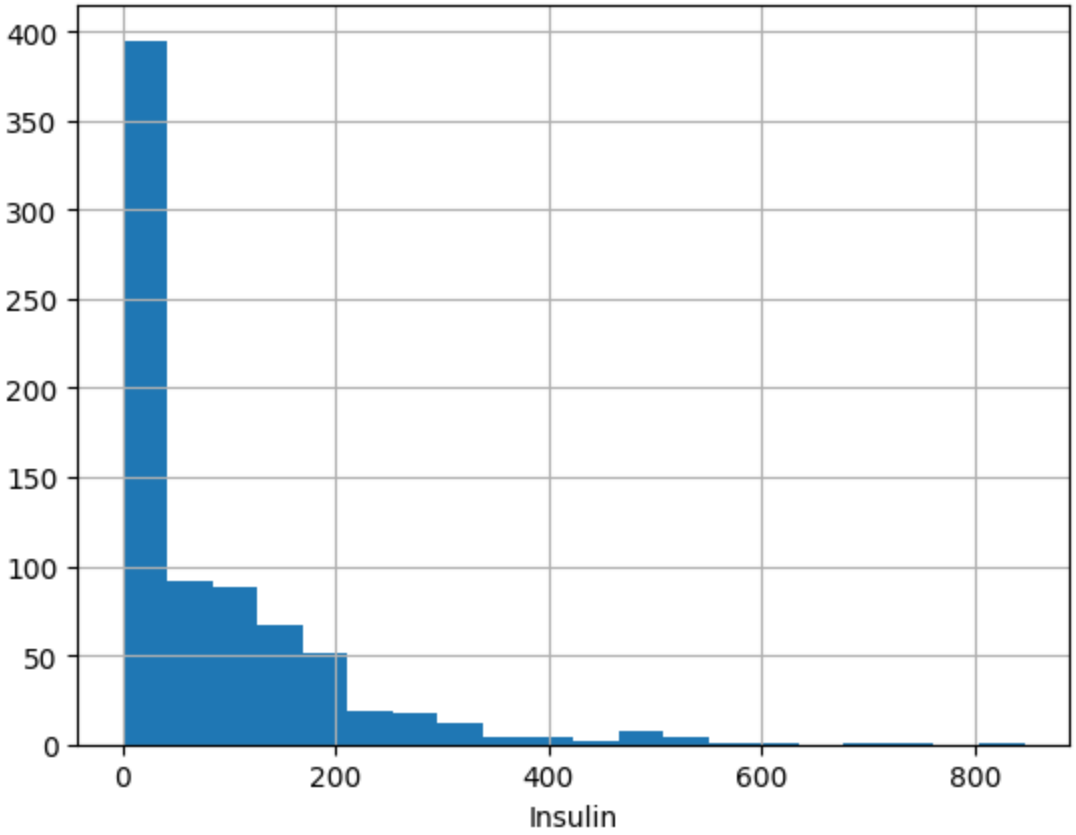


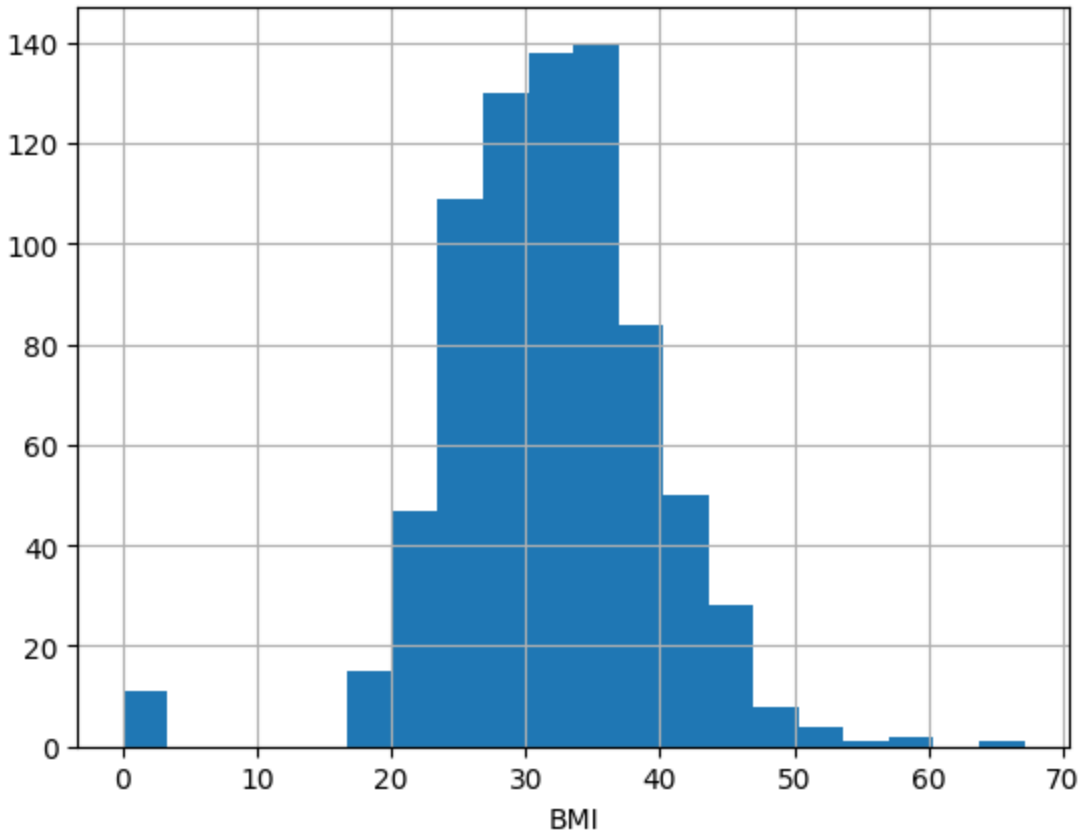

140
120
100
80
60
40
20
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Pregnancies

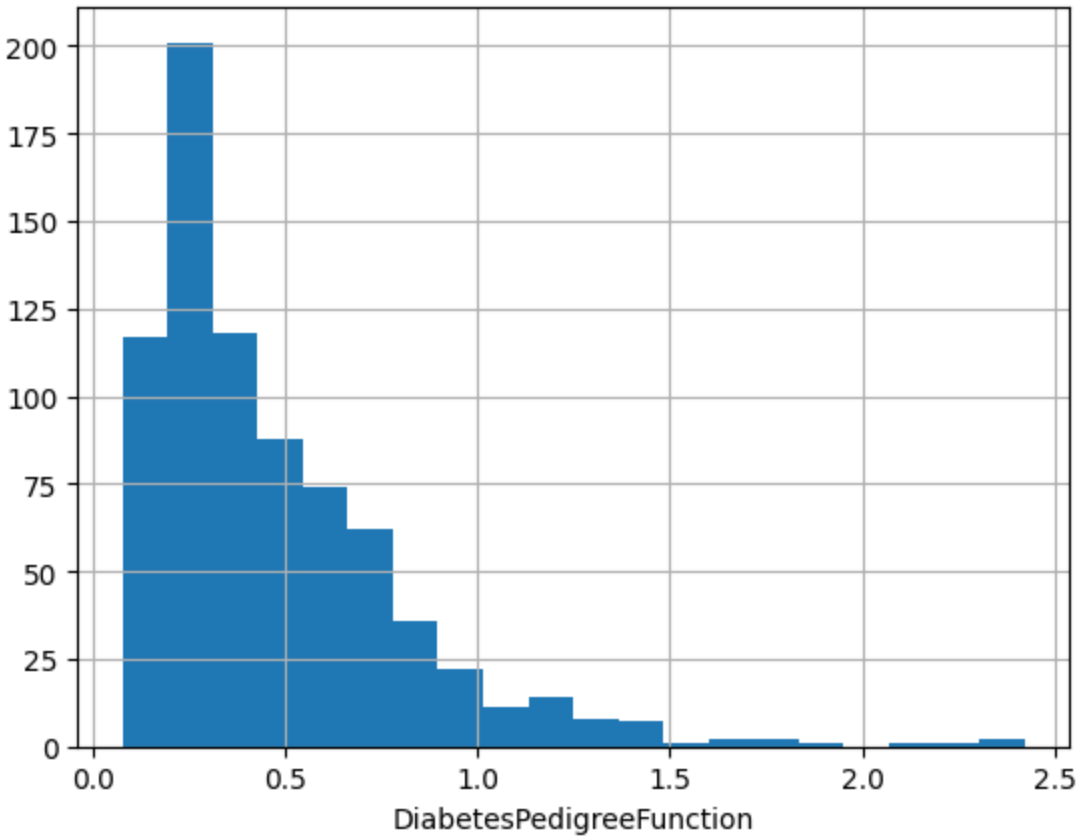

120
100
80
60
40
20
0
0
25
50
75
100
125
150
175
200
Glucose

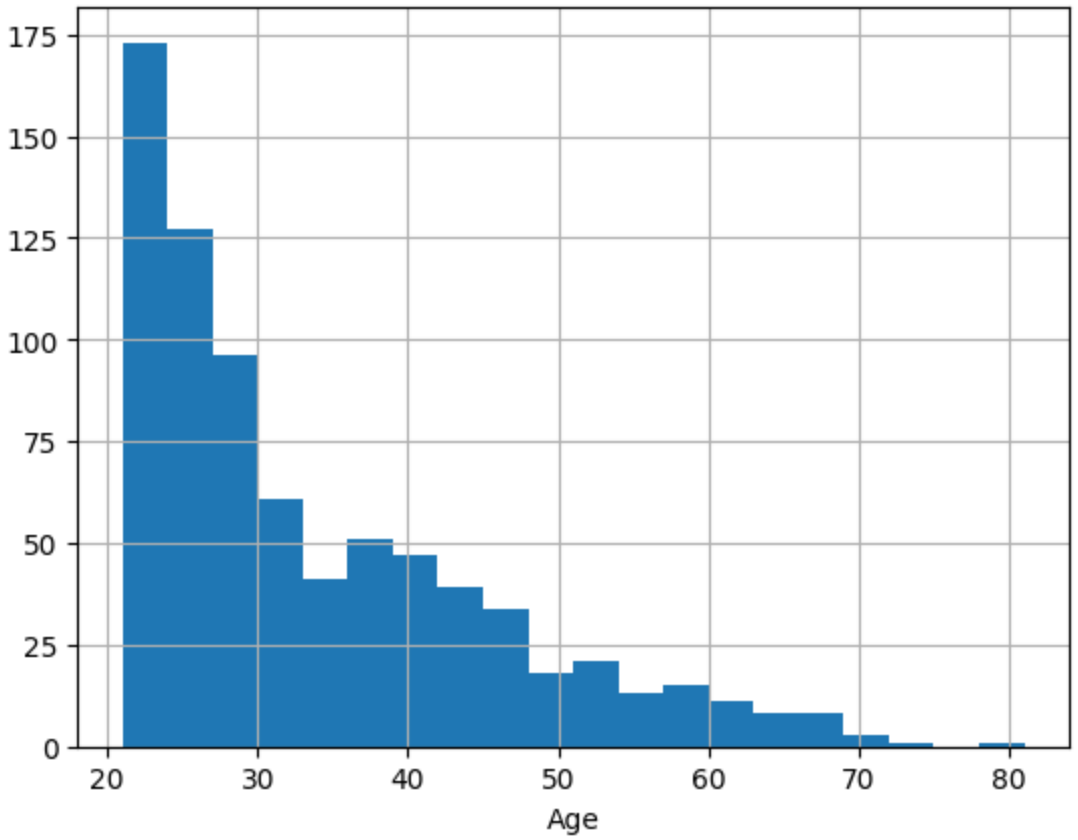

140
120
100
80
60
40
20
0
0
20
40
60
80
100
120
BloodPressure

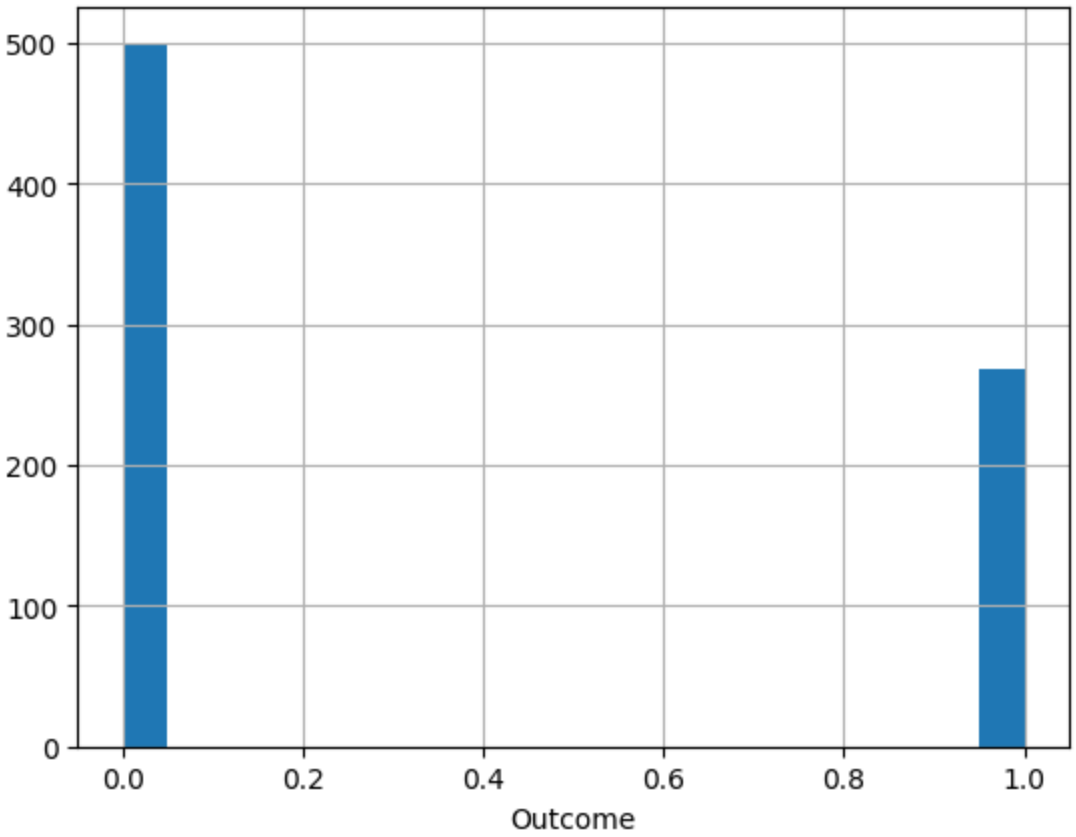

200
150
100
50
0
0
20
40
60
80
100
SkinThickness


400
350
300
250
200
150
100
50
0
0
200
400
600
800
Insulin


140
120
100
80
60
40
20
0
0
10
20
30
40
50
60
70
BMI


200
175
150
125
100
75
50
25
0
0.0
0.5
1.0
1.5
2.0
2.5
DiabetesPedigreeFunction


175
150
125
100
75
50
25
0
20
30
40
50
60
70
80
Age



In [ ]:
```
cols = [col for col in diabetes.columns if "Outcome" not in col]

# for col in cols:  se reviasn datos estadisticos de las caracteristicas o columnas
# plot_numerical_col(diabetes, col)

diabetes.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	m
Pregnancies	768.000	3.845	3.370	0.000	1.000	3.000	6.000	17.0
Glucose	768.000	120.895	31.973	0.000	99.000	117.000	140.250	199.0
BloodPressure	768.000	69.105	19.356	0.000	62.000	72.000	80.000	122.0
SkinThickness	768.000	20.536	15.952	0.000	0.000	23.000	32.000	99.0
Insulin	768.000	79.799	115.244	0.000	0.000	30.500	127.250	846.0
BMI	768.000	31.993	7.884	0.000	27.300	32.000	36.600	67.1
DiabetesPedigreeFunction	768.000	0.472	0.331	0.078	0.244	0.372	0.626	2.4
Age	768.000	33.241	11.760	21.000	24.000	29.000	41.000	81.0
Outcome	768.000	0.349	0.477	0.000	0.000	0.000	1.000	1.0

Target vs Features - Objetivo de la investigacion frente a las características

In [ ]:
```
diabetes.groupby("Outcome").agg({"Pregnancies" : "mean"}) # se agrupan los resultad

def target_summary_with_num(dataframe, target, numerical_col): # se define la funci
    print(dataframe.groupby(target).agg({numerical_col: "mean"}) , end="\n\n\n") #

for col in cols:         # se aplica el cilco for para todas las columnas
    target_summary_with_num(diabetes, "Outcome", col)
```

```
         Pregnancies
Outcome
0              3.298
1              4.866


         Glucose
Outcome
0        109.980
1        141.257


         BloodPressure
Outcome
0               68.184
1               70.825


         SkinThickness
Outcome
0               19.664
1               22.164


         Insulin
Outcome
0         68.792
1        100.336


           BMI
Outcome
0       30.304
1       35.143


         DiabetesPedigreeFunction
Outcome
0                           0.430
1                           0.550


           Age
Outcome
0       31.190
1       37.067
```

Pregnancies = mujeres india pime embarazadas segun los esultados 22.164 = 1 son diabeticas y 3.298 = 0 no padecen la enfermedad.

Glucose segun los resultados de las mujeres indias pime 141.257 = 1 son diabeticas, y 109.980 = 0 no padecen la enfermedad.

BloodPressure (Presión arterial) en mujeres indias pime segun el resultado 70.825 = 1 son diabeticas y 68.184 = 0 no padecen la enfermedad.

SkinThickness (Grosor de la piel) en mujeres indias pime segun el resultado 22.164 = 1 son diabeticas y 19.664 = 0 no padecen la enfermedad.

Insulin (insulina) en mujeres indias pime segun el resultado 100.336 = 1 son diabeticas y 68.792 = 0 no padecen la enfermedad.

BMI (IMC: indice de masa corporal) en mujeres indias pime segun el resultado 35.143 = 1 son diabeticas y 30.304 = 0 no padecen la enfermedad.

DiabetesPedigreeFunction (calcula la probabilidad de tener diabetes según la edad de las personas y sus antecedentes familiares diabéticos) en mujeres indias pime segun el resultado 0.550 = 1 son diabeticas y 0.430 = 0 no padecen la enfermedad.

Age (edad) en mujeres indias pime segun el resultado 37.067 = 1 son diabeticas y 31.190 = 0 no padecen la enfermedad.

Data Preprocessing (Preprocesamiento de datos)

In [ ]:
```
diabetes.shape # tamaño de los datos 768 filas y 9 columnas
```

Out[ ]:
```
(768, 9)
```

In [ ]:
```
diabetes.isnull().sum() # verificar si hay valores nulos con isnull y sum
```

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

In [ ]:
```
for col in cols:    # se sigue verificando datos
    print(col, ": ", check_outlier(diabetes, col))
```

```
Pregnancies :  False
Glucose :  False
BloodPressure :  False
SkinThickness :  False
Insulin :  True
BMI :  False
DiabetesPedigreeFunction :  False
Age :  False
Age :  False
```

In [ ]:
```
replace_with_thresholds(diabetes, "Insulin") # la funcion replace_with_thresholds e

for col in cols:  # ciclo for para transformar todas la columnas
    diabetes[col] = RobustScaler().fit_transform(diabetes[[col]])

diabetes.head() # imprimir resultado
```

```
C:\Users\USUARIO\AppData\Local\Temp\ipykernel_5688\1864853897.py:20: FutureWarning:
Setting an item of incompatible dtype is deprecated and will raise in a future error
of pandas. Value '-439.5' has dtype incompatible with int64, please explicitly cast
to a compatible dtype first.
  dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
```

Out[ ]:

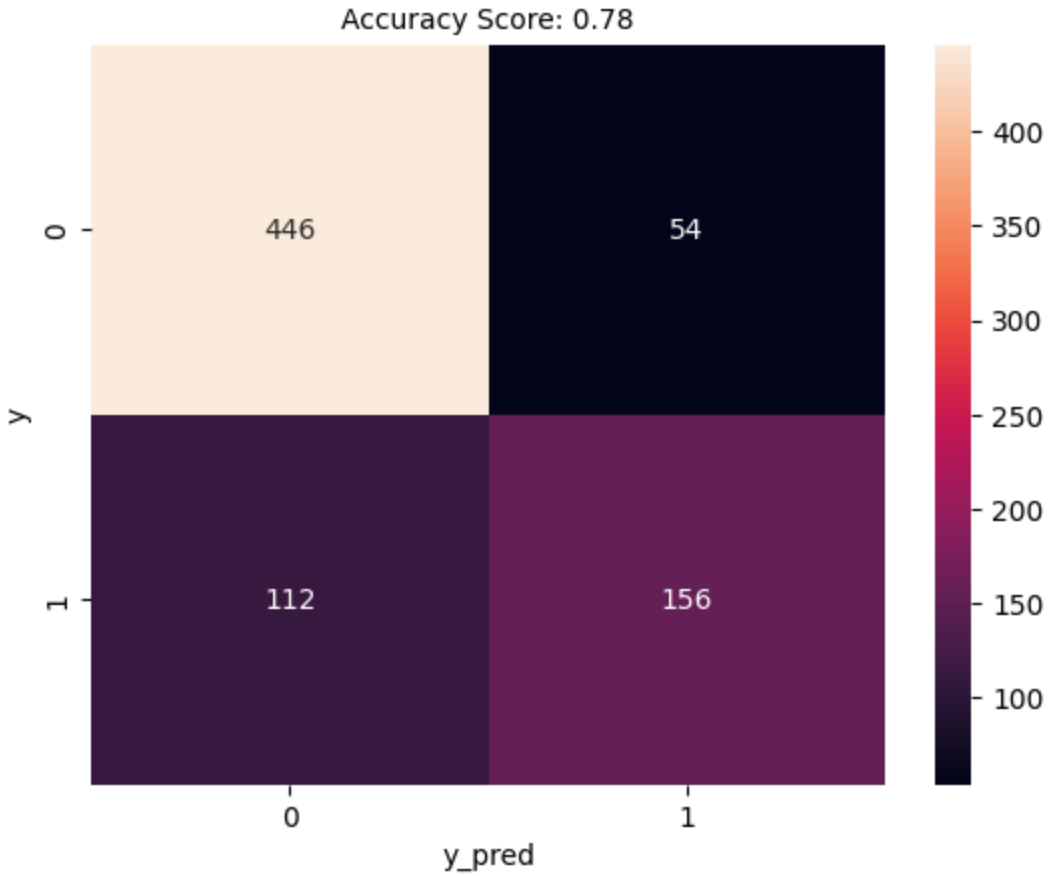
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
0	0.600	0.752	0.000	0.375	-0.240	0.172	
1	-0.400	-0.776	-0.333	0.188	-0.240	-0.581	
2	1.000	1.600	-0.444	-0.719	-0.240	-0.935	
3	-0.400	-0.679	-0.333	0.000	0.499	-0.419	
4	-0.600	0.485	-1.778	0.375	1.081	1.194	

Model & Prediction - Modelo y Prediccion

In [ ]:
```
y = diabetes["Outcome"]

X = diabetes.drop(["Outcome"], axis=1)

log_model = LogisticRegression().fit(X, y)

log_model.intercept_
log_model.coef_

y_pred = log_model.predict(X)
```

Model Evaluation - Evaluación del modelo

In [ ]:
```
def plot_confusion_matrix(y, y_pred):
    acc = round(accuracy_score(y, y_pred), 2)
    cm = confusion_matrix(y, y_pred)
    sns.heatmap(cm, annot=True, fmt=".0f")
    plt.xlabel("y_pred")
    plt.ylabel("y")
    plt.title("Accuracy Score: {0}".format(acc), size=10)
    plt.show()

plot_confusion_matrix(y, y_pred)

print(classification_report(y, y_pred))
```



	precision	recall	f1-score	support
0	0.80	0.89	0.84	500
1	0.74	0.58	0.65	268
accuracy			0.78	768
macro avg	0.77	0.74	0.75	768
weighted avg	0.78	0.78	0.78	768

Los valores de la diagonal principal 446 y 156 se corresponden con los valores estimados de forma correcta por el modelo, tanto los verdaderos positivos_ TP, como los verdaderos negativos_TN. La otra diagonal, por tanto, representa los casos en los que el modelo "se ha equivocado 112 falsos negativos_FN, 54 falsos positivos_FP.

Accuracy (Exactitud) macro avg (macropromedio) muestra el rendimiento promedio entre clases y trata cada clase como igualmente importante. weighted avg (promedio ponderado) la contribución de cada clase al promedio se pondera por su tamaño.

In [ ]:
```
# Accuracy (Exactitud): 0.78 de todos los ejemplos positivos reales que existen, ¿c
# correctamente que eran positivos?: 0.78 representa el porcentaje de predicciones

# Precision: 0.74 se refiere a lo cerca que está el resultado de una predicción del
# positivos previstos, ¿qué porcentaje es realmente positivo?.

# Recall (Recordar o sensibilidad): 0.58 representa la tasa de verdaderos positivos
# todos los ejemplos positivos reales que existen, ¿cuántos de ellos predije correc
```

```
# F1-score (Puntuación F1): 0.65 es la "media armónica" de precisión y sensibilidad
# el rendimiento de un modelo de clasificación..

# ROC Permite comparar diferentes modelos para identificar cual otorga mejor rendim
# AUC El área debajo de la curva puede ser utilizado como resumen de la calidad del
# rendimiento del modelo. Cuanto más esté hacia la izquierda la curva, más área hab
# ende, mejor será el clasificador.

y_prob = log_model.predict_proba(X)[:, 1]
roc_auc_score(y, y_prob)

# 0.83939
```

Out[ ]: 0.8393955223880598

Model Validation: Holdout - Modelo de Validacion

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_st

log_model = LogisticRegression().fit(X_train, y_train)

y_pred = log_model.predict(X_test)
y_prob = log_model.predict_proba(X_test)[:, 1]

# AUC
roc_auc_score(y_test, y_prob)
# 0.8755

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.77      0.92      0.84        97
           1       0.79      0.53      0.63        57

    accuracy                           0.77       154
   macro avg       0.78      0.72      0.73       154
weighted avg       0.78      0.77      0.76       154
```

In [ ]:
```
# Before (Antes)
# Accuracy (Exactitud): 0.78
# Precision: 0.74
# Recall (Recordar o sensibilidad): 0.58
# F1-score (Puntuación F1): 0.65

# After (después)
# Accuracy (Exactitud): 0.77
# Precision: 0.79
# Recall (Recordar sensibilidad): 0.53
# F1-score (Puntuación F1): 0.63
```

In [ ]:
```
# Model Validation: 10-Fold Cross Validation - Validación cruzada de 10 veces
```

```
y = diabetes["Outcome"]

X = diabetes.drop("Outcome", axis=1)

log_model = LogisticRegression().fit(X, y)

cv_results = cross_validate(log_model, X, y,
                            cv=5,
                            scoring=["accuracy", "precision", "recall", "f1", "roc_a
```

In [ ]:
```
# Before (antes)
# Accuracy Exactitud: 0.78
# Precision: 0.74
# Recall (Recordar o sensibilidad): 0.58
# F1-score (puntuación F1): 0.65

# After (después)
# Accuracy (Exactitud): 0.77
# Precision: 0.79
# Recall (Recordar o sensibilidad): 0.53
# F1-score (puntuación F1): 0.63

# Validation After (Validacion antes)

cv_results["test_accuracy"].mean() # test de Exactitud com la media
# Accuracy: 0.77

cv_results["test_precision"].mean() # test de Precision com la media
# Precision: 0.71

cv_results["test_recall"].mean()  # test de Recordar o sensibilidad com la media
# Recall: 0.57

cv_results["test_f1"].mean()     # test de puntuación F1 com la media
# F1-score: 0.63

cv_results["test_roc_auc"].mean() # test de el modelo y su rendimiento com la media
# ROC AUC: 0.83
```

Out[ ]:
```
0.8327295597484277
```

Prediction for A New Observation - Predicción para una nueva observación

In [ ]:
```
X.columns

random_user = X.sample(1, random_state=45)
log_model.predict(random_user)
```

Out[ ]:
```
array([1], dtype=int64)
```