In [1]:
```
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
```

In [2]:
```
df = pd.read_csv("C:\\Users\\Student\\Desktop\\ajinkya mote- 24\\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 [5]:
```
df.describe
```

Out[5]:
```
<bound method NDFrame.describe of      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6   
1              1       85             66             29        0  26.6   
2              8      183             64              0        0  23.3   
3              1       89             66             23       94  28.1   
4              0      137             40             35      168  43.1   
..           ...      ...            ...            ...      ...   ...   
763           10      101             76             48      180  32.9   
764            2      122             70             27        0  36.8   
765            5      121             72             23      112  26.2   
766            1      126             60              0        0  30.1   
767            1       93             70             31        0  30.4   

     Pedigree  Age  Outcome  
0       0.627   50        1  
1       0.351   31        0  
2       0.672   32        1  
3       0.167   21        0  
4       2.288   33        1  
..        ...  ...      ...  
763     0.171   63        0  
764     0.340   27        0  
765     0.245   30        0  
766     0.349   47        1  
767     0.315   23        0  

[768 rows x 9 columns]>
```

In [9]:
```
df.head()
```

Out[9]:

	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

In [10]:
```
df.corr().style.background_gradient(cmap='BuGn')
```

Out[10]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
Pedigree	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

In [11]:

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

In [12]:

```
df.isna().sum()
```

Out[12]:

```
Pregnancies    0
Glucose        0
Insulin        0
BMI            0
Pedigree       0
Age            0
Outcome        0
dtype: int64
```

In [13]:

```
df.describe
```

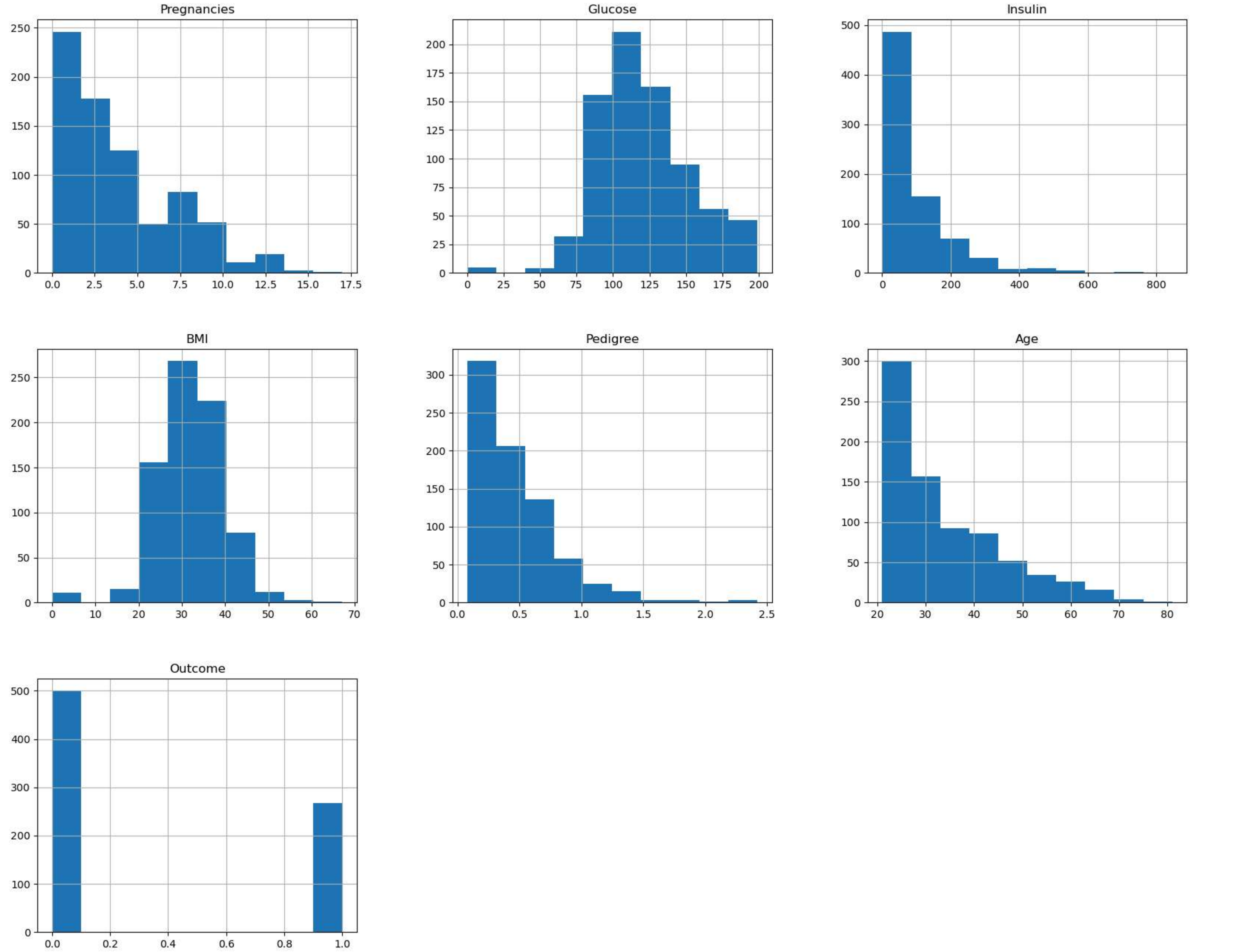
Out[13]:

```
<bound method NDFrame.describe of      Pregnancies  Glucose  Insulin   BMI  Pedigree  Age  Outcome
0              6      148        0  33.6     0.627   50        1
1              1       85        0  26.6     0.351   31        0
2              8      183        0  23.3     0.672   32        1
3              1       89       94  28.1     0.167   21        0
4              0      137      168  43.1     2.288   33        1
..           ...      ...      ...   ...       ...  ...      ...
763           10      101      180  32.9     0.171   63        0
764            2      122        0  36.8     0.340   27        0
765            5      121      112  26.2     0.245   30        0
766            1      126        0  30.1     0.349   47        1
767            1       93        0  30.4     0.315   23        0

[768 rows x 7 columns]>
```

In [14]:

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



In [15]:

```
X=df.iloc[:, :df.shape[1]-1] #Independent Variables
y=df.iloc[:, -1] #Dependent Variable
X.shape, y.shape
```

Out[15]:

```
((768, 6), (768,))
```

In [16]:

```
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)
```

In [23]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

def knn(X_train, X_test, y_train, y_test, neighbors, power):
    model = KNeighborsClassifier(n_neighbors=neighbors, p=power)

    # Fit the model using the training set and make predictions on the test set
    y_pred = model.fit(X_train, y_train).predict(X_test)

    # Print accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy for K-Nearest Neighbors model \t: {accuracy:.4f}")

    # Compute and print confusion matrix
    cm = confusion_matrix(y_test, y_pred)
    print(f'''Confusion matrix :
    | 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''')

    # Compute and print classification report
    cr = classification_report(y_test, y_pred)
    print('Classification report : \n', cr)
```

In [24]:

```
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[24]: (KNeighborsClassifier(n_neighbors=27),
          {'n_neighbors': 27, 'p': 2},
          0.7719845395175262)
```

```
In [27]: knn(X_train, X_test, y_train, y_test, grid.best_params_['n_neighbors'], grid.best_params_['p'])
```

```
Accuracy for K-Nearest Neighbors model  : 0.7987
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
```

```
In [ ]:
```