PROGRAMACIÓN PARA CIENCIAS DE DATOS 2

Juan Fernando Rodríguez Medina

https://media.istockphoto.com/id/1364317541/es/foto/cient%C3%ADficos-de-datos-mano-del-programador-tocando-y-analizando-el-desarrollo-en-diversas.jpg?s=1024x1024&w=is&k=20&c=ydt_7TNUweIb8ebSsDQUybI59ABLI7JriD99NgU-YdQ=

1. Defina un problema específico dentro del campo elegido que pueda abordarse mediante el análisis de datos. Explica el problema y por qué vale la pena investigarlo.

> En el ámbito del sector salud hay muchas problemáticas que afectan a la población, uno de los asuntos más concernientes es que cada año hay una incidencia mayor de enfermedades crónicas que aumentan en gran medida el gasto público, sin embarg, según muchas revistas médicas en casi todos los casos de las enfermedades crónicas éstas de haberse tratado a tiempo se habría podido evitar la enfermedad final o un peor desenlace, en este trabajo se realizarán la comparativa de una serie de relaciones con respecto a la diabetes en una población de un hospital entre algunas de las comparaciones que se harán estás serán la siguientes
>
> 1. ¿Qué relación hay entre el ser diabético (outcome 1 o 0) y el BMI (índice de masa corporal)
> 2. ¿Qué relación hay entre el BMI y la edad?
> 3. ¿Qué relación hay entre el BMI y la diabetes pedigree function (tener diabetes según la historia familiar) Una vez establecida la relación anterior, estudiar qué tan probable es padecer de diabetes dada esta relación puede ser crucial para tener una serie de controles médicos en estas personas para que el riesgo de diabetes sea menor.

1. ¿Cómo el análisis de datos podría contribuir a comprender o resolver el problema definido? ¿Qué ideas espera obtener?

> El análisis de datos permite através de herramientas estadísticas y lenguajes de programación establecer una serie de evidencias reproducibles y contextualizadas, que ayuden a resolver problemas mediante hipótesis y procedimientos científicos matemáticos. De esta manera no solo se resuelve el problema, se puede llegar a realizar predicciones que orienten directrices preventivas hacia lo cual, este proyecto está dirigido.
>
> En el análisis a las preguntas (hipótesis) que he propuesto, espero poder hacer los primeros "scratch" a una problemática interesante en sector salud que pueda ser de ayuda para evitar posibles casos de diabetes que se podrían haber evitado, siendo así una reducción de costos en el sector salud y mejoramiento de calidad de vida de muchos posibles pacientes diabéticos.

In [68]:
```
# Importamos la librerias que pueden ser útiles en el trabajo y las nombramos
# Importamos la base de datos que está en formato .csv

import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.metrics import confusion_matrix
from sklearn import tree
diabetes= pd.read_csv(filepath_or_buffer="D:/Bases_de_datos/input/diabetes.csv")
```

In [69]:
```
diabetes.head(5)
```

Out[69]:

	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

In [70]:
```
diabetes.BloodPressure
```

Out[70]:
```
0      72
1      66
2      64
3      66
4      40
       ..
763    76
764    70
765    72
766    60
767    70
Name: BloodPressure, Length: 768, dtype: int64
```

In [71]:
```
diabetes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [72]:
```
diabetes.dropna(inplace=True)
diabetes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [73]:
```
# Revisamos si la base de datos tiene subcategorías que sean datos con solo 1 variable, lo que significaría que esa variable no es de interés p
```

In [74]:
```
cols_cat = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']

for col in cols_cat:
    print(f'Columna {col}: {diabetes[col].nunique()} subniveles')
```

```
Columna Pregnancies: 17 subniveles
Columna Glucose: 136 subniveles
Columna BloodPressure: 47 subniveles
Columna SkinThickness: 51 subniveles
Columna Insulin: 186 subniveles
Columna BMI: 248 subniveles
Columna DiabetesPedigreeFunction: 517 subniveles
Columna Age: 52 subniveles
Columna Outcome: 2 subniveles
```

3. Realiza un cálculo de al menos dos medidas estadísticas descriptivas clave (por ejemplo, media, mediana, desviación estándar) utilizando NumPy. ¿Qué información te proporcionan estas medidas sobre los datos?

In [75]:
```
# media mediana y moda con numpy

print("Media: ",np.mean(diabetes))
print("Mediana: ",np.median(diabetes))
# numpy parece que no tiene el módulo para sacar la moda de un df "print("Moda: ",np.mode(diabetes))"
print("Desviación Estándar: ",np.std(diabetes))
print("Varianza: ",np.var(diabetes))
```

```
Media:  Pregnancies                   3.845052
Glucose                     120.894531
BloodPressure                69.105469
SkinThickness                20.536458
Insulin                      79.799479
BMI                          31.992578
DiabetesPedigreeFunction      0.471876
Age                          33.240885
Outcome                       0.348958
dtype: float64
Mediana:  24.3
Desviación Estándar:  Pregnancies                  3.367384
Glucose                      31.951796
BloodPressure                19.343202
SkinThickness                15.941829
Insulin                     115.168949
BMI                           7.879026
DiabetesPedigreeFunction      0.331113
Age                          11.752573
Outcome                       0.476641
dtype: float64
Varianza:  Pregnancies                  11.339272
Glucose                      1020.917262
BloodPressure                 374.159449
SkinThickness                 254.141900
Insulin                     13263.886875
BMI                            62.079046
DiabetesPedigreeFunction        0.109636
Age                           138.122964
Outcome                         0.227186
dtype: float64
c:\Users\USUARIO\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3438: FutureWarning: In a future version, DataFrame.mean(axis=None) will return a scalar mean over the entire DataFrame. To retain the old behavior, use 'frame.mean(axis=0)' or just 'frame.mean()'
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
```

> La información que me proporciona y que resulta más interesante la desviación estándar la cual para Diabetes Pedigree function es mínima se explica porque de media los valores son muy pequeños y en los extremos la variación es mínima, y por otra parte vemos que en datos como la glucosa la desviación estándar es muy alta, lo cual se explica porque los no diabéticos y los diabéticos tienen una diferencia marcada, sin embargo hay que mencionar que en estos valores existen valores incoherentes como podrían ser los 0 y valores por encima de los 900, cuando en la teoría se describen hasta 200mg/dl de glucosa en sangre, es decir existen valores muy discordantes para el dataframe conseguido

In [76]:
```
# Sacando la mediana, media, moda, mínimo, máximo desviación estándar y los cuartiles usando simplemente la función .describe de pandas

diabetes.describe()
```

Out[76]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

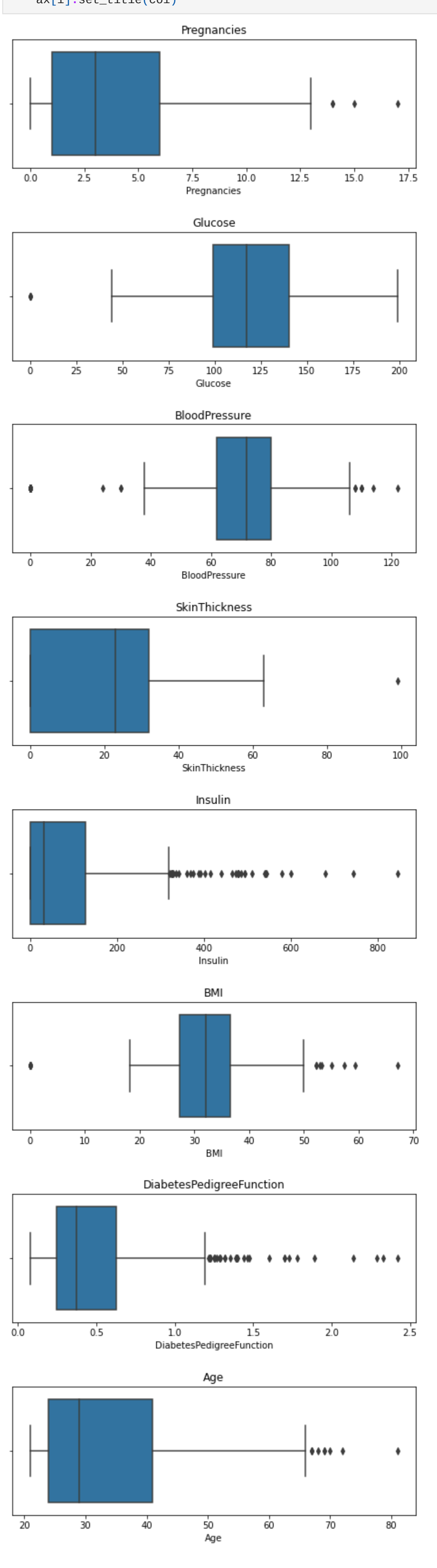
In [77]:
```
# la desviación estándar nos indica que hay valores dispersos en las columnas
```

In [78]:
```
print(f'Tamaño del set antes de eliminar las filas repetidas: {diabetes.shape}')
diabetes.drop_duplicates(inplace=True)
print(f'Tamaño del set después de eliminar las filas repetidas: {diabetes.shape}')
```

```
Tamaño del set antes de eliminar las filas repetidas: (768, 9)
Tamaño del set después de eliminar las filas repetidas: (768, 9)
```

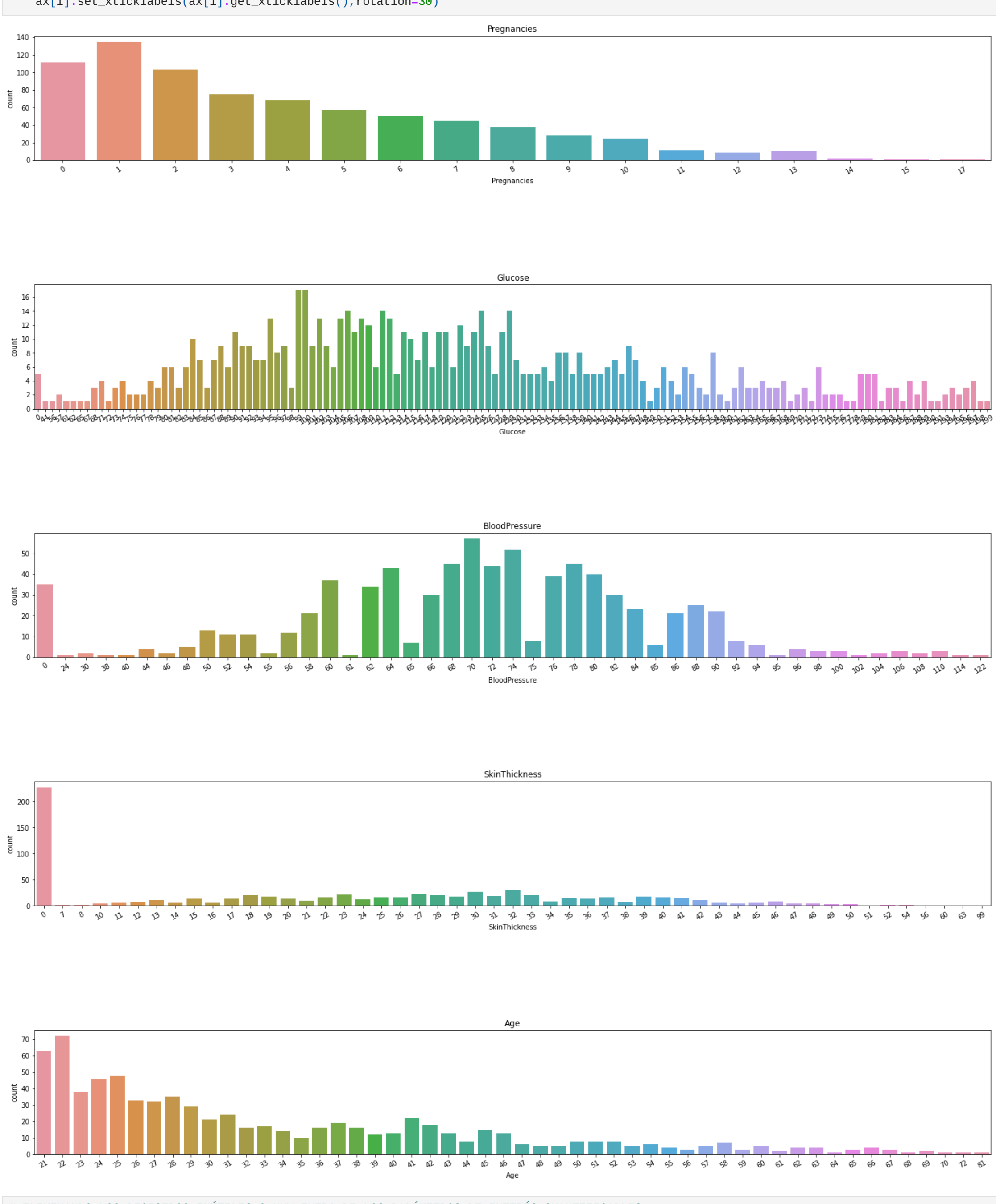
4. Cree una visualización gráfica adecuada para explorar los datos. Puede ser un histograma, un gráfico de dispersión u otro tipo relevante. Adjunta la visualización y explica cómo esta visualización puede ayudarle a abordar el problema inicial.

In [79]:
```
# Generar diagramas de caja individuales pues las variables numéricas
# están en rangos diferentes
cols_num = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age']

fig, ax = plt.subplots(nrows=8, ncols=1, figsize=(8,30))
fig.subplots_adjust(hspace=0.5)

for i, col in enumerate(cols_num):
    sns.boxplot(x=col, data=diabetes,ax=ax[i])
    ax[i].set_title(col)
```



> Los gráficos y precisamente para el caso propuesto, el diagrama de caja ayuda a entender fácilmente en qué valores se encuentran los datos según el campo, y por lo tanto permite hacer un análisis general de cómo están distribuidos los datos y qué tantos datos atípicos existen, es decir todos los datos que salen de los bigotes. Por otra parte también indicará los datos que se encuentran dentro de la media, y qué tan homogéneos o no, son los datos.

In [80]:
```
# Graficar los subniveles de cada variable categórica en forma de histograma
cols_cat =  ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Age']

fig, ax = plt.subplots(nrows=5, ncols=1, figsize=(25,30))
fig.subplots_adjust(hspace=1)

for i, col in enumerate(cols_cat):
    sns.countplot(x=col, data=diabetes, ax=ax[i])
    ax[i].set_title(col)
    ax[i].set_xticklabels(ax[i].get_xticklabels(),rotation=30)
```



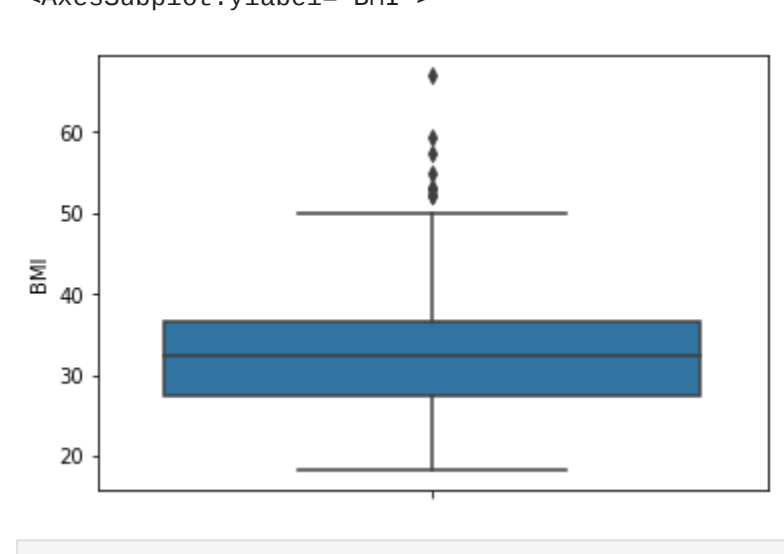
In [81]:
```
# ELIMINANDO LOS REGISTROS INÚTILES O MUY FUERA DE LOS PARÁMETROS DE INTERÉS CUANTIFICABLES
```

In [82]:
```
print(f'Tamaño del set antes de eliminar registros de BMI: {diabetes.shape}')
data = diabetes[diabetes['BMI']>10]
print(f'Tamaño del set después de eliminar registros de BMI: {data.shape}')
```

```
Tamaño del set antes de eliminar registros de BMI: (768, 9)
Tamaño del set después de eliminar registros de BMI: (757, 9)
```

In [83]:
```
sns.boxplot(y="BMI",data=data)
```

Out[83]: `<AxesSubplot:ylabel='BMI'>`

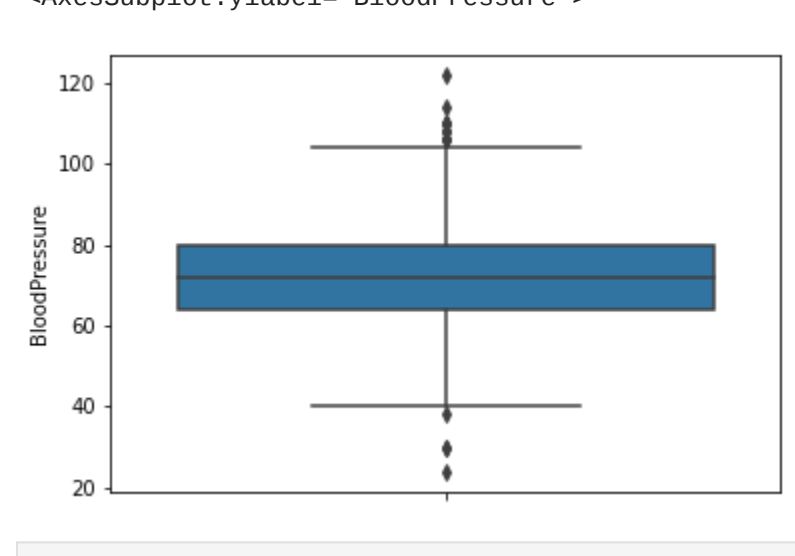


In [84]:
```
print(f'Tamaño del set antes de eliminar registros de Presión Sanguínea: {diabetes.shape}')
data1 = diabetes[diabetes['BloodPressure']>=24]
print(f'Tamaño del set después de eliminar registros de Presión Sanguínea: {data1.shape}')
```

```
Tamaño del set antes de eliminar registros de Presión Sanguínea: (768, 9)
Tamaño del set después de eliminar registros de Presión Sanguínea: (733, 9)
```

In [85]:
```
sns.boxplot(y="BloodPressure",data=data1)
```

Out[85]: `<AxesSubplot:ylabel='BloodPressure'>`

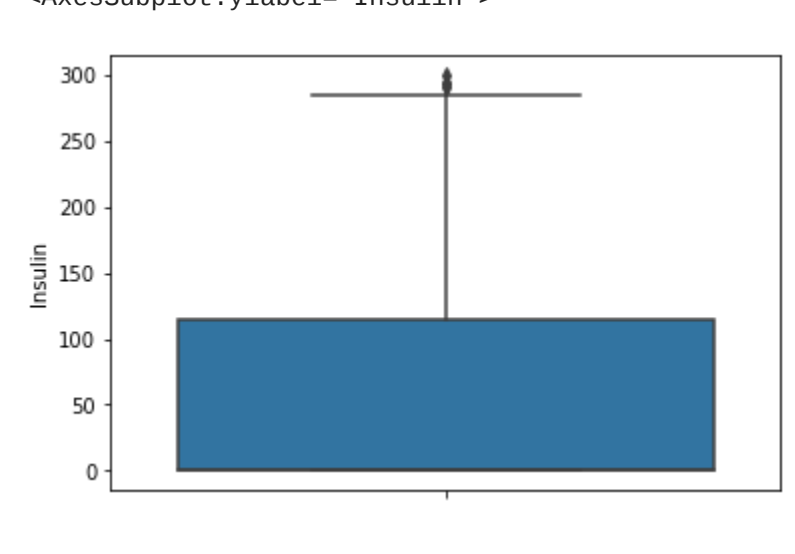


In [86]:
```
print(f'Tamaño del set antes de eliminar registros de Insulina: {diabetes.shape}')
data2 = diabetes[diabetes['Insulin']<=300]
print(f'Tamaño del set después de eliminar registros de Insulina: {data2.shape}')
```

```
Tamaño del set antes de eliminar registros de Insulina: (768, 9)
Tamaño del set después de eliminar registros de Insulina: (731, 9)
```

In [87]:
```
sns.boxplot(y="Insulin",data=data2)
```

Out[87]: `<AxesSubplot:ylabel='Insulin'>`



5. Teniendo en cuenta los resultados del análisis estadístico y la visualización. ¿Qué patrones o tendencias iniciales puedes observar? ¿Hay algún valor atípico o característica inesperada en los datos?

> En el análisis exploratorio que se ha hecho con los datos através de los gráficos de caja de bigotes,vemos que hay en la mayoría de los parámetros muchos datos excéntricos que pueden hacer que futuros procedimientos estadísticos presenten sesgos, por lo tanto enumeraré los pasos a seguir
>
> 1. Eliminar datos que están muy fuera de los bigotes o no tienen coherencia con los el tipo de los datos, por ejemplo, no tienen sentido pacientes con 0 de presión sanguínea si todos son pacientes con vida, lo mismo se podría decir con los valores que tienen 0 en grosor de piel entre otros.
> 2. Hay una relación entre tener diabetes y el IMC (o BMI en inglés)
> 3. Para algunos campos como la glucosa en sangre hay una cantidad de subniveles tan grande que valdría la pena categorizarlos en secciones

6. Seleccione dos variables que podrían estar relacionadas dentro del contexto del problema. Convierta las series correspondientes a estas variables en vectores de NumPy e implemente una regresión lineal utilizando numpy.polyfit().

> 1. Outcome + DPF
> 2. Outcome + Age
> 3. Outcome + BMI

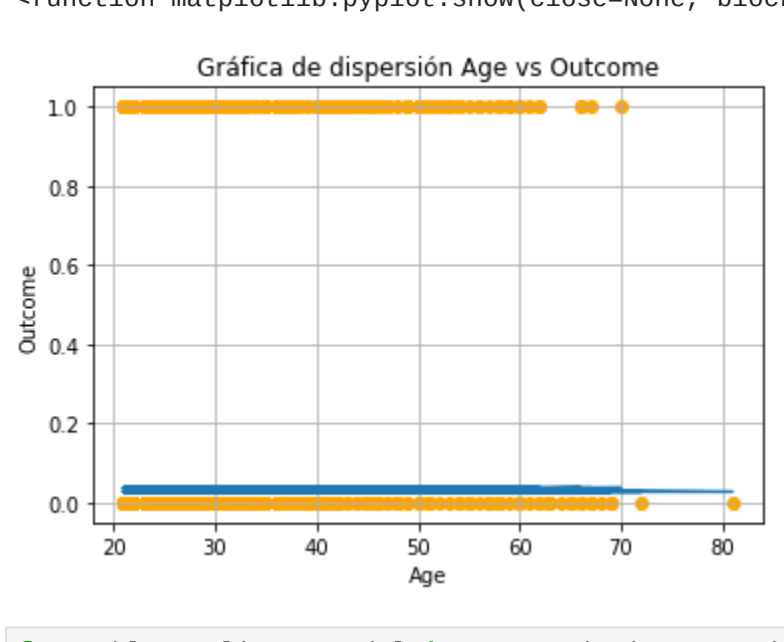
7. Grafica las variables que has seleccionado usando un gráfico de dispersión y añade un gráfico lineal determinado por los coeficientes de la relación de regresión.

In [88]:
```
b1, b0 = np.polyfit(diabetes["Age"],diabetes["Outcome"],1)
print(b1,b0)
```

```
0.009666834703521698 0.02762418861171164
```

In [89]:
```
plt.scatter(diabetes["Age"],diabetes["Outcome"],c="orange")
plt.plot(diabetes["Age"], b0+b1*diabetes["Outcome"])
plt.title("Gráfica de dispersión Age vs Outcome")
plt.xlabel("Age")
plt.ylabel("Outcome")
plt.grid()
plt.show
```

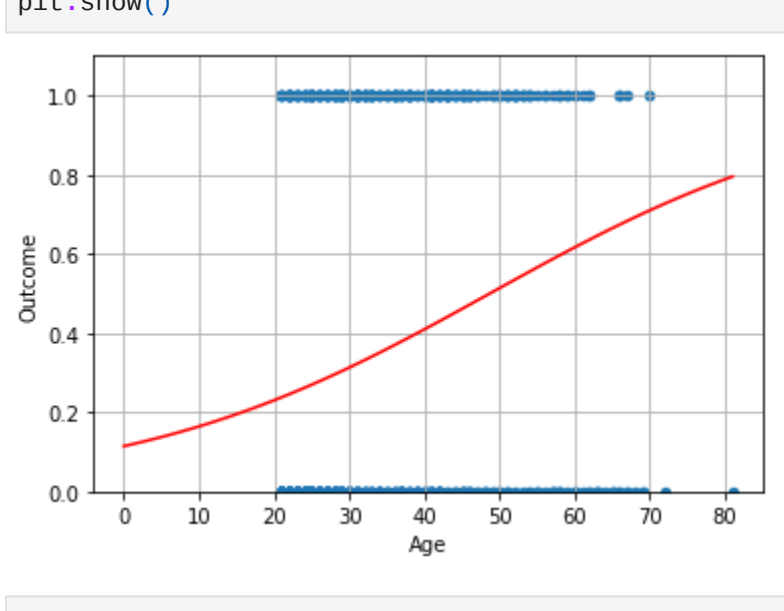
Out[89]: `<function matplotlib.pyplot.show(close=None, block=None)>`



In [90]:
```
from sklearn.linear_model import LogisticRegression

# definiendo input y output
X_train = np.array(diabetes['Age']).reshape((-1, 1))
Y_train = np.array(diabetes['Outcome'])

# creando modelo
model = LogisticRegression()
model.fit(X_train, Y_train)

# imprimiendo parametros
print(f"intercepto (b): {model.intercept_}")
print(f"pendiente (w): {model.coef_}")
```

```
intercepto (b): [-2.04744865]
pendiente (w): [[0.04202466]]
```

In [91]:
```
w = 0.04202466
b = -2.04744865
# puntos de la recta
x = np.linspace(0,diabetes['Age'].max(),100)
y = 1/(1+np.exp(-(w*x+b)))

# grafica de la recta
diabetes.plot.scatter(x='Age',y='Outcome')
plt.plot(x, y, '-r')
plt.ylim(0,diabetes['Outcome'].max()*1.1)
# plt.grid()
plt.grid()
plt.show()
```

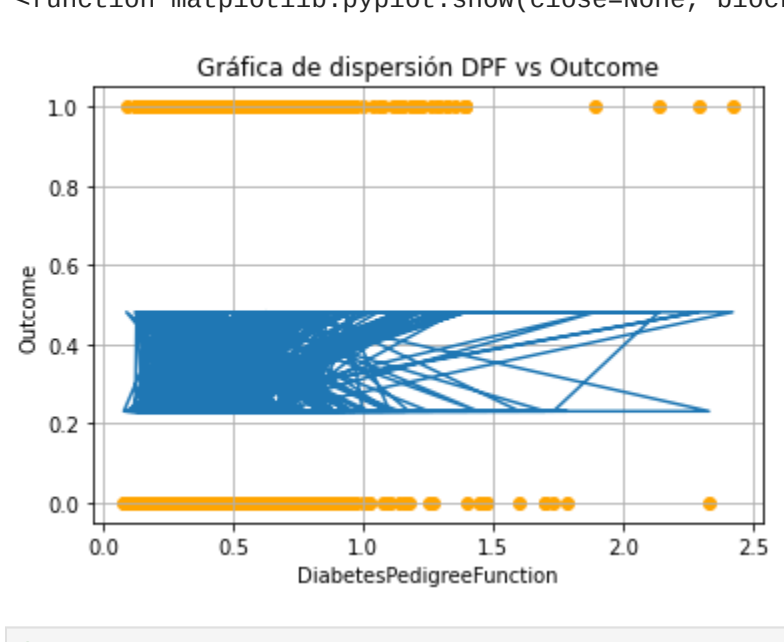


In [92]:
```
b1, b0 = np.polyfit(diabetes["DiabetesPedigreeFunction"],diabetes["Outcome"],1)
print(b1,b0)
```

```
0.250250560279803 0.2308710243542176
```

In [93]:
```
plt.scatter(diabetes["DiabetesPedigreeFunction"],diabetes["Outcome"],c="orange")
plt.plot(diabetes["DiabetesPedigreeFunction"], b0+b1*diabetes["Outcome"])
plt.title("Gráfica de dispersión DPF vs Outcome")
plt.xlabel("DiabetesPedigreeFunction")
plt.ylabel("Outcome")
plt.grid()
plt.show
```

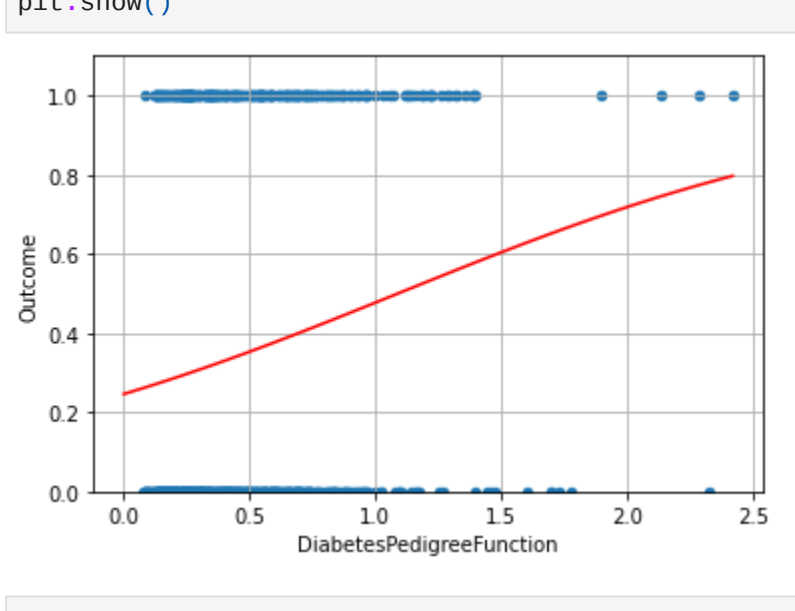
Out[93]: `<function matplotlib.pyplot.show(close=None, block=None)>`



In [94]:
```
from sklearn.linear_model import LogisticRegression

# definiendo input y output
X_train = np.array(diabetes['DiabetesPedigreeFunction']).reshape((-1, 1))
Y_train = np.array(diabetes['Outcome'])

# creando modelo
model = LogisticRegression()
model.fit(X_train, Y_train)

# imprimiendo parametros
print(f"intercepto (b): {model.intercept_}")
print(f"pendiente (w): {model.coef_}")
```

```
intercepto (b): [-1.12000907]
pendiente (w): [[1.02697365]]
```

In [95]:
```
w = 1.02697365
b = -1.12000907
# puntos de la recta
x = np.linspace(0,diabetes['DiabetesPedigreeFunction'].max(),100)
y = 1/(1+np.exp(-(w*x+b)))

# grafica de la recta
diabetes.plot.scatter(x='DiabetesPedigreeFunction',y='Outcome')
plt.plot(x, y, '-r')
plt.ylim(0,diabetes['Outcome'].max()*1.1)
plt.grid()
# plt.grid()
plt.show()
```

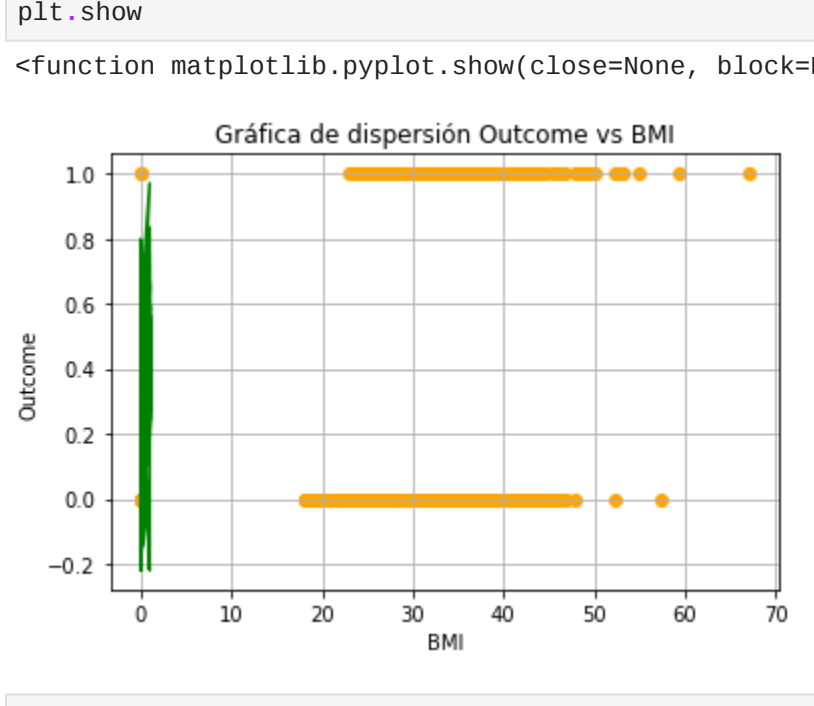


In [96]:
```
b1, b0 = np.polyfit(diabetes["BMI"],diabetes["Outcome"],1)
print(b1,b0)
```

```
0.01770653015503836 -0.2175192159743995
```

In [97]:
```
plt.scatter(diabetes["BMI"],diabetes["Outcome"],c="orange")
plt.plot(diabetes["Outcome"], b0+b1*diabetes["BMI"], c="green")
plt.title("Gráfica de dispersión Outcome vs BMI")
plt.xlabel("BMI")
plt.ylabel("Outcome")
plt.grid()
plt.show
```

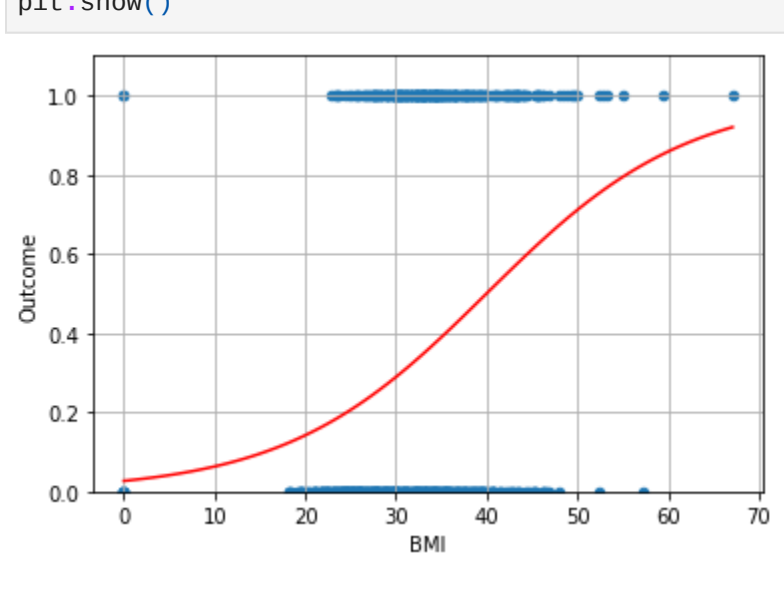
Out[97]: `<function matplotlib.pyplot.show(close=None, block=None)>`



In [98]:
```
from sklearn.linear_model import LogisticRegression

# definiendo input y output
X_train = np.array(diabetes['BMI']).reshape((-1, 1))
Y_train = np.array(diabetes['Outcome'])

# creando modelo
model = LogisticRegression()
model.fit(X_train, Y_train)

# imprimiendo parametros
print(f"intercepto (b): {model.intercept_}")
print(f"pendiente (w): {model.coef_}")
```

```
intercepto (b): [-3.68596089]
pendiente (w): [[0.09351691]]
```

In [99]:
```
w = 0.09
b = -3.6
# puntos de la recta
x = np.linspace(0,diabetes['BMI'].max(),100)
y = 1/(1+np.exp(-(w*x+b)))

# grafica de la recta
diabetes.plot.scatter(x='BMI',y='Outcome')
plt.plot(x, y, '-r')
plt.ylim(0,diabetes['Outcome'].max()*1.1)
# plt.grid()
plt.grid()
plt.show()
```



8. Considerando la calidad del ajuste de la regresión lineal a los datos, ¿crees que este modelo es suficiente para describir completamente la relación entre las variables? ¿Qué podrías hacer para mejorar la precisión de la predicción?

> No es suficiente la regresión Lineal hay que aplicar otro tipo de herramientas de estadísticas para poder realizar un análisis de datos suficiente para hallar una relación entre los diferentes datos que se quieren estudiar. La otra herramienta estadística que utilicé fue la de regresión logística o función sigmoide que permite predecir clases binarias, de manera que la relación que existe entre dos variables es de sí o no, una vez que pasa o no el 0.5 en los casos del Outcome que me dice si un paciente tiene o no diabetes. Este procedimiento de comparar el Outcome con las diferentes variables para predecir qué valores me determinan si un paciente que tiene determinadas características tiene tendencias de padecer diabetes lo realicé con variables como la edad, el BMI y Diabetes Pedigree Function, sin embargo se puede aplicar a cualquier otro parámetro de interés de estudio.

9. Basándote en el gráfico de los datos y la regresión lineal, ¿qué conclusiones puedes extraer sobre la relación entre las dos variables? ¿Observas una tendencia clara en los datos que respalda la línea de regresión?

> En el caso de Outcome vs BMI, Outcome vs Age, Outcome vs DPF se observa una relación que no puede ser explicada a través de una regresión lineal, significando que los datos no tienen una relación inversa o directamente proporcional *aparente*. En este orden de ideas, para poder hallar una relación entre las variables he realizado la regresión logística, y los resultados plasmados en la gráfica indican que sí hay una relación que podríamos denominar directamente proporcional
>
> 1. Outcome vs BMI: los resultados alrededor de entre 40 BMI podemos determinar que el Outcome de ser diabético o no empieza a incrementar bastante siendo éste el punto donde aumenta del 0.5 del Outcome.
> 2. Outcome vs Age: para los resultados entre 40 y 50 años la probabilidad de tener diabetes aumenta considerablemente y la misma tiene una tendencia exponencial
> 3. Outcome vs DPF: para resultados donde el diabetes pedigree function es 1 o superior, hay una tendencia de considerar que esa persona llegue a tener diabetes.