K Nearest Neighbors Classification

1) Iris data set with KNN

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import ListedColormap
from sklearn import neighbors, datasets
```

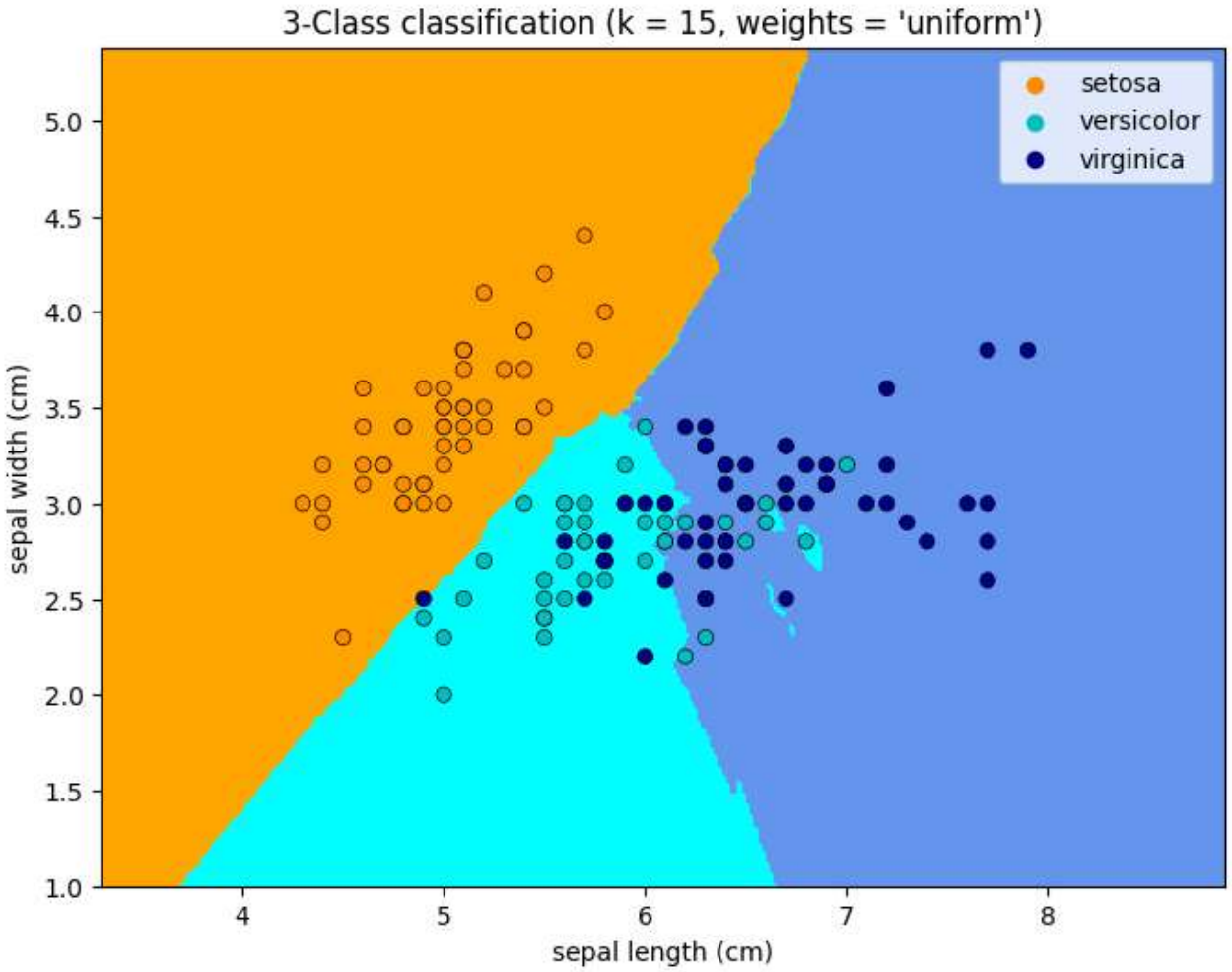
In [2]:
```
n_neighbors = 15

iris = datasets.load_iris()
```

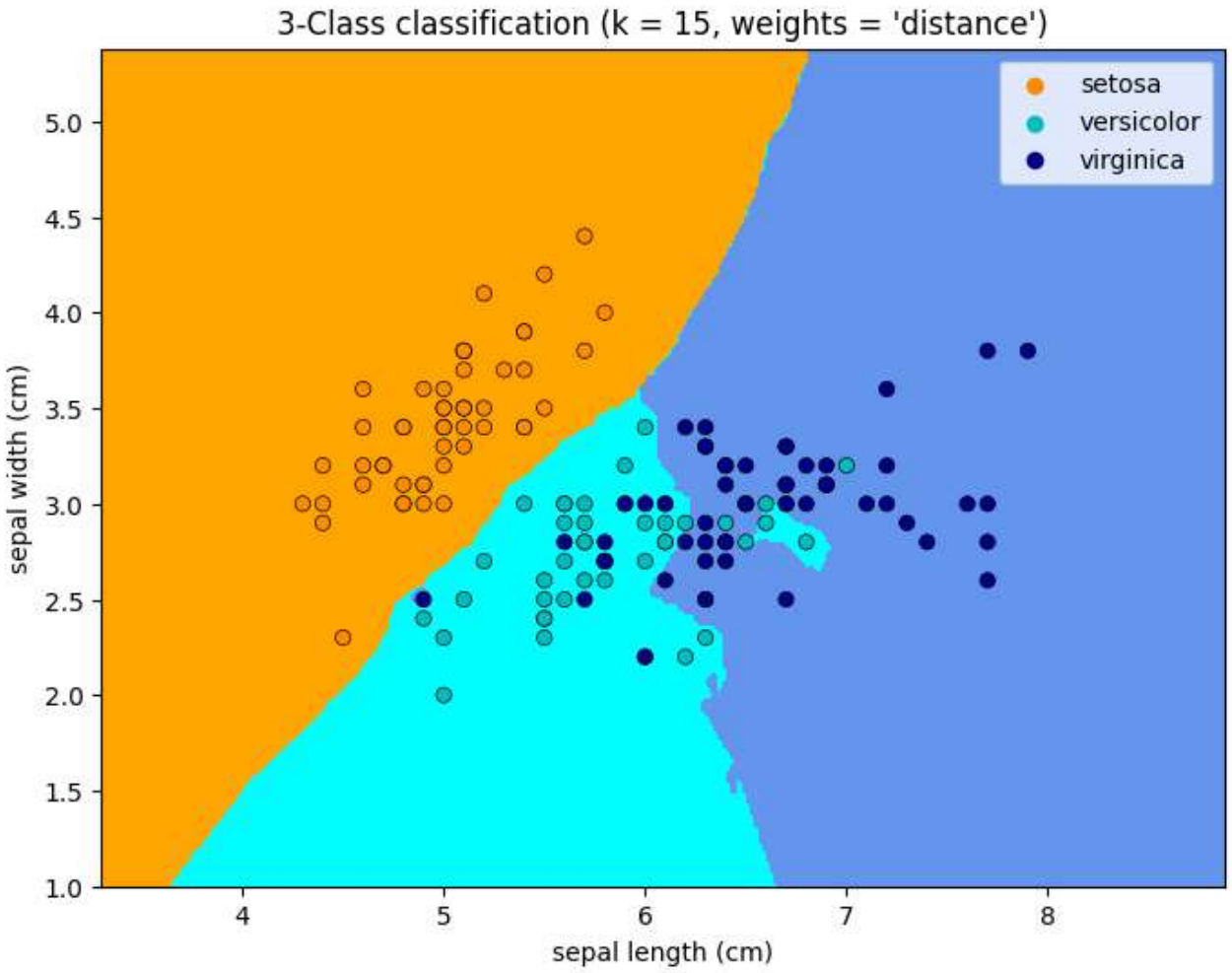
In [3]:
```
X = iris.data[:, :2]
y = iris.target
```

In [4]:
```
h = .02

cmap_light = ListedColormap(['orange', 'cyan', 'cornflowerblue'])
cmap_bold = ['darkorange', 'c', 'darkblue']
```

In [5]:
```
for weights in ['uniform', 'distance']:
    clf = neighbors.KNeighborsClassifier(n_neighbors, weights=weights)
    clf.fit(X, y)


    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])



    Z = Z.reshape(xx.shape)
    plt.figure(figsize=(8, 6))
    plt.contourf(xx, yy, Z, cmap=cmap_light)



    sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=iris.target_names[y],
                    palette=cmap_bold, alpha=1.0, edgecolor="black")
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.title("3-Class classification (k = %i, weights = '%s')"
```

```
        % (n_neighbors, weights))
    plt.xlabel(iris.feature_names[0])
    plt.ylabel(iris.feature_names[1])

plt.show()
```





2) Diabetes data set with KNN

In [6]:
```
from mlxtend.plotting import plot_decision_regions
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [7]:
```
diabetes_data = pd.read_csv('diabetes.csv')

diabetes_data.head()
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	C
0	6	148	72	35	0	33.6	0.627	50	
1	1	85	66	29	0	26.6	0.351	31	
2	8	183	64	0	0	23.3	0.672	32	

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	C
3	1	89	66	23	94	28.1	0.167	21	
4	0	137	40	35	168	43.1	2.288	33	

In [8]:
```
diabetes_data.info(verbose=True)
```

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

In [9]:
```
diabetes_data.describe()
```

Out[9]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [10]:
```
diabetes_data.describe().T
```

Out[10]:

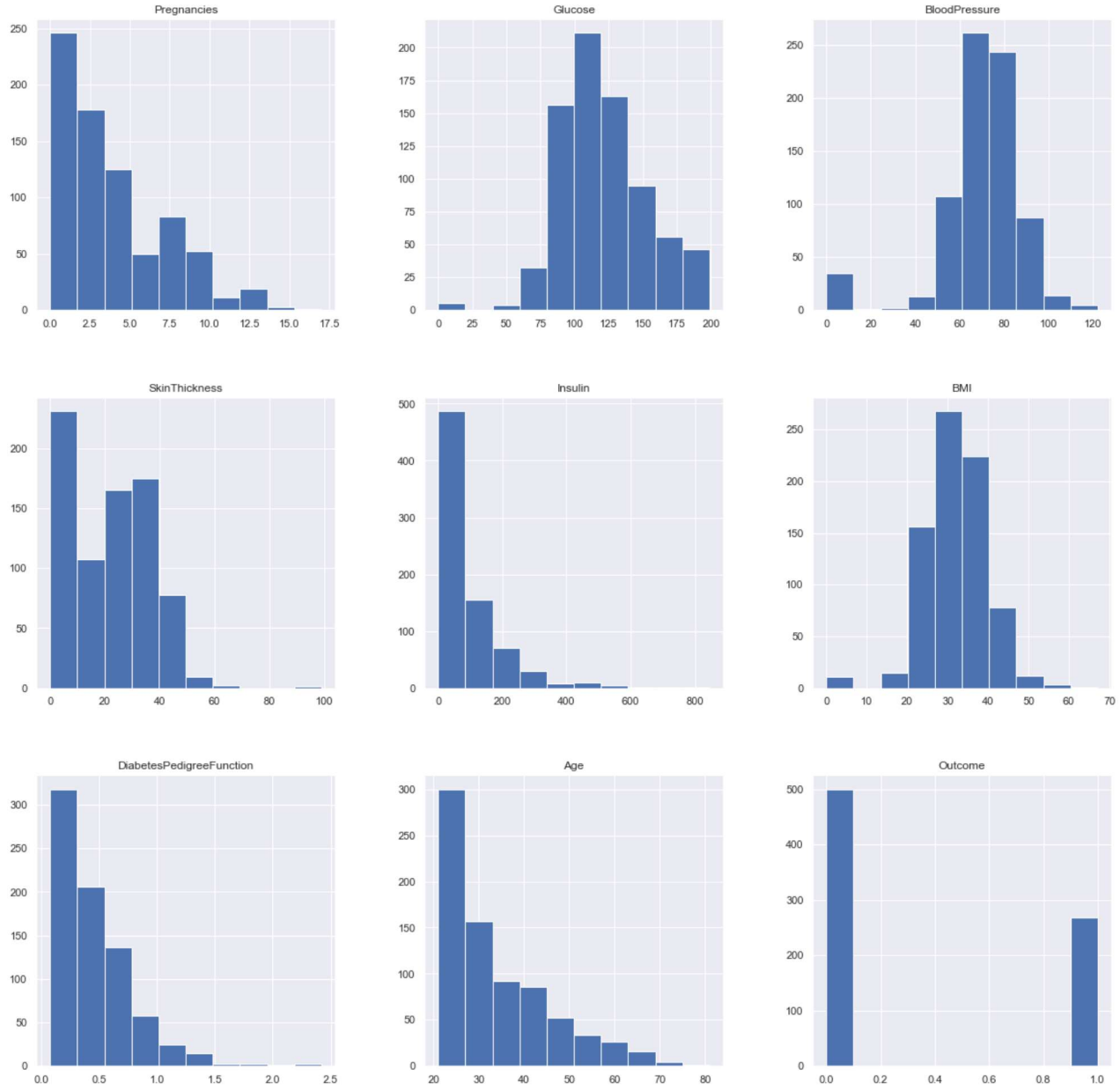
	count	mean	std	min	25%	50%	75%	r
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99

	count	mean	std	min	25%	50%	75%	r
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1

In [11]:
```
diabetes_data_copy = diabetes_data.copy(deep = True)
diabetes_data_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = diabetes_data_copy[['Glucose','Blo


print(diabetes_data_copy.isnull().sum())
```

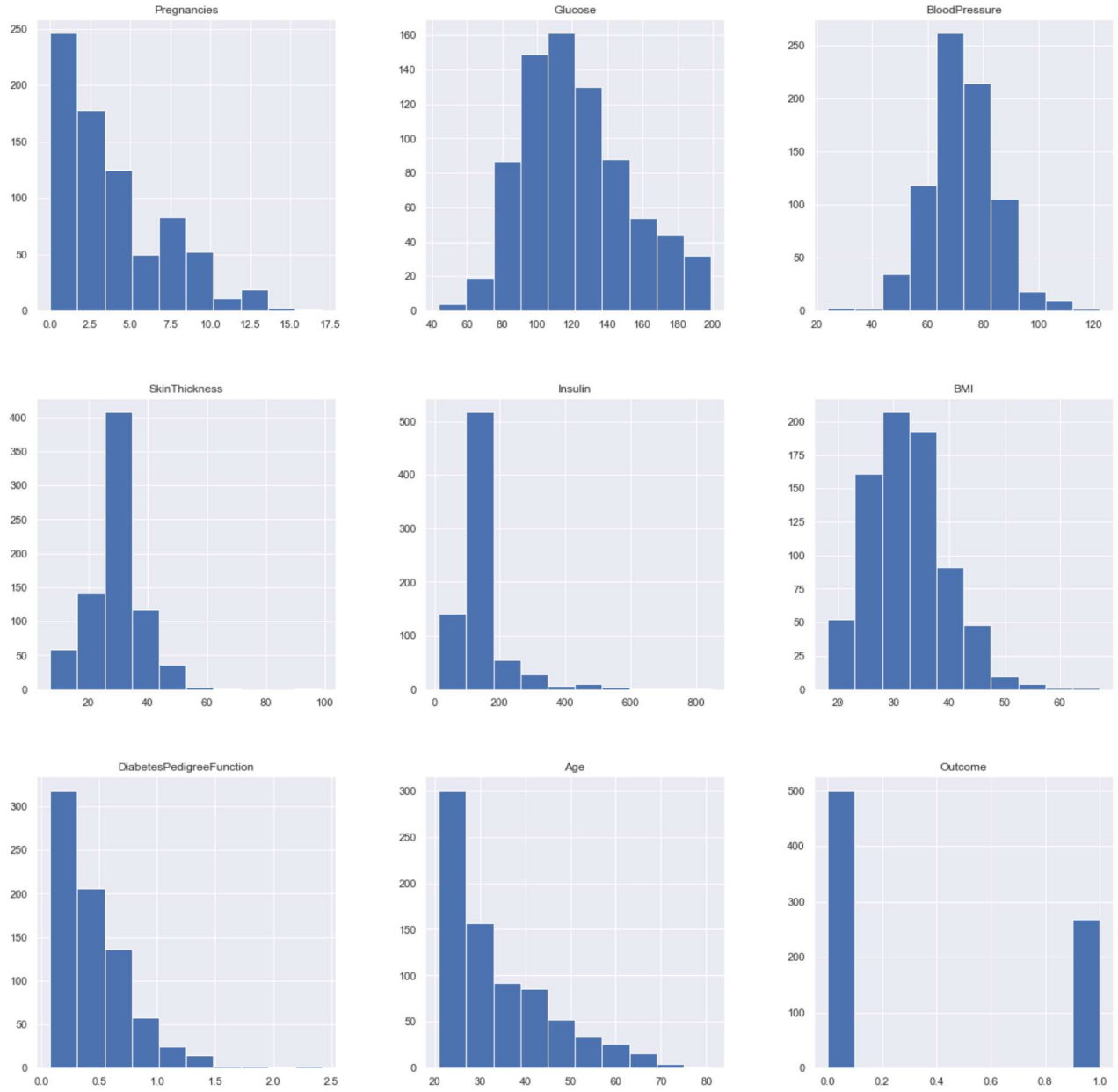
```
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [12]:
```
p = diabetes_data.hist(figsize = (20,20))
```



In [13]:
```
diabetes_data_copy['Glucose'].fillna(diabetes_data_copy['Glucose'].mean(), inplace = True)
diabetes_data_copy['BloodPressure'].fillna(diabetes_data_copy['BloodPressure'].mean(), inplace = True)
diabetes_data_copy['SkinThickness'].fillna(diabetes_data_copy['SkinThickness'].median(), inplace = True)
diabetes_data_copy['Insulin'].fillna(diabetes_data_copy['Insulin'].median(), inplace = True)
diabetes_data_copy['BMI'].fillna(diabetes_data_copy['BMI'].median(), inplace = True)
```

In [14]:
```
p = diabetes_data_copy.hist(figsize = (20,20))
```

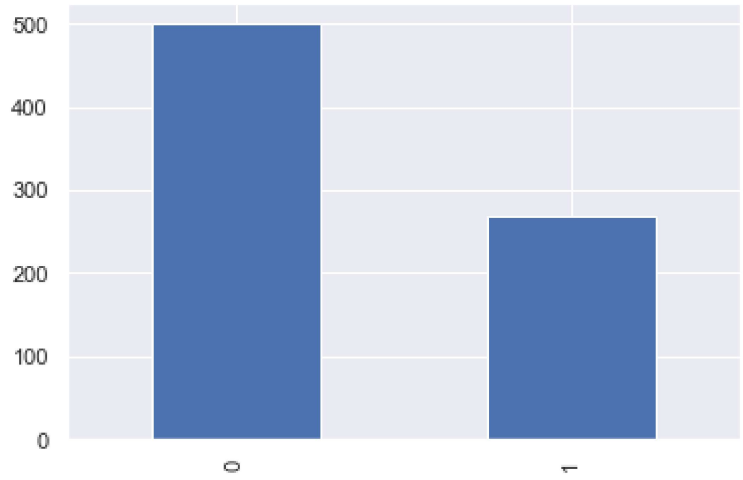


In [15]:
```
diabetes_data.shape
```

Out[15]: (768, 9)

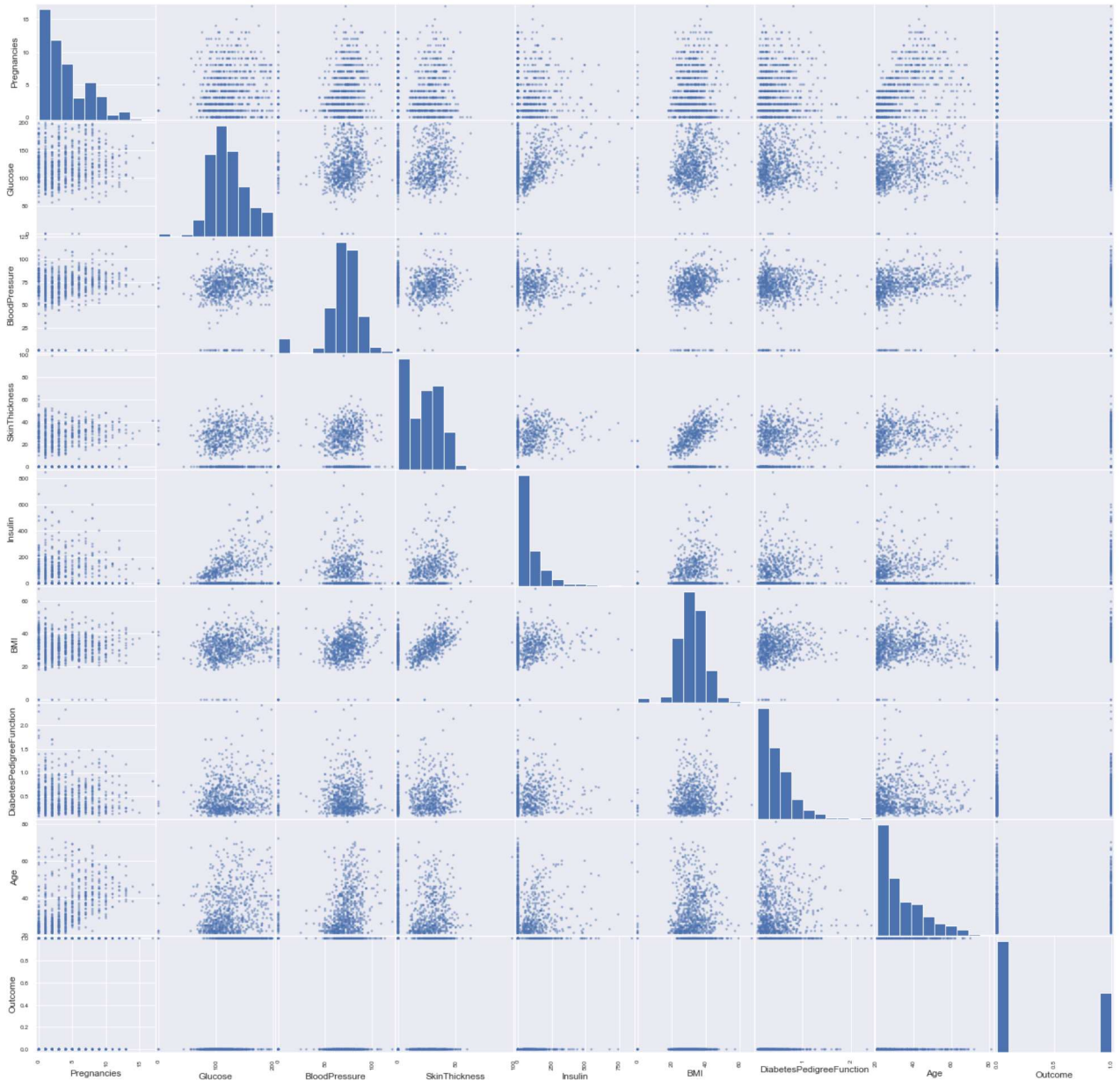
In [16]:
```
color_wheel = {1: "#0392cf", 2: "#7bc043"}
colors = diabetes_data["Outcome"].map(lambda x: color_wheel.get(x + 1))
print(diabetes_data.Outcome.value_counts())
p=diabetes_data.Outcome.value_counts().plot(kind="bar")
```

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



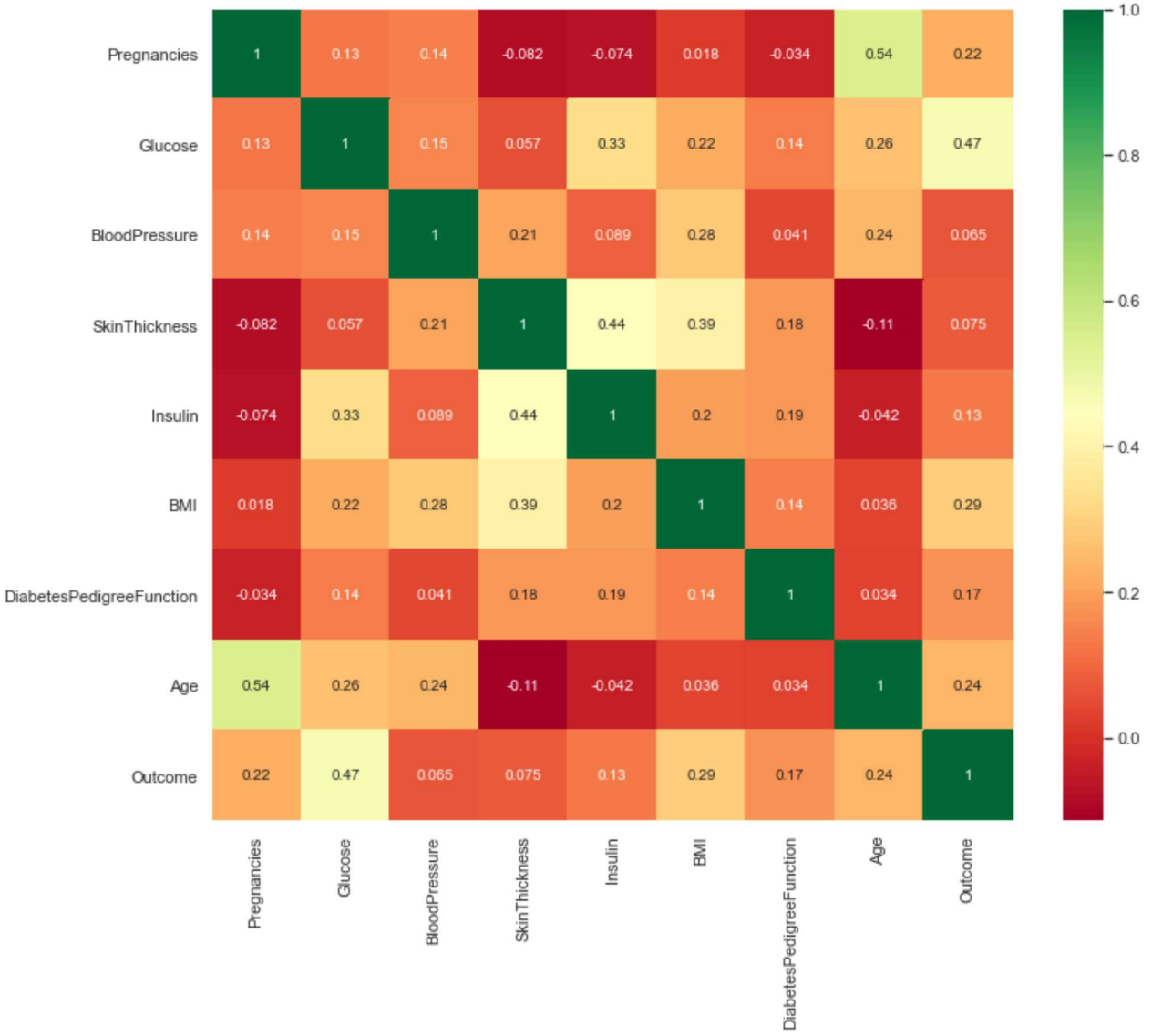
In [17]:

```
from pandas.plotting import scatter_matrix
p=scatter_matrix(diabetes_data,figsize=(25, 25))
```



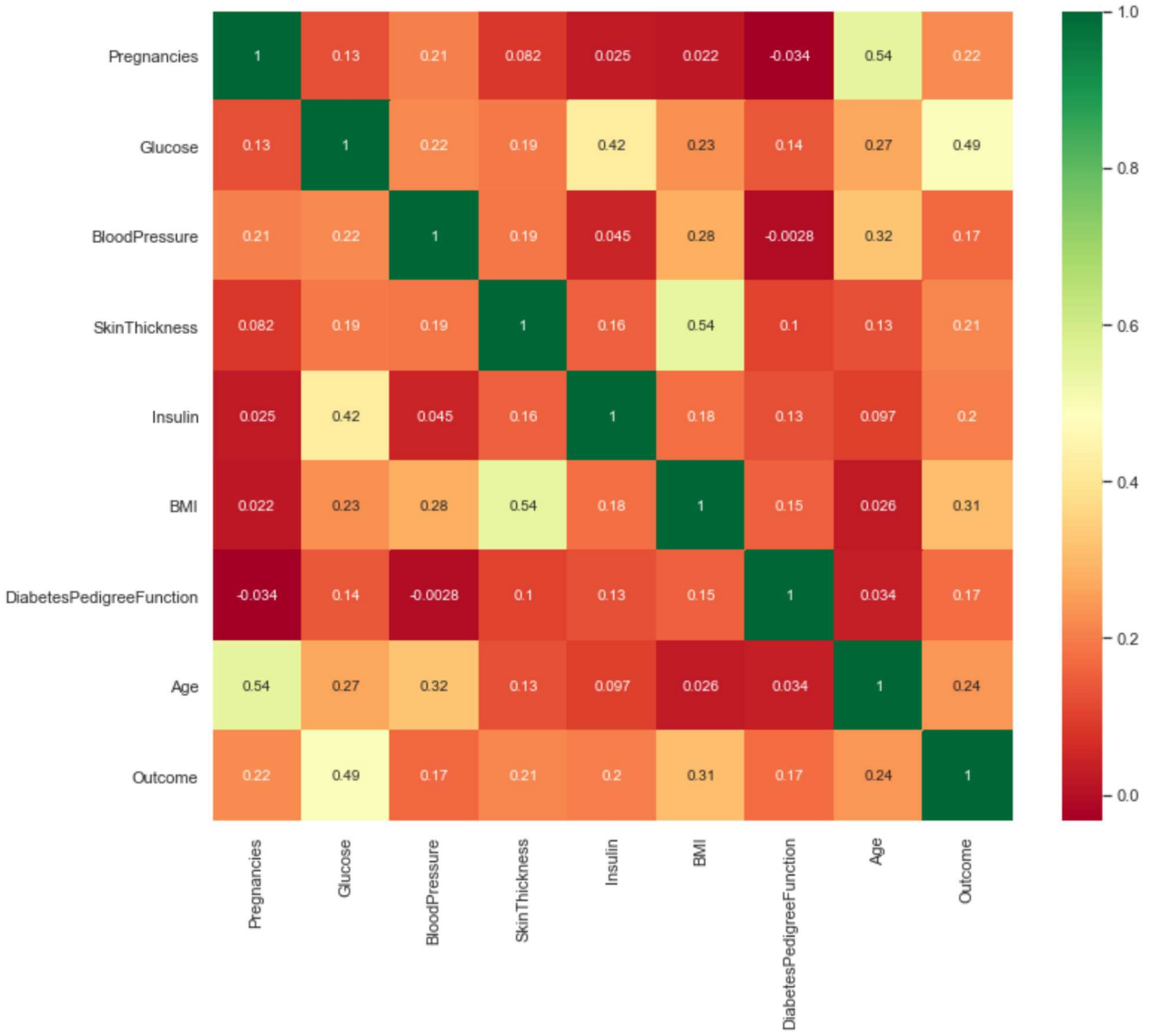
In [18]:
```
p=sns.pairplot(diabetes_data_copy, hue = 'Outcome')
```

In [19]:
```
plt.figure(figsize=(12,10))
p=sns.heatmap(diabetes_data.corr(), annot=True,cmap ='RdYlGn')
```



In [20]:

```
plt.figure(figsize=(12,10))
p=sns.heatmap(diabetes_data_copy.corr(), annot=True,cmap ='RdYlGn')
```



In [21]:

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X =  pd.DataFrame(sc_X.fit_transform(diabetes_data_copy.drop(["Outcome"],axis = 1),),
      columns=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
     'BMI', 'DiabetesPedigreeFunction', 'Age'])
```

In [22]:

```
X.head()
```

Out[22]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctic
0	0.639947	0.865108	-0.033518	0.670643	-0.181541	0.166619	0.46849
1	-0.844885	-1.206162	-0.529859	-0.012301	-0.181541	-0.852200	-0.36506
2	1.233880	2.015813	-0.695306	-0.012301	-0.181541	-1.332500	0.60439
3	-0.844885	-1.074652	-0.529859	-0.695245	-0.540642	-0.633881	-0.92076
4	-1.141852	0.503458	-2.680669	0.670643	0.316566	1.549303	5.48490

In [23]:
```
y = diabetes_data_copy.Outcome
```

In [24]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=1/3,random_state=42, stratify=y)
```

In [25]:
```
from sklearn.neighbors import KNeighborsClassifier


test_scores = []
train_scores = []

for i in range(1,15):

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

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

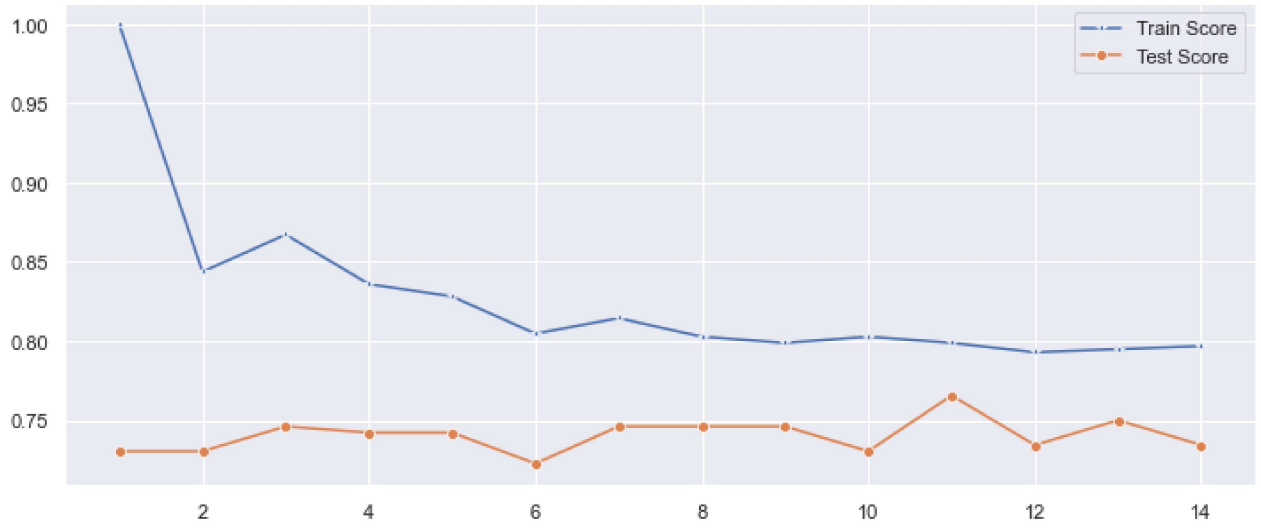
In [26]:
```
max_train_score = max(train_scores)
train_scores_ind = [i for i, v in enumerate(train_scores) if v == max_train_score]
print('Max train score {} % and k = {}'.format(max_train_score*100,list(map(lambda x: x+1, train_scores_ind))))
```

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

In [27]:
```
max_test_score = max(test_scores)
test_scores_ind = [i for i, v in enumerate(test_scores) if v == max_test_score]
print('Max test score {} % and k = {}'.format(max_test_score*100,list(map(lambda x: x+1, test_scores_ind))))
```

```
Max test score 76.5625 % and k = [11]
```

In [28]:
```
plt.figure(figsize=(12,5))
p = sns.lineplot(range(1,15),train_scores,marker='*',label='Train Score')
p = sns.lineplot(range(1,15),test_scores,marker='o',label='Test Score')
```

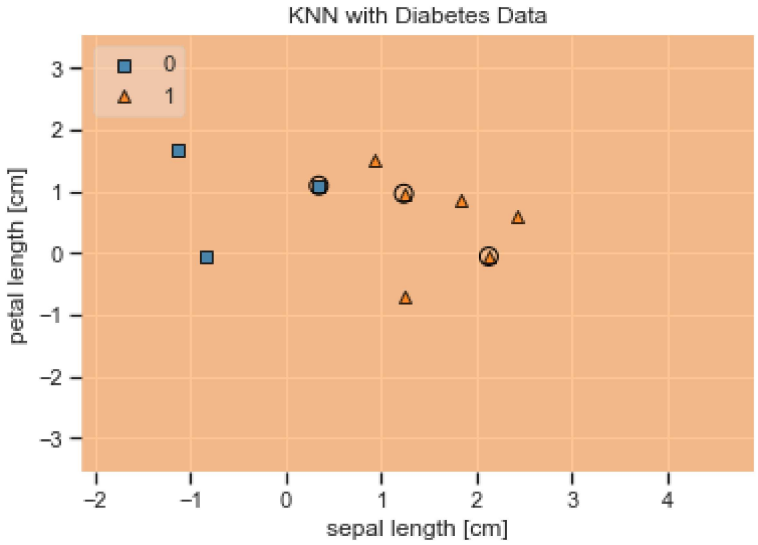


In [29]:
```
knn = KNeighborsClassifier(11)
```

```
knn.fit(X_train,y_train)
knn.score(X_test,y_test)
```

Out[29]: 0.765625

In [30]:
```
value = 20000
width = 20000
plot_decision_regions(X.values, y.values, clf=knn, legend=2,
                      filler_feature_values={2: value, 3: value, 4: value, 5: value, 6: value, 7: value},
                      filler_feature_ranges={2: width, 3: width, 4: width, 5: width, 6: width, 7: width},
                      X_highlight=X_test.values)

plt.xlabel('sepal length [cm]')
plt.ylabel('petal length [cm]')
plt.title('KNN with Diabetes Data')
plt.show()
```



In [31]:
```
from sklearn.metrics import confusion_matrix

y_pred = knn.predict(X_test)
confusion_matrix(y_test,y_pred)
pd.crosstab(y_test, y_pred, rownames=['True'], colnames=['Predicted'], margins=True)
```

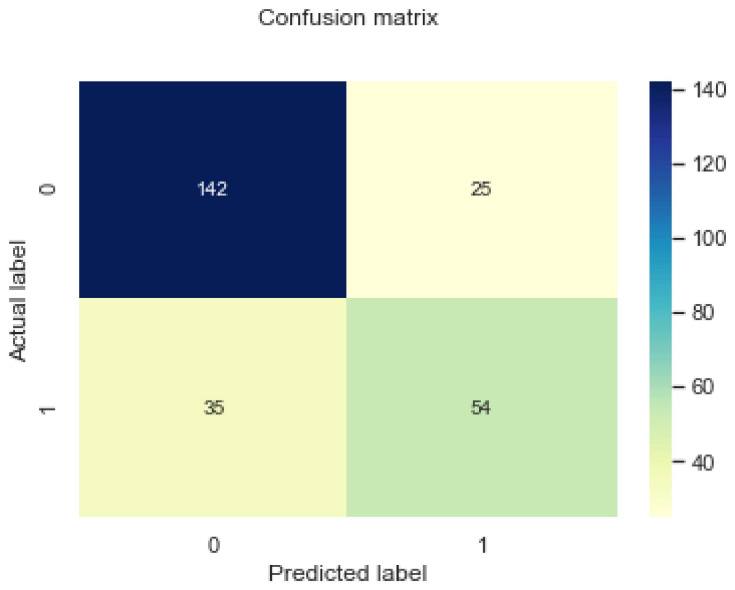
Out[31]:

Predicted	0	1	All
True			
0	142	25	167
1	35	54	89
All	177	79	256

In [32]:
```
y_pred = knn.predict(X_test)
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
p = sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
```

```
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[32]: Text(0.5, 12.5, 'Predicted label')



In [ ]: