In [1]:
```
# Load libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import ttest_ind

from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import linkage, fcluster
from sklearn.cluster import KMeans, DBSCAN
from sklearn import metrics
```

In [2]:
```
# Load dataset
diabetes_data_binary = pd.read_csv('diabetes_data_upload.csv')
diabetes_data_floats = pd.read_csv('diabetes-dataset.csv')
```

In [3]:
```
# Display original binary dataset
print(diabetes_data_binary.info())
diabetes_data_binary.head(10)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 520 entries, 0 to 519
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Age                 520 non-null    int64 
 1   Gender              520 non-null    object
 2   Polyuria            520 non-null    object
 3   Polydipsia          520 non-null    object
 4   sudden weight loss  520 non-null    object
 5   weakness            520 non-null    object
 6   Polyphagia          520 non-null    object
 7   Genital thrush      520 non-null    object
 8   visual blurring     520 non-null    object
 9   Itching             520 non-null    object
 10  Irritability        520 non-null    object
 11  delayed healing     520 non-null    object
 12  partial paresis     520 non-null    object
 13  muscle stiffness    520 non-null    object
 14  Alopecia            520 non-null    object
 15  Obesity             520 non-null    object
 16  class               520 non-null    object
dtypes: int64(1), object(16)
memory usage: 69.2+ KB
None
```

Out[3]:

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching
0	40	Male	No	Yes	No	Yes	No	No	No	Yes
1	58	Male	No	No	No	Yes	No	No	Yes	No
2	41	Male	Yes	No	No	Yes	Yes	No	No	Yes
3	45	Male	No	No	Yes	Yes	Yes	Yes	No	Yes
4	60	Male	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
5	55	Male	Yes	Yes	No	Yes	Yes	No	Yes	Yes
6	57	Male	Yes	Yes	No	Yes	Yes	Yes	No	No
7	66	Male	Yes	Yes	Yes	Yes	No	No	Yes	Yes
8	67	Male	Yes	Yes	No	Yes	Yes	Yes	No	Yes
9	70	Male	No	Yes	Yes	Yes	Yes	No	Yes	Yes

In [ ]:
```
```

In [4]:
```
# Map string values to int for the binary dataset

# Convert Yes/No values to 1/0 values
diabetes_data_binary = diabetes_data_binary.applymap(lambda x: 1 if x=='Yes' e
lse x)
diabetes_data_binary = diabetes_data_binary.applymap(lambda x: 0 if x=='No' el
se x)

# Convert Pos/Neg values to 1/0 values
diabetes_data_binary = diabetes_data_binary.applymap(lambda x: 1 if x=='Positi
ve' else x)
diabetes_data_binary = diabetes_data_binary.applymap(lambda x: 0 if x=='Negati
ve' else x)

# Rename Gender column to Male
diabetes_data_binary = diabetes_data_binary.rename(columns={'Gender': 'Male'})

# Convert Male/Female values to 1/0 values
diabetes_data_binary['Male'] = diabetes_data_binary['Male'].map({'Male': 1, 'F
emale': 0})
```

In [ ]:
```

```

In [5]:
```
# Display binary dataset after data preparation
print(diabetes_data_binary.info())
diabetes_data_binary
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 520 entries, 0 to 519
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Age                 520 non-null    int64
 1   Male                520 non-null    int64
 2   Polyuria            520 non-null    int64
 3   Polydipsia          520 non-null    int64
 4   sudden weight loss  520 non-null    int64
 5   weakness            520 non-null    int64
 6   Polyphagia          520 non-null    int64
 7   Genital thrush      520 non-null    int64
 8   visual blurring     520 non-null    int64
 9   Itching             520 non-null    int64
 10  Irritability        520 non-null    int64
 11  delayed healing     520 non-null    int64
 12  partial paresis     520 non-null    int64
 13  muscle stiffness    520 non-null    int64
 14  Alopecia            520 non-null    int64
 15  Obesity             520 non-null    int64
 16  class               520 non-null    int64
dtypes: int64(17)
memory usage: 69.2 KB
None
```

Out[5]:

	Age	Male	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching
0	40	1	0	1	0	1	0	0	0	1
1	58	1	0	0	0	1	0	0	1	0
2	41	1	1	0	0	1	1	0	0	1
3	45	1	0	0	1	1	1	1	0	1
4	60	1	1	1	1	1	1	0	1	1
...	...	...	...	...	...	...	...	...	...	...
515	39	0	1	1	1	0	1	0	0	1
516	48	0	1	1	1	1	1	0	0	1
517	58	0	1	1	1	1	1	0	1	0
518	32	0	0	0	0	1	0	0	1	1
519	42	1	0	0	0	0	0	0	0	0

520 rows × 17 columns

In [ ]:
```
```

In [ ]:

In [6]:
```
# Display original integer/float dataset
print(diabetes_data_floats.info())
diabetes_data_floats.head(10)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               2000 non-null   int64  
 1   Glucose                   2000 non-null   int64  
 2   BloodPressure             2000 non-null   int64  
 3   SkinThickness             2000 non-null   int64  
 4   Insulin                   2000 non-null   int64  
 5   BMI                       2000 non-null   float64
 6   DiabetesPedigreeFunction  2000 non-null   float64
 7   Age                       2000 non-null   int64  
 8   Outcome                   2000 non-null   int64  
dtypes: float64(2), int64(7)
memory usage: 140.8 KB
None
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	2	138	62	35	0	33.6	0.127
1	0	84	82	31	125	38.2	0.233
2	0	145	0	0	0	44.2	0.630
3	0	135	68	42	250	42.3	0.365
4	1	139	62	41	480	40.7	0.536
5	0	173	78	32	265	46.5	1.159
6	4	99	72	17	0	25.6	0.294
7	8	194	80	0	0	26.1	0.551
8	2	83	65	28	66	36.8	0.629
9	2	89	90	30	0	33.5	0.292

In [ ]:

In [7]:
```
# Count missing values for integers/floats dataset
```

In [8]:
```
diabetes_data_floats[diabetes_data_floats['Glucose']==0].shape[0]
```

Out[8]: 13

In [9]:
```
diabetes_data_floats[diabetes_data_floats['BloodPressure']==0].shape[0]
```

Out[9]: 90

In [10]:
```
diabetes_data_floats[diabetes_data_floats['SkinThickness']==0].shape[0]
```

Out[10]: 573

In [11]:
```
diabetes_data_floats[diabetes_data_floats['Insulin']==0].shape[0]
```

Out[11]: 956

In [12]:
```
diabetes_data_floats[diabetes_data_floats['BMI']==0].size
```

Out[12]: 252

In [ ]:
```

```

In [13]:
```
# Remove all observations with missing values
diabetes_data_floats = diabetes_data_floats[diabetes_data_floats['Glucose']!=0
]
diabetes_data_floats = diabetes_data_floats[diabetes_data_floats['BloodPressur
e']!=0]
diabetes_data_floats = diabetes_data_floats[diabetes_data_floats['SkinThicknes
s']!=0]
diabetes_data_floats = diabetes_data_floats[diabetes_data_floats['Insulin']!=0
]
diabetes_data_floats = diabetes_data_floats[diabetes_data_floats['BMI']!=0]
```

In [ ]:
```

```

In [14]:
```
# Display integer/float dataset after data cleaning
print(diabetes_data_floats.info())
diabetes_data_floats.head(10)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1035 entries, 1 to 1999
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               1035 non-null   int64  
 1   Glucose                   1035 non-null   int64  
 2   BloodPressure             1035 non-null   int64  
 3   SkinThickness             1035 non-null   int64  
 4   Insulin                   1035 non-null   int64  
 5   BMI                       1035 non-null   float64
 6   DiabetesPedigreeFunction  1035 non-null   float64
 7   Age                       1035 non-null   int64  
 8   Outcome                   1035 non-null   int64  
dtypes: float64(2), int64(7)
memory usage: 80.9 KB
None
```

Out[14]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
1	0	84	82	31	125	38.2	0.23
3	0	135	68	42	250	42.3	0.36
4	1	139	62	41	480	40.7	0.53
5	0	173	78	32	265	46.5	1.15
8	2	83	65	28	66	36.8	0.62
11	4	125	70	18	122	28.9	1.14
15	2	81	72	15	76	30.1	0.54
16	7	195	70	33	145	25.1	0.16
17	6	154	74	32	193	29.3	0.83
18	2	117	90	19	71	25.2	0.31

In [ ]:

In [ ]:

In [15]:
```
# Computing averages for diabetic and non-diabetic patients using integer/floa
ts dataset
```

In [16]:
```
print("Pregnancies")
print("Non-Diabetic Mean:", diabetes_data_floats[diabetes_data_floats['Outcom
e']==0]['Pregnancies'].mean())
print("Diabetic Mean    :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==1]['Pregnancies'].mean())
print()
print("Non-Diabetic std :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==0]['Pregnancies'].std())
print("Diabetic std     :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==1]['Pregnancies'].std())
print()
ttest_ind(diabetes_data_floats[diabetes_data_floats['Outcome']==1]['Pregnancie
s'], diabetes_data_floats[diabetes_data_floats['Outcome']==0]['Pregnancies'],
equal_var=False)
```

```
Pregnancies
Non-Diabetic Mean: 2.603151862464183
Diabetic Mean    : 4.3916913946587535

Non-Diabetic std : 2.5541686931203627
Diabetic std     : 3.8960819512872114
```

Out[16]:
```
Ttest_indResult(statistic=7.669055764939621, pvalue=9.717868756815654e-14)
```

In [ ]:
```

```

In [17]:
```
print("Glucose")
print("Non-Diabetic Mean:", diabetes_data_floats[diabetes_data_floats['Outcom
e']==0]['Glucose'].mean())
print("Diabetic Mean    :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==1]['Glucose'].mean())
print()
print("Non-Diabetic std :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==0]['Glucose'].std())
print("Diabetic std     :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==1]['Glucose'].std())
print()
ttest_ind(diabetes_data_floats[diabetes_data_floats['Outcome']==1]['Glucose'],
diabetes_data_floats[diabetes_data_floats['Outcome']==0]['Glucose'], equal_var
=False)
```

```
Glucose
Non-Diabetic Mean: 111.7378223495702
Diabetic Mean    : 145.84272997032642

Non-Diabetic std : 24.606662425248683
Diabetic std     : 29.136106337756964
```

Out[17]:
```
Ttest_indResult(statistic=18.53284052946368, pvalue=1.862453776362188e-60)
```

In [ ]:
```

```

In [18]:
```
print("BloodPressure ")
print("Non-Diabetic Mean:", diabetes_data_floats[diabetes_data_floats['Outcom
e']==0]['BloodPressure'].mean())
print("Diabetic Mean    :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==1]['BloodPressure'].mean())
print()
print("Non-Diabetic std :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==0]['BloodPressure'].std())
print("Diabetic std     :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==1]['BloodPressure'].std())
print()
ttest_ind(diabetes_data_floats[diabetes_data_floats['Outcome']==1]['BloodPress
ure'], diabetes_data_floats[diabetes_data_floats['Outcome']==0]['BloodPressur
e'], equal_var=False)
```

```
BloodPressure
Non-Diabetic Mean: 68.98424068767908
Diabetic Mean    : 74.5727002967359

Non-Diabetic std : 11.7922491534455
Diabetic std     : 12.599422327405149
```

Out[18]:
```
Ttest_indResult(statistic=6.825979203640128, pvalue=2.0666122575138323e-11)
```

In [ ]:
```

```

In [19]:
```
print("SkinThickness ")
print("Non-Diabetic Mean:", diabetes_data_floats[diabetes_data_floats['Outcom
e']==0]['SkinThickness'].mean())
print("Diabetic Mean    :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==1]['SkinThickness'].mean())
print()
print("Non-Diabetic std :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==0]['SkinThickness'].std())
print("Diabetic std     :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==1]['SkinThickness'].std())
print()
ttest_ind(diabetes_data_floats[diabetes_data_floats['Outcome']==1]['SkinThickn
ess'], diabetes_data_floats[diabetes_data_floats['Outcome']==0]['SkinThicknes
s'], equal_var=False)
```

```
SkinThickness
Non-Diabetic Mean: 27.302292263610315
Diabetic Mean    : 33.32640949554896

Non-Diabetic std : 10.291672911794015
Diabetic std     : 9.941809106994036
```

Out[19]:
```
Ttest_indResult(statistic=9.0301455996166, pvalue=1.7188263087543889e-18)
```

In [ ]:
```

```

In [20]:
```
print("Insulin ")
print("Non-Diabetic Mean:", diabetes_data_floats[diabetes_data_floats['Outcom
e']==0]['Insulin'].mean())
print("Diabetic Mean    :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==1]['Insulin'].mean())
print()
print("Non-Diabetic std :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==0]['Insulin'].std())
print("Diabetic std     :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==1]['Insulin'].std())
print()
ttest_ind(diabetes_data_floats[diabetes_data_floats['Outcome']==1]['Insulin'],
diabetes_data_floats[diabetes_data_floats['Outcome']==0]['Insulin'], equal_var
=False)
```

```
Insulin
Non-Diabetic Mean: 131.3595988538682
Diabetic Mean    : 200.7299703264095

Non-Diabetic std : 99.73279034570396
Diabetic std     : 119.84536697766426
```

Out[20]:
```
Ttest_indResult(statistic=9.198817762526524, pvalue=6.848789084574853e-19)
```

In [ ]:
```

```

In [21]:
```
print("BMI ")
print("Non-Diabetic Mean:", diabetes_data_floats[diabetes_data_floats['Outcom
e']==0]['BMI'].mean())
print("Diabetic Mean    :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==1]['BMI'].mean())
print()
print("Non-Diabetic std :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==0]['BMI'].std())
print("Diabetic std     :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==1]['BMI'].std())
print()
ttest_ind(diabetes_data_floats[diabetes_data_floats['Outcome']==1]['BMI'], dia
betes_data_floats[diabetes_data_floats['Outcome']==0]['BMI'], equal_var=False)
```

```
BMI
Non-Diabetic Mean: 32.08939828080229
Diabetic Mean    : 35.83086053412463

Non-Diabetic std : 6.847662059242316
Diabetic std     : 6.949854197349751
```

Out[21]:
```
Ttest_indResult(statistic=8.154776053796434, pvalue=1.7833653506931635e-15)
```

In [ ]:
```

```

In [22]:
```
print("Diabetes Pedigree Function")
print("Non-Diabetic Mean:", diabetes_data_floats[diabetes_data_floats['Outcom
e']==0]['DiabetesPedigreeFunction'].mean())
print("Diabetic Mean    :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==1]['DiabetesPedigreeFunction'].mean())
print()
print("Non-Diabetic std :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==0]['DiabetesPedigreeFunction'].std())
print("Diabetic std     :", diabetes_data_floats[diabetes_data_floats['Outcom
e']==1]['DiabetesPedigreeFunction'].std())
print()
ttest_ind(diabetes_data_floats[diabetes_data_floats['Outcome']==1]['DiabetesPe
digreeFunction'], diabetes_data_floats[diabetes_data_floats['Outcome']==0]['Di
abetesPedigreeFunction'], equal_var=False)
```

```
Diabetes Pedigree Function
Non-Diabetic Mean: 0.47991977077363895
Diabetic Mean    : 0.6117329376854599

Non-Diabetic std : 0.2939575738895091
Diabetic std     : 0.38573929468677604
```

Out[22]:
```
Ttest_indResult(statistic=5.543824040855261, pvalue=4.670348451789813e-08)
```

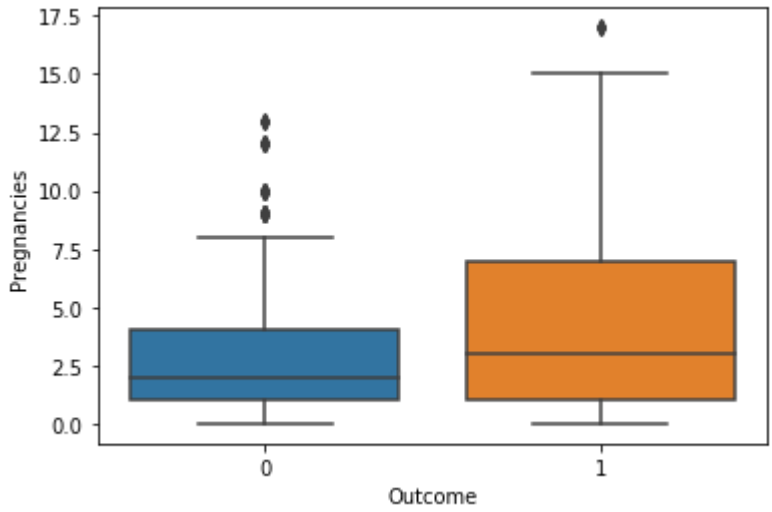
In [ ]:

In [ ]:

In [23]:
```
# Visualize feature statistics for non-diabetic and diabetic patients using in
teger/floats dataset
```

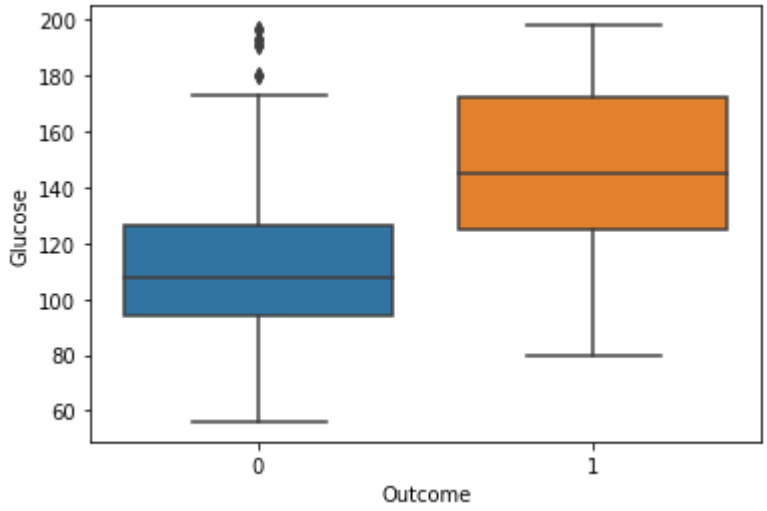
In [24]:
```
sns.boxplot(x='Outcome',y='Pregnancies',data=diabetes_data_floats)
```

Out[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ace9121c08>
```



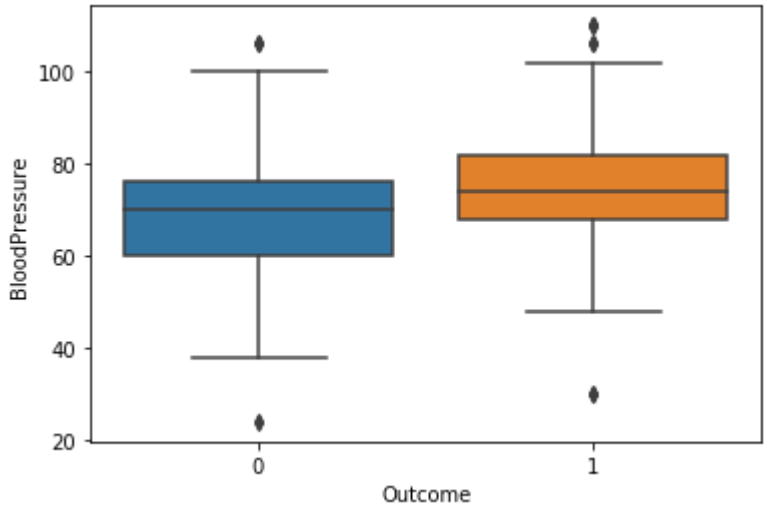
In [25]:
```
sns.boxplot(x='Outcome',y='Glucose',data=diabetes_data_floats)
```

Out[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1aceb230748>
```



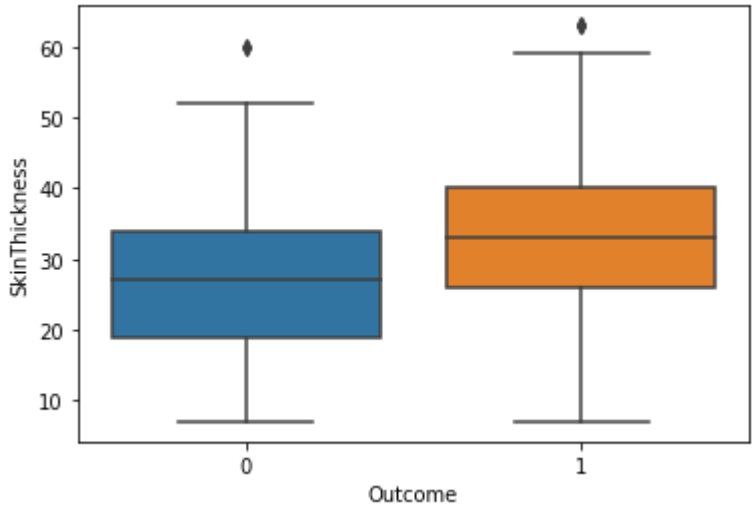
In [26]:
```
sns.boxplot(x='Outcome',y='BloodPressure',data=diabetes_data_floats)
```

Out[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ace730cb08>
```



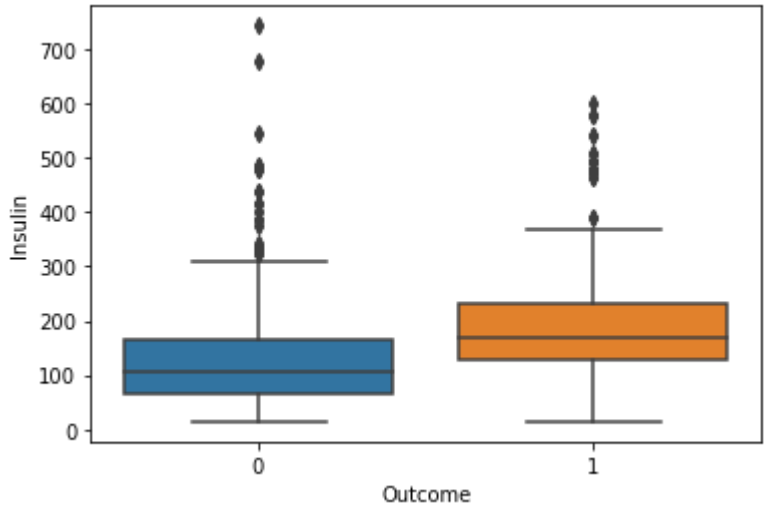
In [27]:
```
sns.boxplot(x='Outcome',y='SkinThickness',data=diabetes_data_floats)
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1aceb340c48>
```



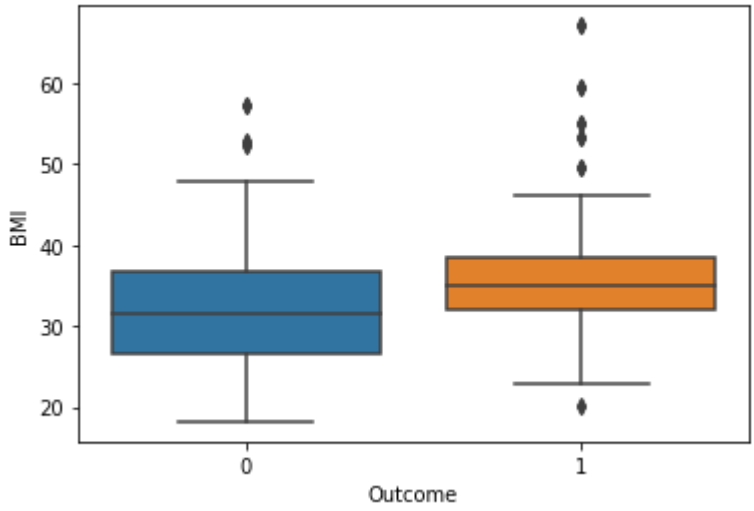
In [28]:
```
sns.boxplot(x='Outcome',y='Insulin',data=diabetes_data_floats)
```

Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1aceb3c7148>
```



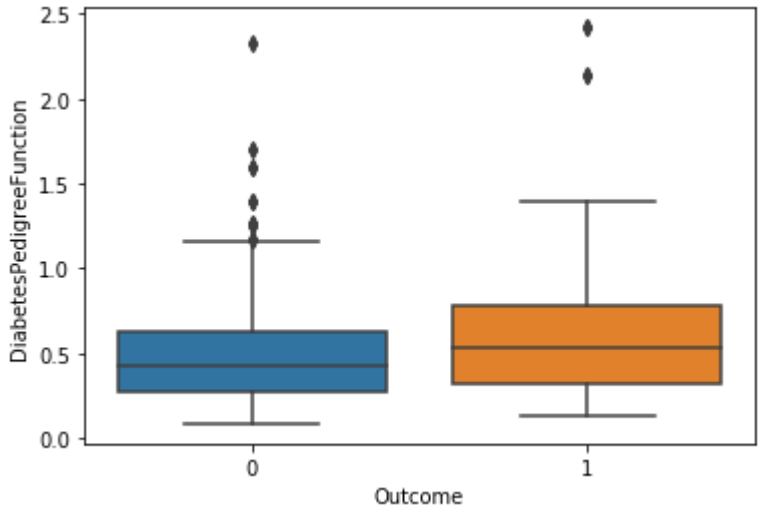
In [29]:
```
sns.boxplot(x='Outcome',y='BMI',data=diabetes_data_floats)
```

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x1aceb445d88>



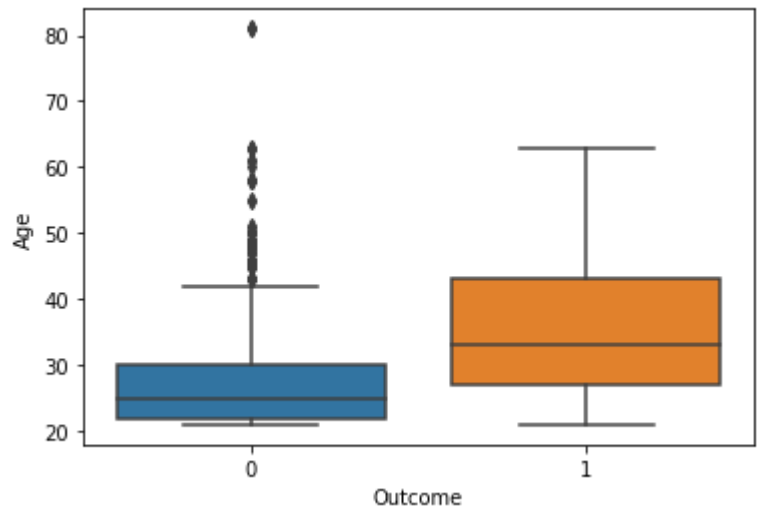
In [30]:
```
sns.boxplot(x='Outcome',y='DiabetesPedigreeFunction',data=diabetes_data_floats
)
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x1aceb4be388>



In [31]:
```
sns.boxplot(x='Outcome',y='Age',data=diabetes_data_floats)
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x1aceb531148>



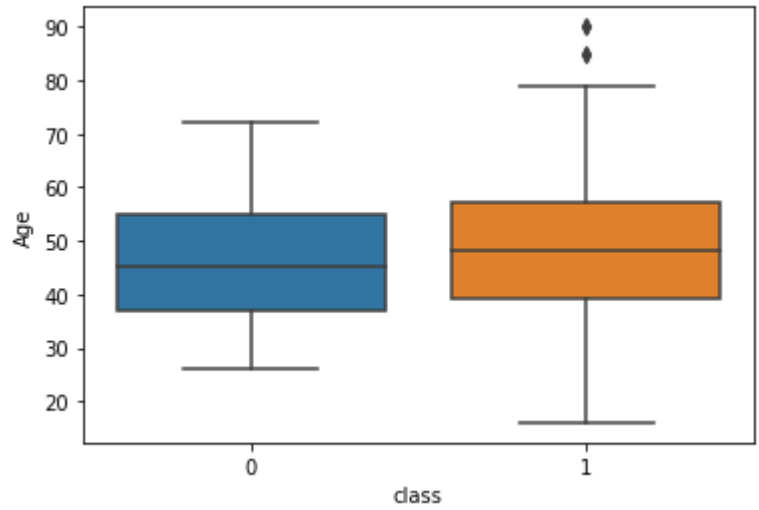
In [ ]:

In [ ]:

In [32]:
```
# Visualize feature statistics for non-diabetic and diabetic patients using bi
nary dataset
```

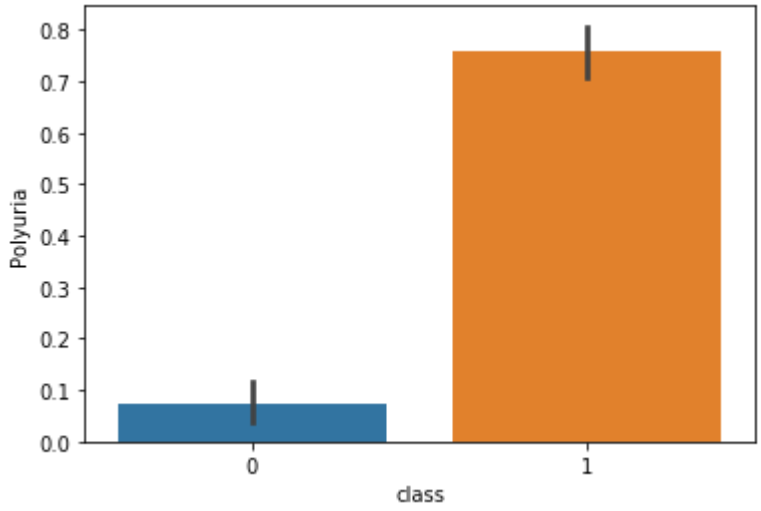
In [33]:
```
sns.boxplot(x='class',y='Age',data=diabetes_data_binary)
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x1aceb5b9f88>



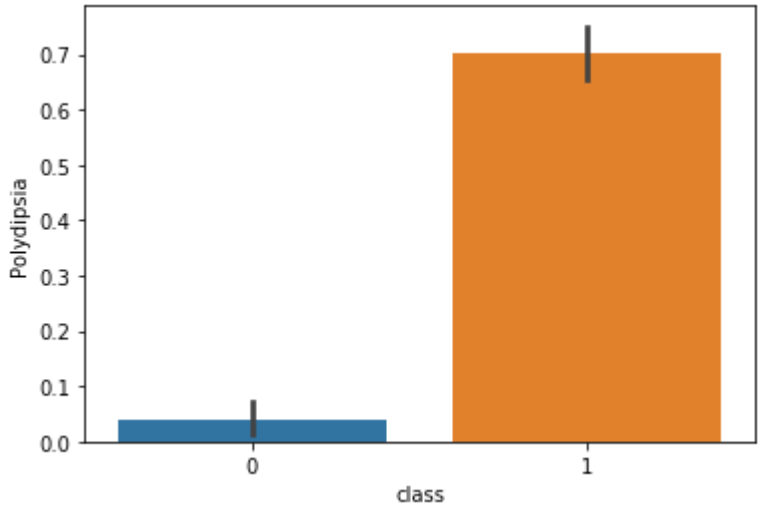
In [34]:
```
sns.barplot(x='class',y='Polyuria',data=diabetes_data_binary)
```

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x1aceb631908>



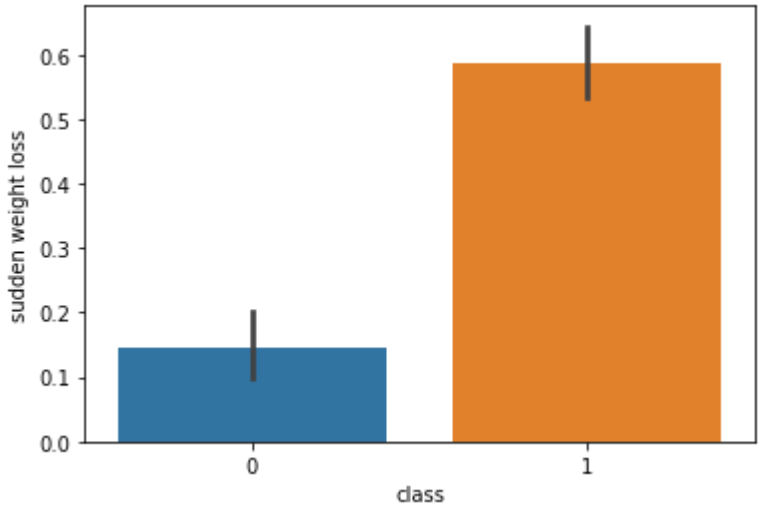
In [35]:
```
sns.barplot(x='class',y='Polydipsia',data=diabetes_data_binary)
```

Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0x1aceb69f8c8>



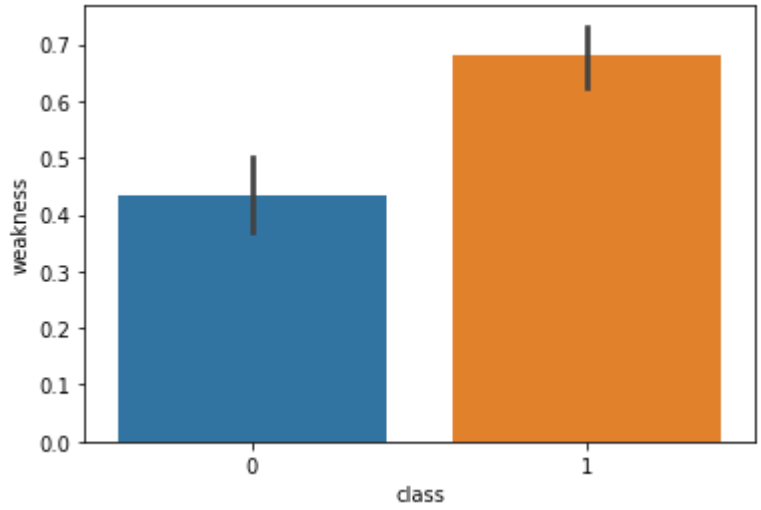
In [36]:
```
sns.barplot(x='class',y='sudden weight loss',data=diabetes_data_binary)
```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x1aceb702148>



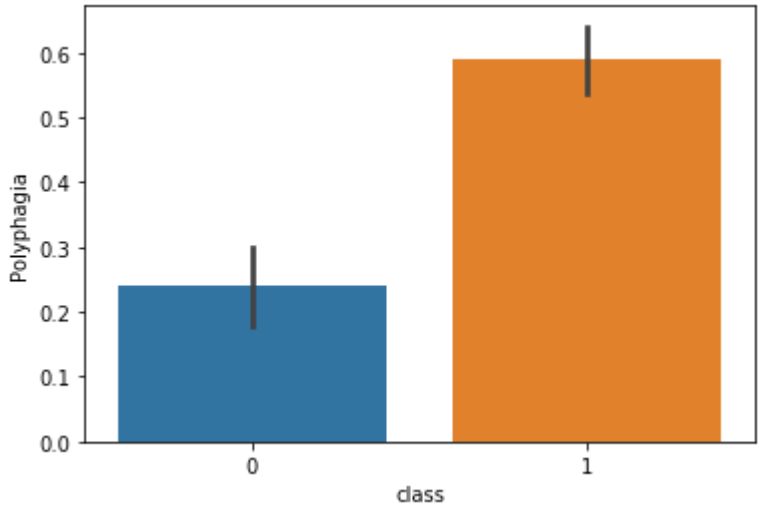
In [37]:
```
sns.barplot(x='class',y='weakness',data=diabetes_data_binary)
```

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x1aceb75fc88>



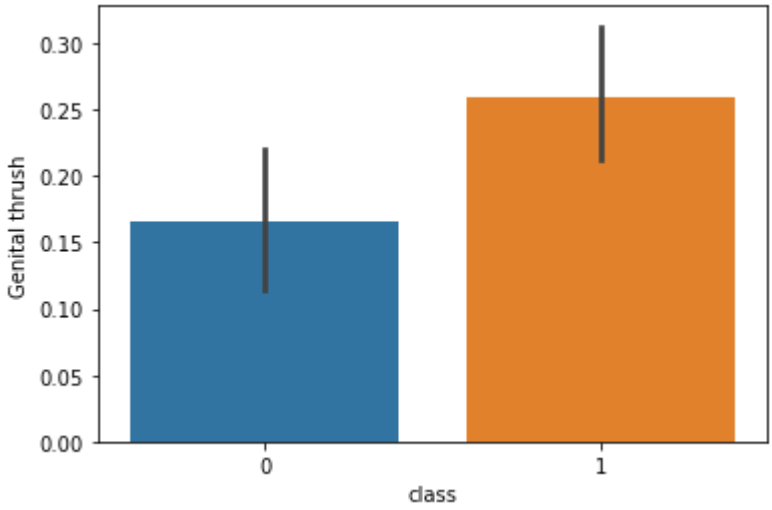
In [38]:
```
sns.barplot(x='class',y='Polyphagia',data=diabetes_data_binary)
```

Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x1aceb7acf08>



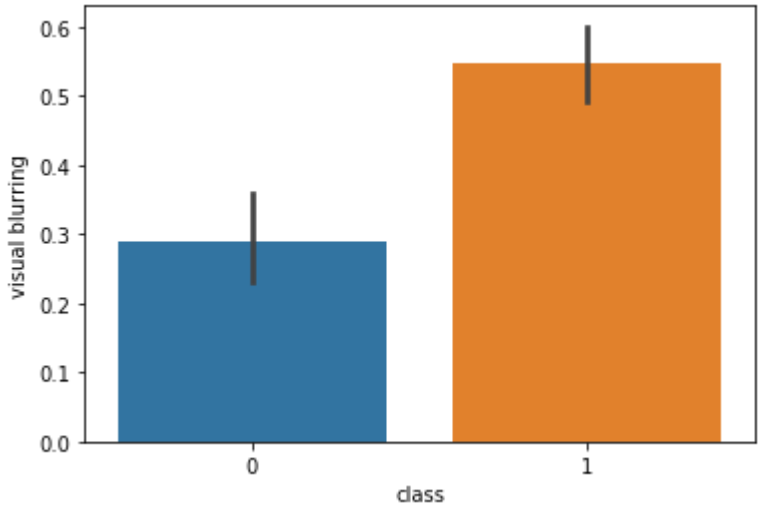
In [39]:
```
sns.barplot(x='class',y='Genital thrush',data=diabetes_data_binary)
```

Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x1aceb8282c8>



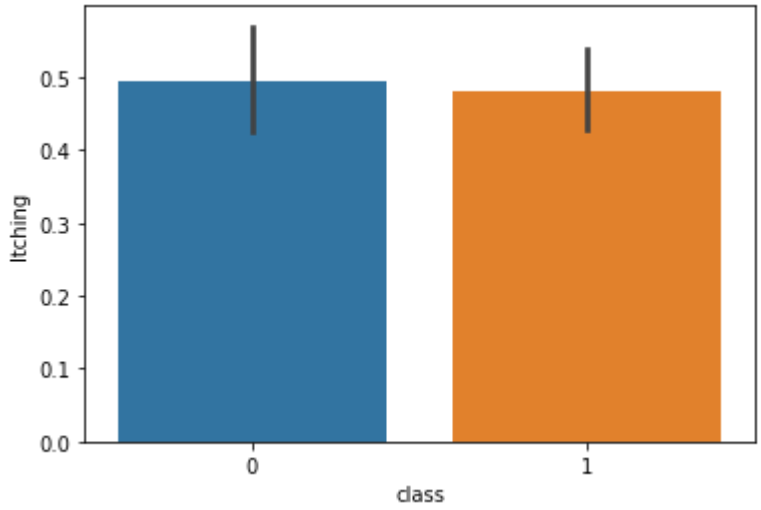
In [40]:
```
sns.barplot(x='class',y='visual blurring',data=diabetes_data_binary)
```

Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x1acec852d08>



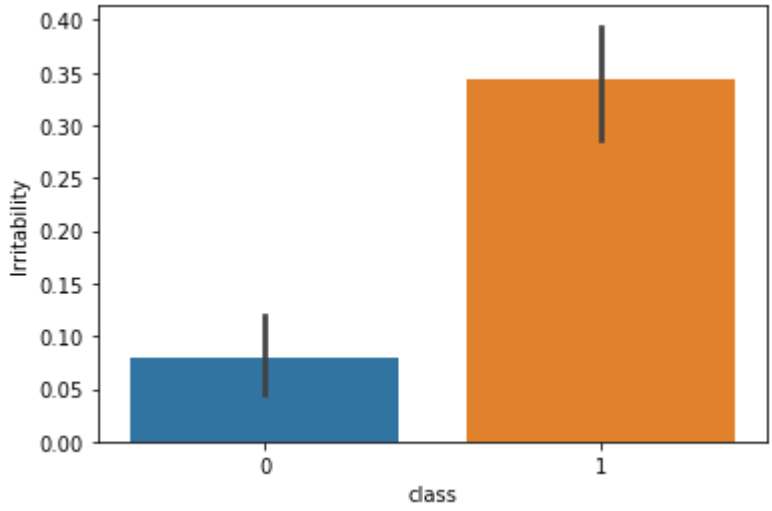
In [41]:
```
sns.barplot(x='class',y='Itching',data=diabetes_data_binary)
```

Out[41]: <matplotlib.axes._subplots.AxesSubplot at 0x1acec8b59c8>



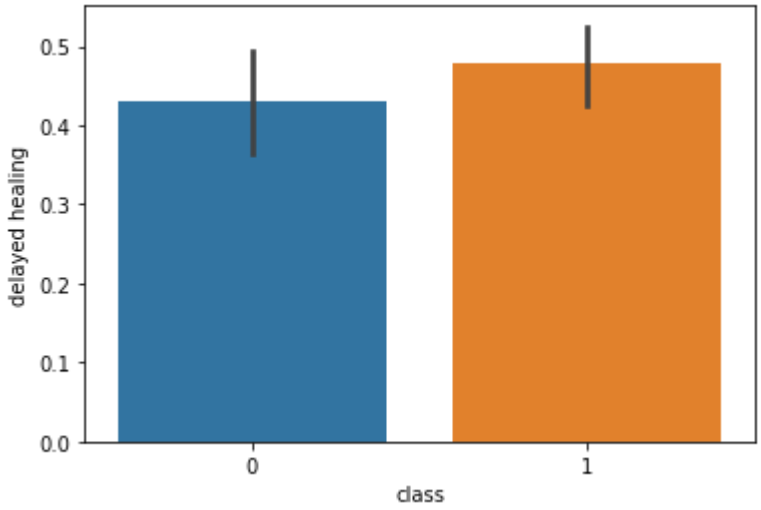
In [42]:
```
sns.barplot(x='class',y='Irritability',data=diabetes_data_binary)
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x1acec910388>



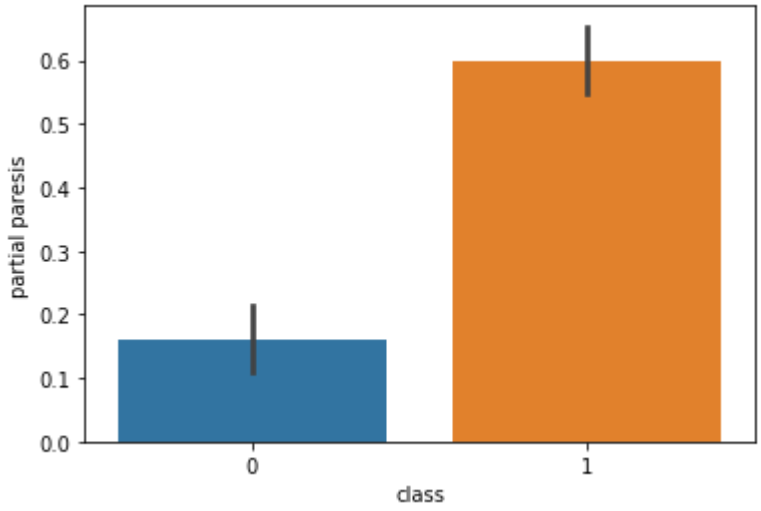
In [43]:
```
sns.barplot(x='class',y='delayed healing',data=diabetes_data_binary)
```

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x1acec8ae248>



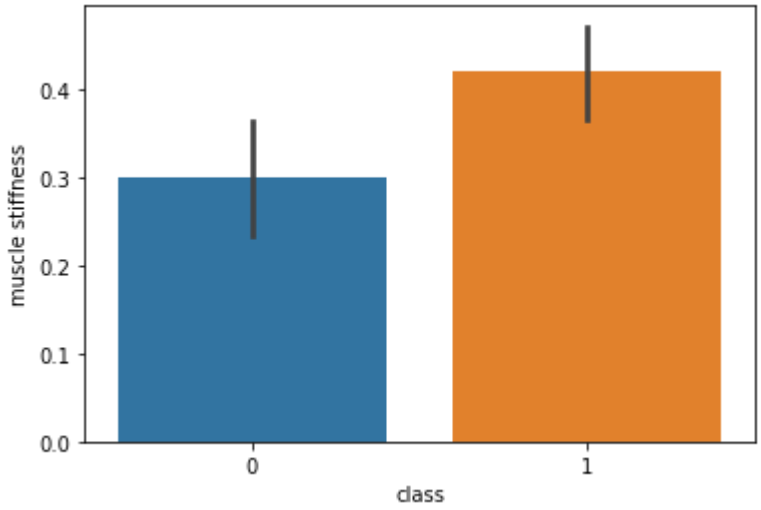
In [44]:
```
sns.barplot(x='class',y='partial paresis',data=diabetes_data_binary)
```

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x1acec915b88>



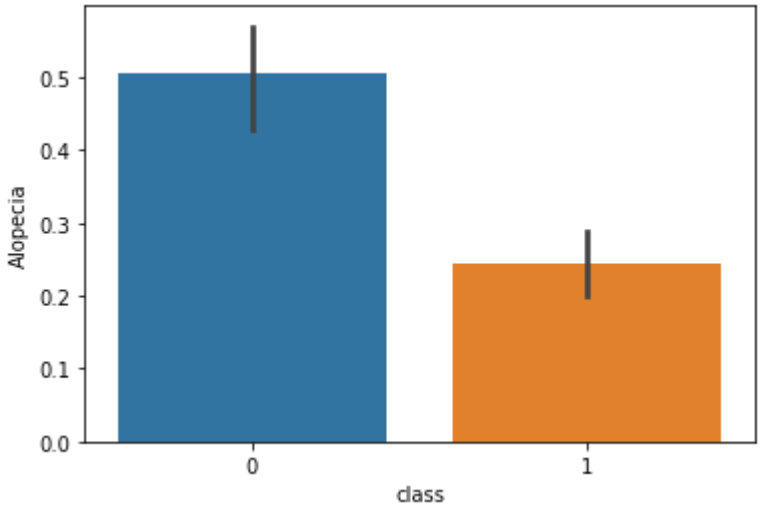
In [45]:
```
sns.barplot(x='class',y='muscle stiffness',data=diabetes_data_binary)
```

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x1aceca373c8>



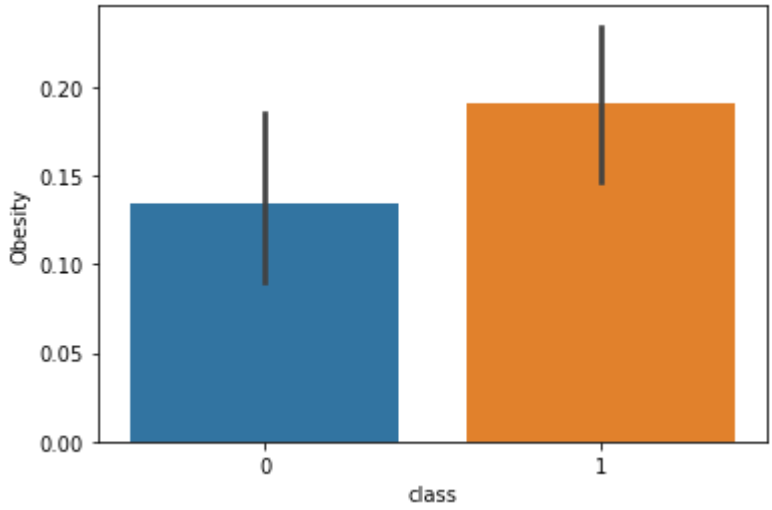
In [46]:
```
sns.barplot(x='class',y='Alopecia',data=diabetes_data_binary)
```

Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x1aceca94548>



In [47]:
```
sns.barplot(x='class',y='Obesity',data=diabetes_data_binary)
```

Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x1acecad5648>



In [ ]:
```
```

In [48]:
```
#Clustering using all the variables for floats
```

In [49]:
```
# Drop the outcome for X and our Y will be outcome
X1_total = diabetes_data_floats.drop("Outcome",axis = 1)
Y1_total = diabetes_data_floats['Outcome']
```

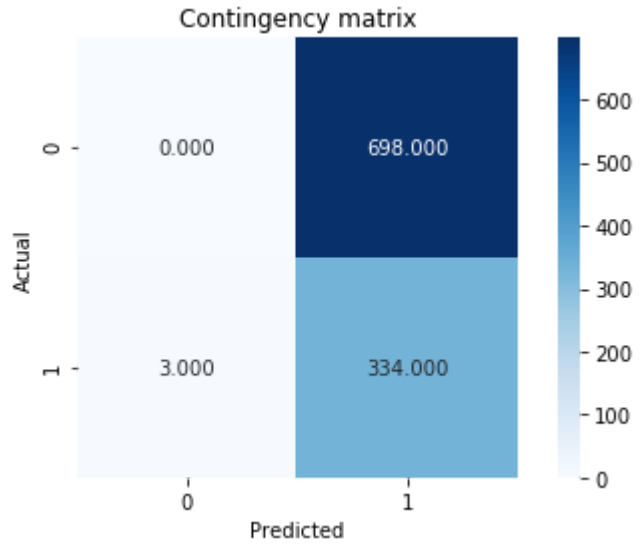
In [50]:
```
#Scale data in X
scaler = StandardScaler()
scaler.fit(X1_total)
X1_total_scaled = scaler.transform(X1_total)
```

In [ ]:
```
```

In [51]:
```
#Hierarchical Clustering with single linkage
```

In [52]:
```
clustering = linkage(X1_total_scaled,method = 'single',metric = 'euclidean')
clusters = fcluster(clustering, 2,criterion = 'maxclust')
```

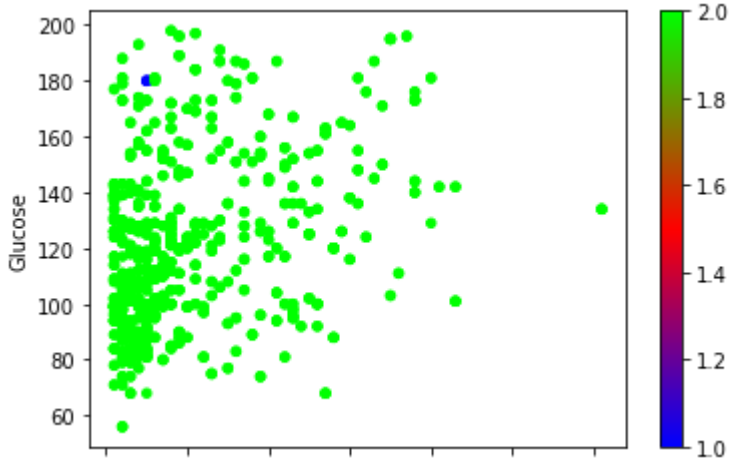
In [53]:
```
#Plot contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y1_total,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

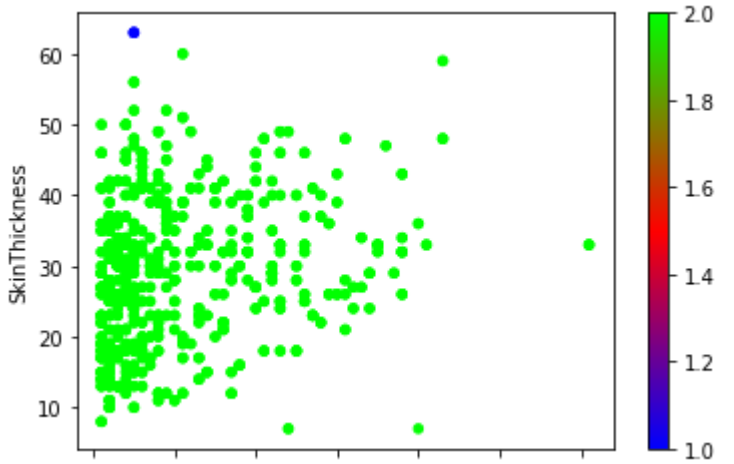


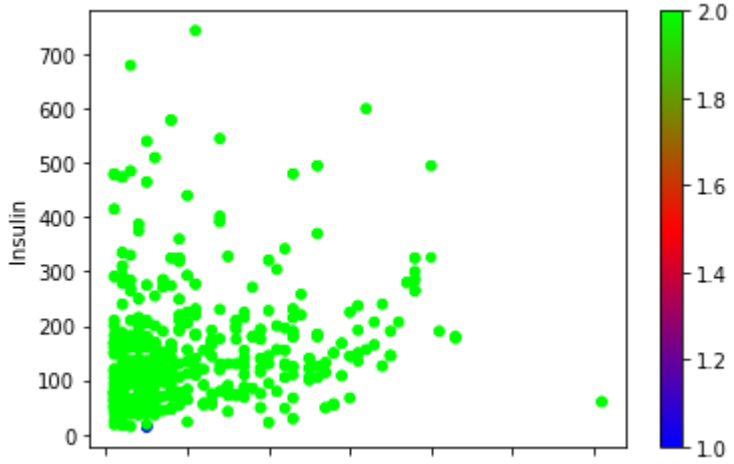
In [54]:
```
#Calculate the adjusted rand index and silhouette coefficient
adjusted_rand_index = metrics.adjusted_rand_score(Y1_total, clusters)
silhouette_coefficient = metrics.silhouette_score(X1_total_scaled, clusters, m
etric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

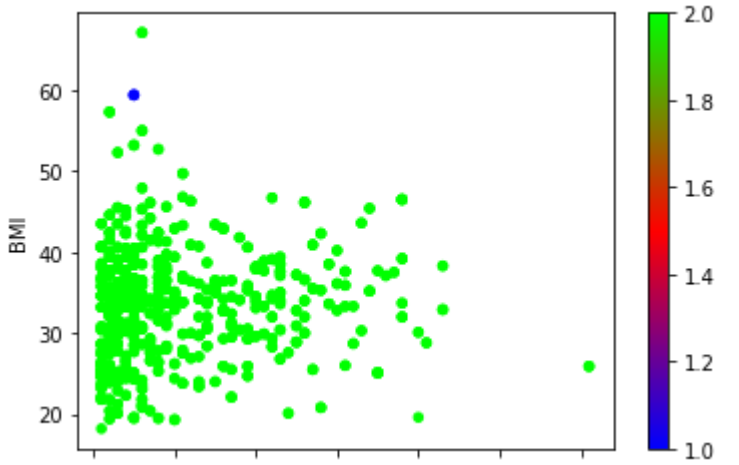
```
[0.0062224833722829085, 0.5472517225709225]
```

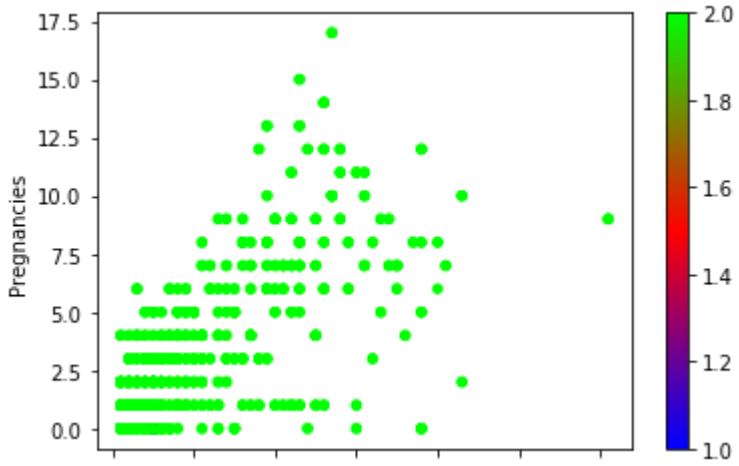
In [55]:
```
#Plot the data using the clusters formed by Hierarchical clustering
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Glucose', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'SkinThicknes
s', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Insulin', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'BMI', c = clu
sters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Pregnancies',
c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'BloodPressur
e', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'DiabetesPedig
reeFunction', c = clusters, colormap = plt.cm.brg)
```

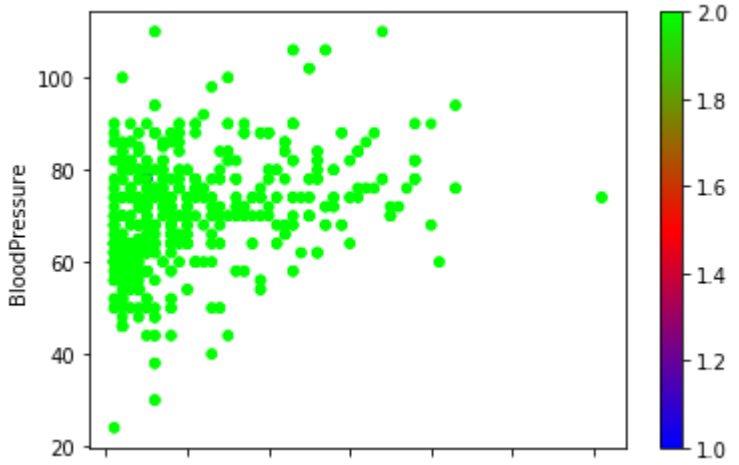

Glucose
200
180
160
140
120
100
80
60
2.0
1.8
1.6
1.4
1.2
1.0

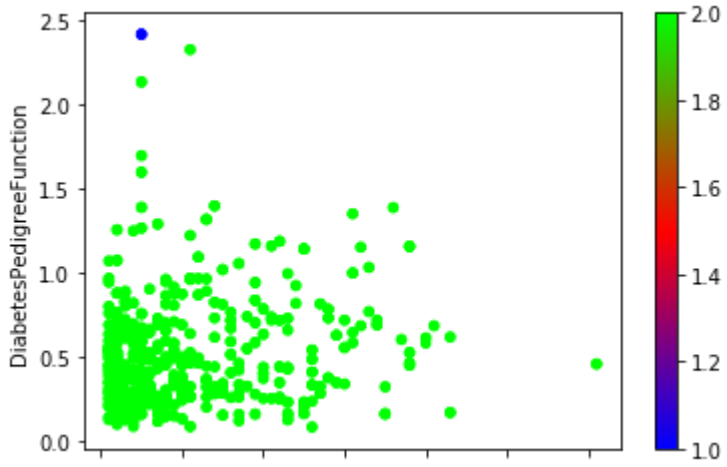

SkinThickness
60
50
40
30
20
10
2.0
1.8
1.6
1.4
1.2
1.0


Insulin
700
600
500
400
300
200
100
0
2.0
1.8
1.6
1.4
1.2
1.0


BMI
60
50
40
30
20
2.0
1.8
1.6
1.4
1.2
1.0


Pregnancies
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
2.0
1.8
1.6
1.4
1.2
1.0


BloodPressure
100
80
60
40
20
2.0
1.8
1.6
1.4
1.2
1.0

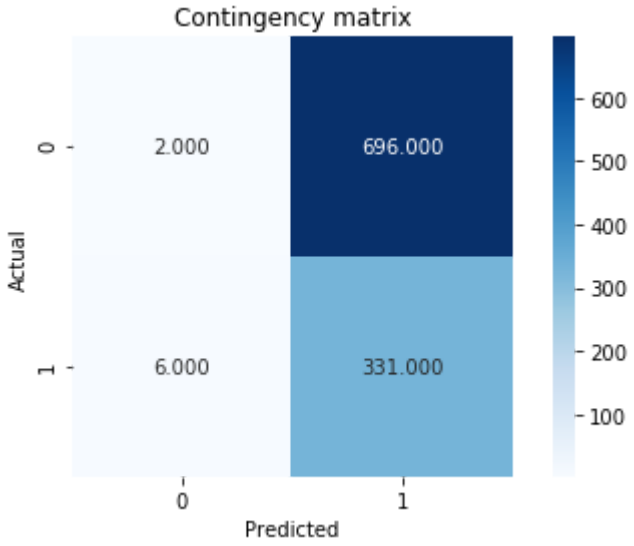


In [ ]:

In [56]:
```
#Hierarchical Clustering with complete linkage
```

In [57]:
```
#Complete linkage clustering
clustering = linkage(X1_total_scaled, method = "complete", metric = "euclidea
n")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
```

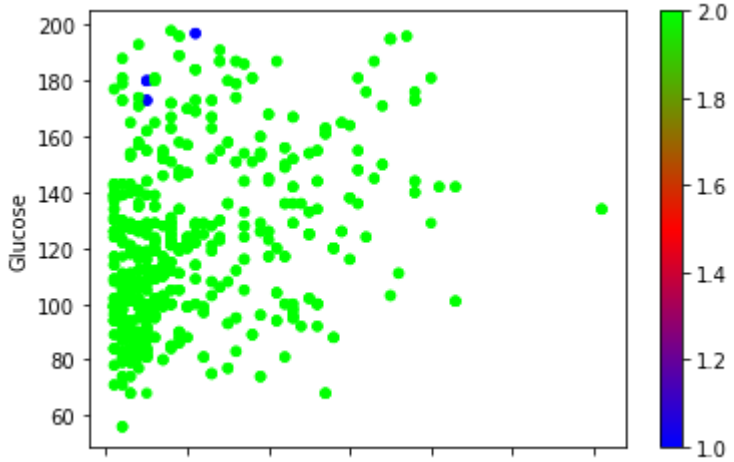
In [58]:
```
#Plot the contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y1_total,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

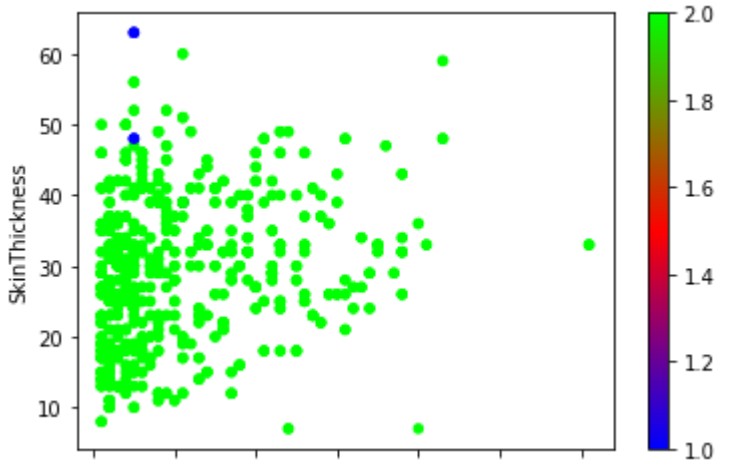


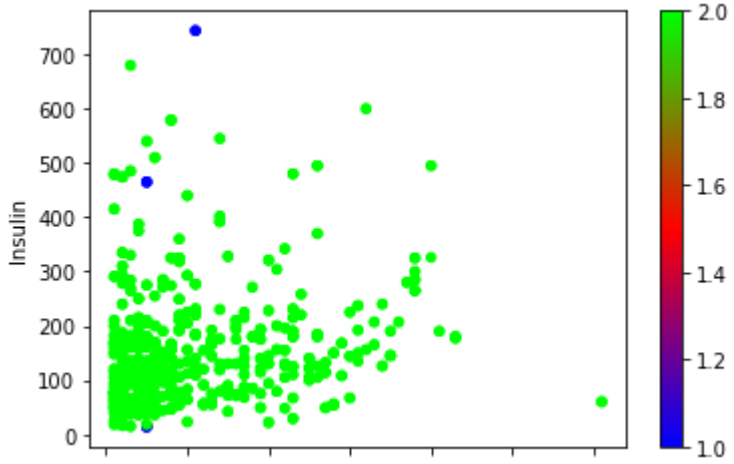
In [59]:
```
#Calculate adjusted rand index and silhouette coefficinet
adjusted_rand_index = metrics.adjusted_rand_score(Y1_total, clusters)
silhouette_coefficient = metrics.silhouette_score(X1_total_scaled, clusters, m
etric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

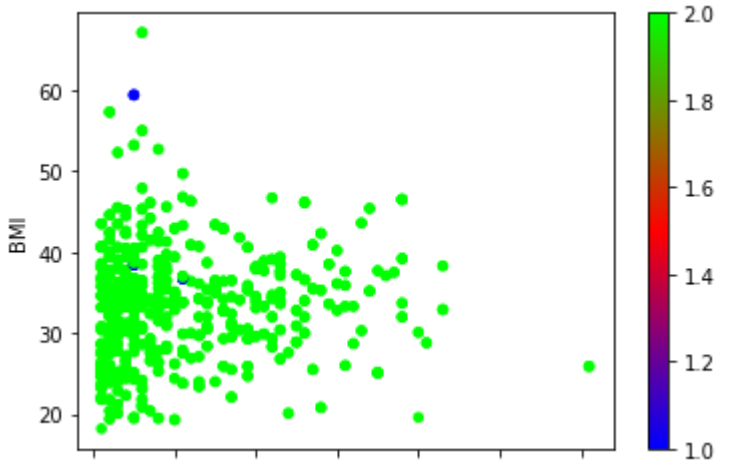
```
[0.010382533211682976, 0.517795170823348]
```

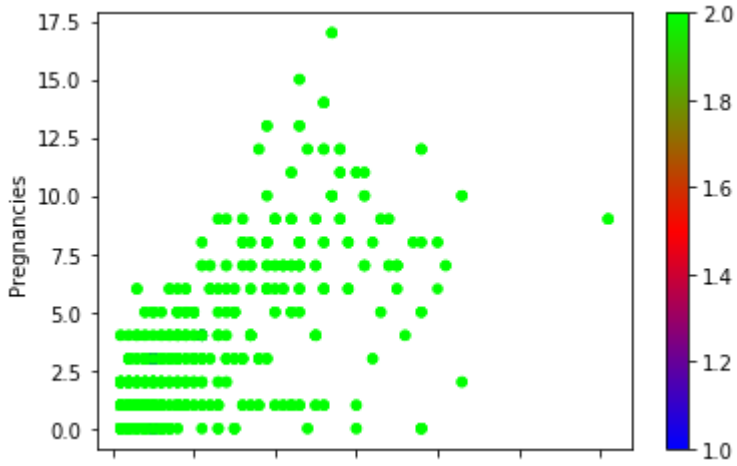
In [60]:
```
#Plot the data using the clusters formed by Hierarchical clustering
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Glucose', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'SkinThicknes
s', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Insulin', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'BMI', c = clu
sters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Pregnancies',
c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'BloodPressur
e', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'DiabetesPedig
reeFunction', c = clusters, colormap = plt.cm.brg)
```

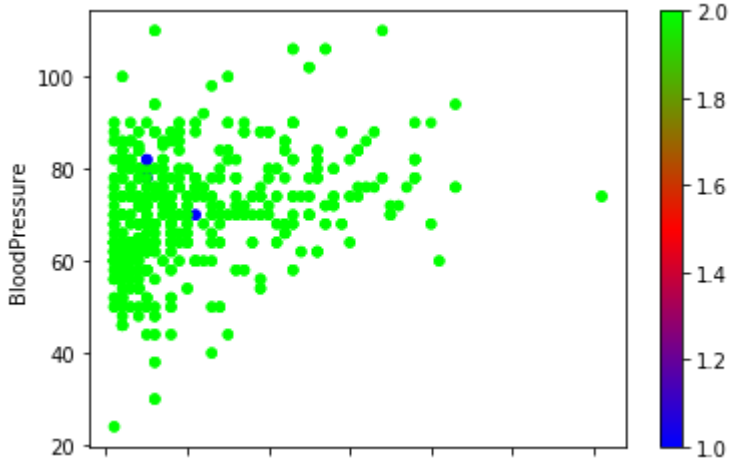

Glucose
200
180
160
140
120
100
80
60
2.0
1.8
1.6
1.4
1.2
1.0

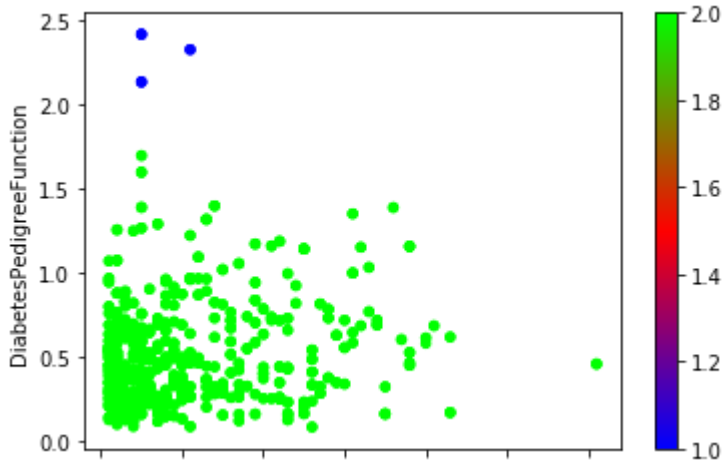

SkinThickness
60
50
40
30
20
10
2.0
1.8
1.6
1.4
1.2
1.0


Insulin
700
600
500
400
300
200
100
0
2.0
1.8
1.6
1.4
1.2
1.0


BMI
60
50
40
30
20
2.0
1.8
1.6
1.4
1.2
1.0


Pregnancies
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
2.0
1.8
1.6
1.4
1.2
1.0

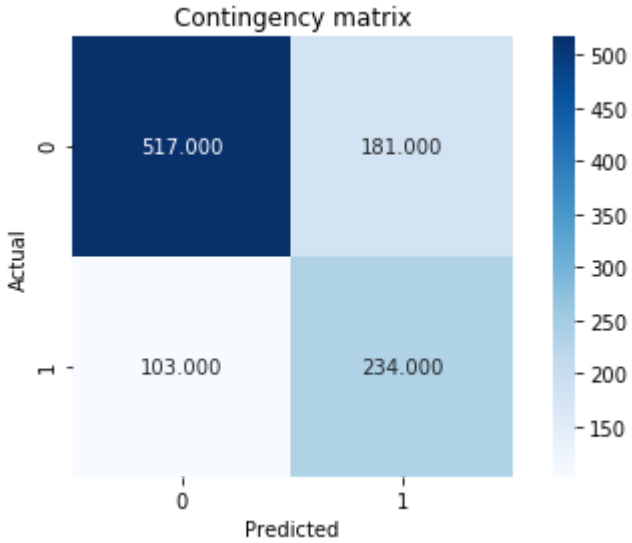

BloodPressure
100
80
60
40
20
2.0
1.8
1.6
1.4
1.2
1.0



In [61]:
```
#KMeans Clustering iteration = 2 and choosing the centroids randomly
```

In [62]:
```
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 2, random_state
= 0).fit(X1_total_scaled)
clusters = clustering.labels_
```

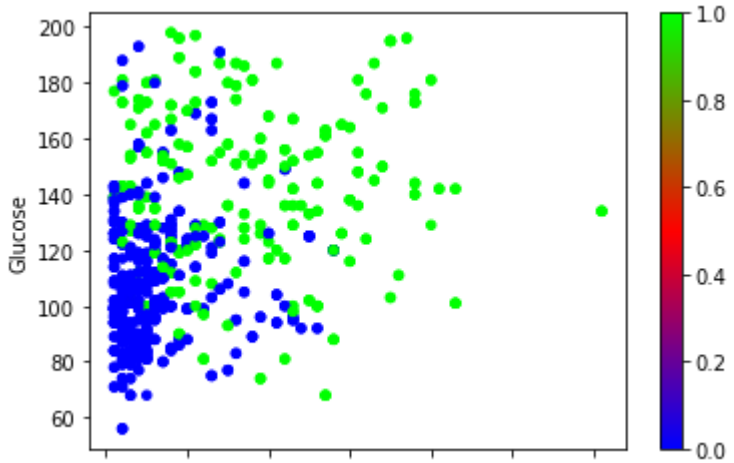
In [63]:
```
#Plot the contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y1_total,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

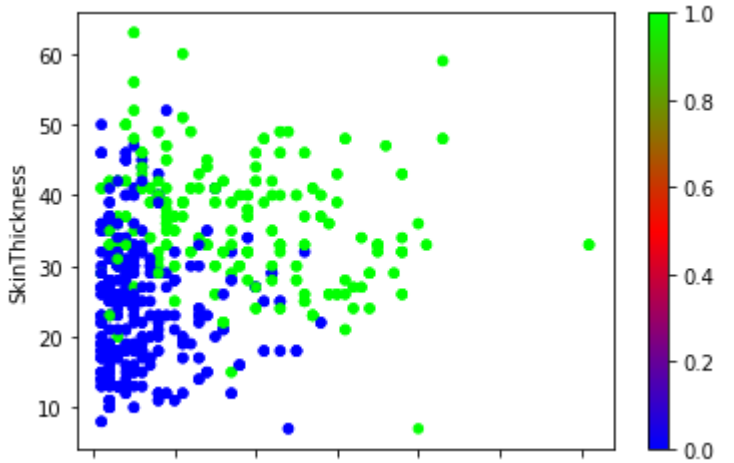


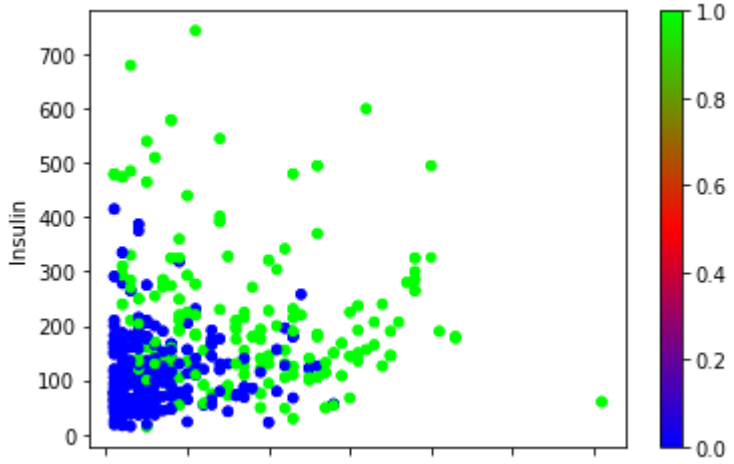
In [64]:
```
#calculate the adjusted rand index and silohouette coeff
adjusted_rand_index = metrics.adjusted_rand_score(Y1_total, clusters)
silhouette_coefficient = metrics.silhouette_score(X1_total_scaled, clusters, m
etric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

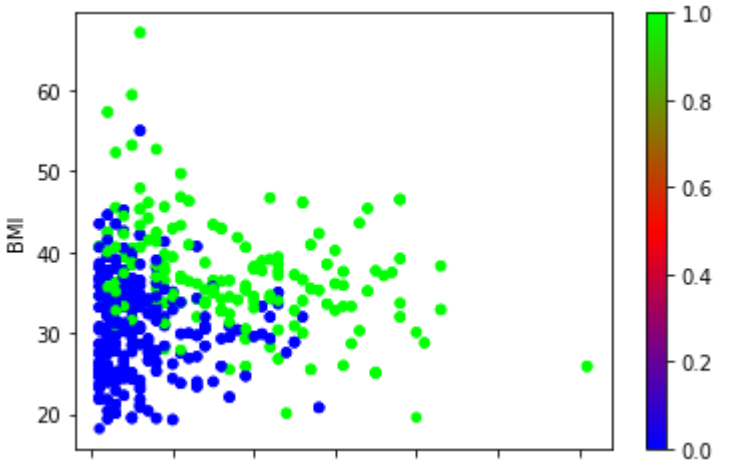
```
[0.19911241484319336, 0.23302435850048664]
```

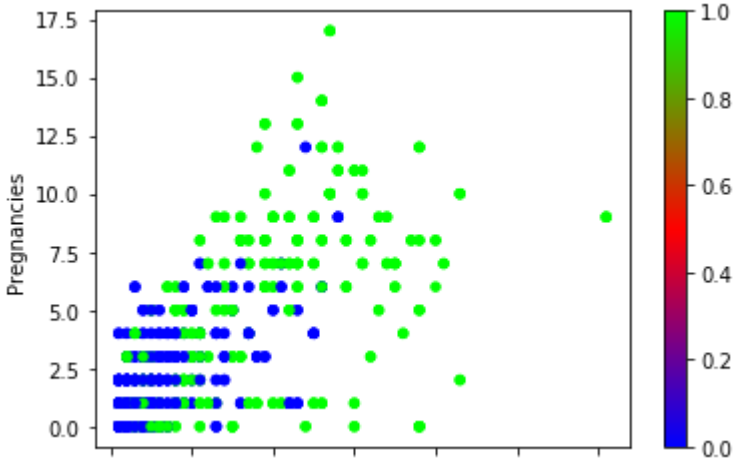
In [65]:
```
#Plot the clusters formed by KMeans clustering
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Glucose', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'SkinThicknes
s', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Insulin', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'BMI', c = clu
sters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Pregnancies',
c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'BloodPressur
e', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'DiabetesPedig
reeFunction', c = clusters, colormap = plt.cm.brg)
```

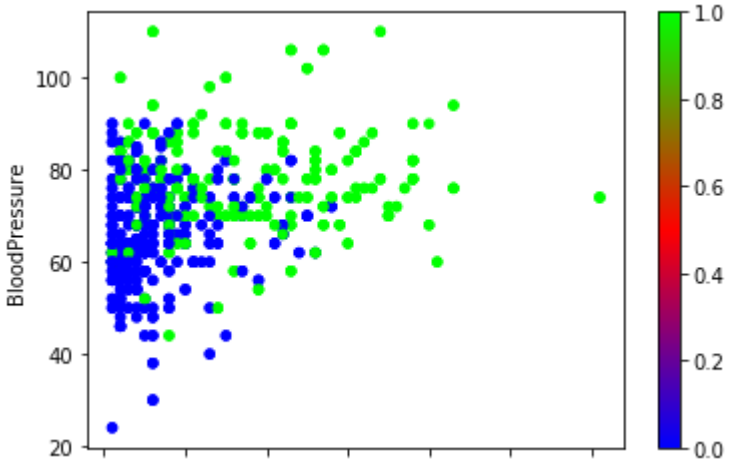

Glucose
200
180
160
140
120
100
80
60
1.0
0.8
0.6
0.4
0.2
0.0

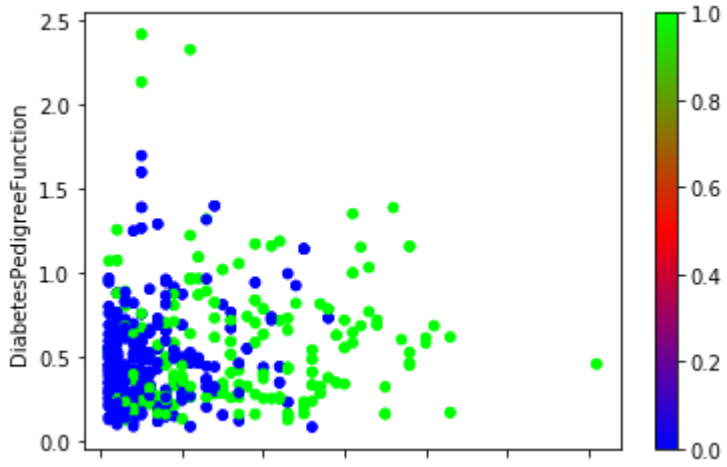

SkinThickness
60
50
40
30
20
10
1.0
0.8
0.6
0.4
0.2
0.0


Insulin
700
600
500
400
300
200
100
0
1.0
0.8
0.6
0.4
0.2
0.0


BMI
60
50
40
30
20
1.0
0.8
0.6
0.4
0.2
0.0


Pregnancies
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1.0
0.8
0.6
0.4
0.2
0.0


BloodPressure
100
80
60
40
20
1.0
0.8
0.6
0.4
0.2
0.0



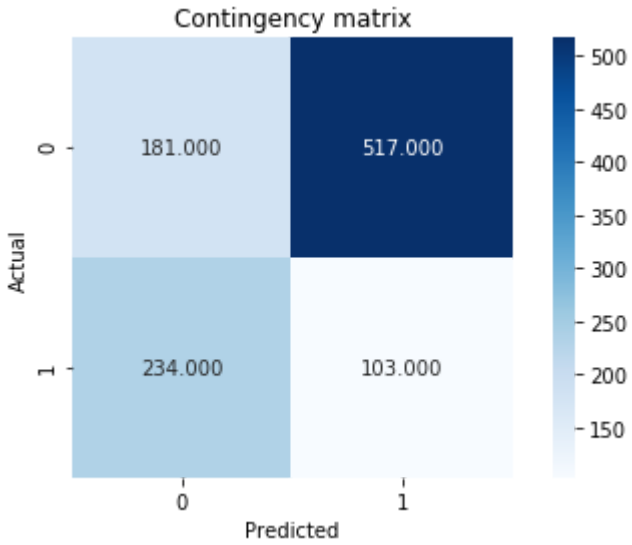
In [ ]:

In [66]:
```
#KMeans using k-means++ and 20 iteration
```

In [ ]:

In [67]:
```
clustering = KMeans(n_clusters = 2, init = 'k-means++', n_init = 20,random_sta
te = 0).fit(X1_total_scaled)
clusters = clustering.labels_
```

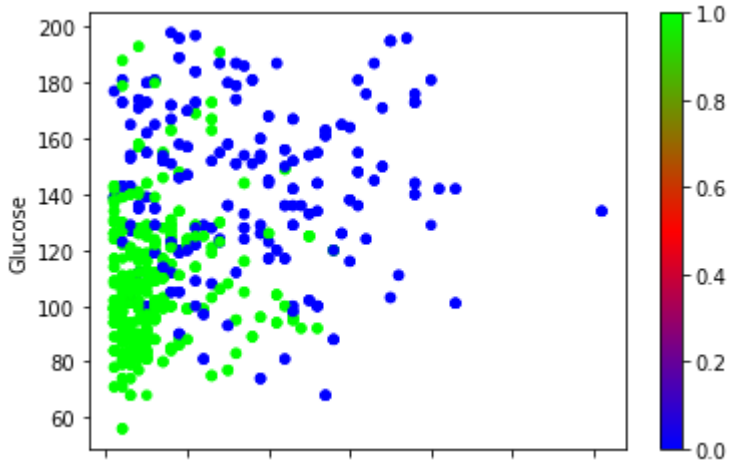
In [68]:
```
#Plot the concingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y1_total,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

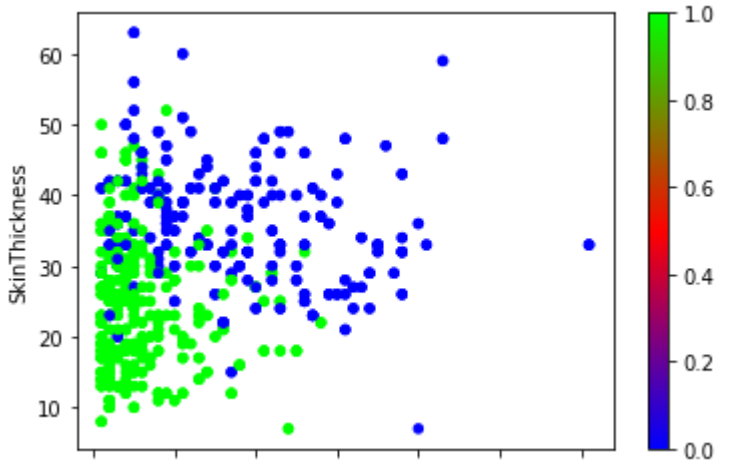


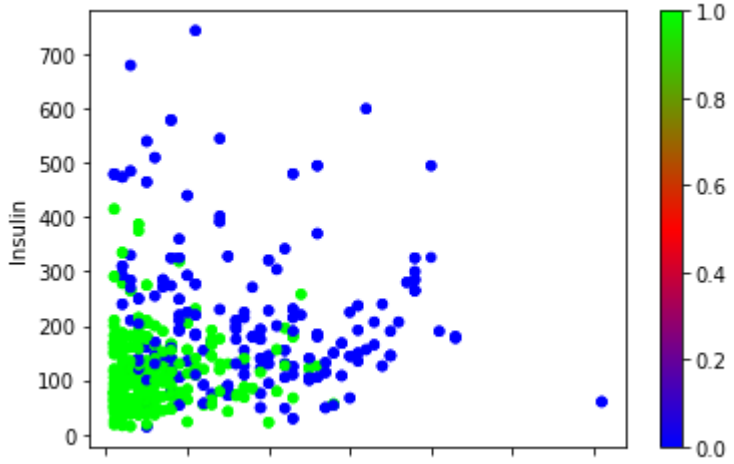
In [69]:
```
#Calculate the adjusted rand index and silhouette coeff
adjusted_rand_index = metrics.adjusted_rand_score(Y1_total, clusters)
silhouette_coefficient = metrics.silhouette_score(X1_total_scaled, clusters, m
etric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

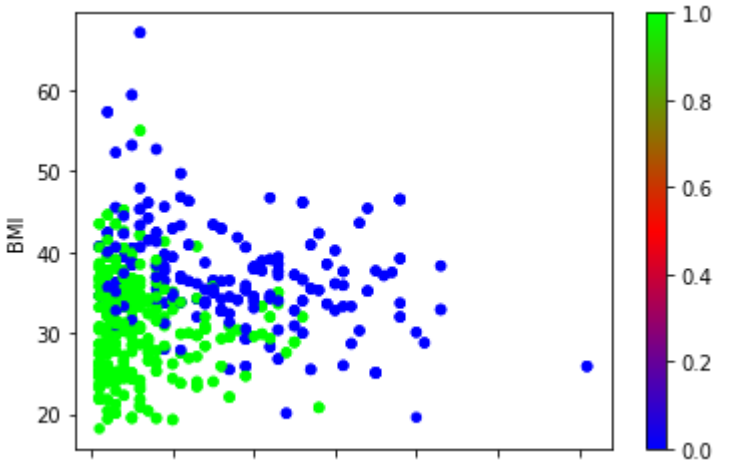
```
[0.19911241484319336, 0.23302435850048664]
```

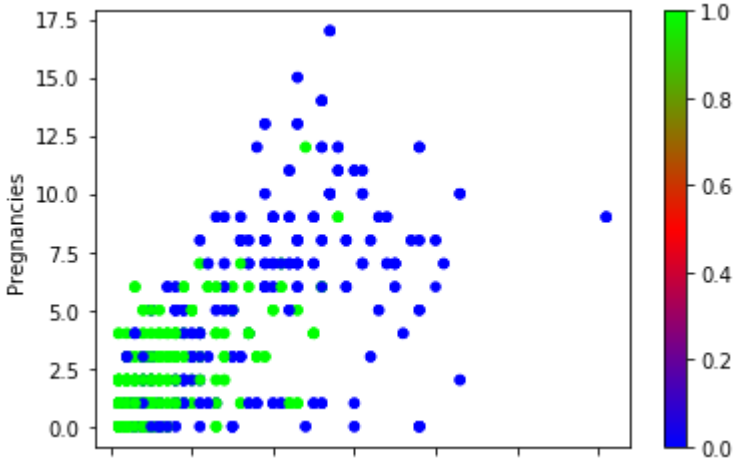
In [70]:
```
#Plot the clusters formed by KMeans clustering
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Glucose', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'SkinThicknes
s', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Insulin', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'BMI', c = clu
sters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Pregnancies',
c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'BloodPressur
e', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'DiabetesPedig
reeFunction', c = clusters, colormap = plt.cm.brg)
```

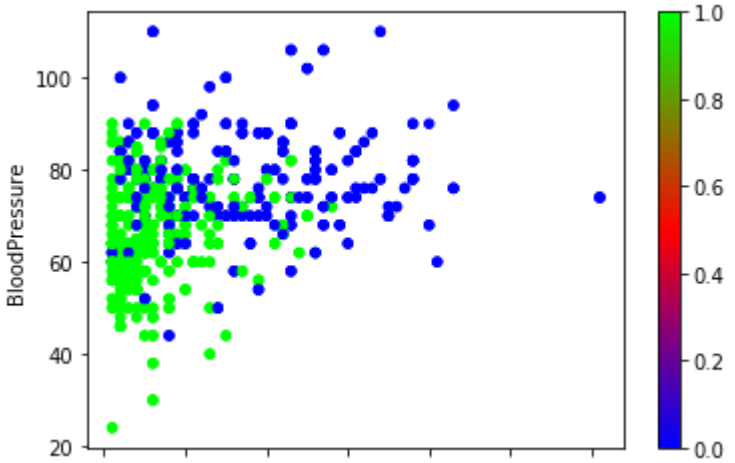

Glucose
200
180
160
140
120
100
80
60
1.0
0.8
0.6
0.4
0.2
0.0

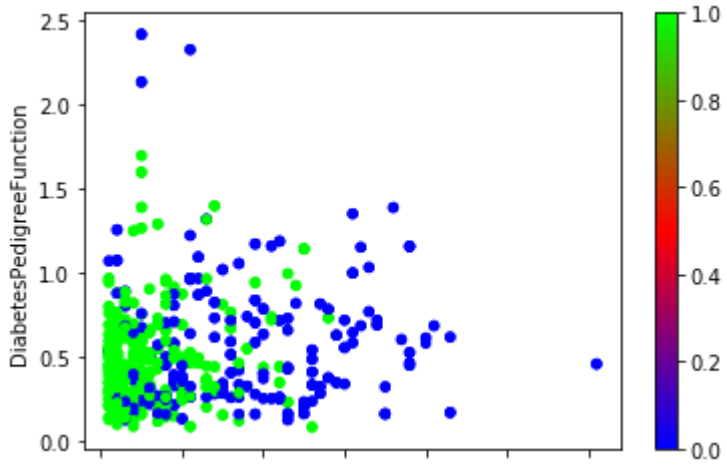

SkinThickness
60
50
40
30
20
10
1.0
0.8
0.6
0.4
0.2
0.0


Insulin
700
600
500
400
300
200
100
0
1.0
0.8
0.6
0.4
0.2
0.0


BMI
60
50
40
30
20
1.0
0.8
0.6
0.4
0.2
0.0


Pregnancies
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1.0
0.8
0.6
0.4
0.2
0.0


BloodPressure
100
80
60
40
20
1.0
0.8
0.6
0.4
0.2
0.0

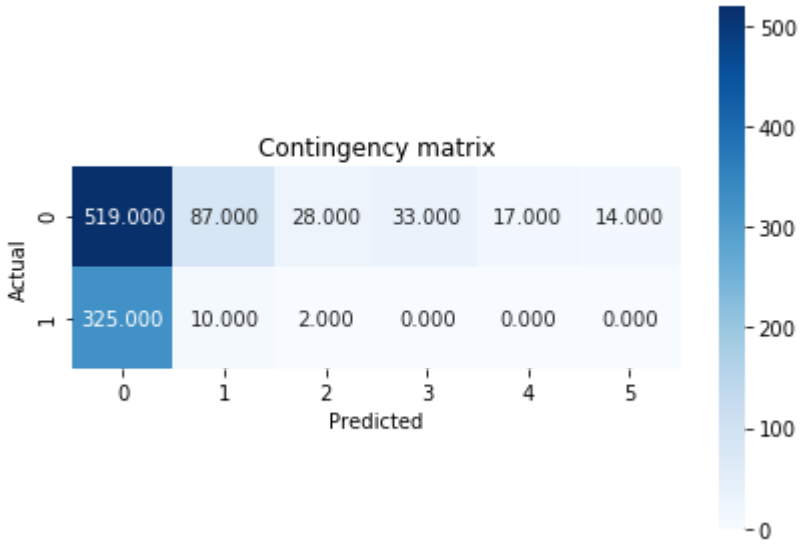


In [ ]:

In [71]:
```
#DBSCAN for eps 1 and min_sample 10
```

In [72]:
```
clustering = DBSCAN(eps = 1, min_samples = 10, metric = "euclidean").fit(X1_to
tal_scaled)
clusters = clustering.labels_
```

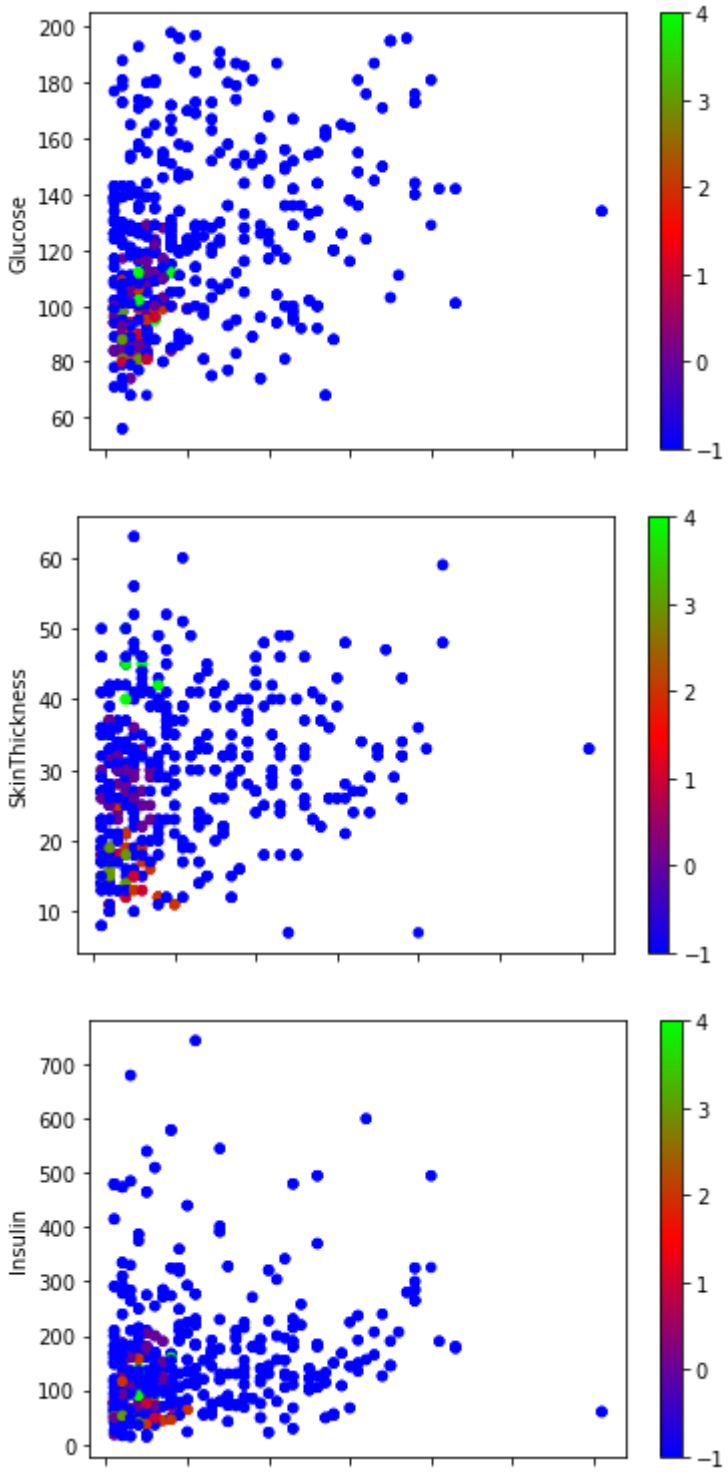
In [73]:
```
#Plot the contingency matrix and we have 3 clusters forms because our min_ sam
ple is small for the eps value
cont_matrix = metrics.cluster.contingency_matrix(Y1_total, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

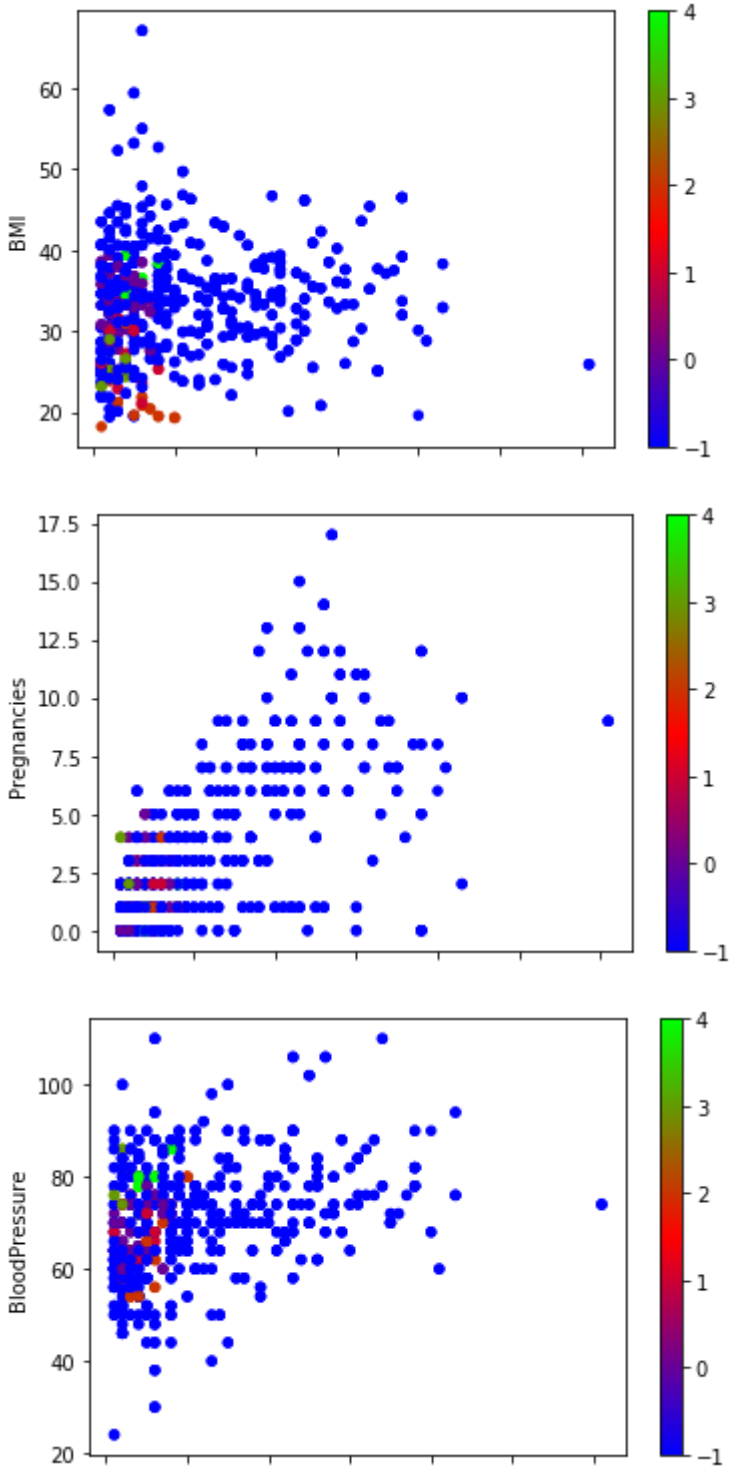


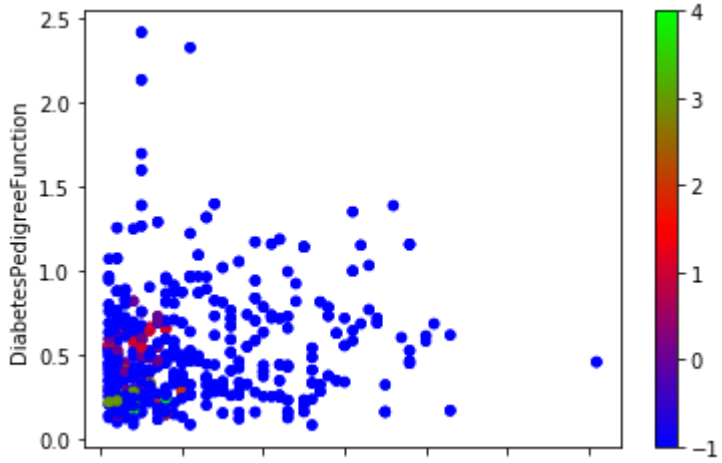
In [74]:
```
#Calculate the rand index
adjusted_rand_index = metrics.adjusted_rand_score(Y1_total, clusters)
silhouette_coefficient = metrics.silhouette_score(X1_total_scaled, clusters, m
etric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[-0.08311410303273457, -0.16423138081070643]
```

In [75]:
```
#Plot the clusters formed by
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Glucose', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'SkinThicknes
s', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Insulin', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'BMI', c = clu
sters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Pregnancies',
c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'BloodPressur
e', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'DiabetesPedig
reeFunction', c = clusters, colormap = plt.cm.brg)
```


Glucose
SkinThickness
Insulin

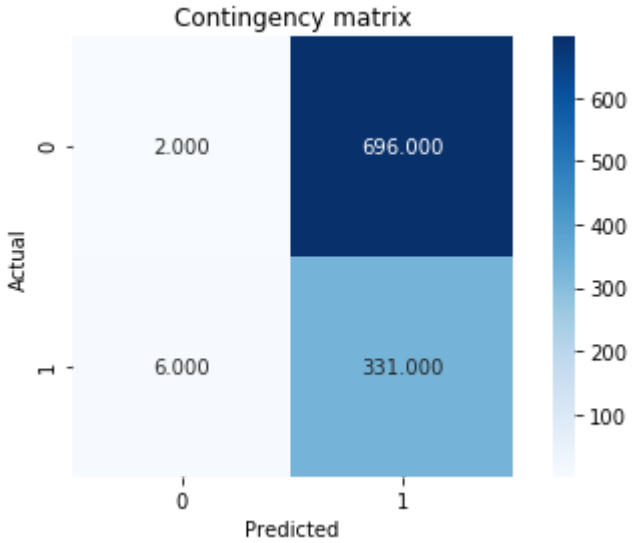

BMI
60
50
40
30
20
4
3
2
1
0
−1
Pregnancies
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
4
3
2
1
0
−1
BloodPressure
100
80
60
40
20
4
3
2
1
0
−1



In [76]:
```
#DBSCAN with increase in min_sample and eps
```

In [77]:
```
clustering = DBSCAN(eps = 3, min_samples = 20, metric = "euclidean").fit(X1_to
tal_scaled)
clusters = clustering.labels_
```

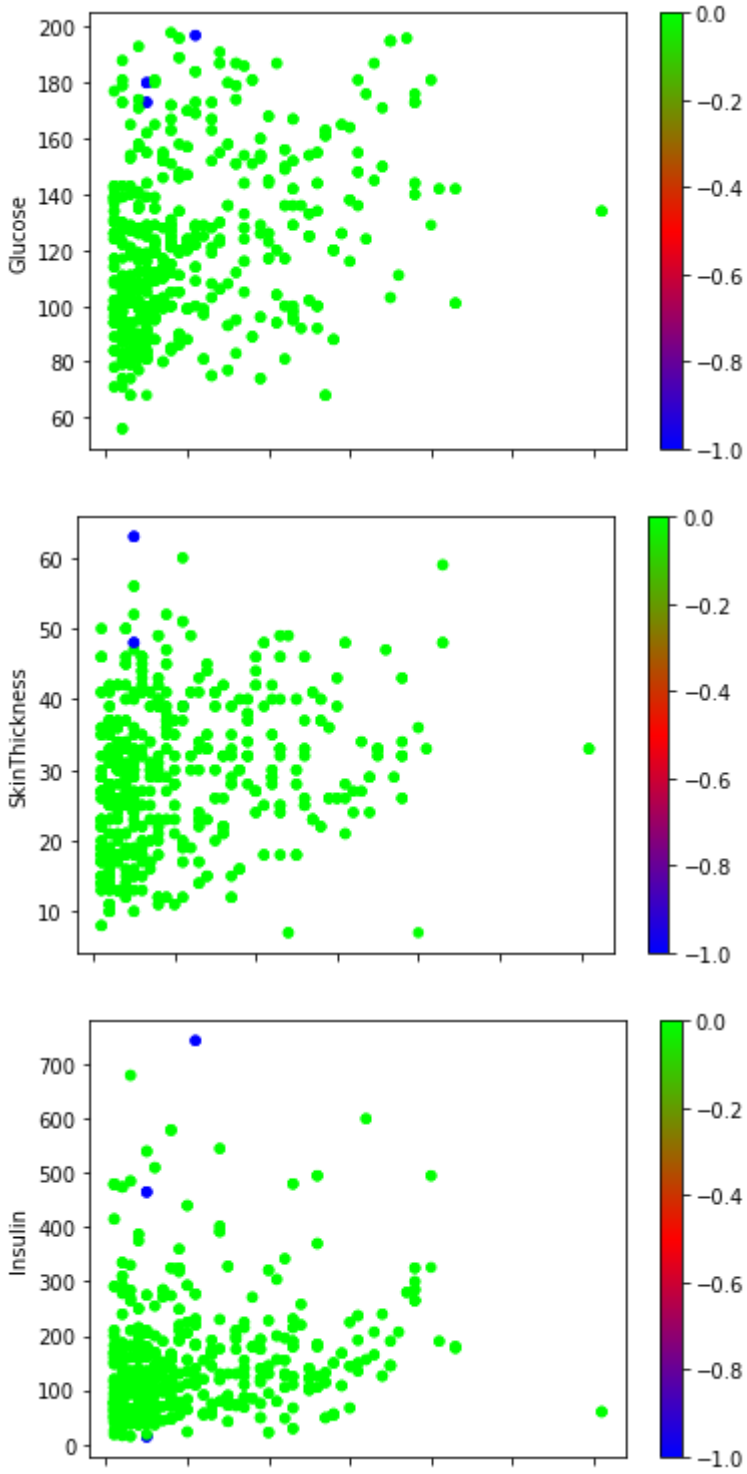
In [78]:
```
#Plot the contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y1_total, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

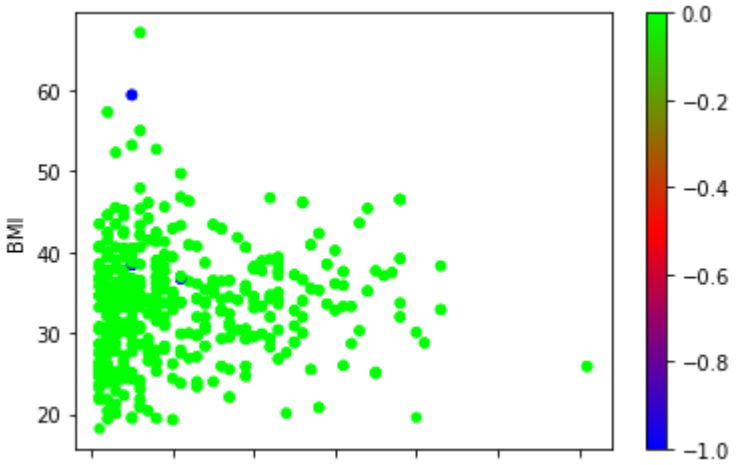


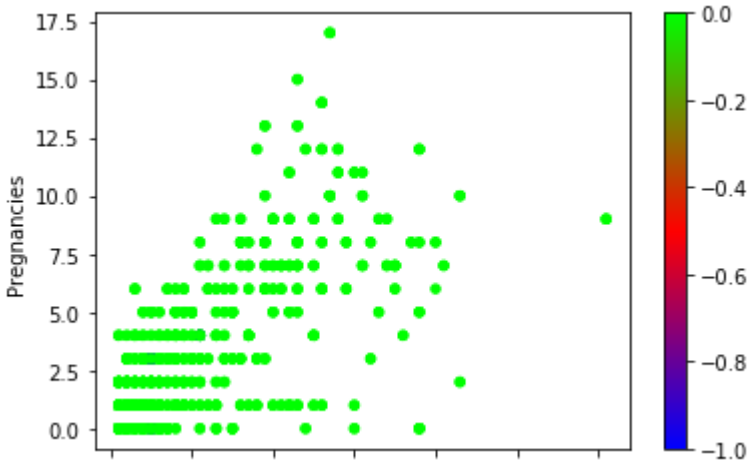
In [79]:
```
#Calculate the values of adjusted rand index and  silhouette coeff
adjusted_rand_index = metrics.adjusted_rand_score(Y1_total, clusters)
silhouette_coefficient = metrics.silhouette_score(X1_total_scaled, clusters, m
etric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

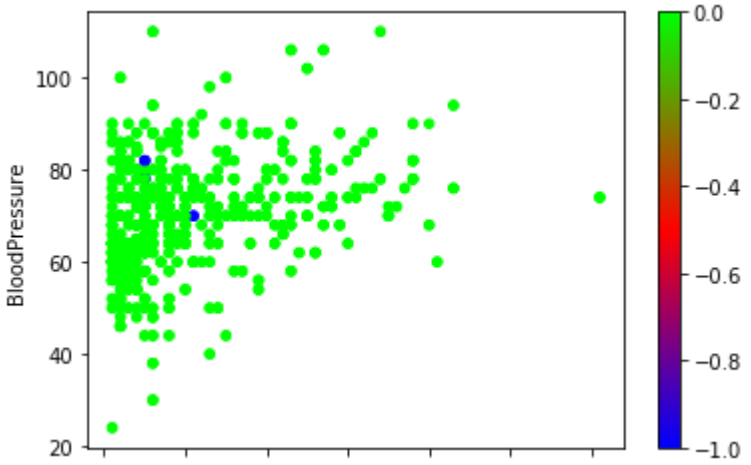
```
[0.010382533211682976, 0.517795170823348]
```

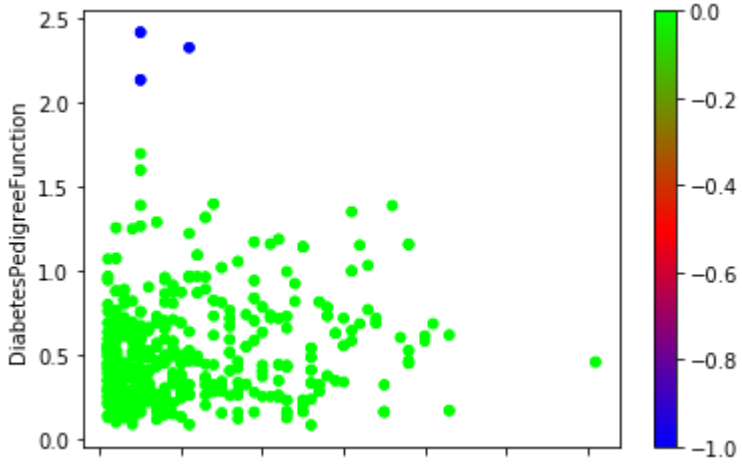
In [80]:
```
#Plot the clusters formed by DBSCAN
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Glucose', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'SkinThickness', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Insulin', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'BMI', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Pregnancies', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'BloodPressure', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'DiabetesPedigreeFunction', c = clusters, colormap = plt.cm.brg)
```


Glucose
200
180
160
140
120
100
80
60
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
SkinThickness
60
50
40
30
20
10
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
Insulin
700
600
500
400
300
200
100
0
0.0
−0.2
−0.4
−0.6
−0.8
−1.0


BMI
60
50
40
30
20
0.0
−0.2
−0.4
−0.6
−0.8
−1.0


Pregnancies
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
−0.2
−0.4
−0.6
−0.8
−1.0


BloodPressure
100
80
60
40
20
0.0
−0.2
−0.4
−0.6
−0.8
−1.0



In [ ]:

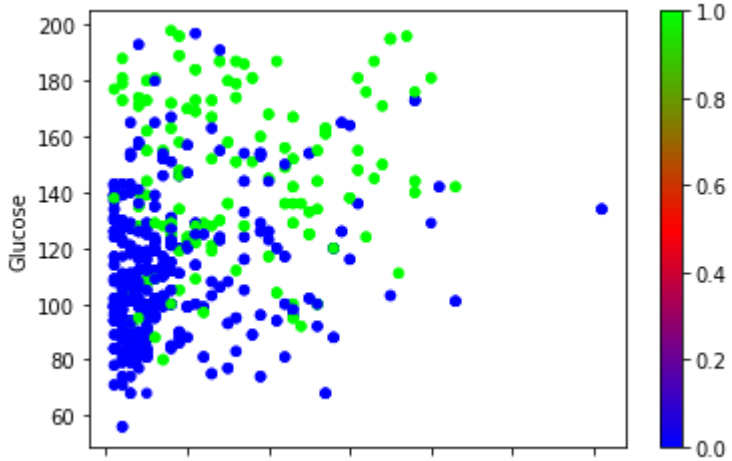
In [ ]:

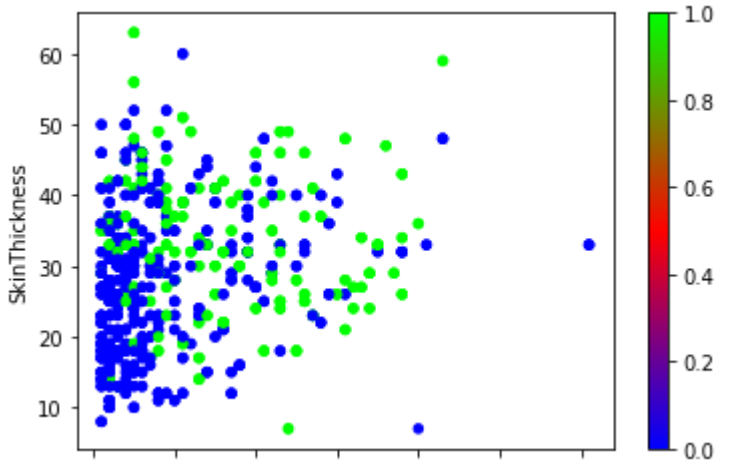
In [81]:
```
#Evaluation metrix for true clusters of data
```

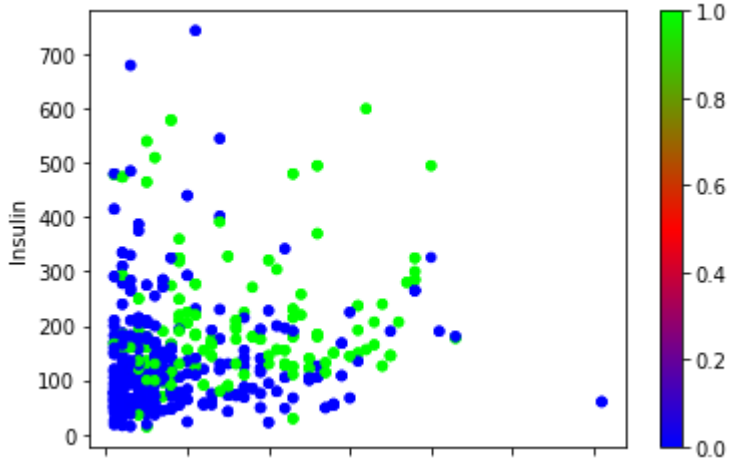
In [82]:
```
silhouette_coefficient = metrics.silhouette_score(X1_total, Y1_total, metric =
"euclidean")
print(silhouette_coefficient)
```

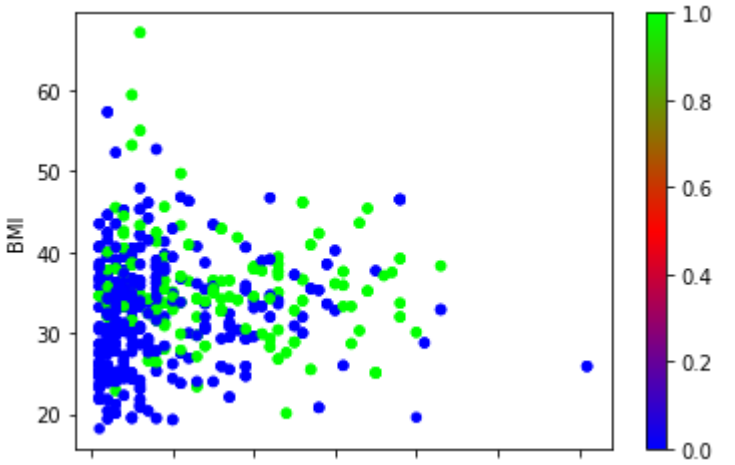
```
0.15444319085109812
```

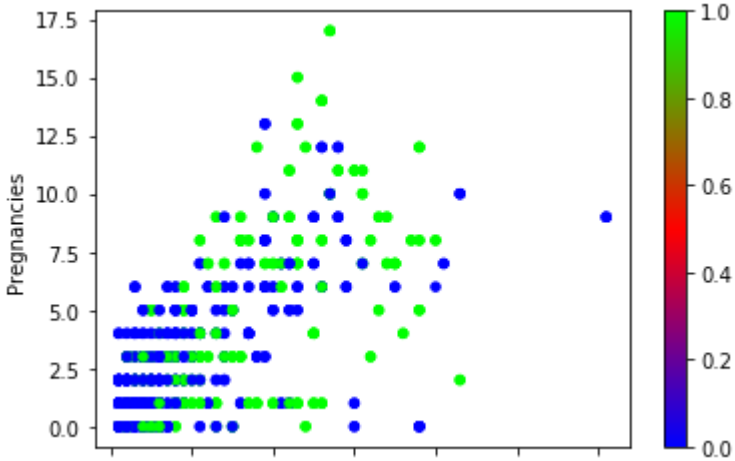
In [83]:
```
#Plot true clusters
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Glucose', c = Y1_total, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'SkinThickness', c = Y1_total, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Insulin', c = Y1_total, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'BMI', c = Y1_total, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Pregnancies', c = Y1_total, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'BloodPressure', c = Y1_total, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'DiabetesPedigreeFunction', c = Y1_total, colormap = plt.cm.brg)
```

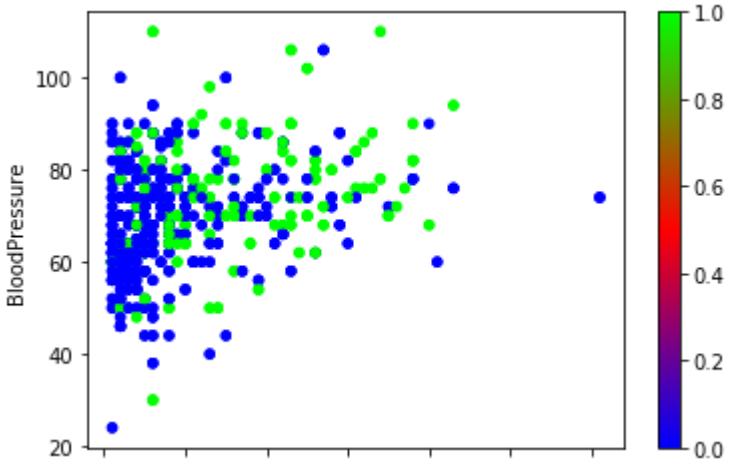

Glucose
200
180
160
140
120
100
80
60
1.0
0.8
0.6
0.4
0.2
0.0

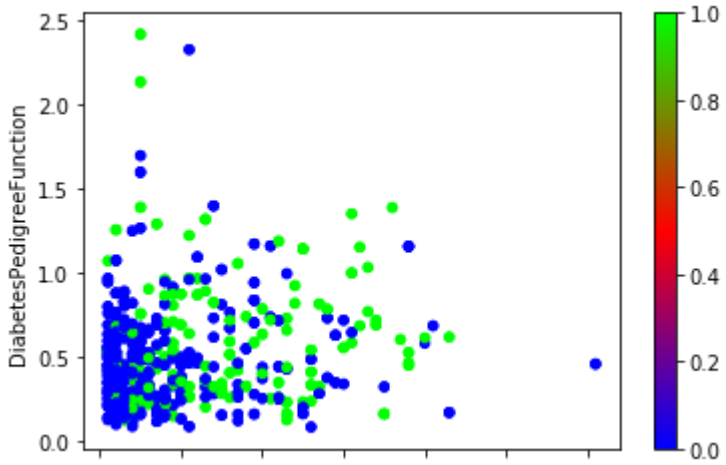

SkinThickness
60
50
40
30
20
10
1.0
0.8
0.6
0.4
0.2
0.0


Insulin
700
600
500
400
300
200
100
0
1.0
0.8
0.6
0.4
0.2
0.0


BMI
60
50
40
30
20
1.0
0.8
0.6
0.4
0.2
0.0


Pregnancies
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1.0
0.8
0.6
0.4
0.2
0.0


BloodPressure
100
80
60
40
20
1.0
0.8
0.6
0.4
0.2
0.0



In [ ]:

In [ ]:

In [84]:
```
#Clustering for majorlying impacting float data variables
```

In [ ]:

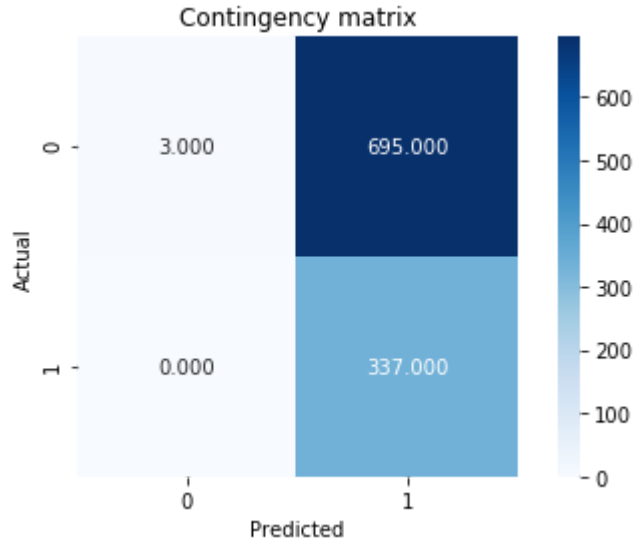
In [85]:
```
X1 = diabetes_data_floats[['Glucose','Age','Insulin','SkinThickness']]
Y1 = diabetes_data_floats['Outcome']
```

In [86]:
```
scaler = StandardScaler()
scaler.fit(X1)
X1_scaled = scaler.transform(X1)
```

In [87]:
```
#Hierarchical Clustering with single linkage
```

In [88]:
```
clustering = linkage(X1_scaled,method = 'single',metric = 'euclidean')
clusters = fcluster(clustering, 2,criterion = 'maxclust')
```

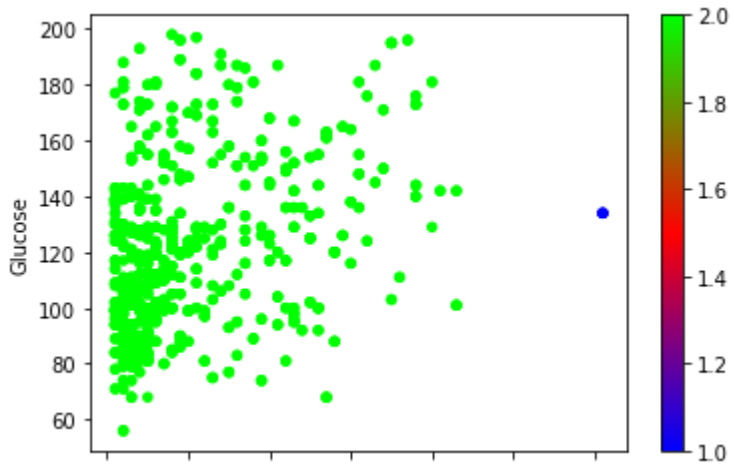
In [89]:
```
#Plot the contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y1,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

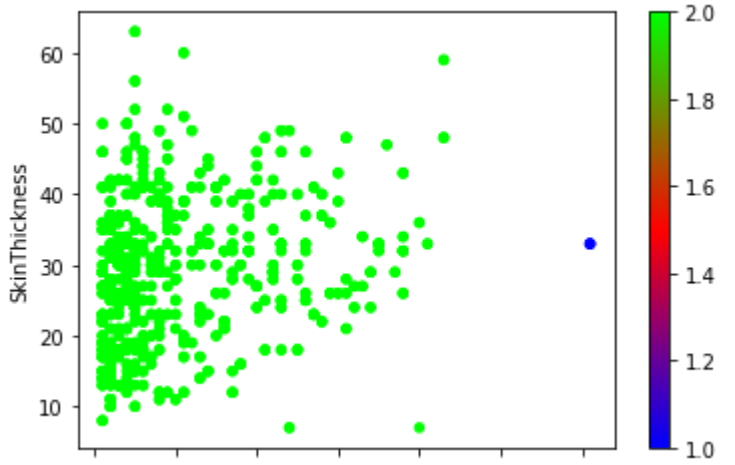


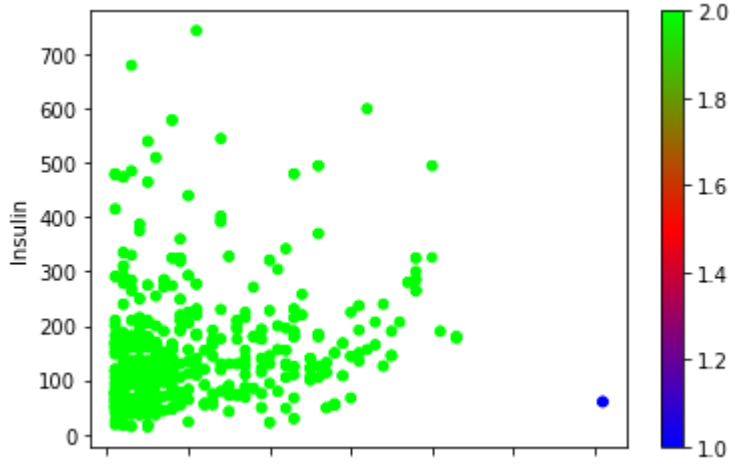
In [90]:
```
#Calculate the adjusted rand index and silohouette coeff
adjusted_rand_index = metrics.adjusted_rand_score(Y1, clusters)
silhouette_coefficient = metrics.silhouette_score(X1_scaled, clusters, metric
= "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[-0.002971064344048915, 0.5068727410342162]
```

In [91]:
```
#Plot clusters formed by hierarchical clustering
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Glucose', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'SkinThicknes
s', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Insulin', c =
clusters, colormap = plt.cm.brg)
```



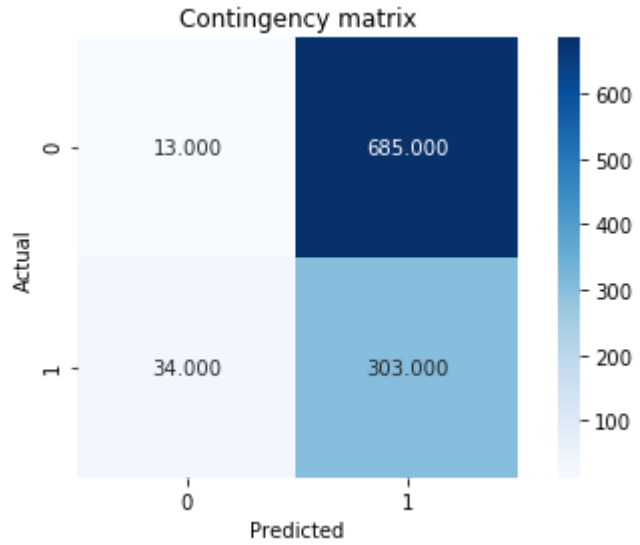




In [92]:
```
#Hierarchical Clustering with complete linkage
```

In [93]:
```
clustering = linkage(X1_scaled, method = "complete", metric = "euclidean")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
```

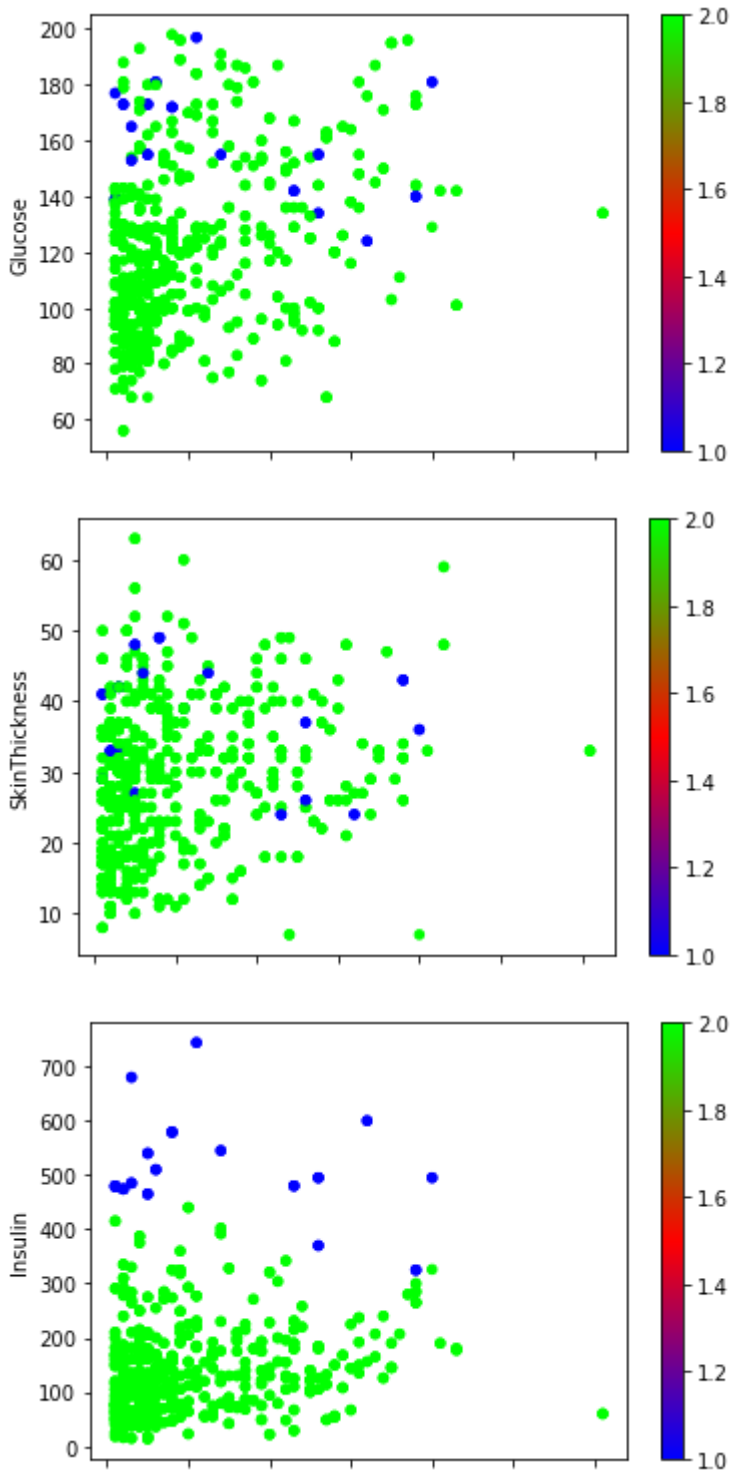
In [94]:
```
#Plot the contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y1,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [95]:
```
#Calculate the values for adjusted rand index and silhouette coeff
adjusted_rand_index = metrics.adjusted_rand_score(Y1, clusters)
silhouette_coefficient = metrics.silhouette_score(X1_scaled, clusters, metric
= "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.056601950592570116, 0.4229687239625707]
```

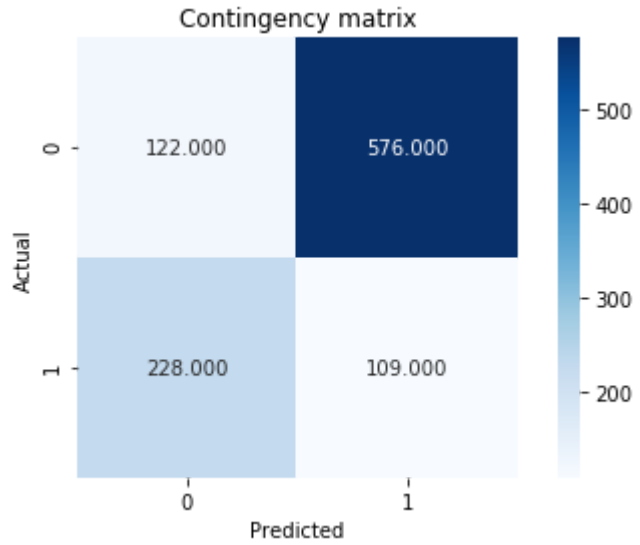
In [96]:
```
#Plot the clusters formed by Hierarchical clusters
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Glucose', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'SkinThicknes
s', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Insulin', c =
clusters, colormap = plt.cm.brg)
```



In [97]:
```
#KMeans Clustering iteration = 2
```

In [98]:
```
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 2, random_state
= 0).fit(X1_scaled)
clusters = clustering.labels_
```

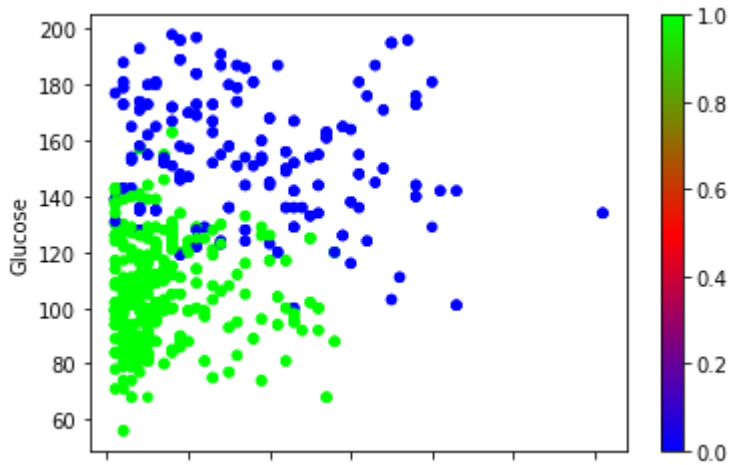
In [99]:
```
#Plot the contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y1,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

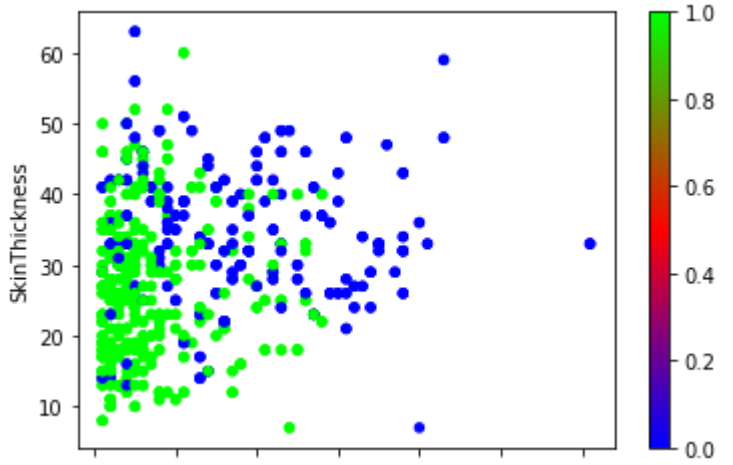


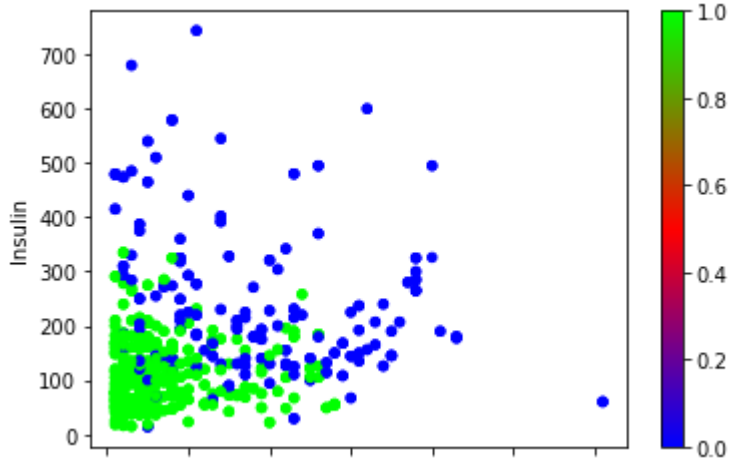
In [100]:
```
adjusted_rand_index = metrics.adjusted_rand_score(Y1, clusters)
silhouette_coefficient = metrics.silhouette_score(X1_scaled, clusters, metric
= "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.29700429006533774, 0.3330387674335957]
```

In [101]:
```
#Plot the clusters formed by KMeans clustering
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Glucose', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'SkinThicknes
s', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Insulin', c =
clusters, colormap = plt.cm.brg)
```



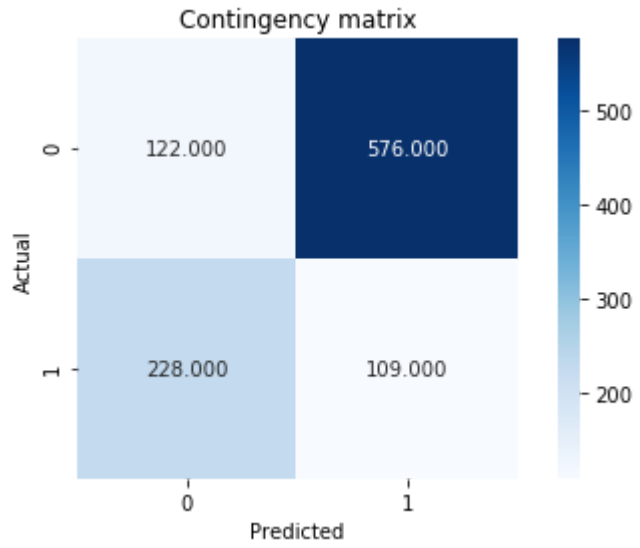




In [102]:
```
#KMeans using k-means++ iteration
```

In [103]:
```
clustering = KMeans(n_clusters = 2, init = 'k-means++', n_init = 20,random_sta
te = 0).fit(X1_scaled)
# clustering = KMeans(n_clusters = 4, init = 'random', n_init = 20, random_sta
te = 0).fit(X)
clusters = clustering.labels_
```

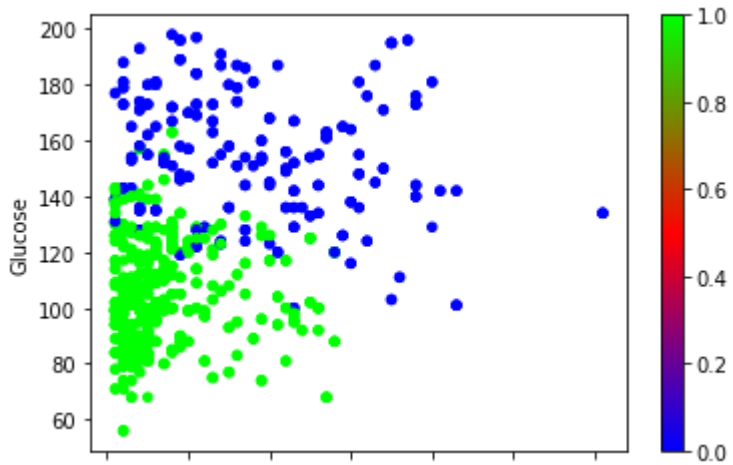
In [104]:
```
#Plot the contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y1,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

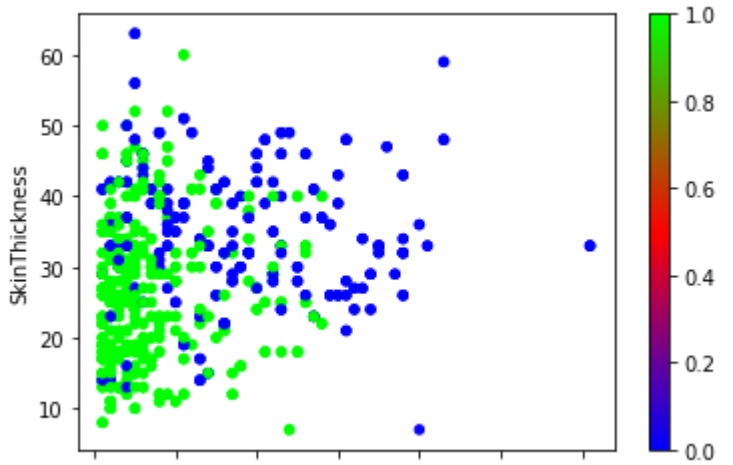


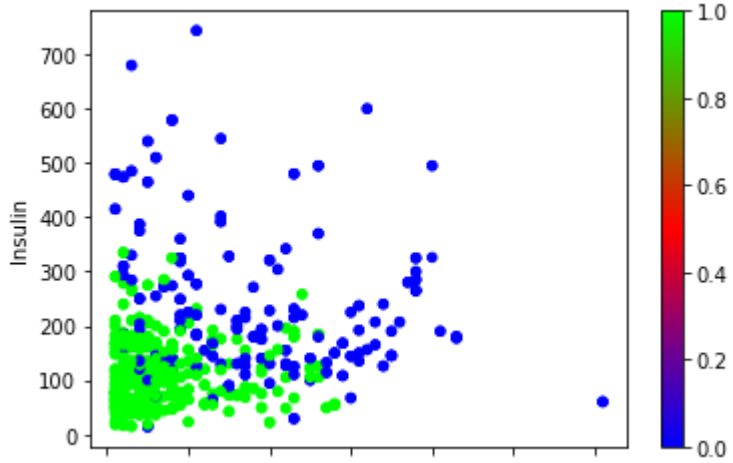
In [105]:
```
#Calculate the adjusted rand index and silhouette coeff
adjusted_rand_index = metrics.adjusted_rand_score(Y1, clusters)
silhouette_coefficient = metrics.silhouette_score(X1_scaled, clusters, metric
= "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.29700429006533774, 0.3330387674335957]
```

In [106]:
```
#Plot the clusters formed by KMeans clustering
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Glucose', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'SkinThicknes
s', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Insulin', c =
clusters, colormap = plt.cm.brg)
```



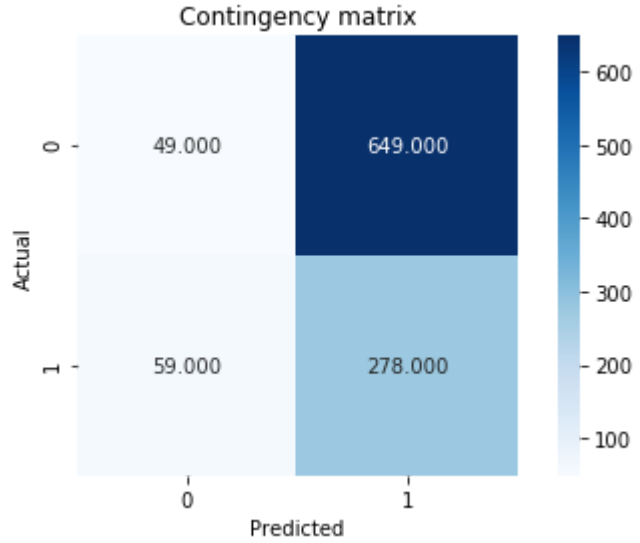




In [107]:
```
#DBSCAN Clustering
```

In [108]:
```
clustering = DBSCAN(eps = 1, min_samples = 20, metric = "euclidean").fit(X1_sc
aled)
clusters = clustering.labels_
```

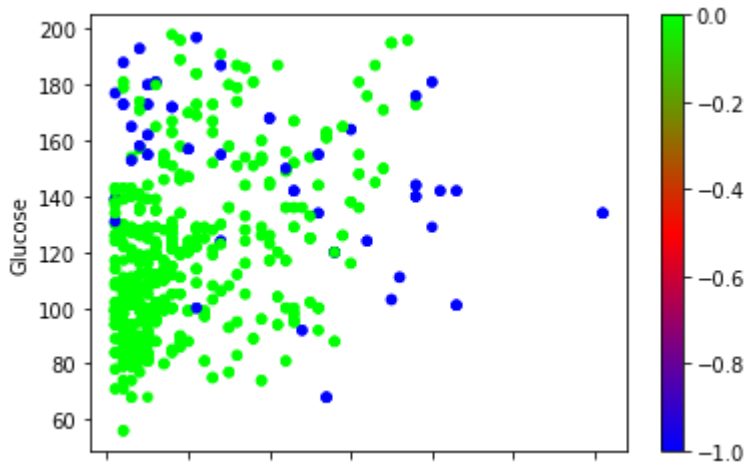
In [109]:
```
# Plot contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y1, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

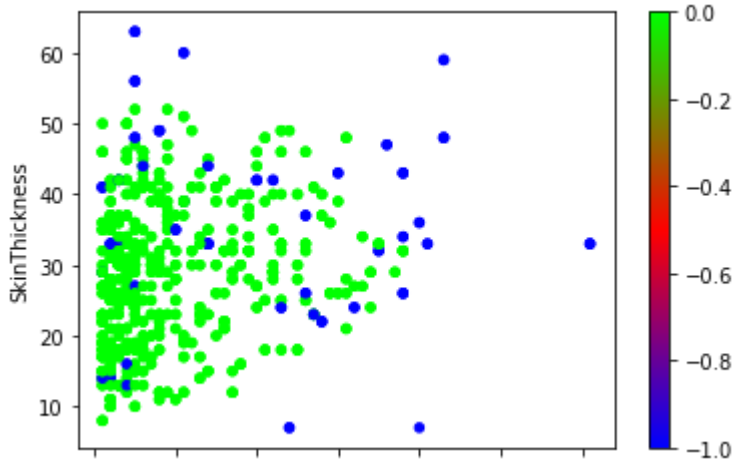


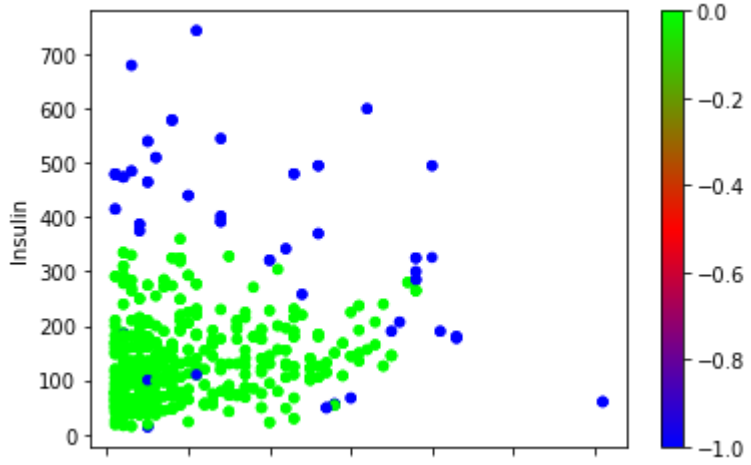
In [110]:
```
#Calculate the adjusted rand index and silohouette coeff
adjusted_rand_index = metrics.adjusted_rand_score(Y1, clusters)
silhouette_coefficient = metrics.silhouette_score(X1_scaled, clusters, metric
= "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.06390356294089194, 0.37679957531587693]
```

In [111]:
```
#Plot the clusters formed by DBSCAN
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Glucose', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'SkinThicknes
s', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Insulin', c =
clusters, colormap = plt.cm.brg)
```



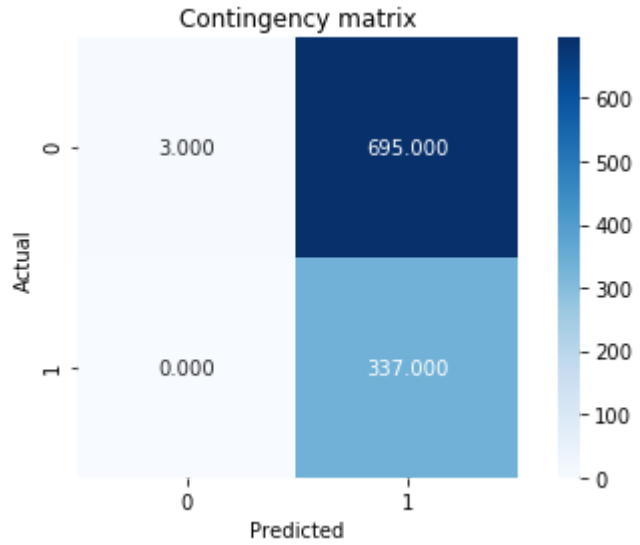




In [112]:
```
#DBSCAN with change eps value and min_sample
```

In [113]:
```
clustering = DBSCAN(eps = 2, min_samples = 10, metric = "euclidean").fit(X1_sc
aled)
clusters = clustering.labels_
```

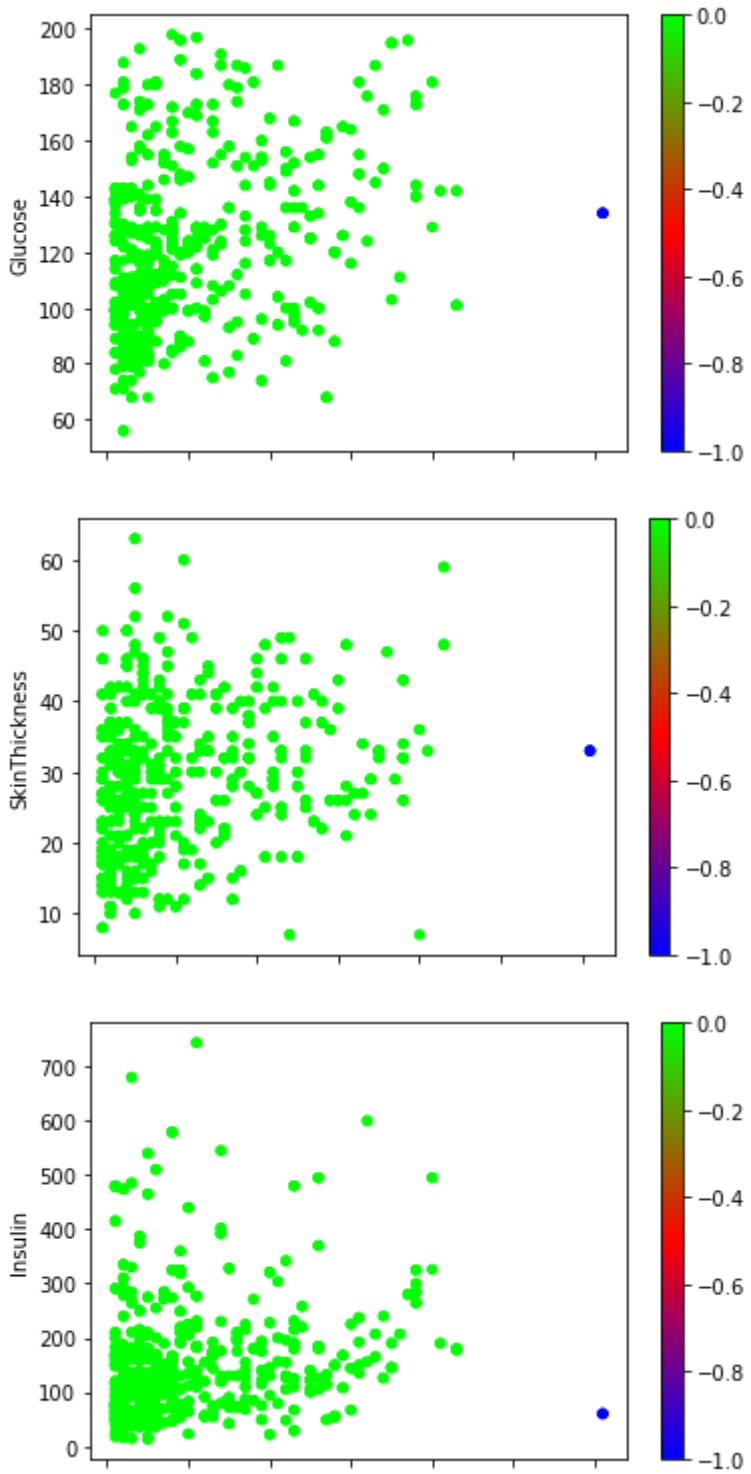
In [114]:
```
# Plot contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y1, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```



In [115]:
```
#Calculate the adjusted rand index and silohouette coeff
adjusted_rand_index = metrics.adjusted_rand_score(Y1, clusters)
silhouette_coefficient = metrics.silhouette_score(X1_scaled, clusters, metric
= "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[-0.002971064344048915, 0.5068727410342162]
```

In [116]:
```
#Plot the clusters formed by DBSCAN
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Glucose', c =
clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'SkinThicknes
s', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Insulin', c =
clusters, colormap = plt.cm.brg)
```



In [ ]:

In [ ]:

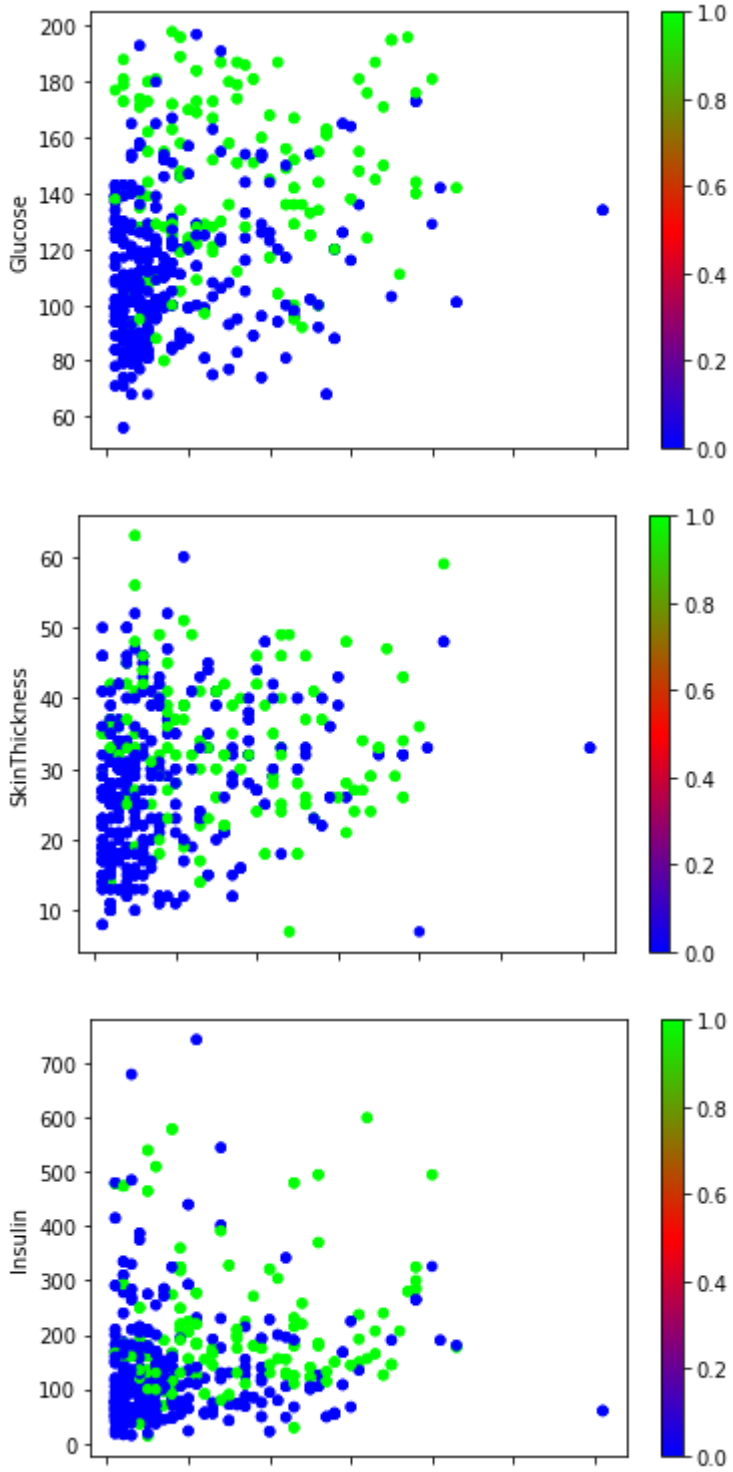
In [ ]:

In [117]:
```
# Evaluation Matrix
```

In [118]:
```
silhouette_coefficient = metrics.silhouette_score(X1, Y1, metric = "euclidean"
)
print(silhouette_coefficient)
```

```
0.15827210564465985
```

In [119]:
```
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Glucose', c = Y1, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'SkinThickness', c = Y1, colormap = plt.cm.brg)
ax = diabetes_data_floats.plot(kind = 'scatter', x = 'Age', y = 'Insulin', c = Y1, colormap = plt.cm.brg)
```



In [ ]:

In [120]:
```
# Clustering using all variables for binary clusters
```

In [121]:
```
X2_total = diabetes_data_binary.drop('class',axis = 1)
```

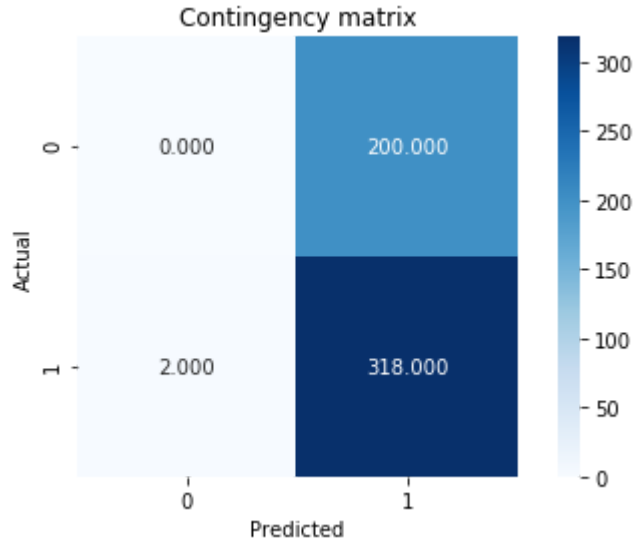
In [122]:
```
Y2_total = diabetes_data_binary["class"]
```

In [123]:
```
scaler = StandardScaler()
scaler.fit(X2_total)
X2_total_scaled = scaler.transform(X2_total)
```

In [124]:
```
#Hierarchical Clustering with single linkage
```

In [125]:
```
clustering = linkage(X2_total_scaled,method = 'single',metric = 'euclidean')
clusters = fcluster(clustering, 2,criterion = 'maxclust')
```

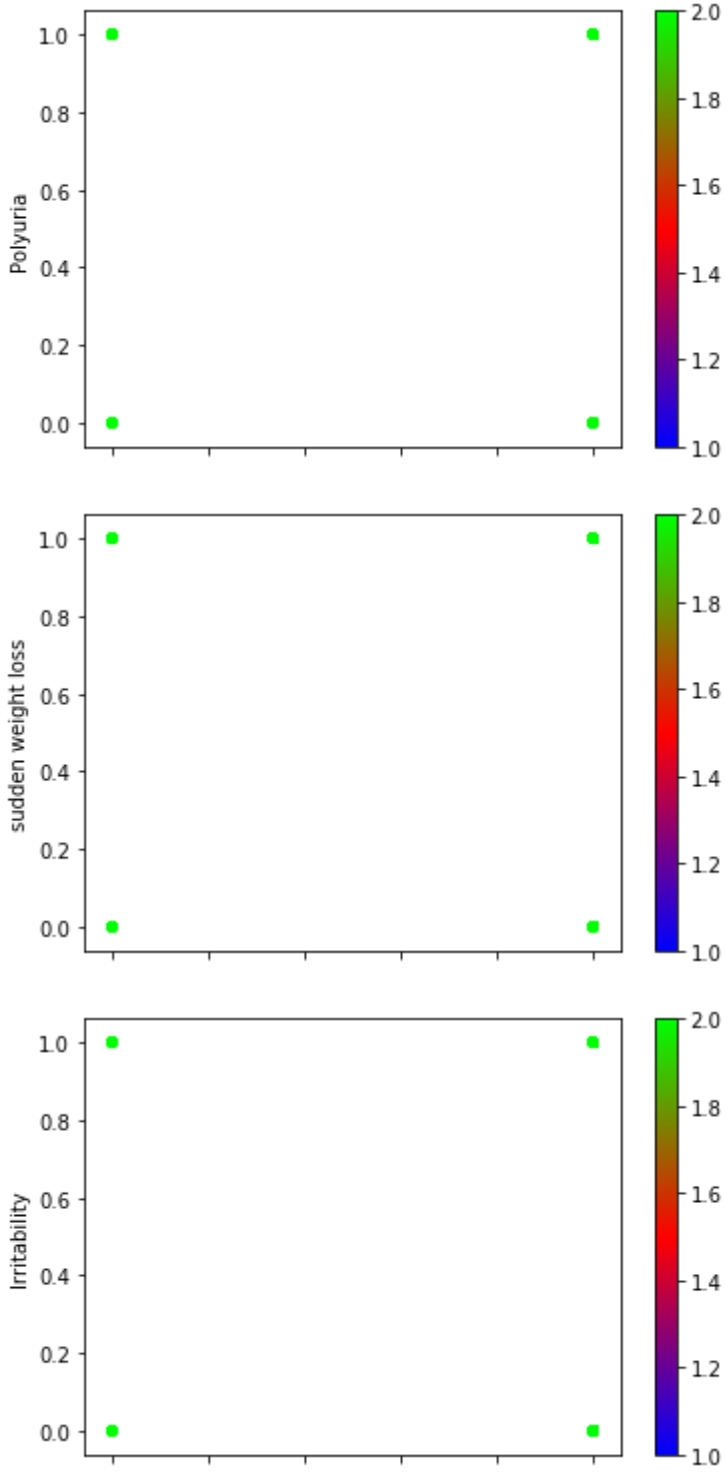
In [126]:
```
#Plot the contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y2_total,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

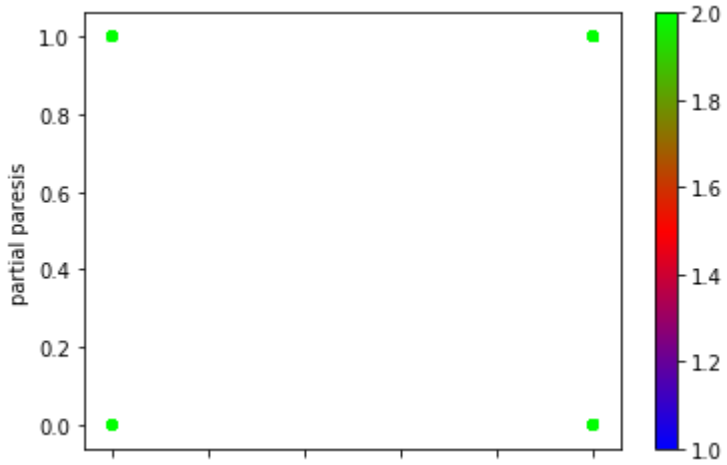


In [127]:
```
#Calculate the adjusted rand index and silhouette coeff
adjusted_rand_index = metrics.adjusted_rand_score(Y2_total, clusters)
silhouette_coefficient = metrics.silhouette_score(X2_total_scaled, clusters, m
etric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[-0.0028525977145914877, 0.18717183855304528]
```

In [128]:
```
#Plot the clusters formed by Hierarchical clustering
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Polyur
ia', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = "sudden
weight loss", c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Irrita
bility', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'partia
l paresis', c = clusters, colormap = plt.cm.brg)
```


Polyuria
sudden weight loss
Irritability
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0

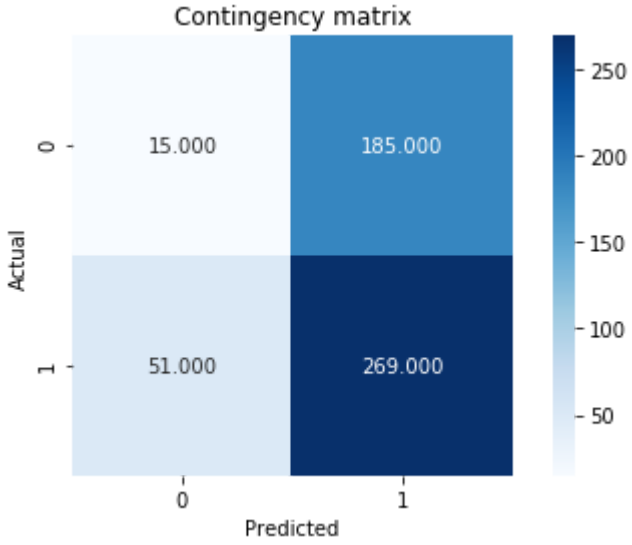


In [ ]:

In [129]:
```
#Hierarchical Clustering with complete linkage
```

In [130]:
```
clustering = linkage(X2_total_scaled, method = "complete", metric = "euclidea
n")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
```

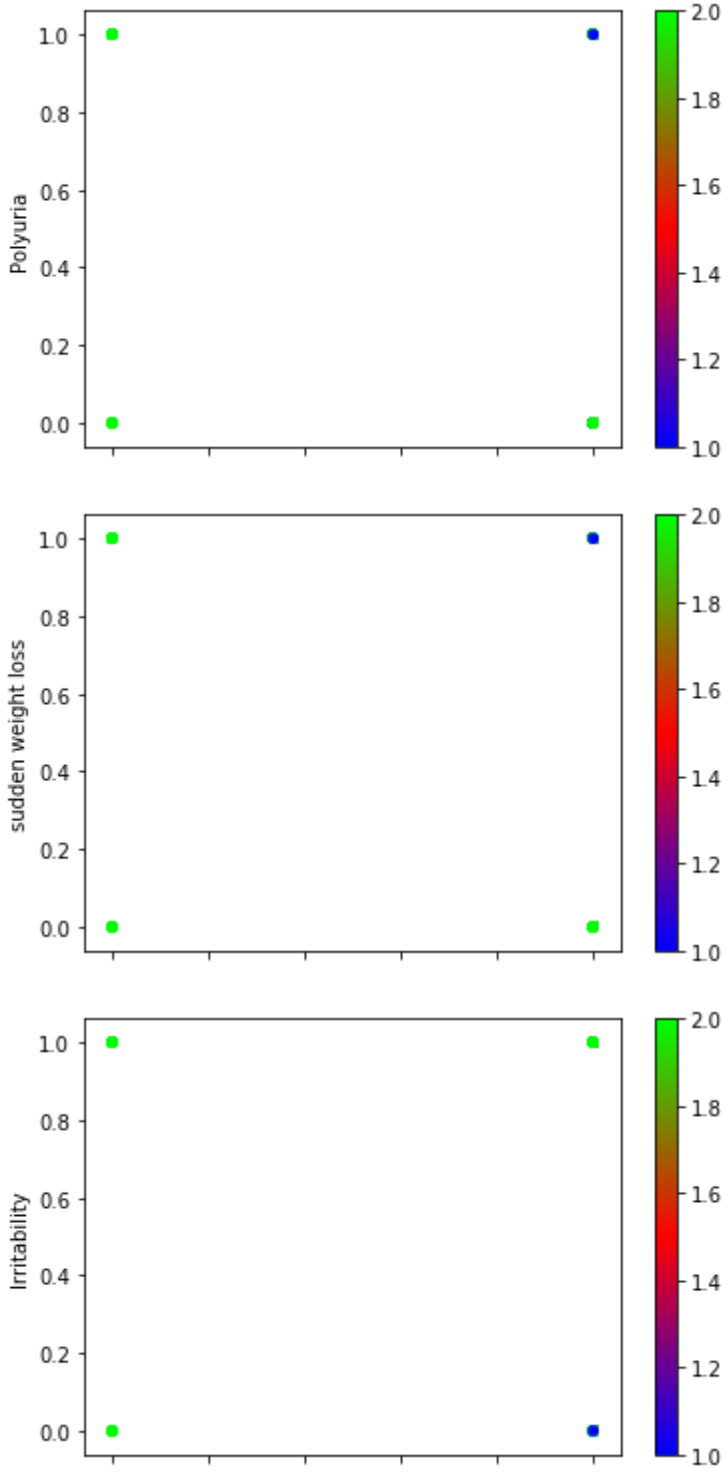
In [131]:
```
#Plot the contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y2_total,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

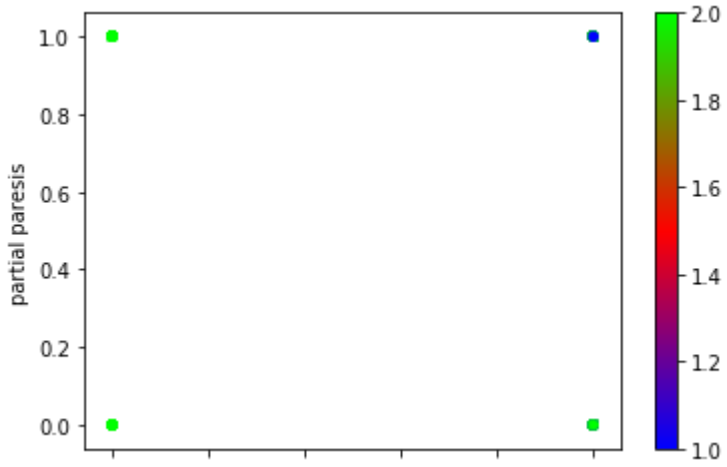


In [132]:
```
#Calculate the adjusted rand index and silhouette coeff
adjusted_rand_index = metrics.adjusted_rand_score(Y2_total, clusters)
silhouette_coefficient = metrics.silhouette_score(X2_total_scaled, clusters, m
etric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[-0.02262617678656324, 0.10424018167505285]
```

In [133]:
```
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Polyuria', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = "sudden weight loss", c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Irritability', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'partial paresis', c = clusters, colormap = plt.cm.brg)
```


Polyuria
sudden weight loss
Irritability
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0

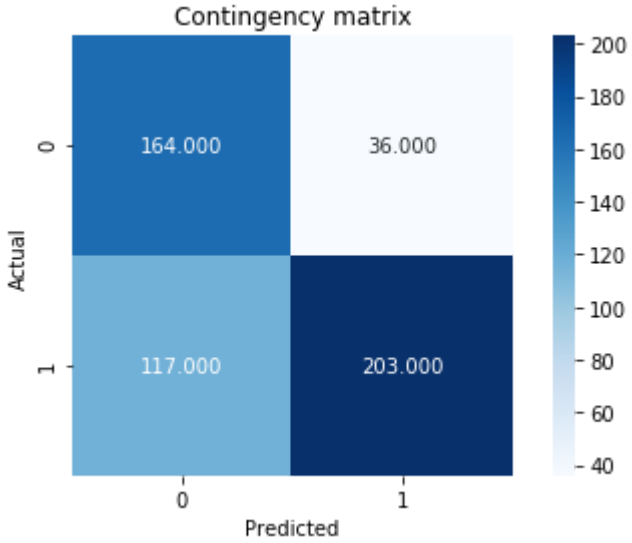


In [ ]:

In [134]:
```
# KMeans clustering by randon choice of centroids and 1 iteration
```

In [135]:
```
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 1, random_state
= 0).fit(X2_total_scaled)
clusters = clustering.labels_
```

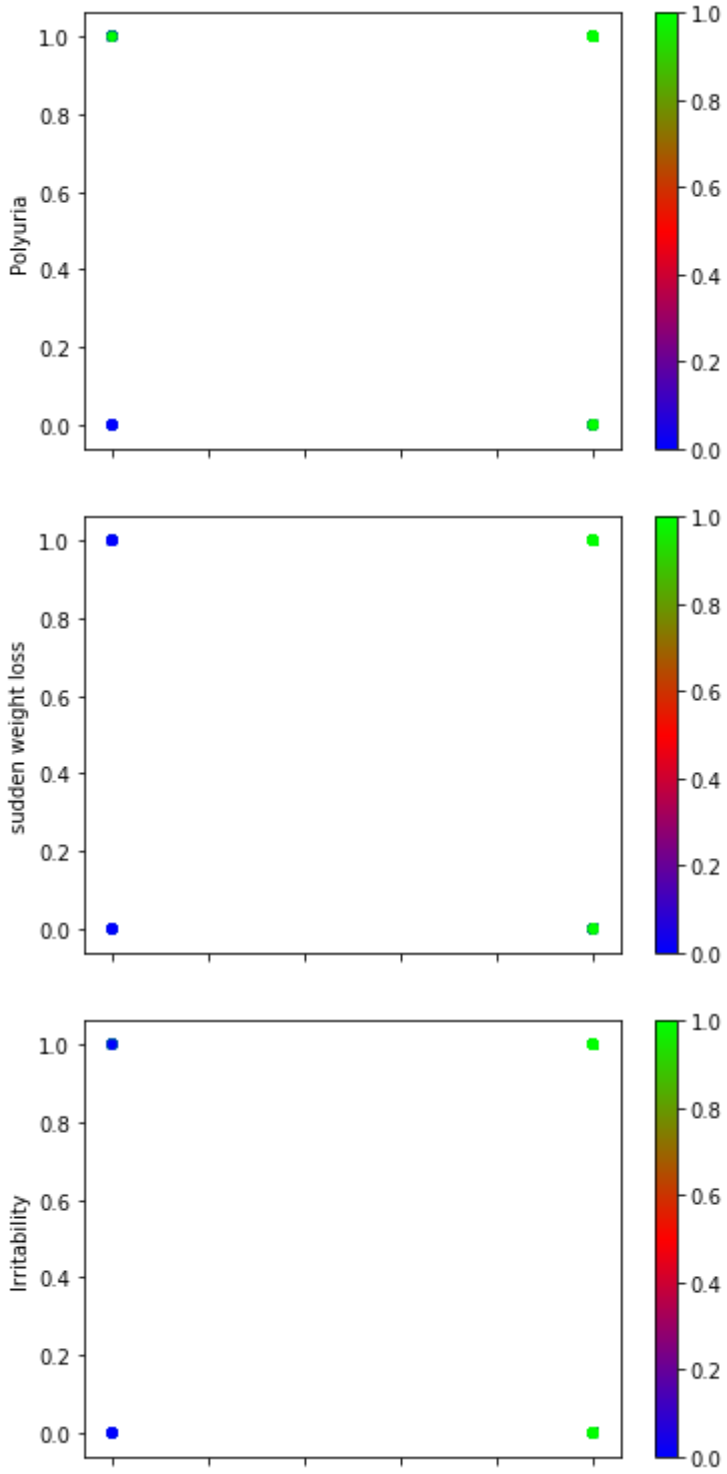
In [136]:
```
#Plot the contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y2_total,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

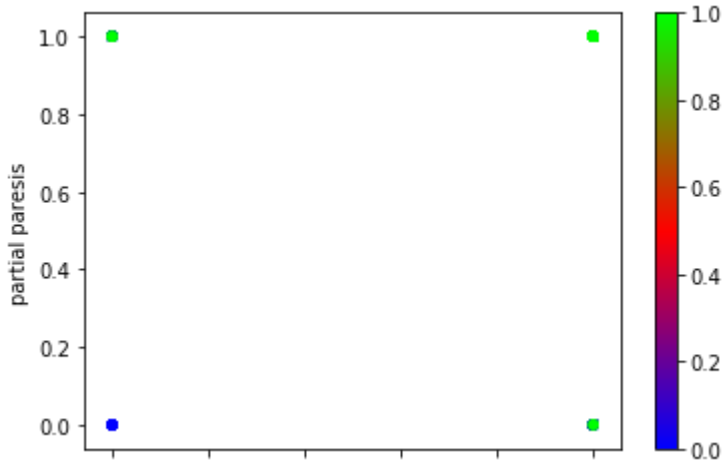


In [137]:
```
#Calculate the adjusted rand index and silhouette coeff
adjusted_rand_index = metrics.adjusted_rand_score(Y2_total, clusters)
silhouette_coefficient = metrics.silhouette_score(X2_total_scaled, clusters, m
etric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.16756610902604885, 0.18494243619206485]
```

In [138]:
```
#Plot the clusters using the clusters formed by KMeans clusetring
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Polyur
ia', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = "sudden
weight loss", c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Irrita
bility', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'partia
l paresis', c = clusters, colormap = plt.cm.brg)
```


Polyuria
sudden weight loss
Irritability
0.0
0.2
0.4
0.6
0.8
1.0

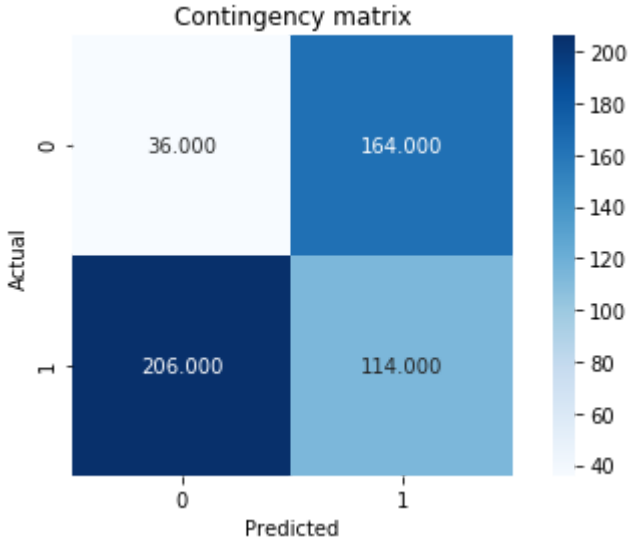


In [ ]:

In [139]:
```
#KMeans clusteirng using 10 iteration and the centroids are chosen iteratively
```

In [140]:
```
clustering = KMeans(n_clusters = 2, init = 'k-means++', n_init = 10,random_sta
te = 0).fit(X2_total_scaled)
clusters = clustering.labels_
```

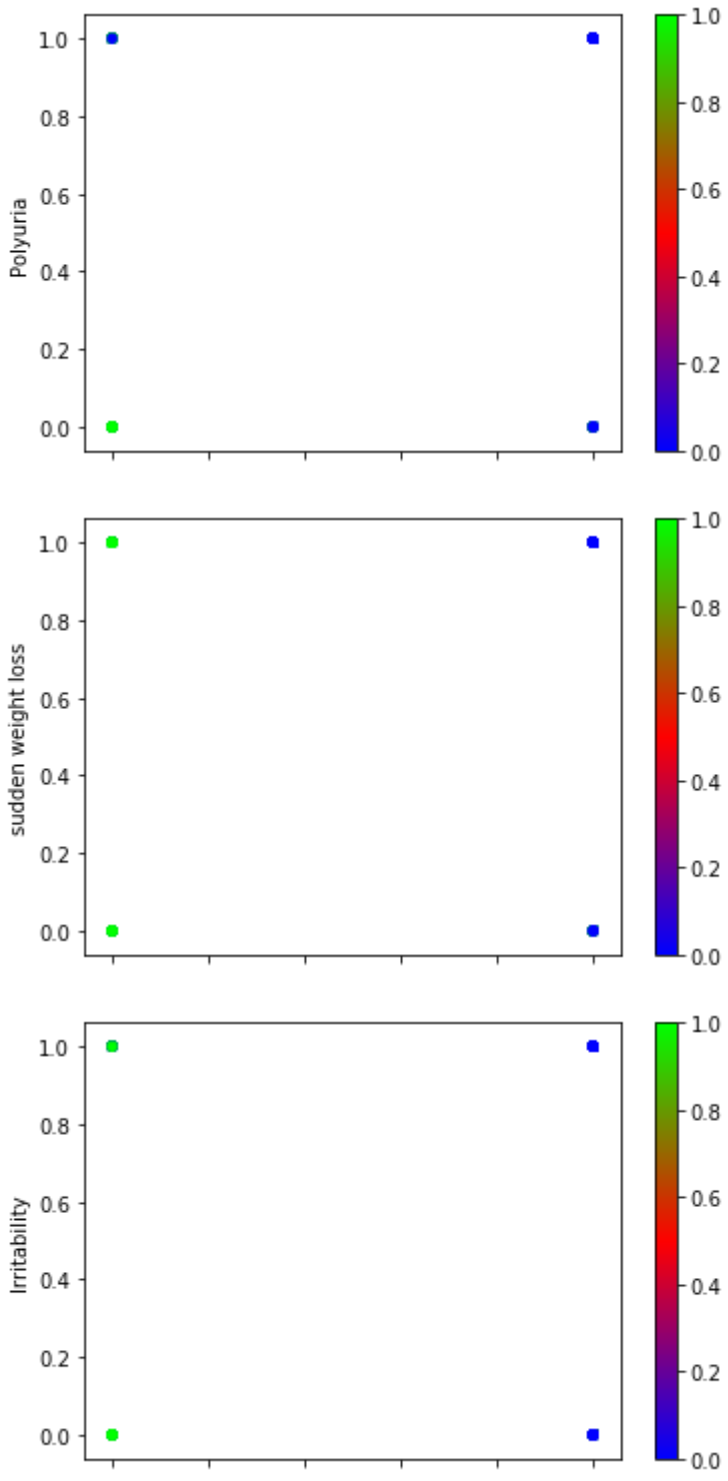
In [141]:
```
#Plot the contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y2_total,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

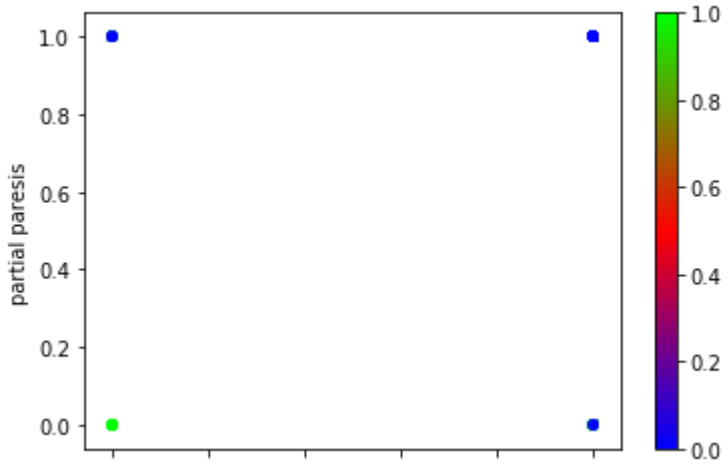


In [142]:
```
#Calculate the adjusted rand index and silhouette coeff
adjusted_rand_index = metrics.adjusted_rand_score(Y2_total, clusters)
silhouette_coefficient = metrics.silhouette_score(X2_total_scaled, clusters, m
etric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.1772905208187989, 0.18464233649706438]
```

In [143]:
```
#Plot the clusters formed by KMeans clustering
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Polyur
ia', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = "sudden
weight loss", c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Irrita
bility', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'partia
l paresis', c = clusters, colormap = plt.cm.brg)
```


Polyuria
sudden weight loss
Irritability
0.0
0.2
0.4
0.6
0.8
1.0



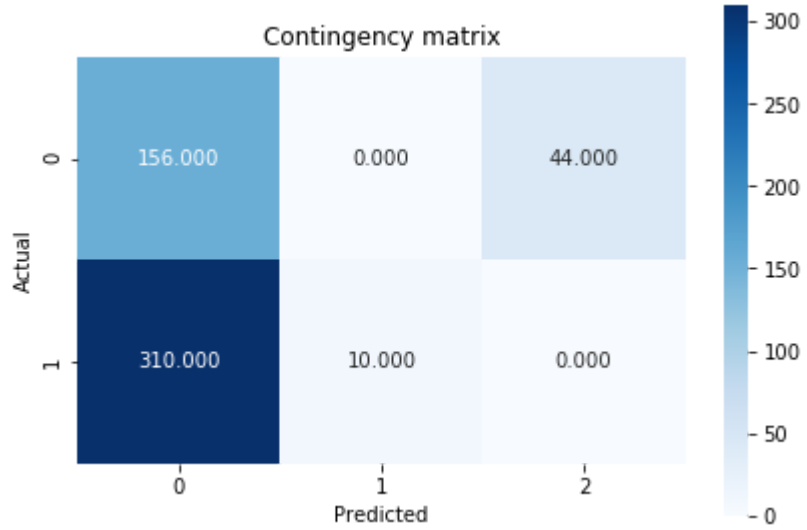
In [ ]:

In [144]:
```
#DBSCAN using eps = 1 and min_sample = 20
```

In [145]:
```
clustering = DBSCAN(eps = 1, min_samples = 10, metric = "euclidean").fit(X2_to
tal_scaled)
clusters = clustering.labels_
print(clusters)
```

```
[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1  0 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1  0 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1
 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
  0 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1  1  1  1 -1 -1 -1 -1 -1 -1  1  1  1
 -1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1
 -1  0 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1
 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1
 -1  0 -1 -1 -1 -1 -1 -1  1  1  1 -1 -1 -1 -1 -1 -1 -1  1  1  1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1  1
  1  1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1]
```

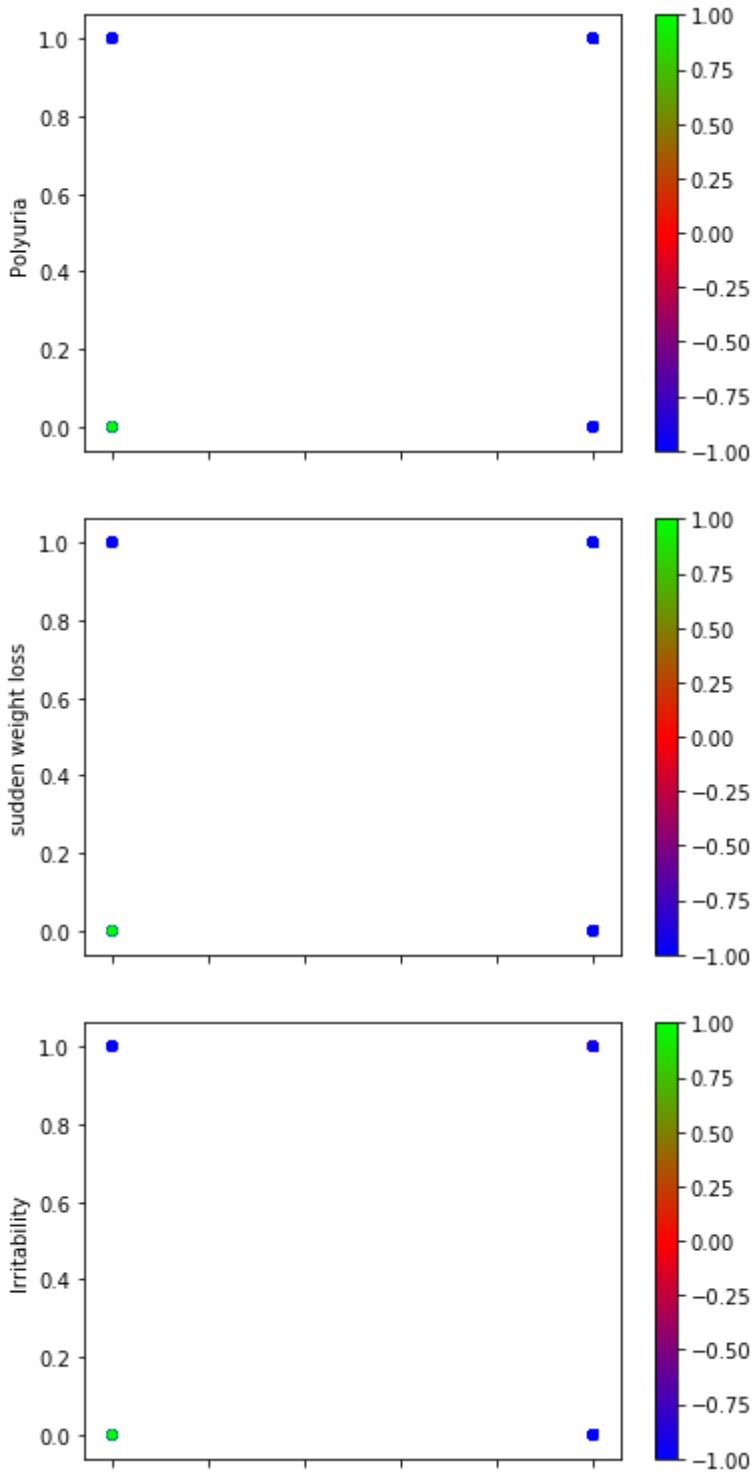
In [146]:
```
#Plot the contingency matrix and we notice that we have 3 clusters because the
min_Sample value is low for eps value
cont_matrix = metrics.cluster.contingency_matrix(Y2_total,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

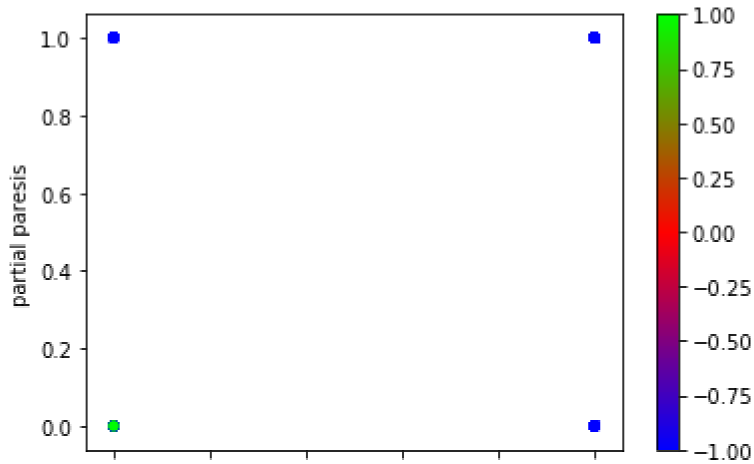


In [147]:
```
adjusted_rand_index = metrics.adjusted_rand_score(Y2_total, clusters)
silhouette_coefficient = metrics.silhouette_score(X2_total_scaled, clusters, m
etric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.10706002616045818, -0.07833480748384714]
```

In [148]:
```
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Polyuria', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = "sudden weight loss", c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Irritability', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'partial paresis', c = clusters, colormap = plt.cm.brg)
```


Polyuria
sudden weight loss
Irritability
1.0
0.8
0.6
0.4
0.2
0.0
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00



In [ ]:

```

```

In [149]:

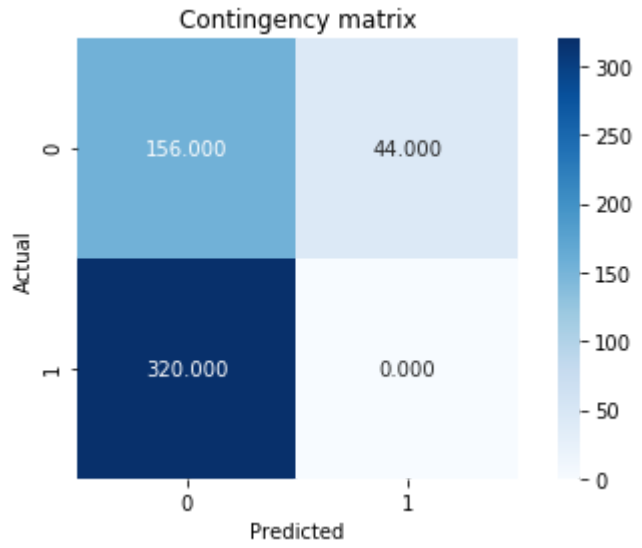
```
#DBSCAN using eps = 2 and min_Sample = 15 and we achive 2 clusters
```

In [150]:

```
clustering = DBSCAN(eps = 2, min_samples = 15, metric = "euclidean").fit(X2_to
tal_scaled)
clusters = clustering.labels_
print(clusters)
```

```
[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1  0  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1
 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0 -1 -1 -1 -1 -1 -1  0  0  0
 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1
 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0 -1 -1 -1 -1 -1 -1  0  0  0 -1 -1 -1  0
 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1  0
  0  0 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1  0]
```

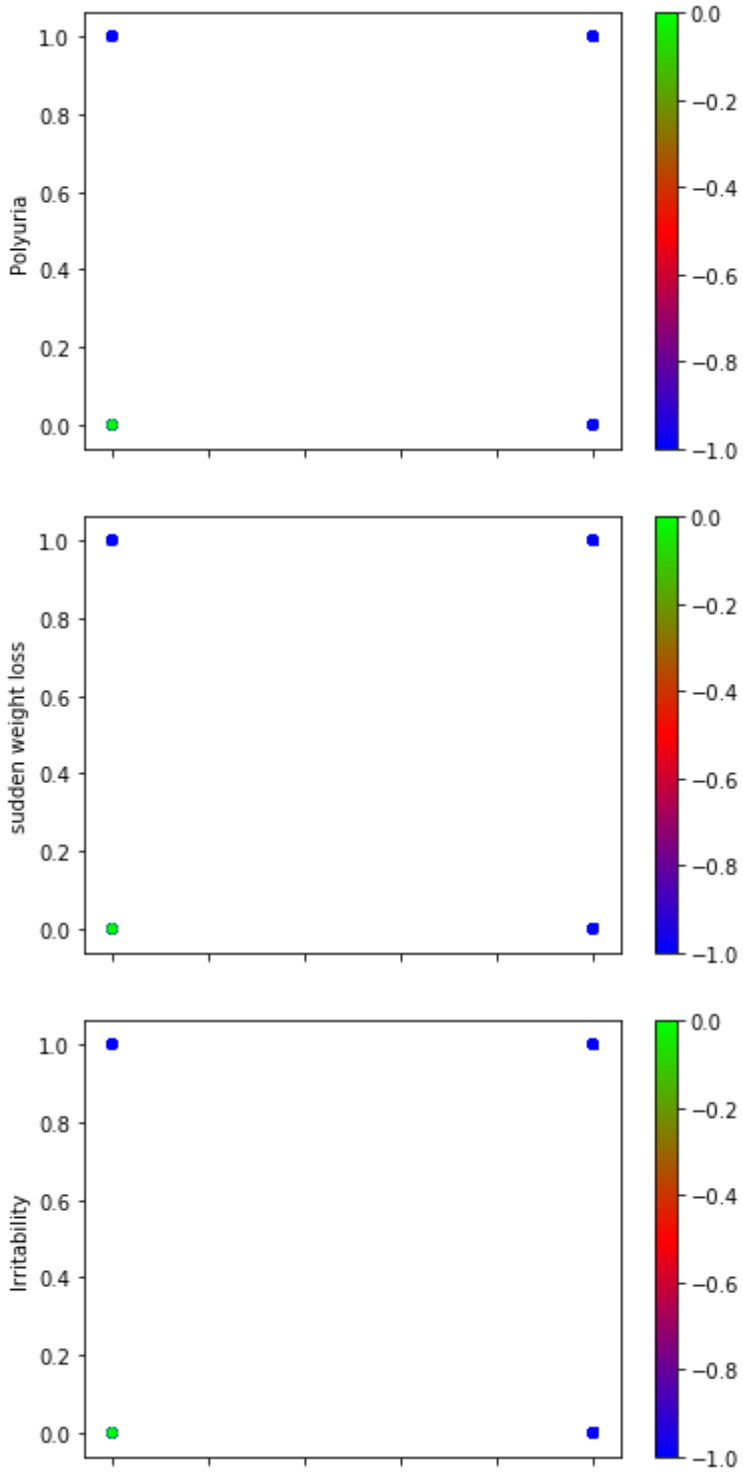
In [151]:
```
cont_matrix = metrics.cluster.contingency_matrix(Y2_total,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

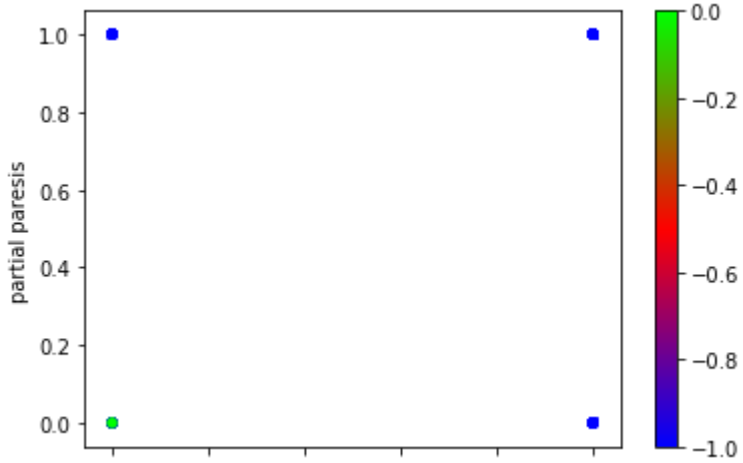


In [152]:
```
adjusted_rand_index = metrics.adjusted_rand_score(Y2_total, clusters)
silhouette_coefficient = metrics.silhouette_score(X2_total_scaled, clusters, m
etric = "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.12743574017309023, 0.030035718688800037]
```

In [153]:
```
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Polyuria', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = "sudden weight loss", c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Irritability', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'partial paresis', c = clusters, colormap = plt.cm.brg)
```


Polyuria
sudden weight loss
Irritability
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0



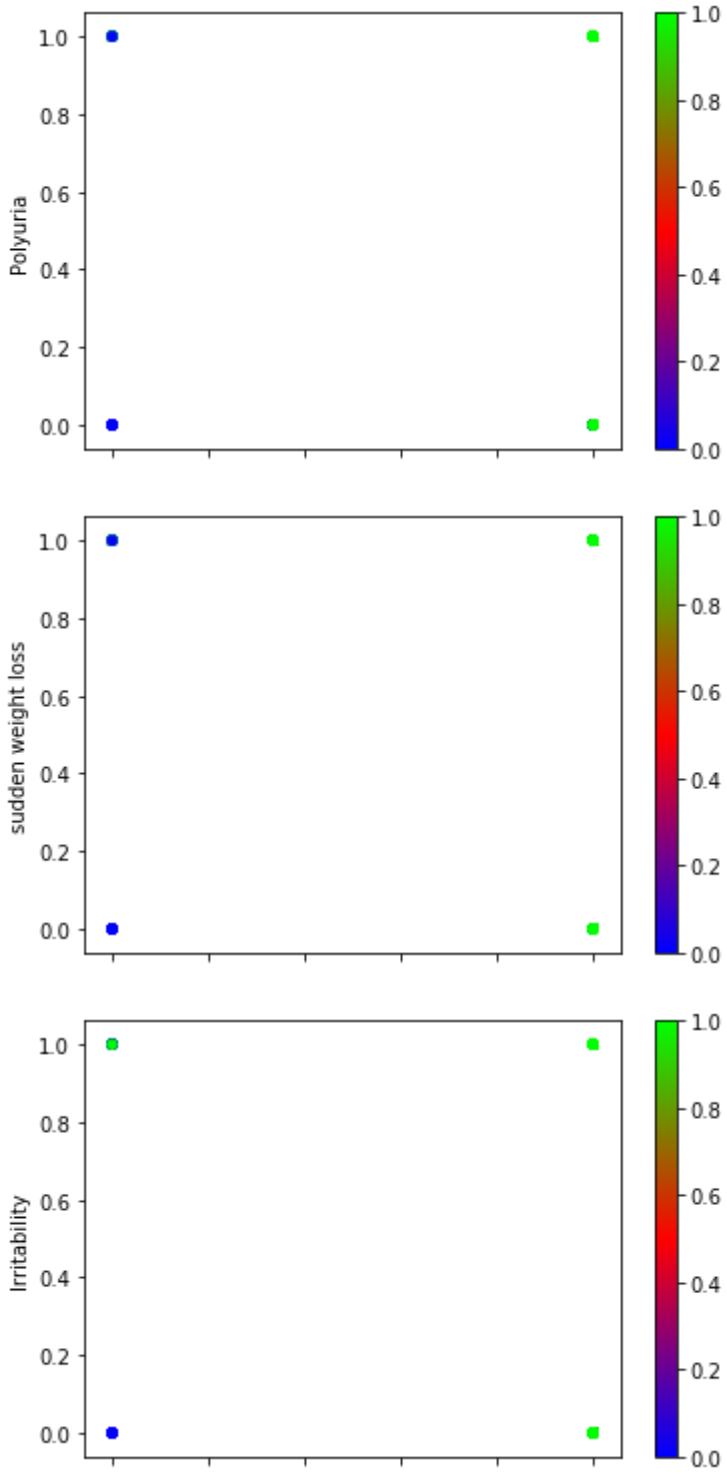
In [ ]:

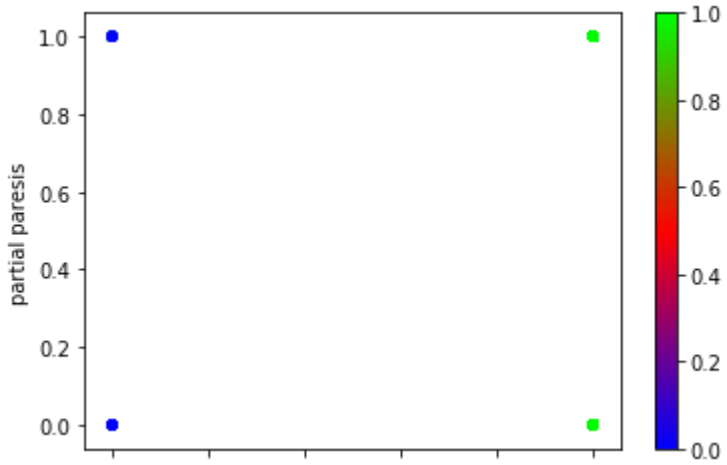
In [154]:
```
#Evaluation metrics
```

In [155]:
```
silhouette_coefficient = metrics.silhouette_score(X2_total_scaled, Y2_total, m
etric = "euclidean")
print(silhouette_coefficient)
```

```
0.11930836400822513
```

In [156]:
```
#Plot true clusters
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Polyuria', c = Y2_total, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = "sudden weight loss", c = Y2_total, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Irritability', c = Y2_total, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'partial paresis', c = Y2_total, colormap = plt.cm.brg)
```


Polyuria
sudden weight loss
Irritability
0.0
0.2
0.4
0.6
0.8
1.0



In [ ]:

In [157]:
```
#Clustering for Binary Data
```

In [158]:
```
#Choosing the variables that have higher impact on diabetic vs non-diabetic fo
r binary values
```

In [159]:
```
X2 = diabetes_data_binary[["Polyuria", "Polydipsia","sudden weight loss",'Irri
tability','partial paresis']]
```

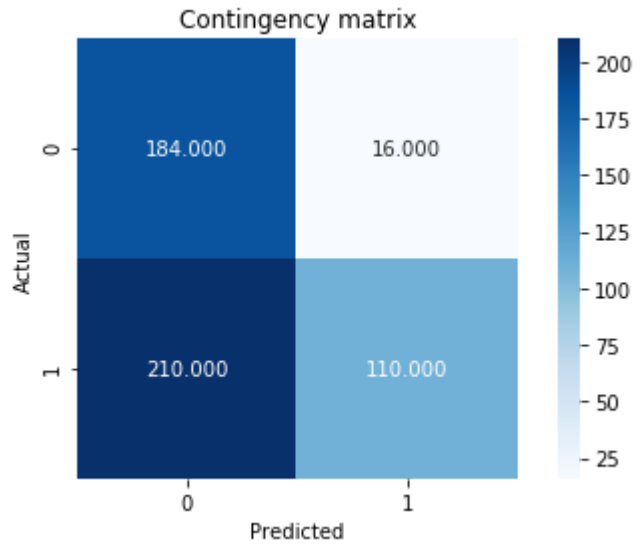
In [160]:
```
Y2 = diabetes_data_binary["class"]
```

In [161]:
```
scaler = StandardScaler()
scaler.fit(X2)
X2_scaled = scaler.transform(X2)
```

In [162]:
```
#Hierarchical Clustering with single linkage
```

In [163]:
```
clustering = linkage(X2_scaled,method = 'single',metric = 'euclidean')
clusters = fcluster(clustering, 2,criterion = 'maxclust')
```

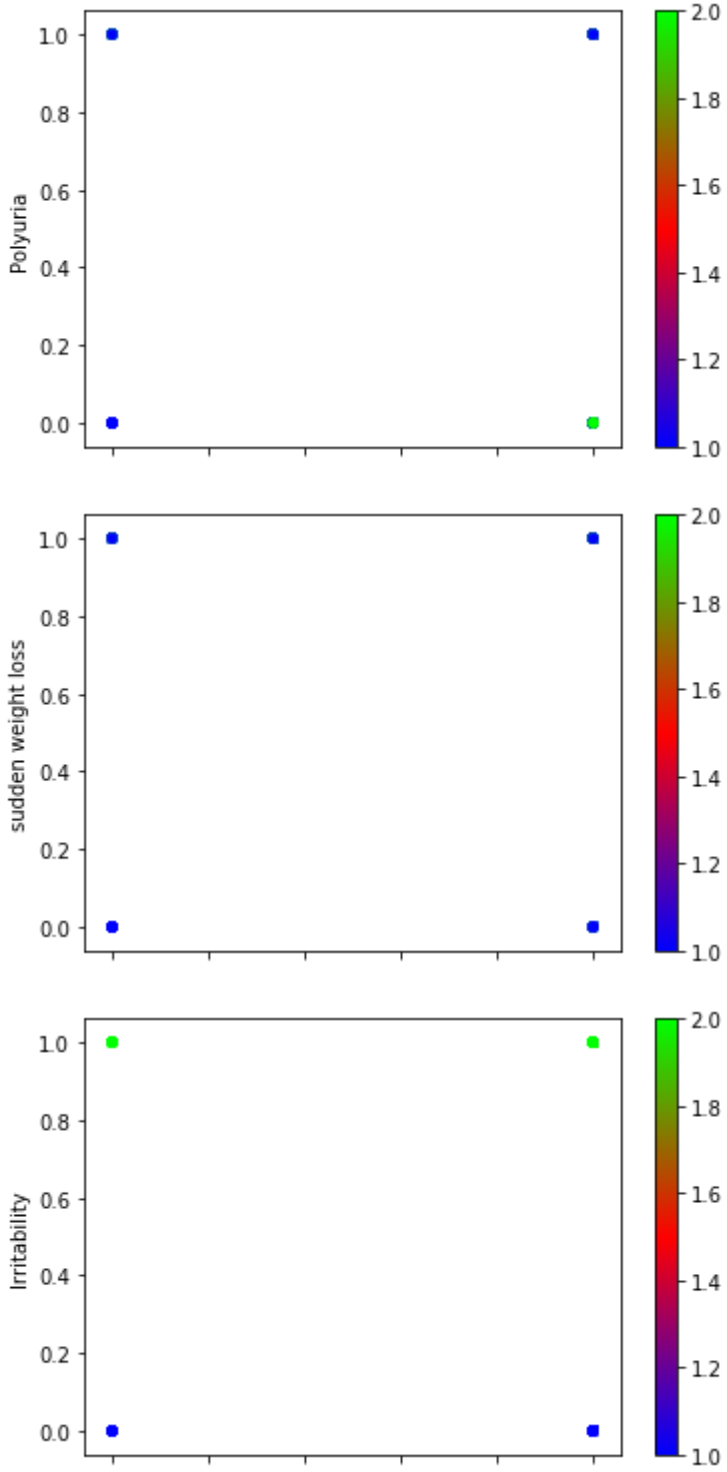
In [164]:
```
#Plot the contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y2,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

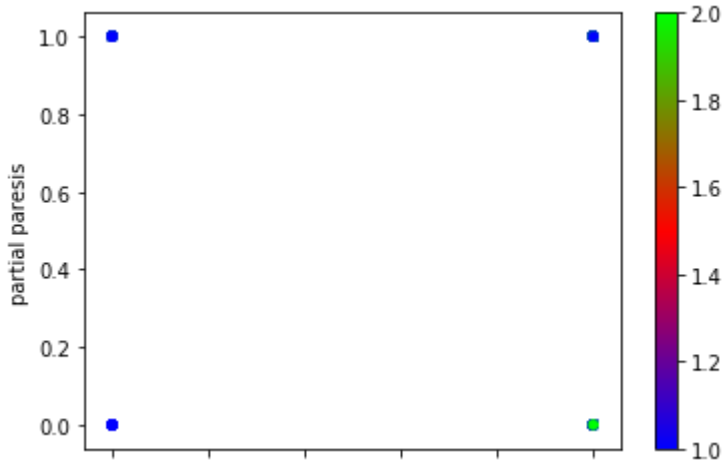


In [165]:
```
#Calculate the adjusted rand index and silhouette coeff
adjusted_rand_index = metrics.adjusted_rand_score(Y2, clusters)
silhouette_coefficient = metrics.silhouette_score(X2_scaled, clusters, metric
= "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.0016408539934176282, 0.3183722153202761]
```

In [166]:
```
#Plot the clusters formed by Hierarchical clustering
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Polyur
ia', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = "sudden
weight loss", c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Irrita
bility', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'partia
l paresis', c = clusters, colormap = plt.cm.brg)
```


Polyuria
sudden weight loss
Irritability
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0

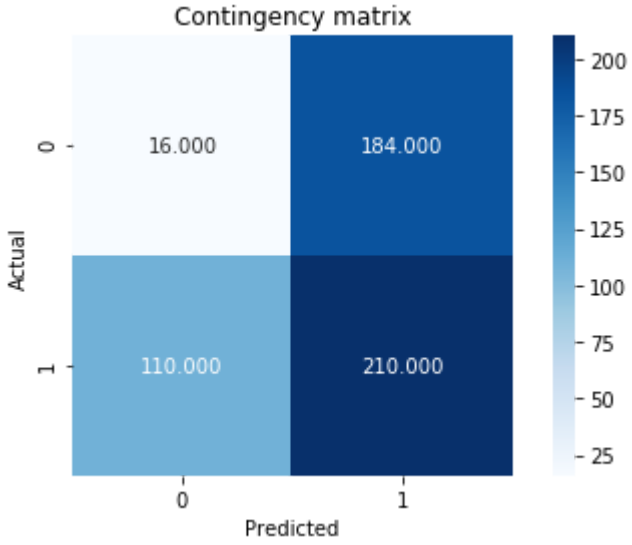


In [ ]:

In [167]:
```
#Hierarchical Clustering with complete linkage
```

In [168]:
```
clustering = linkage(X2_scaled, method = "complete", metric = "euclidean")
clusters = fcluster(clustering, 2, criterion = 'maxclust')
```

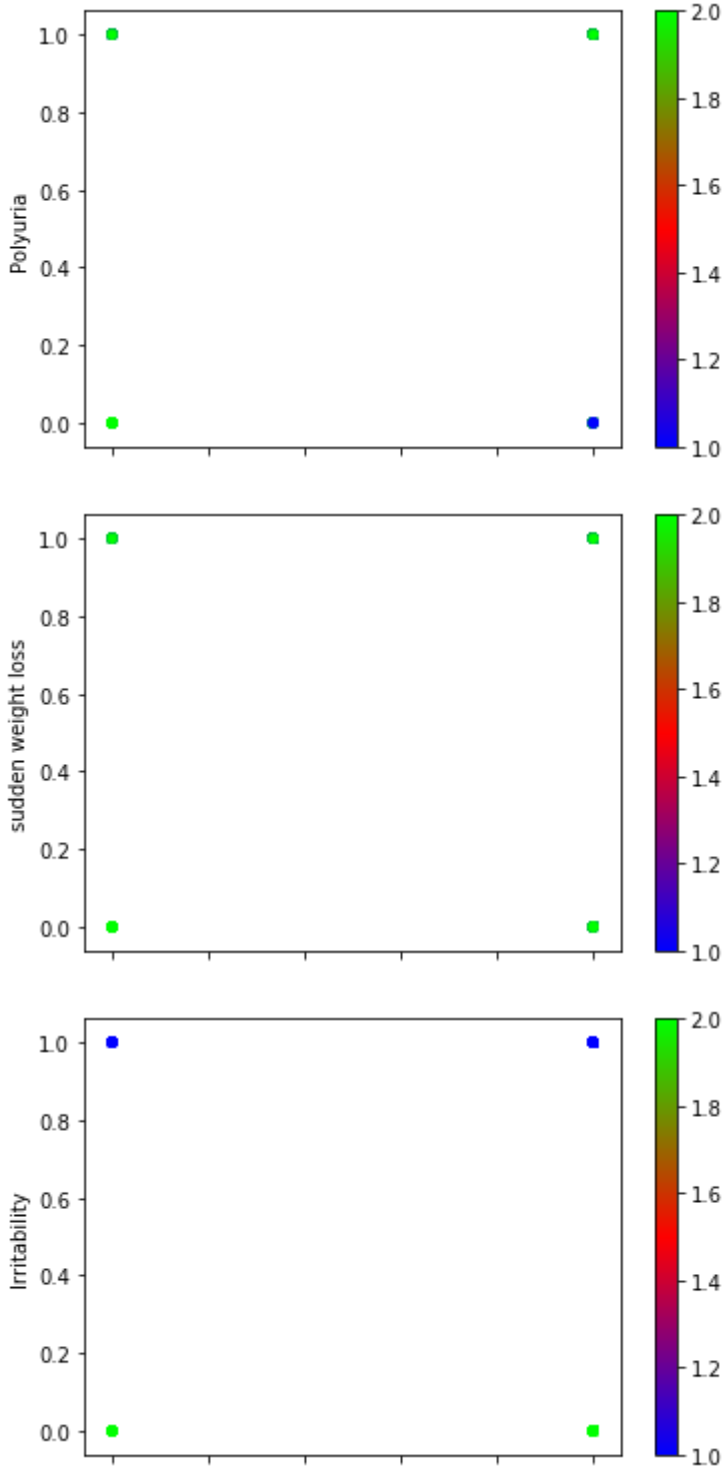
In [169]:
```
#Plot teh contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y2,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

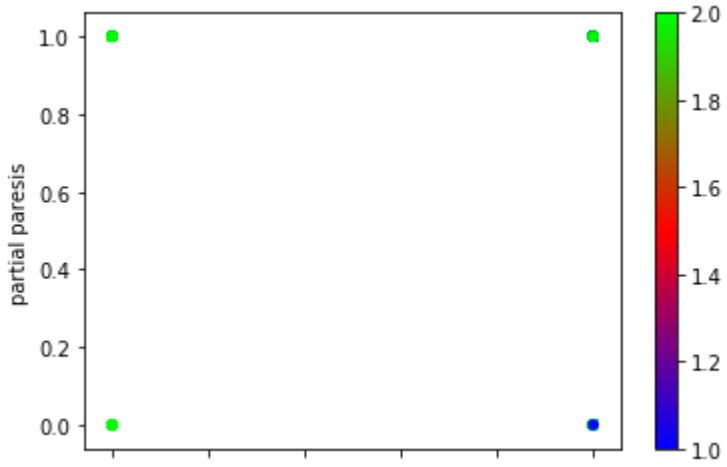


In [170]:
```
adjusted_rand_index = metrics.adjusted_rand_score(Y2, clusters)
silhouette_coefficient = metrics.silhouette_score(X2_scaled, clusters, metric
= "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.0016408539934176282, 0.3183722153202761]
```

In [171]:
```
#Plot the clusters formed by Hierarchical clustering
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Polyur
ia', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = "sudden
weight loss", c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Irrita
bility', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'partia
l paresis', c = clusters, colormap = plt.cm.brg)
```


Polyuria
sudden weight loss
Irritability
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.2
1.4
1.6
1.8
2.0

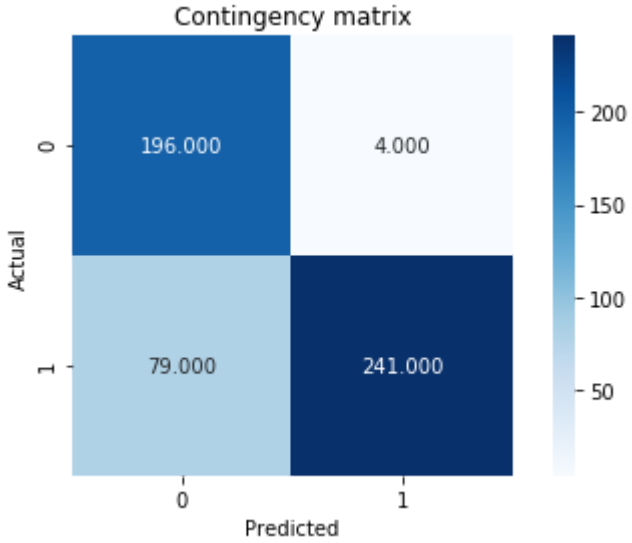


In [ ]:

In [172]:
```
#KMeans clustering with random centroids and 1 iteration
```

In [173]:
```
clustering = KMeans(n_clusters = 2, init = 'random', n_init = 1, random_state
= 0).fit(X2_scaled)
clusters = clustering.labels_
```

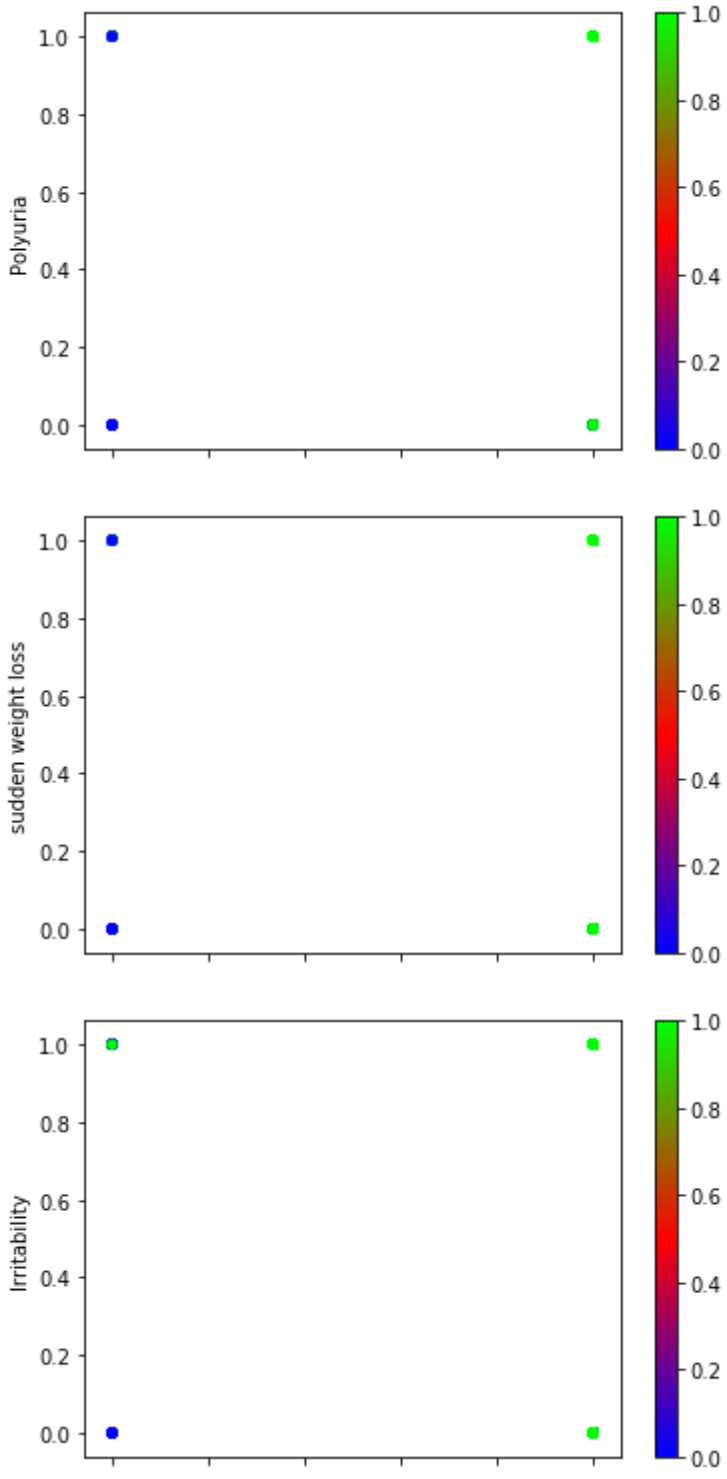
In [174]:
```
#Plot the contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y2,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

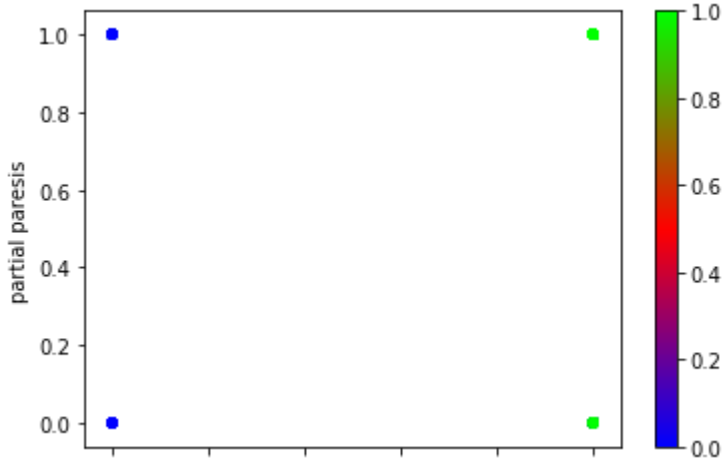


In [175]:
```
adjusted_rand_index = metrics.adjusted_rand_score(Y2, clusters)
silhouette_coefficient = metrics.silhouette_score(X2_scaled, clusters, metric
= "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.4623739917951326, 0.4088798328173262]
```

In [176]:
```
#Plot the clusters formed by KMeans clustering
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Polyur
ia', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = "sudden
weight loss", c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Irrita
bility', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'partia
l paresis', c = clusters, colormap = plt.cm.brg)
```


Polyuria
sudden weight loss
Irritability
0.0
0.2
0.4
0.6
0.8
1.0

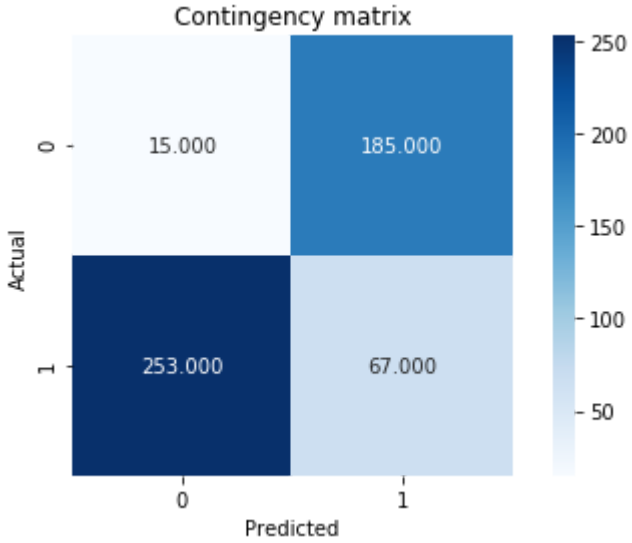


In [ ]:

In [177]:
```
#KMeans with 20 iterations and the centroids are chosen iteratively
```

In [178]:
```
clustering = KMeans(n_clusters = 2, init = 'k-means++', n_init = 20,random_sta
te = 0).fit(X2_scaled)
clusters = clustering.labels_
```

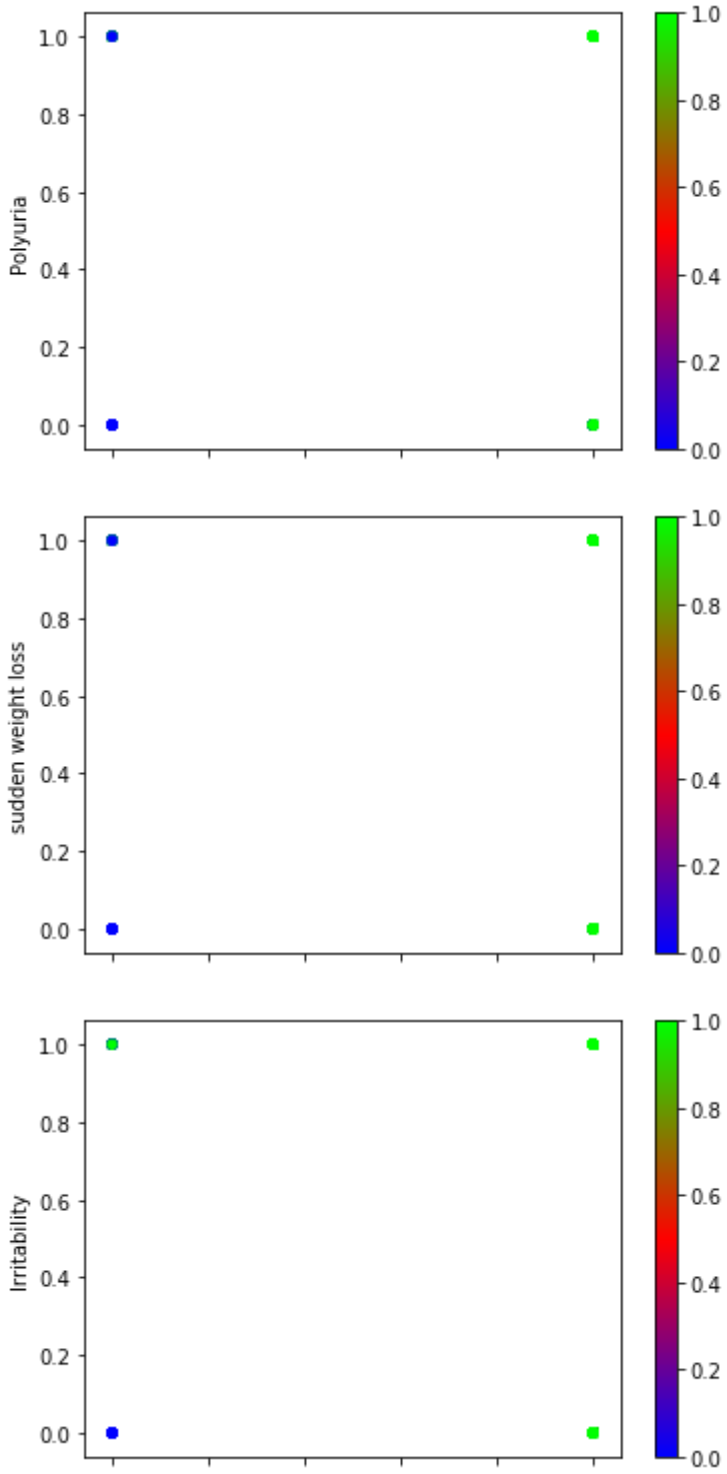
In [179]:
```
#Plot the contingency matrix
cont_matrix = metrics.cluster.contingency_matrix(Y2,clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

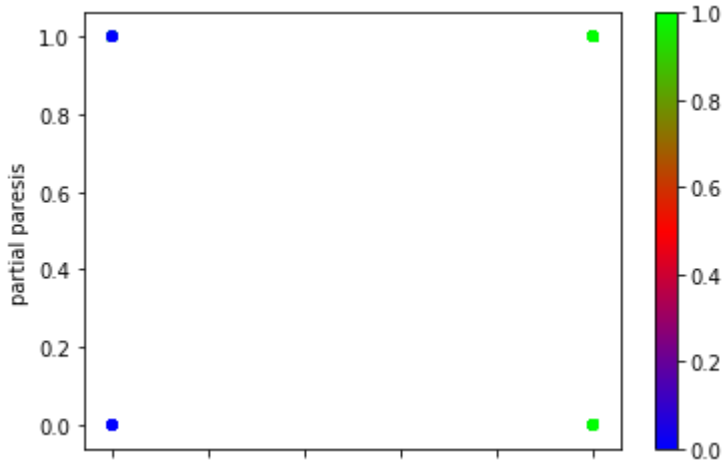


In [180]:
```
#Calculate the adjusted rand index and silhouette coeff
adjusted_rand_index = metrics.adjusted_rand_score(Y2, clusters)
silhouette_coefficient = metrics.silhouette_score(X2_scaled, clusters, metric
= "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.46770130181377945, 0.4109783250553093]
```

In [181]:
```
#Plot the clusters formed by KMeans
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Polyur
ia', c = Y2, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = "sudden
weight loss", c = Y2, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Irrita
bility', c = Y2, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'partia
l paresis', c = Y2, colormap = plt.cm.brg)
```


Polyuria
sudden weight loss
Irritability
0.0
0.2
0.4
0.6
0.8
1.0



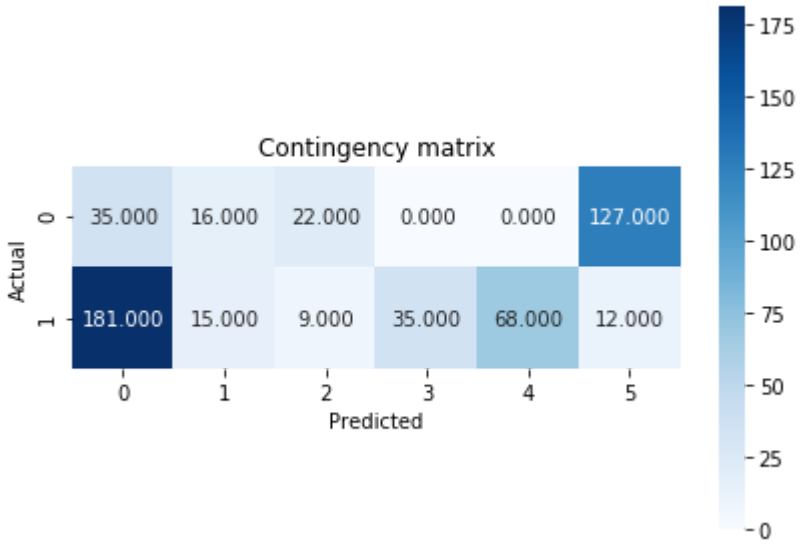
In [ ]:

In [182]:
```
#DBSCAN for eps = 1 and min_Sample = 30
```

In [183]:
```
clustering = DBSCAN(eps = 1, min_samples = 30, metric = "euclidean").fit(X2_sc
aled)
clusters = clustering.labels_
print(clusters)
```

```
[-1  0 -1  1  2 -1 -1  2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
  3 -1 -1  3 -1  2  3 -1  4  2 -1 -1 -1 -1 -1 -1 -1  3 -1  3  3 -1  3  3
 -1 -1  2  3 -1 -1  0  4  3  0  4 -1 -1  3 -1 -1  0 -1 -1 -1  4  3  1 -1
  4  3  3  3  3  3  2  3 -1  1 -1 -1  3  1  2  0 -1  4  0  2  3 -1 -1 -1
  3  2 -1 -1 -1  3 -1 -1 -1 -1 -1 -1 -1 -1  1  2 -1 -1 -1  2  2 -1 -1 -1
 -1 -1 -1 -1 -1  3 -1 -1 -1  3 -1 -1 -1 -1 -1 -1 -1  2  2 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  2 -1 -1  2  0 -1  4  0  2  3  3 -1
  4  2 -1 -1 -1 -1 -1 -1 -1  3 -1  3  3  2 -1 -1 -1  3 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1  0 -1 -1 -1 -1  1  4  4  4 -1  0  4  1  1  4 -1 -1 -1  4  4
  4 -1  4  4  4  4  1  4  4  4  4  1 -1  4  1  4 -1  0  4  4  4  0  4  4
  0 -1  4  4  4  4  4  0  4  3  3  3  2  3 -1  1 -1 -1  3 -1 -1  2 -1 -1
  2  0 -1  4  0  2  3  3  4  4  4 -1  4  4  4  4  1  4  4  4 -1  4  4  4
  4  1  4  4  4  0  4  3  3  3  2  3 -1  1 -1 -1 -1 -1  1  4  4  4 -1  0
  4  1  1  4  0 -1  0  1  4  4 -1 -1  4  4  4 -1 -1  1  4  4  4  4  4  0
  4  3  3  3  2  3 -1  1 -1 -1 -1 -1  1  4  4  4 -1  0  4  1  1  4  2 -1
 -1  2  0 -1  4  0  2  3  3  4  4  4 -1  4  4  4  4  1 -1 -1 -1 -1 -1 -1
 -1 -1 -1  4  4  4 -1 -1  1  4  4  4  4  4  0 -1 -1  4  4  4 -1 -1  1  4
  4  4  4  4  0  4  3  3  3  2  3 -1  1 -1 -1 -1 -1 -1 -1 -1 -1  3 -1 -1
  3 -1  2  3 -1  4  2 -1 -1 -1 -1 -1 -1 -1  3 -1  3  3 -1  3  2  0 -1  4
  0  2  3  3  4  4  4 -1  4  4  4  4  1  4  4  4 -1  4  4  4  4  1  4  4
  4  0  4  3  3  3  2  3  4  4  4  0 -1 -1  4  4  4 -1 -1 -1 -1  4 -1  4
  4  4  4  1  4  4  4  0  4  3  3  3  2  3  4  4]
```

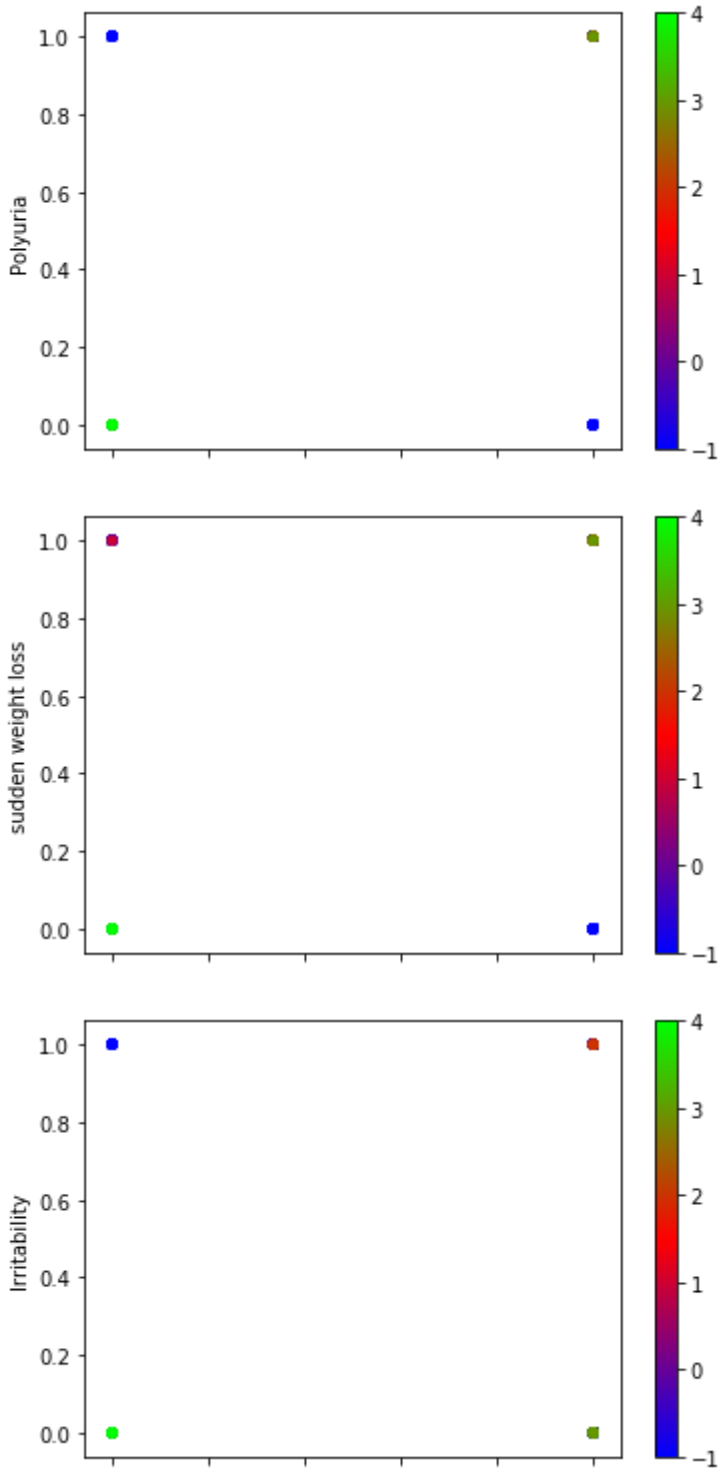
In [184]:
```
#Plot the contingency matrix and we see that we have 6 clusters formed for the
following eps and min_sample values
cont_matrix = metrics.cluster.contingency_matrix(Y2, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

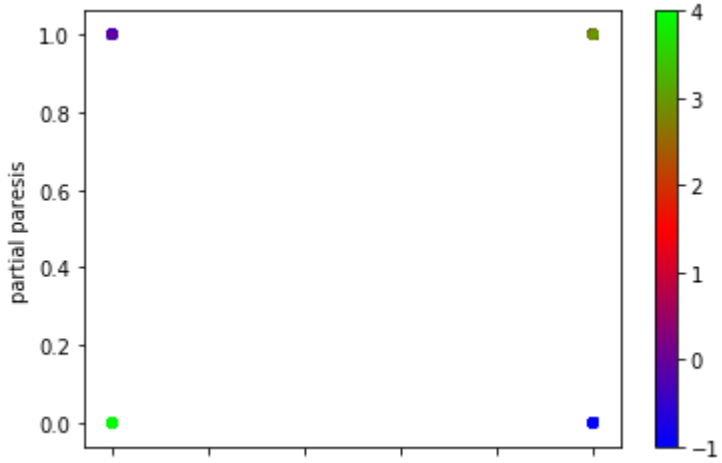


In [185]:
```
adjusted_rand_index = metrics.adjusted_rand_score(Y2, clusters)
silhouette_coefficient = metrics.silhouette_score(X2_scaled, clusters, metric
= "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.26263738731225866, 0.48709627008582834]
```

In [186]:
```
#Plot the clusters formed by DBSCAN
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Polyur
ia', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = "sudden
weight loss", c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Irrita
bility', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'partia
l paresis', c = clusters, colormap = plt.cm.brg)
```


Polyuria
sudden weight loss
Irritability
1.0
0.8
0.6
0.4
0.2
0.0
4
3
2
1
0
−1



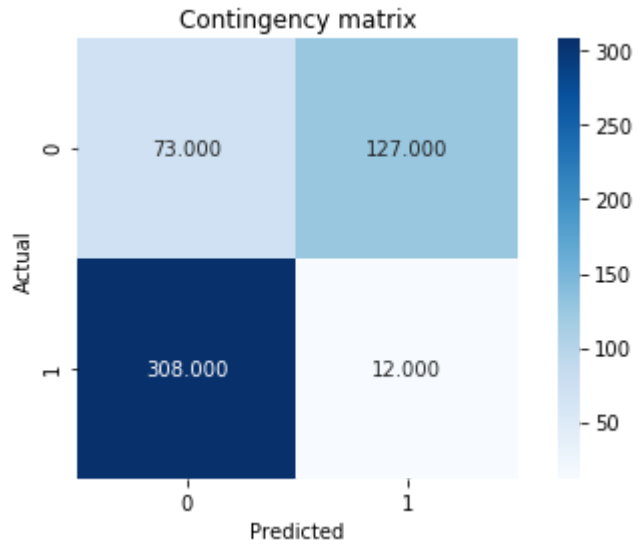
In [ ]:

In [187]:
```
# DBSCAN modified to form two clusters
```

In [188]:
```
clustering = DBSCAN(eps = 2, min_samples = 70, metric = "euclidean").fit(X2_sc
aled)
clusters = clustering.labels_
print(clusters)
```

```
[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1  0 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1
  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1
  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0 -1 -1  0 -1 -1  0 -1 -1 -1  0  0
  0 -1  0  0  0  0 -1  0  0  0  0 -1 -1  0 -1  0 -1 -1  0  0  0 -1  0  0
 -1 -1  0  0  0  0  0 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1  0 -1 -1 -1 -1  0  0  0 -1  0  0  0  0 -1  0  0  0 -1  0  0  0
  0 -1  0  0  0 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0 -1 -1
  0 -1 -1  0 -1 -1 -1 -1  0  0 -1 -1  0  0  0 -1 -1 -1  0  0  0  0  0 -1
  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0  0  0 -1 -1  0 -1 -1  0 -1 -1
 -1 -1 -1 -1  0 -1 -1 -1 -1  0  0  0 -1  0  0  0  0 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1  0  0  0 -1 -1 -1  0  0  0  0  0 -1 -1 -1  0  0  0 -1 -1 -1  0
  0  0  0  0 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0
 -1 -1 -1 -1  0  0  0 -1  0  0  0  0 -1  0  0  0 -1  0  0  0  0 -1  0  0
  0 -1  0 -1 -1 -1 -1 -1  0  0  0 -1 -1 -1  0  0  0 -1 -1 -1 -1  0 -1  0
  0  0  0 -1  0  0  0 -1  0 -1 -1 -1 -1 -1  0  0]
```

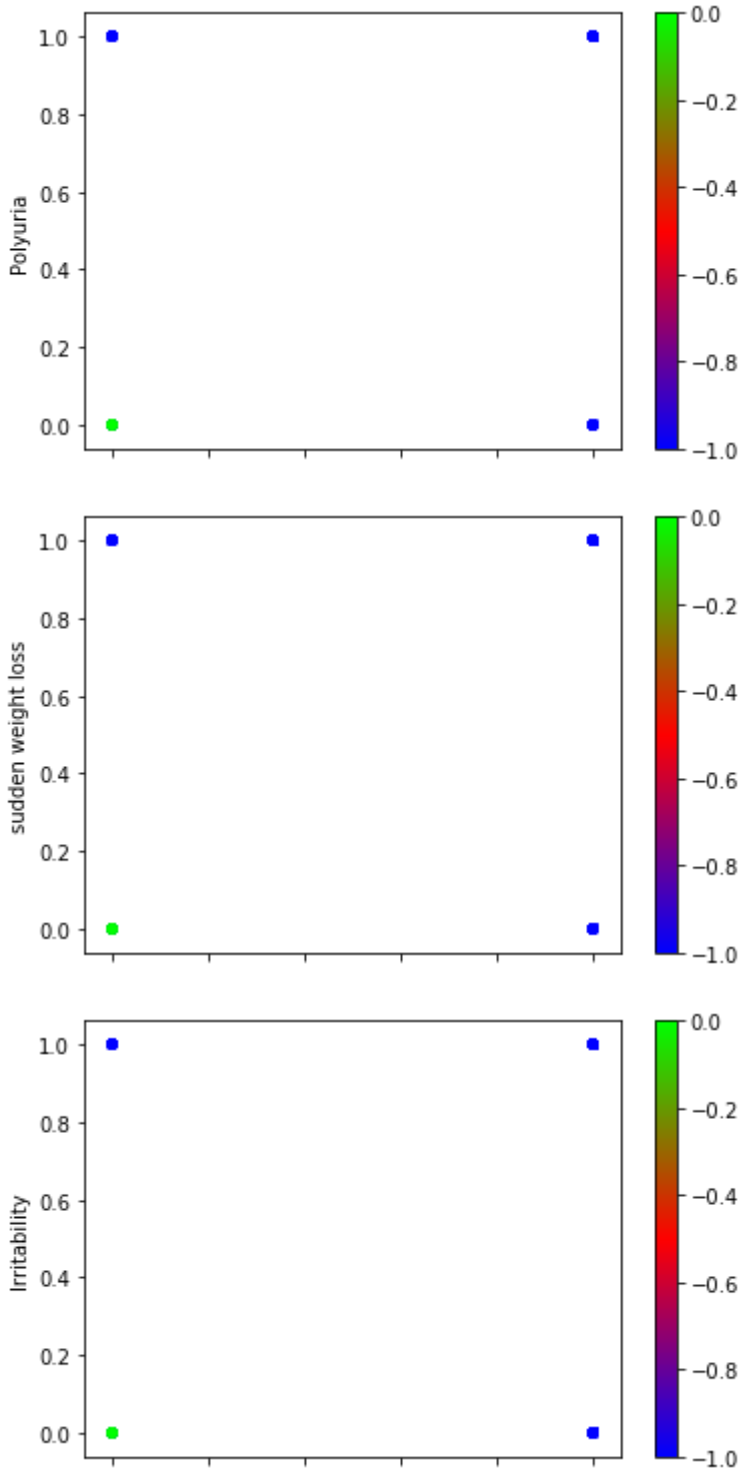
In [189]:
```
cont_matrix = metrics.cluster.contingency_matrix(Y2, clusters)
sns.heatmap(cont_matrix, annot = True, fmt = ".3f", square = True, cmap = plt.
cm.Blues)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Contingency matrix')
plt.tight_layout()
```

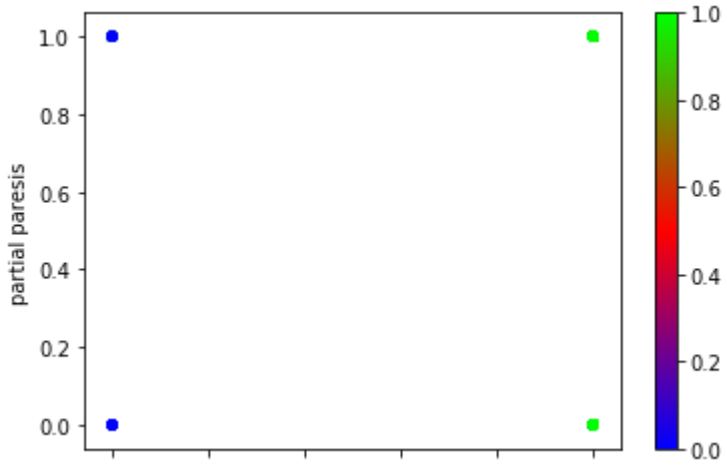


In [190]:
```
adjusted_rand_index = metrics.adjusted_rand_score(Y2, clusters)
silhouette_coefficient = metrics.silhouette_score(X2_scaled, clusters, metric
= "euclidean")
print([adjusted_rand_index, silhouette_coefficient])
```

```
[0.44584903807930437, 0.3100785597576417]
```

In [191]:
```
#Plot clusters formed by DBSCAN
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Polyur
ia', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = "sudden
weight loss", c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Irrita
bility', c = clusters, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'partia
l paresis', c = Y2, colormap = plt.cm.brg)
```


Polyuria
sudden weight loss
Irritability
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0



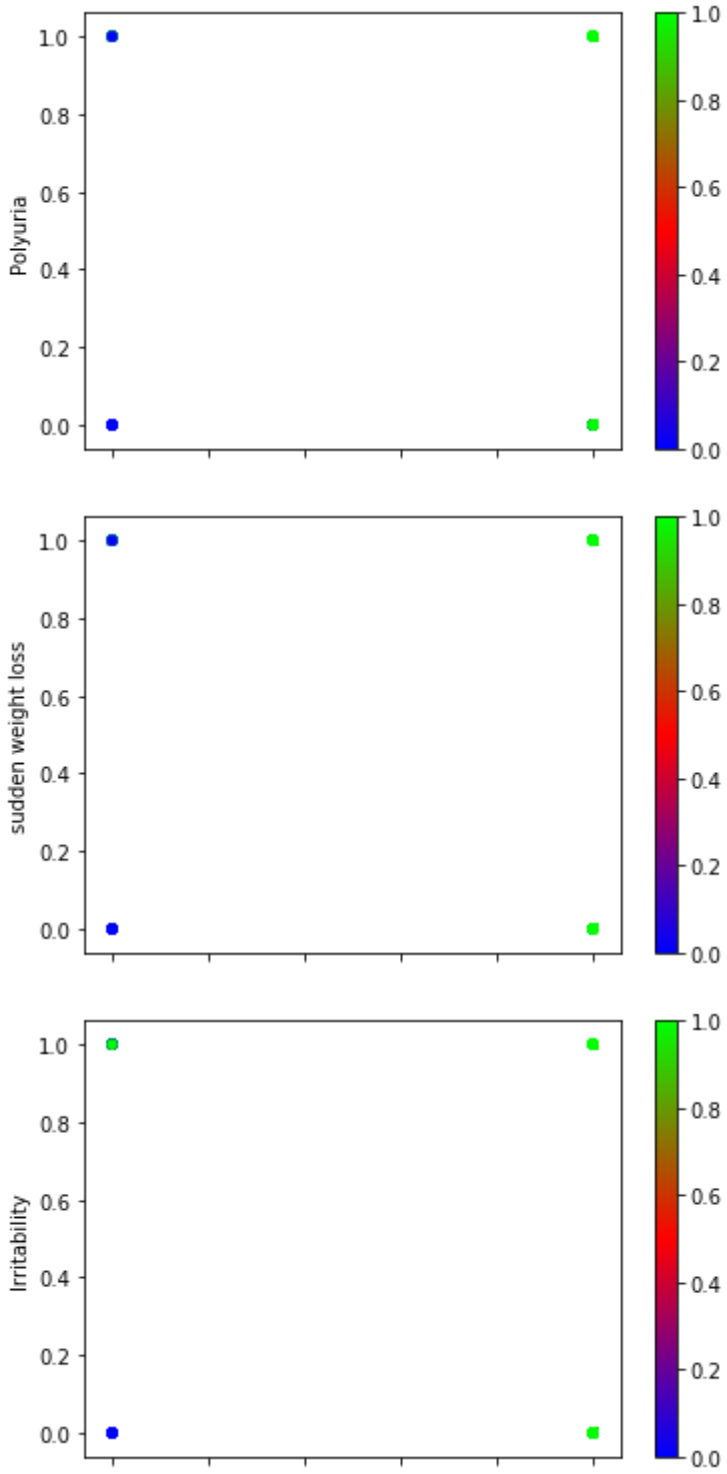
In [ ]:

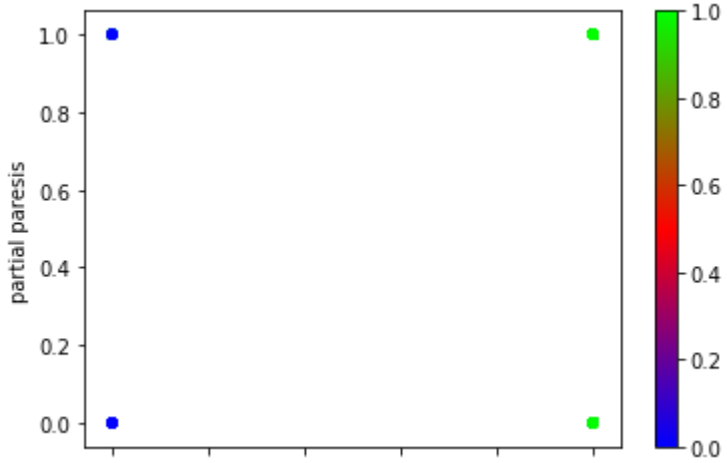
In [192]:
```
#Evaluation Matrix for true clusters for binary data
```

In [193]:
```
silhouette_coefficient = metrics.silhouette_score(X2_scaled, Y2, metric = "euc
lidean")
print(silhouette_coefficient)
```

```
0.28719132813462656
```

In [194]:
```
# Plot true clusters
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Polyuria', c = Y2, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = "sudden weight loss", c = Y2, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'Irritability', c = Y2, colormap = plt.cm.brg)
ax = diabetes_data_binary.plot(kind = 'scatter', x = 'Polydipsia', y = 'partial paresis', c = Y2, colormap = plt.cm.brg)
```


Polyuria
sudden weight loss
Irritability
0.0
0.2
0.4
0.6
0.8
1.0



In [ ]: