In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from mlxtend.plotting import plot_decision_regions
import missingno as msno
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

In [2]:
```
diabetes_df = pd.read_csv('D:\diabetes.csv')
```

In [3]:
```
diabetes_df.head()
```

Out[3]:

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

In [4]:
```
diabetes_df.columns
```

Out[4]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [5]:
```
diabetes_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   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 [39]:
```
diabetes_df.describe()
```

Out[39]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
count	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	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

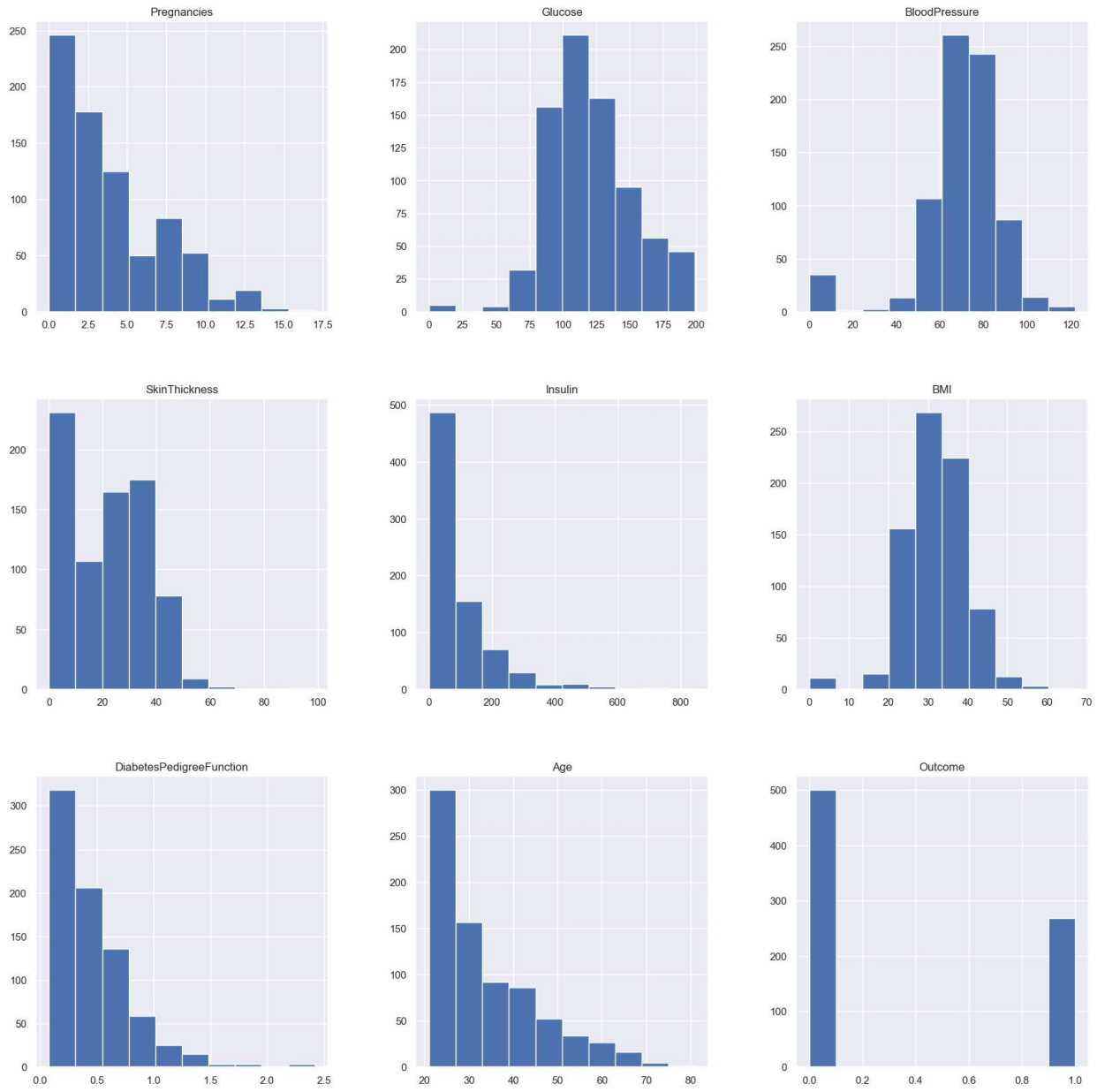
In [28]:
```
diabetes_df.isnull().sum()
```

Out[28]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [29]:
```
diabetes_df_copy = diabetes_df.copy(deep = True)
diabetes_df_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = diabet

print(diabetes_df_copy.isnull().sum())
```

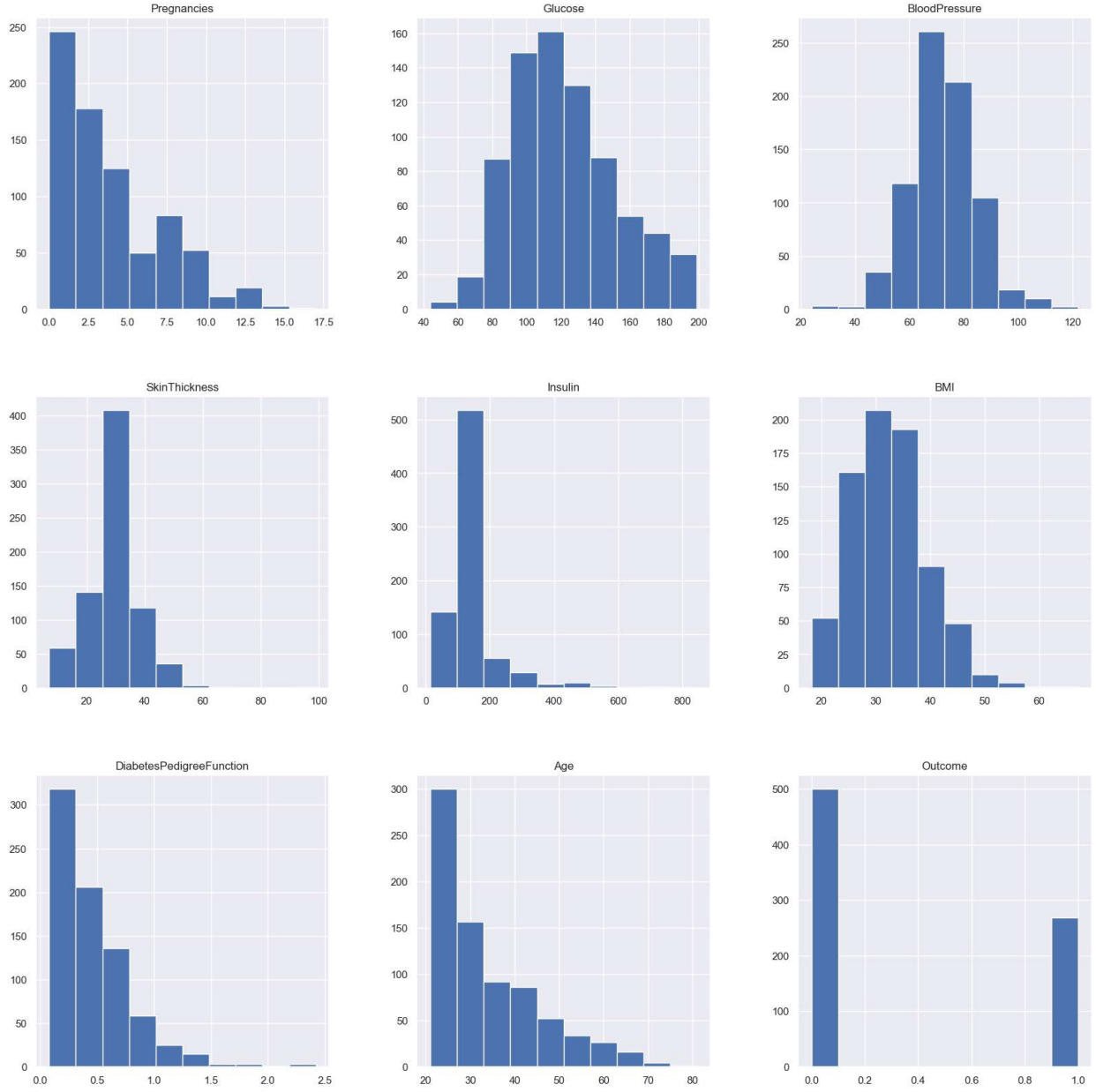
```
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [30]:
```
p = diabetes_df.hist(figsize = (20,20))
```



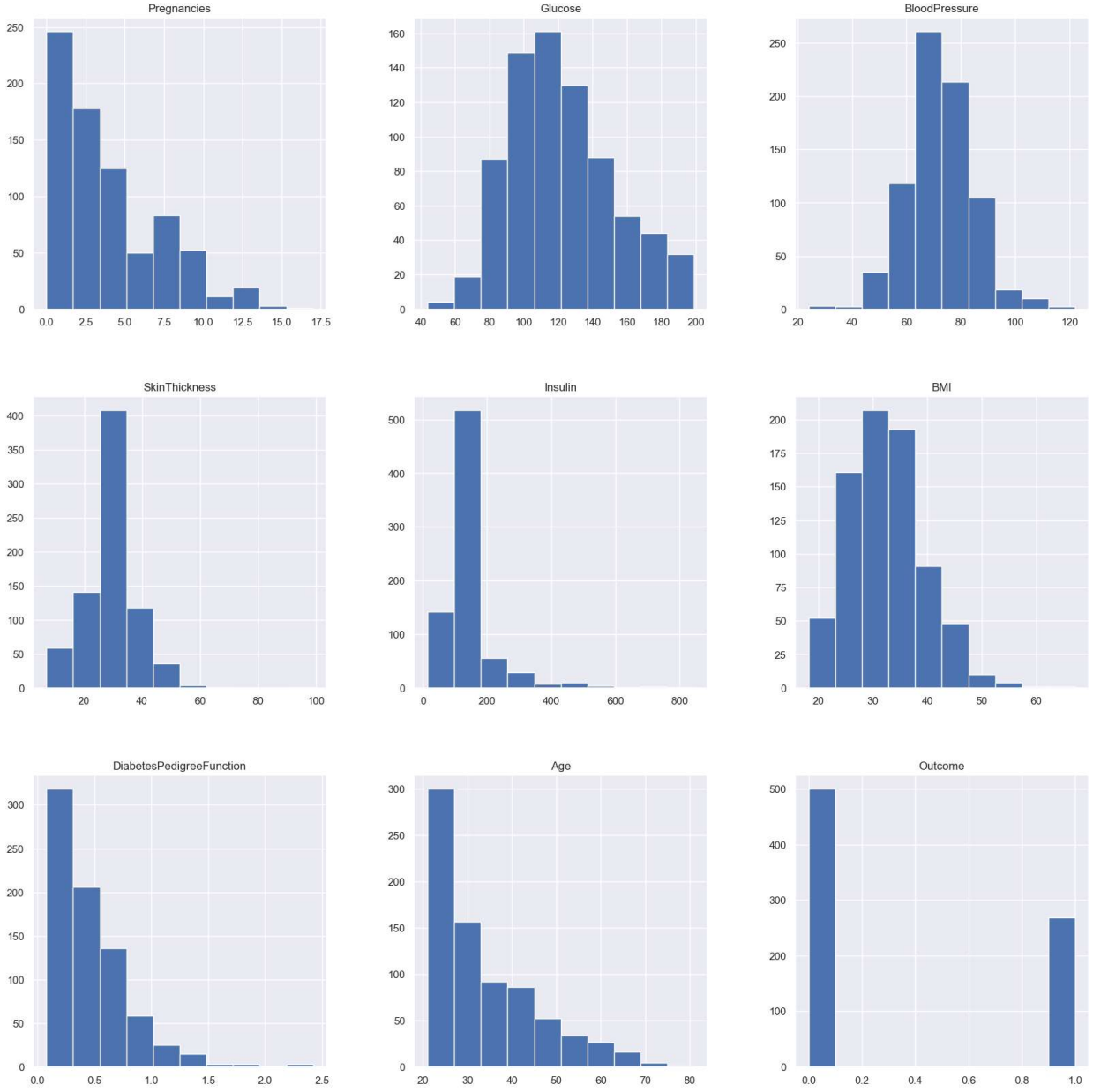
In [31]:
```
diabetes_df_copy['Glucose'].fillna(diabetes_df_copy['Glucose'].mean(), inplace = True)
diabetes_df_copy['BloodPressure'].fillna(diabetes_df_copy['BloodPressure'].mean(), inp
diabetes_df_copy['SkinThickness'].fillna(diabetes_df_copy['SkinThickness'].mean(), inp
diabetes_df_copy['Insulin'].fillna(diabetes_df_copy['Insulin'].mean(), inplace = True)
diabetes_df_copy['BMI'].fillna(diabetes_df_copy['BMI'].mean(), inplace = True)
```

In [32]:
```
p = diabetes_df_copy.hist(figsize = (20,20))
```

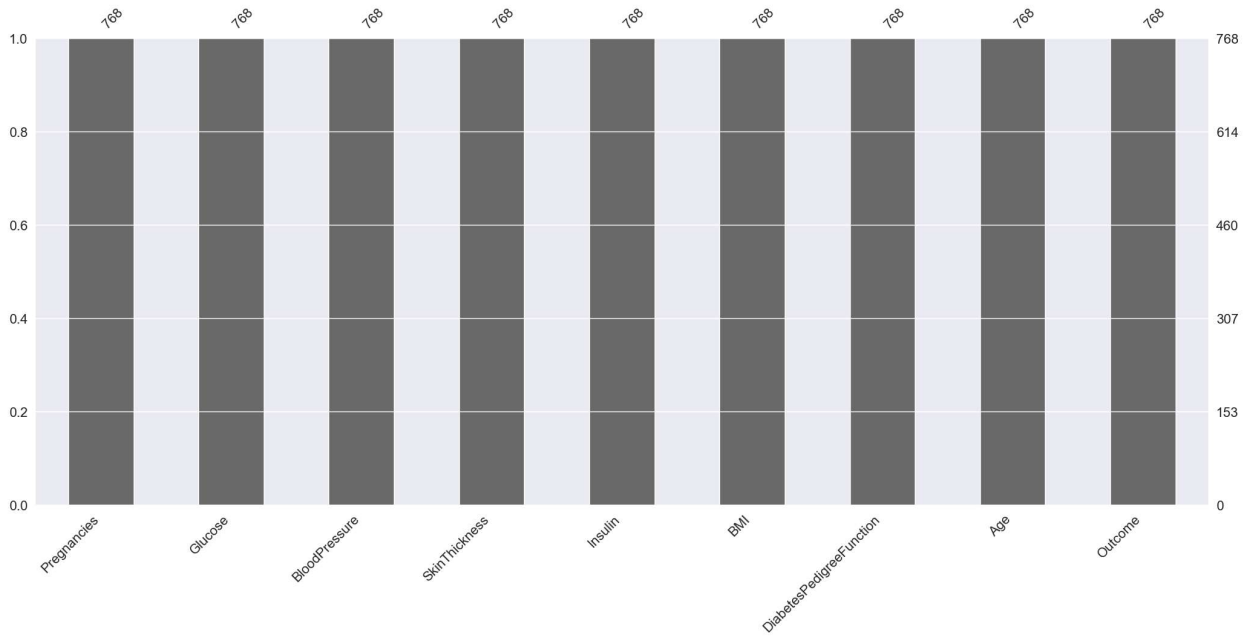


In [33]:
```
diabetes_df_copy['Glucose'].fillna(diabetes_df_copy['Glucose'].mean(), inplace = True)
diabetes_df_copy['BloodPressure'].fillna(diabetes_df_copy['BloodPressure'].mean(), inp
diabetes_df_copy['SkinThickness'].fillna(diabetes_df_copy['SkinThickness'].median(), i
diabetes_df_copy['Insulin'].fillna(diabetes_df_copy['Insulin'].median(), inplace = Tru
diabetes_df_copy['BMI'].fillna(diabetes_df_copy['BMI'].median(), inplace = True)
```

In [34]:
```
p = diabetes_df_copy.hist(figsize = (20,20))
```

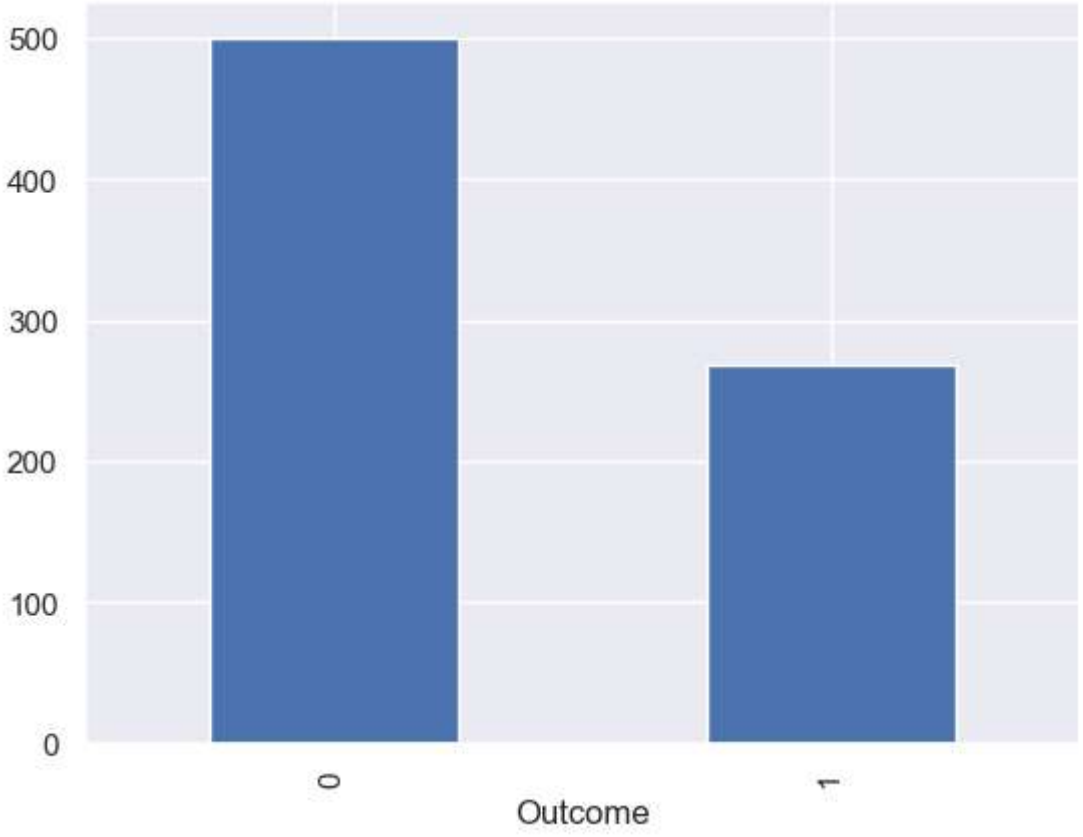


In [35]:
```
p = msno.bar(diabetes_df)
```

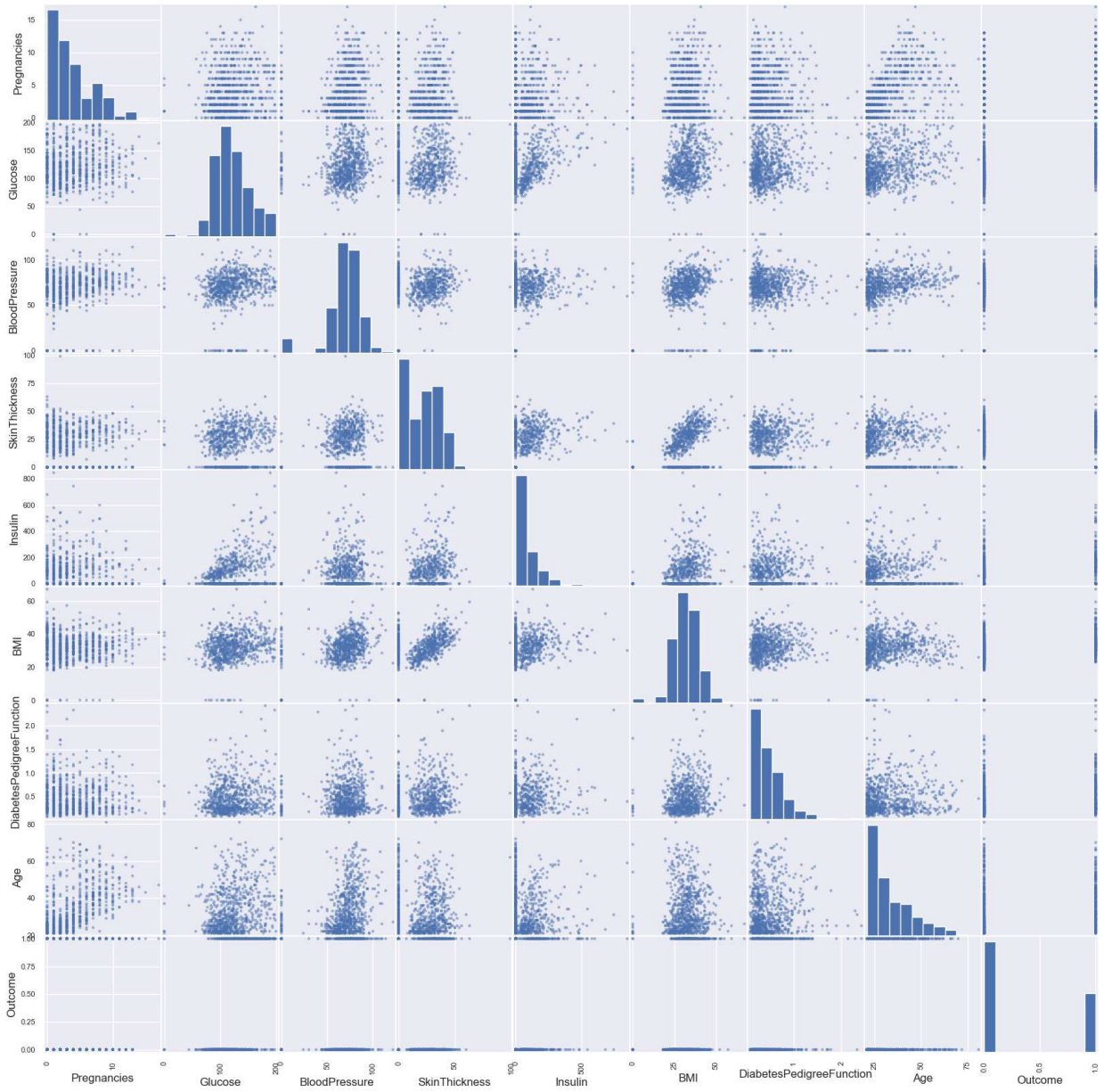


In [36]:
```
color_wheel = {1: "#0392cf", 2: "#7bc043"}
colors = diabetes_df["Outcome"].map(lambda x: color_wheel.get(x + 1))
print(diabetes_df.Outcome.value_counts())
p=diabetes_df.Outcome.value_counts().plot(kind="bar")
```

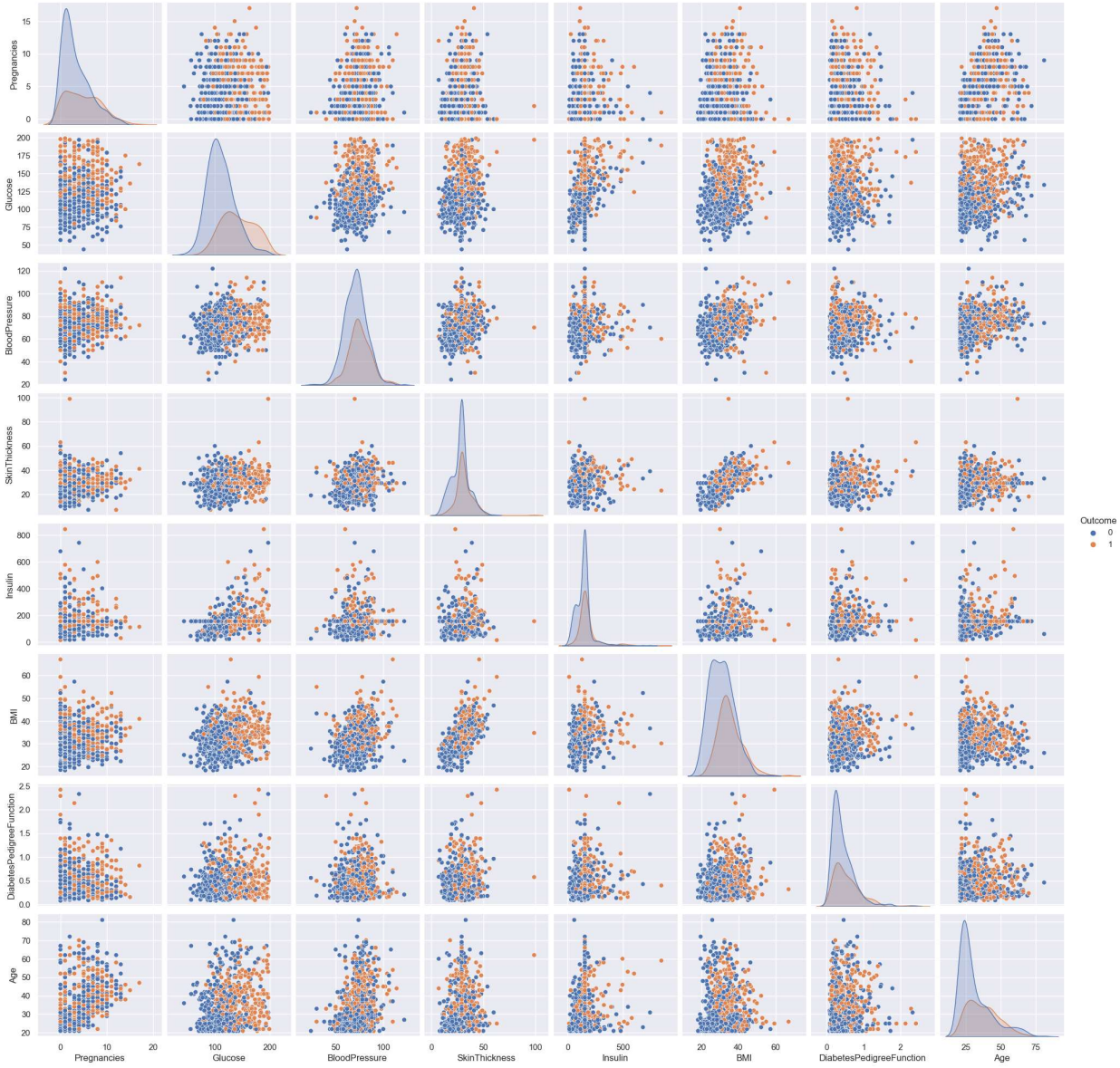
```
Outcome
0    500
1    268
Name: count, dtype: int64
```



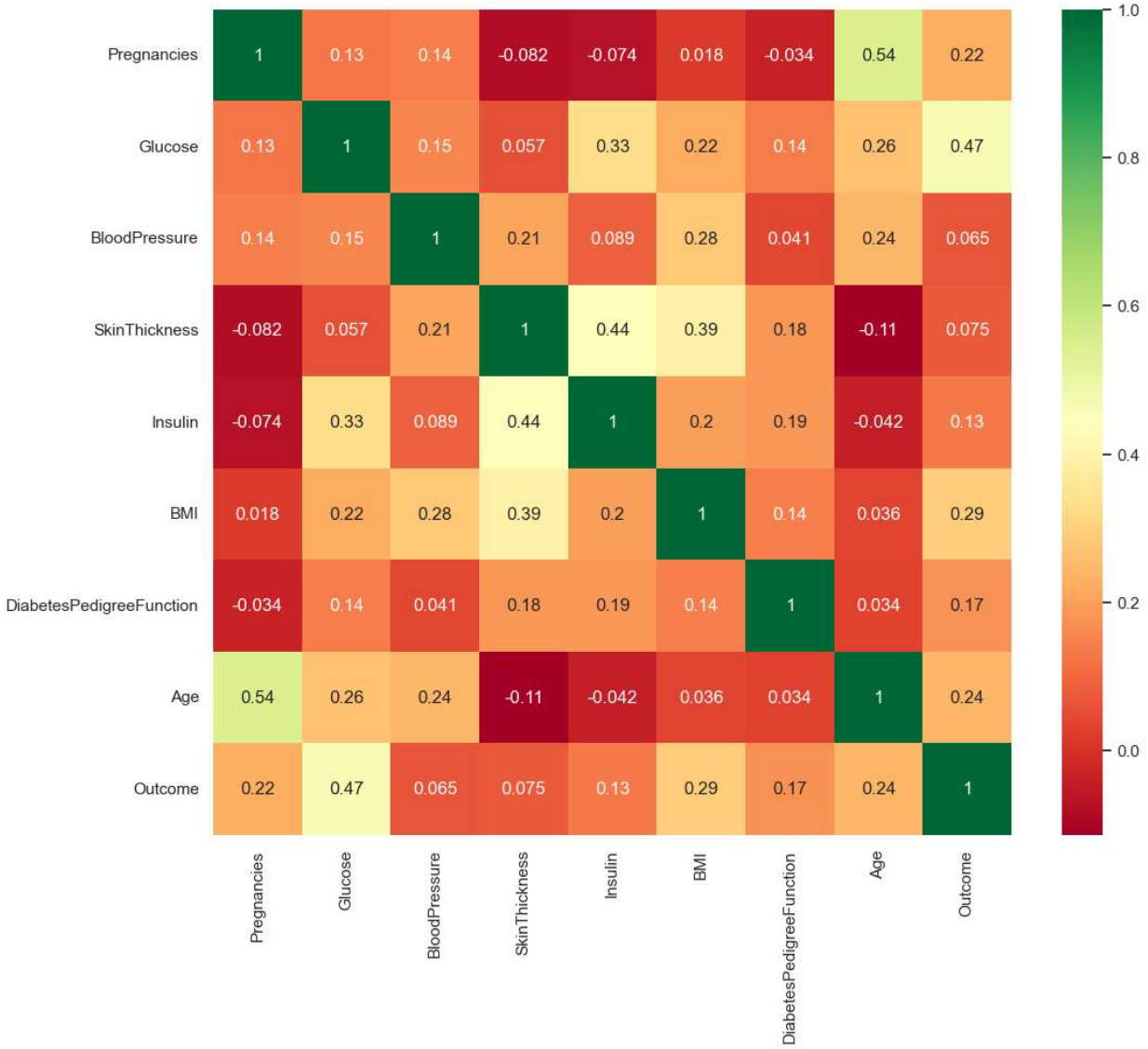
In [37]:
```
p = scatter_matrix(diabetes_df, figsize= (20,20))
```



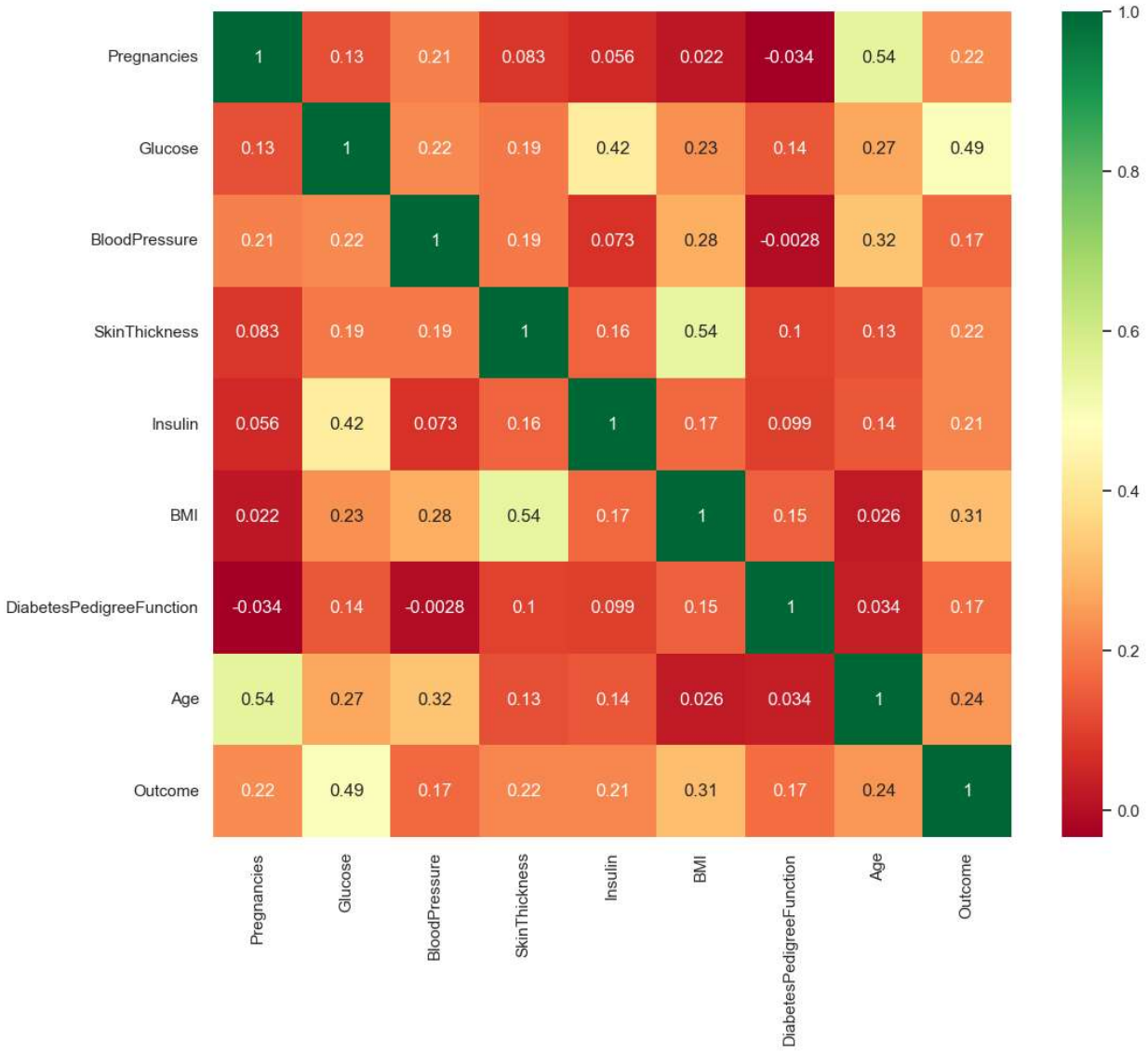
In [40]:
```
p=sns.pairplot(diabetes_df_copy, hue = 'Outcome')
```



In [42]:
```
plt.figure(figsize =(12,10))
g=sns.heatmap(diabetes_df.corr(),annot=True,cmap="RdYlGn")
```



In [43]:
```
plt.figure(figsize =(12,10))
g=sns.heatmap(diabetes_df_copy.corr(),annot=True,cmap="RdYlGn")
```



In [44]:
```
diabetes_df_copy.head()
```

Out[44]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148.0	72.0	35.00000	155.548223	33.6	0.627	
1	1	85.0	66.0	29.00000	155.548223	26.6	0.351	
2	8	183.0	64.0	29.15342	155.548223	23.3	0.672	
3	1	89.0	66.0	23.00000	94.000000	28.1	0.167	
4	0	137.0	40.0	35.00000	168.000000	43.1	2.288	

In [47]:
```
sc_X = StandardScaler()
X = pd.DataFrame(sc_X.fit_transform(diabetes_df_copy.drop(["Outcome"],axis = 1),),colu
X.head()
```

Out[47]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	0.639947	0.865108	-0.033518	6.655021e-01	-3.345079e-16	0.166292	0.468
1	-0.844885	-1.206162	-0.529859	-1.746338e-02	-3.345079e-16	-0.852531	-0.365
2	1.233880	2.015813	-0.695306	8.087936e-16	-3.345079e-16	-1.332833	0.604
3	-0.844885	-1.074652	-0.529859	-7.004289e-01	-7.243887e-01	-0.634212	-0.920
4	-1.141852	0.503458	-2.680669	6.655021e-01	1.465506e-01	1.548980	5.484

In [48]:
```
y = diabetes_df_copy.Outcome
```

In [49]:
```
y
```

Out[49]:
```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [52]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=1/3,random_state=42,str
```

Model Training

In [ ]:
```
test_scores = []
train_scores = []

for i in range(1,15):

    knn = KNeighborsClassifier(i)
    knn.fit(X_train,y_train)

    train_scores.append(knn.score(X_train,y_train))
    test_scores.append(knn.score(X_test,y_test))
```

In [56]:
```
train_scores
```

Out[56]:
```
[1.0,
 0.84765625,
 0.865234375,
 0.83203125,
 0.8359375,
 0.806640625,
 0.81640625,
 0.8046875,
 0.802734375,
 0.79296875,
 0.80859375,
 0.794921875,
 0.796875,
 0.79296875]
```

In [57]:
```
test_scores
```

Out[57]:
```
[0.7265625,
 0.73046875,
 0.75390625,
 0.74609375,
 0.734375,
 0.7265625,
 0.74609375,
 0.73046875,
 0.73046875,
 0.72265625,
 0.76171875,
 0.7265625,
 0.75390625,
 0.73828125]
```

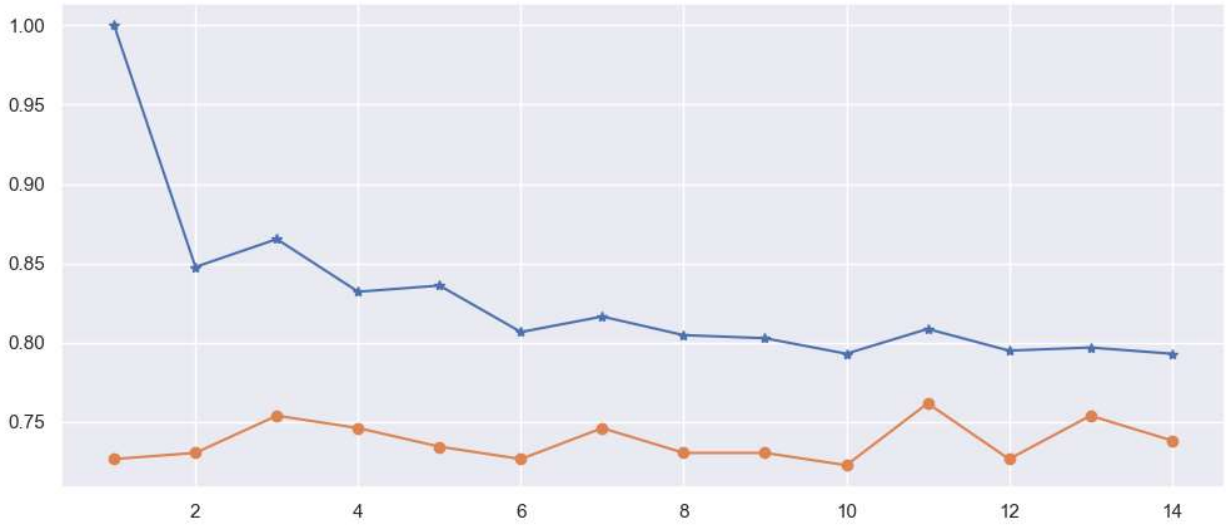
In [60]:
```
max_train_score = max(train_scores)
train_scores_ind = [i for i, v in enumerate(train_scores) if v == max_train_score]
print('Max train score {} % and k = {}'.format(max_train_score*100,list(map(lambda x:
```

```
Max train score 100.0 % and k = [1]
```

In [61]:
```
max_test_score = max(test_scores)
test_scores_ind = [i for i, v in enumerate(test_scores) if v == max_test_score]
print('Max test score {} % and k = {}'.format(max_test_score*100,list(map(lambda x: x+
```

```
Max test score 76.171875 % and k = [11]
```

In [80]:
```
plt.figure(figsize=(12, 5))
x_values = range(1, 15)  # Assuming this is the correct range for your data
plt.plot(x_values, train_scores, marker='*', label='Train Score')
plt.plot(x_values, test_scores, marker='o', label='Test Score')
plt.show()
```



In [81]:
```
knn = KNeighborsClassifier(11)
knn.fit(X_train,y_train)
knn.score(X_test,y_test)
```

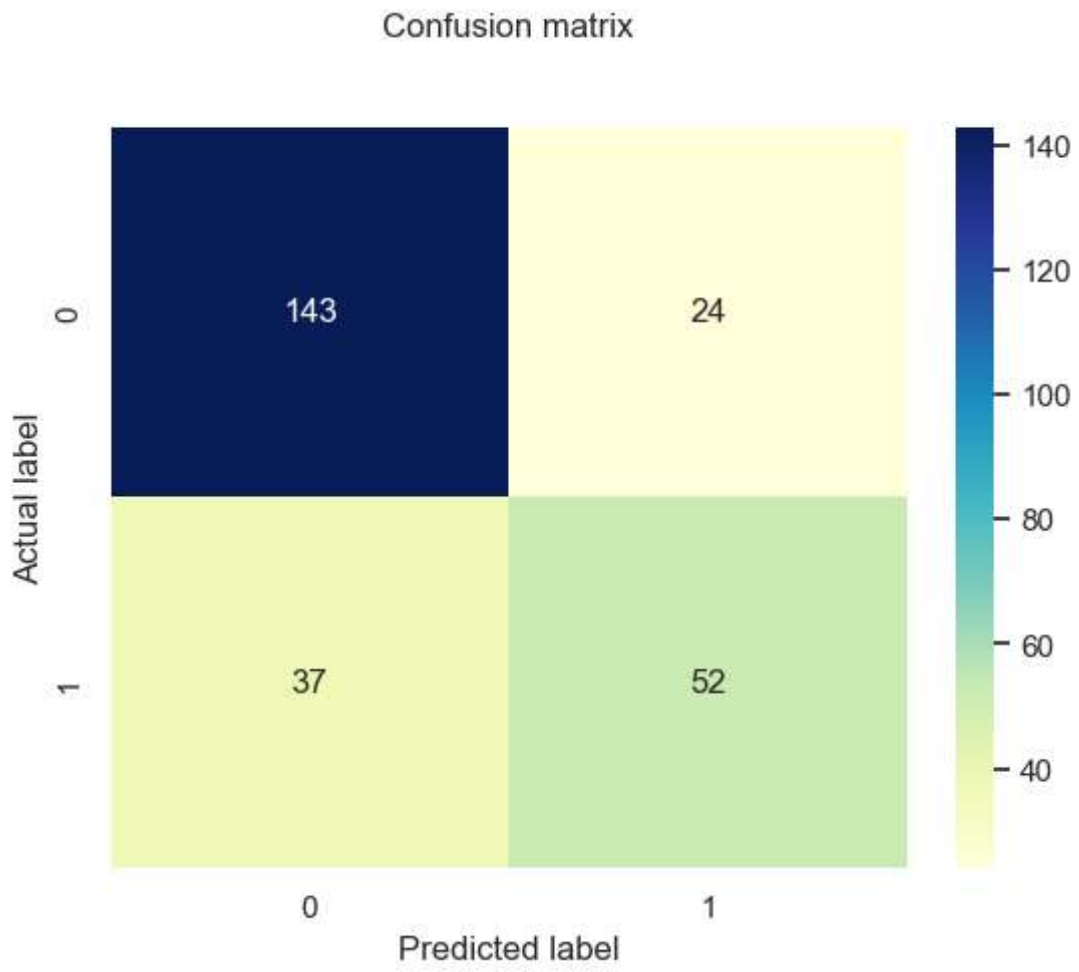
Out[81]:
```
0.76171875
```

In [84]:
```
'''value = 20000
width = 20000
plot_decision_regions(X.values, y.values, clf=knn, legend=2,
                      filler_feature_values=(2: value, 3: value, 4: value, 5: value)
                      filler_feature_ranges=(2: width, 3: width, 4: width, 5: width)
                      X_highlight=X_test.values)

plt.title('KNN with Diabetes Data')
plt.show()'''
```

Out[84]:
```
"value = 20000\nwidth = 20000\nplot_decision_regions(X.values, y.values, clf=knn, leg
end=2,\n                      filler_feature_values=(2: value, 3: value, 4: value, 5:
value)\n                      filler_feature_ranges=(2: width, 3: width, 4: width, 5:
width)\n                      X_highlight=X_test.values)\n\nplt.title('KNN with Diabet
es Data')\nplt.show()"
```

In [87]:
```
y_pred = knn.predict(X_test)

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
p = sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu", fmt='g')
plt.title('Confusion matrix',y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[87]:
```
Text(0.5, 20.049999999999997, 'Predicted label')
```

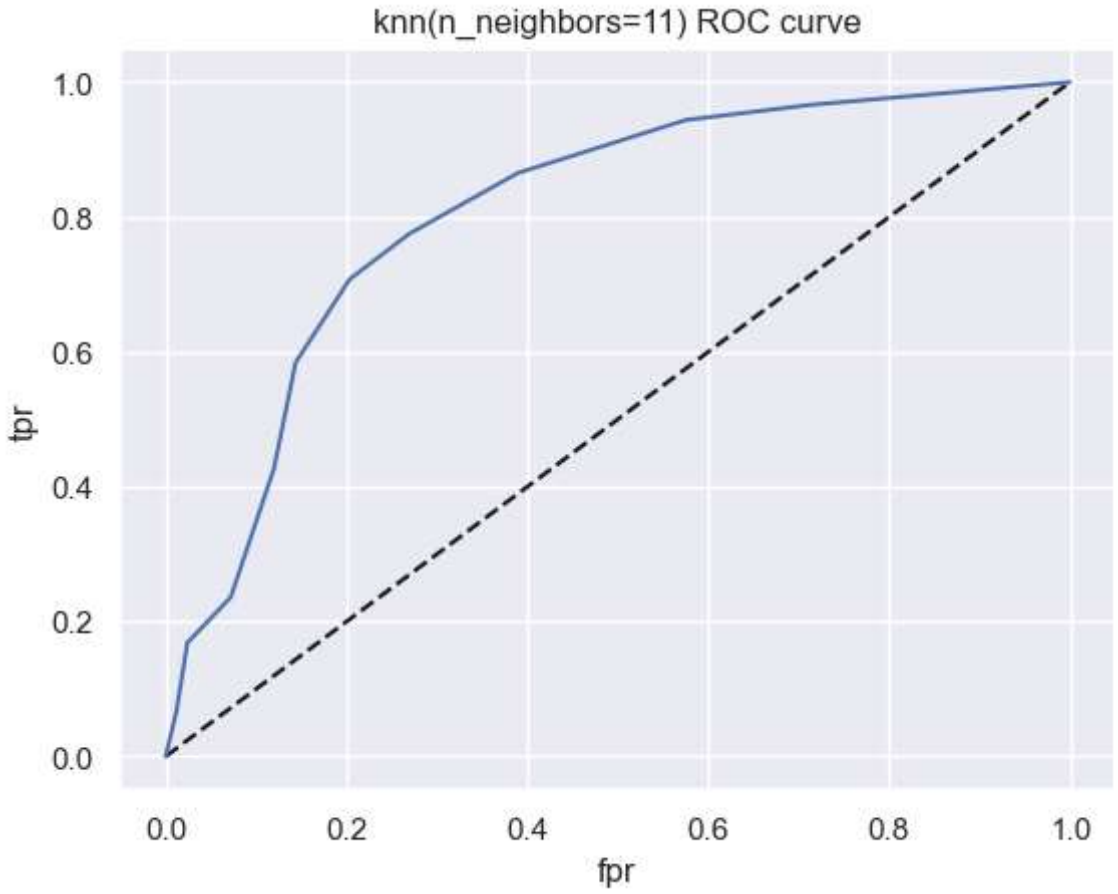


In [88]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.86      0.82       167
           1       0.68      0.58      0.63        89

    accuracy                           0.76       256
   macro avg       0.74      0.72      0.73       256
weighted avg       0.76      0.76      0.76       256
```

In [89]:
```
y_pred_proba = knn.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
```

In [90]:
```
plt.plot([0,1],[0,1],'k--')
plt.plot(fpr,tpr, label='Knn')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title('knn(n_neighbors=11) ROC curve')
plt.show()
```



In [91]:
```
roc_auc_score(y_test,y_pred_proba)
```

Out[91]:
```
0.8072730942609163
```

In [94]:
```
param_grid={'n_neighbors':np.arange(1,50)}
knn =KNeighborsClassifier()
knn_cv=GridSearchCV(knn,param_grid, cv=5)
knn_cv.fit(X,y)

print("Best Score:"+ str(knn_cv.best_score_))
print("best Parameters:"+ str(knn_cv.best_params_))
```

```
Best Score:0.7748068924539513
best Parameters:{'n_neighbors': 25}
```