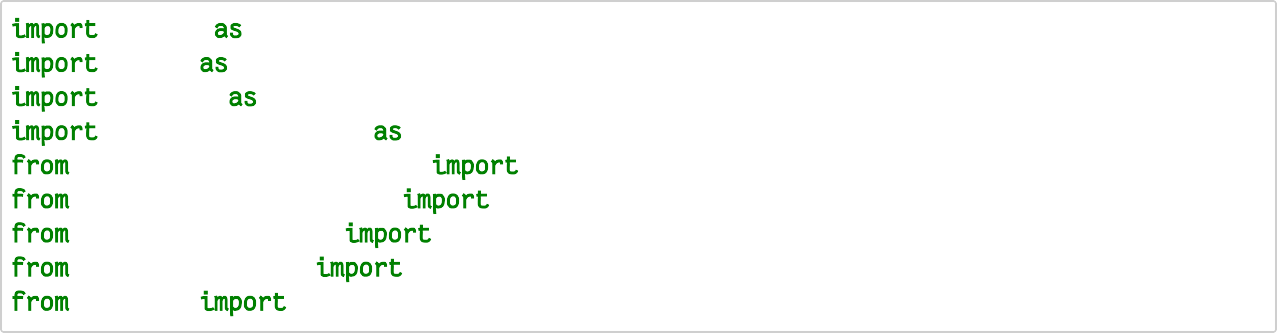
In [27]:



import pandas as pd import numpy as np import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split, GridSearchCV from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix, classification\_report, accuracy\_score from sklearn import preprocessing

## Loading the Dataset

First we load the dataset and find out the number of columns, rows, NULL values, etc.

In [2]:

df = pd.read\_csv('diabetes.csv')

In [3]:

df.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 |  | Pedigree | 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 [4]:

df.head()

Out[4]:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Pregnancies** | **Glucose** | **BloodPressure** | **SkinThickness** | **Insulin** | **BMI** | **Pedigree** | **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 |

## Cleaning

In [11]:

df.corr().style.background\_gradient(cmap='BuGn')

Out[11]:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Pregnancies** | **Glucose** | **BloodPressure** | **SkinThickness** | **Insulin** | **BMI** | **Pedigree** |
| **Pregnancies** | 1.000000 | 0.129459 | 0.141282 | -0.081672 | -0.073535 | 0.017683 | -0.033523 |
| **Glucose** | 0.129459 | 1.000000 | 0.152590 | 0.057328 | 0.331357 | 0.221071 | 0.137337 |
| **BloodPressure** | 0.141282 | 0.152590 | 1.000000 | 0.207371 | 0.088933 | 0.281805 | 0.041265 |
| **SkinThickness** | -0.081672 | 0.057328 | 0.207371 | 1.000000 | 0.436783 | 0.392573 | 0.183928 |
| **Insulin** | -0.073535 | 0.331357 | 0.088933 | 0.436783 | 1.000000 | 0.197859 | 0.185071 |
| **BMI** | 0.017683 | 0.221071 | 0.281805 | 0.392573 | 0.197859 | 1.000000 | 0.140647 |
| **Pedigree** | -0.033523 | 0.137337 | 0.041265 | 0.183928 | 0.185071 | 0.140647 | 1.000000 |
| **Age** | 0.544341 | 0.263514 | 0.239528 | -0.113970 | -0.042163 | 0.036242 | 0.033561 |
| **Outcome** | 0.221898 | 0.466581 | 0.065068 | 0.074752 | 0.130548 | 0.292695 | 0.173844 |

In [13]:

df.drop(['BloodPressure', 'SkinThickness'], axis=1, inplace=True)

In [14]:

Out[14]:

df.isna().sum()

Pregnancies 0

Glucose 0

Insulin 0

BMI 0

Pedigree 0

Age 0

Outcome 0

dtype: int64

In [15]:

Out[15]:

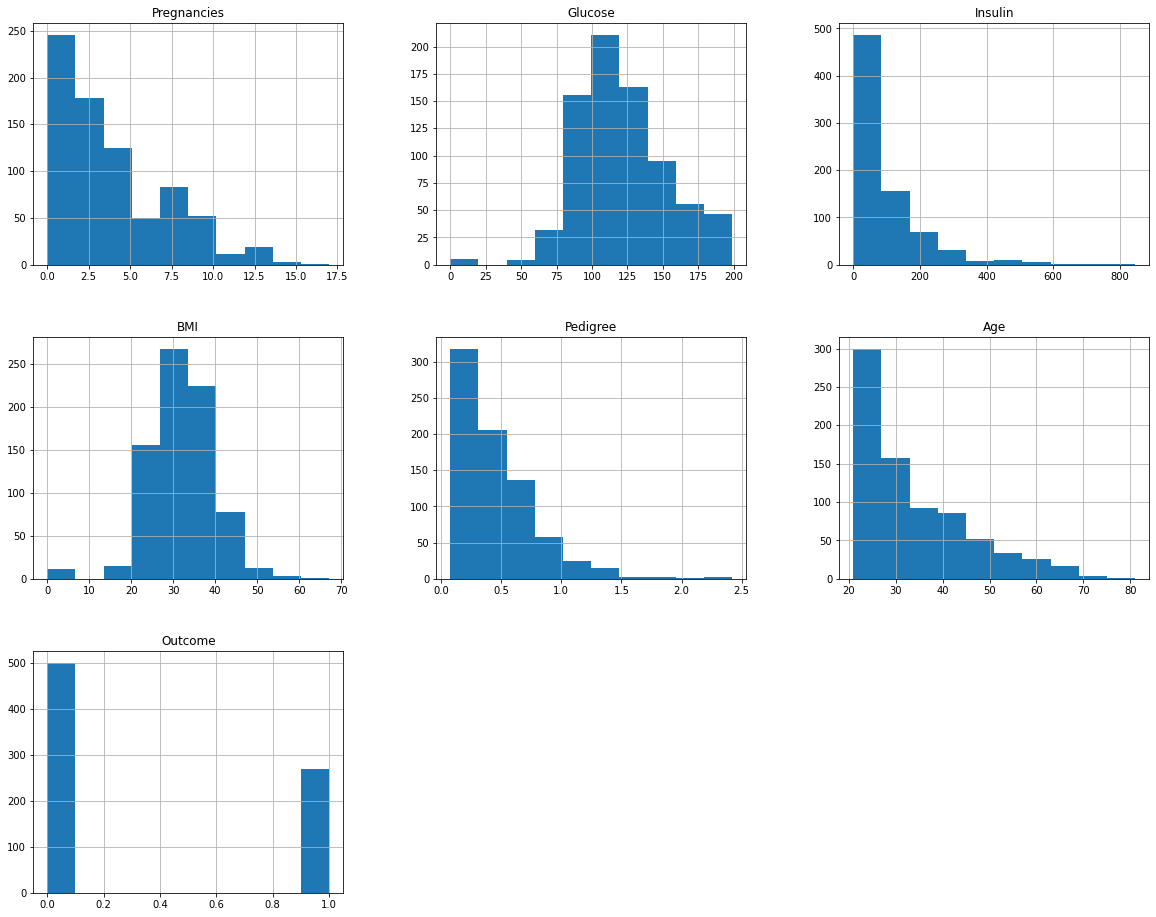
df.describe()

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Pregnancies** | **Glucose** | **Insulin** | **BMI** | **Pedigree** | **Age** | **Outcome** |
| **count** | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 |
| **mean** | 3.845052 | 120.894531 | 79.799479 | 31.992578 | 0.471876 | 33.240885 | 0.348958 |
| **std** | 3.369578 | 31.972618 | 115.244002 | 7.884160 | 0.331329 | 11.760232 | 0.476951 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.078000 | 21.000000 | 0.000000 |
| **25%** | 1.000000 | 99.000000 | 0.000000 | 27.300000 | 0.243750 | 24.000000 | 0.000000 |
| **50%** | 3.000000 | 117.000000 | 30.500000 | 32.000000 | 0.372500 | 29.000000 | 0.000000 |
| **75%** | 6.000000 | 140.250000 | 127.250000 | 36.600000 | 0.626250 | 41.000000 | 1.000000 |
| **max** | 17.000000 | 199.000000 | 846.000000 | 67.100000 | 2.420000 | 81.000000 | 1.000000 |

## Visualization

In [16]:

hist = df.hist(figsize=(20,16))



# Separating the features and the labels

In [17]:

X=df.iloc[:, :df.shape[1]~~-~~1]

y=df.iloc[:, ~~-~~1] X.shape, y.shape

*#Independent Variables*

*#Dependent Variable*

Out[17]: ((768, 6), (768,))

# Splitting the Dataset

Training and Test Set

In [21]:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=8 scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train) X\_test = scaler.transform(X\_test)

# Machine Learning model

In [30]:



def knn(X\_train, X\_test, y\_train, y\_test, neighbors, power): model = KNeighborsClassifier(n\_neighbors=neighbors, p=power) *# Fit and predict on model*

*# Model is trained using the train set and predictions are made based on the test s*

y\_pred=model.fit(X\_train, y\_train).predict(X\_test)

print(f"Accuracy for K-Nearest Neighbors model \t: {accuracy\_score(y\_test, y\_pred)}

cm = confusion\_matrix(y\_test, y\_pred) print(f'''Confusion matrix :\n

| Positive Prediction\t| Negative Prediction

---------------+------------------------+----------------------

Positive Class | True Positive (TP) {cm[0, 0]}\t| False Negative (FN) {cm[0, 1]}

---------------+------------------------+----------------------

Negative Class | False Positive (FP) {cm[1, 0]}\t| True Negative (TN) {cm[1, 1]}\n' cr = classification\_report(y\_test, y\_pred)

print('Classification report : \n', cr)

## Hyperparameter tuning

In [28]:

param\_grid = {

'n\_neighbors': range(1, 51),

'p': range(1, 4)

}

grid = GridSearchCV(estimator=KNeighborsClassifier(), param\_grid=param\_grid, cv=5) grid.fit(X\_train, y\_train)

grid.best\_estimator\_, grid.best\_params\_, grid.best\_score\_

Out[28]: (KNeighborsClassifier(n\_neighbors=27),

{'n\_neighbors': 27, 'p': 2},

0.7719845395175262)

In [31]:

knn(X\_train, X\_test, y\_train, y\_test, grid.best\_params\_['n\_neighbors'], grid.best\_param

Accuracy for K-Nearest Neighbors model : 0.7987012987012987 Confusion matrix :

| Positive Prediction | Negative Prediction

---------------+------------------------+----------------------

Positive Class | True Positive (TP) 91 | False Negative (FN) 11

---------------+------------------------+----------------------

|  |  |  |  |
| --- | --- | --- | --- |
| Negative Class | False | Positive | (FP) 20 | | True Negative (TN) 32 |
| Classification report :  precision | recall | f1-score | support |
| 0 0.82 | 0.89 | 0.85 | 102 |
| 1 0.74 | 0.62 | 0.67 | 52 |
| accuracy |  | 0.80 | 154 |
| macro avg 0.78 | 0.75 | 0.76 | 154 |
| weighted avg 0.79 | 0.80 | 0.79 | 154 |