Diabetes Risk Analysis

Attributes :-

(A)Glucose-level in the body (B) No. of Pregnancies (C)Blood Pressure (D)Skin thickness (E)Insulin level in the body (F)Diabetes Prediction function (G)Age

problem statement- To cluster the group of people into various clusters into risk level groups with the help of different Attributes.

In [321]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
```

In [259]:
```
dataset = pd.read_csv("diabetes_data_upload.csv")
```

In [260]:
```
dataset.head()
```

Out[260]:

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

In [261]:
```
dataset.shape
```

Out[261]: (768, 8)

In [262]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 8 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  
dtypes: float64(2), int64(6)
memory usage: 48.1 KB
```

In [263]:
```
dataset.describe()
```

Out[263]:

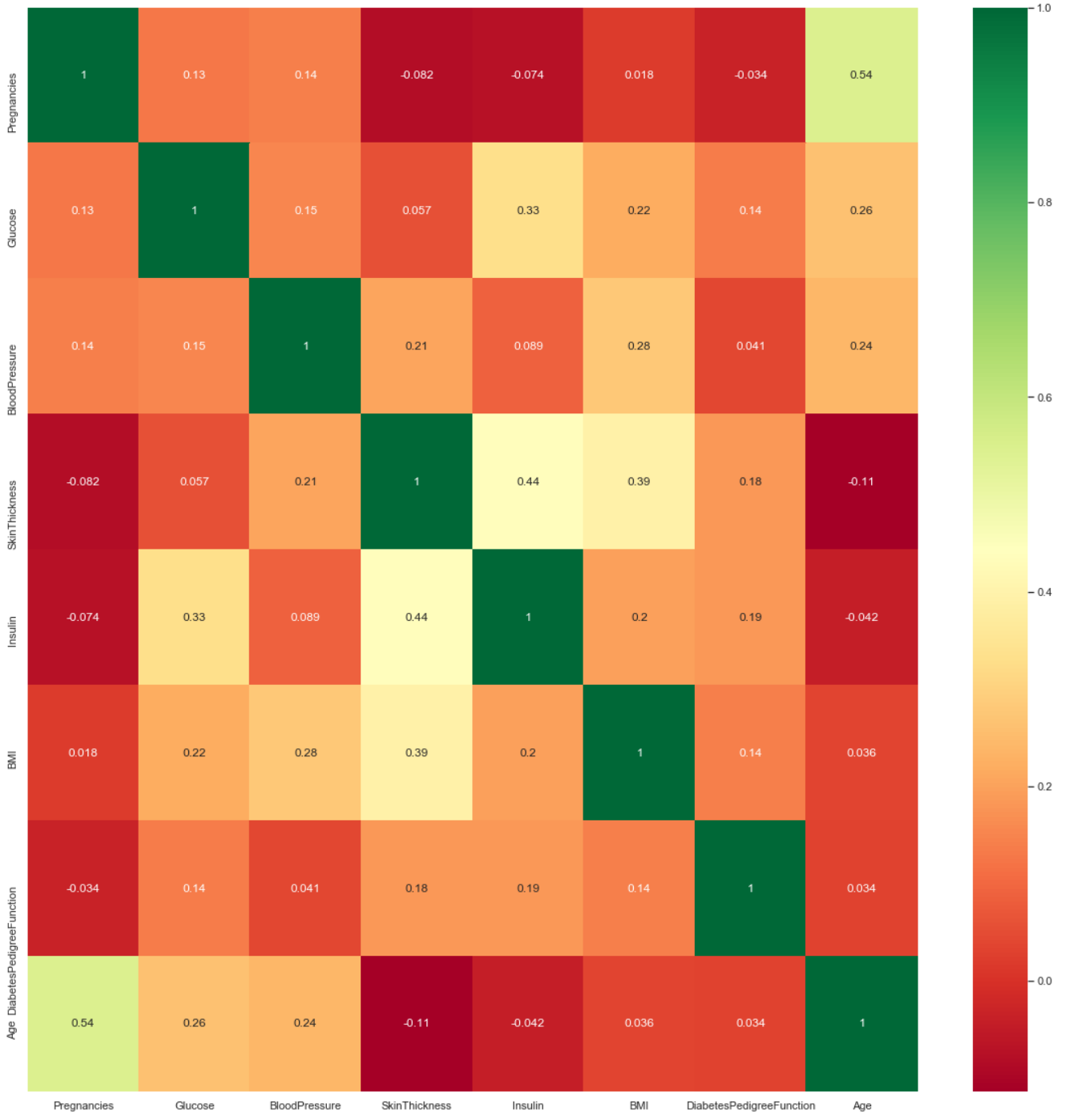
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeF
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2

In [264]:
```
dataset.isnull().values.any()
```

Out[264]:
```
False
```

In [283]:

```
#get correlations of each features in dataset
corrmat = dataset.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(dataset[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



In [285]:
```
dataset.corr()
```

Out[285]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diab
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	

In [265]:
```
print("number of rows missing glucose_conc: {0}".format(len(dataset.loc[dataset['Glucos
print("number of rows missing diastolic_bp: {0}".format(len(dataset.loc[dataset['BloodP
print("number of rows missing insulin: {0}".format(len(dataset.loc[dataset['Insulin'] =
print("number of rows missing bmi: {0}".format(len(dataset.loc[dataset['BMI'] == 0])))
print("number of rows missing diab_pred: {0}".format(len(dataset.loc[dataset['DiabetesP
print("number of rows missing age: {0}".format(len(dataset.loc[dataset['Age'] == 0])))
print("number of rows missing skin: {0}".format(len(dataset.loc[dataset['SkinThickness'
```

```
number of rows missing glucose_conc: 5
number of rows missing diastolic_bp: 35
number of rows missing insulin: 374
number of rows missing bmi: 11
number of rows missing diab_pred: 0
number of rows missing age: 0
number of rows missing skin: 227
```

In [ ]:

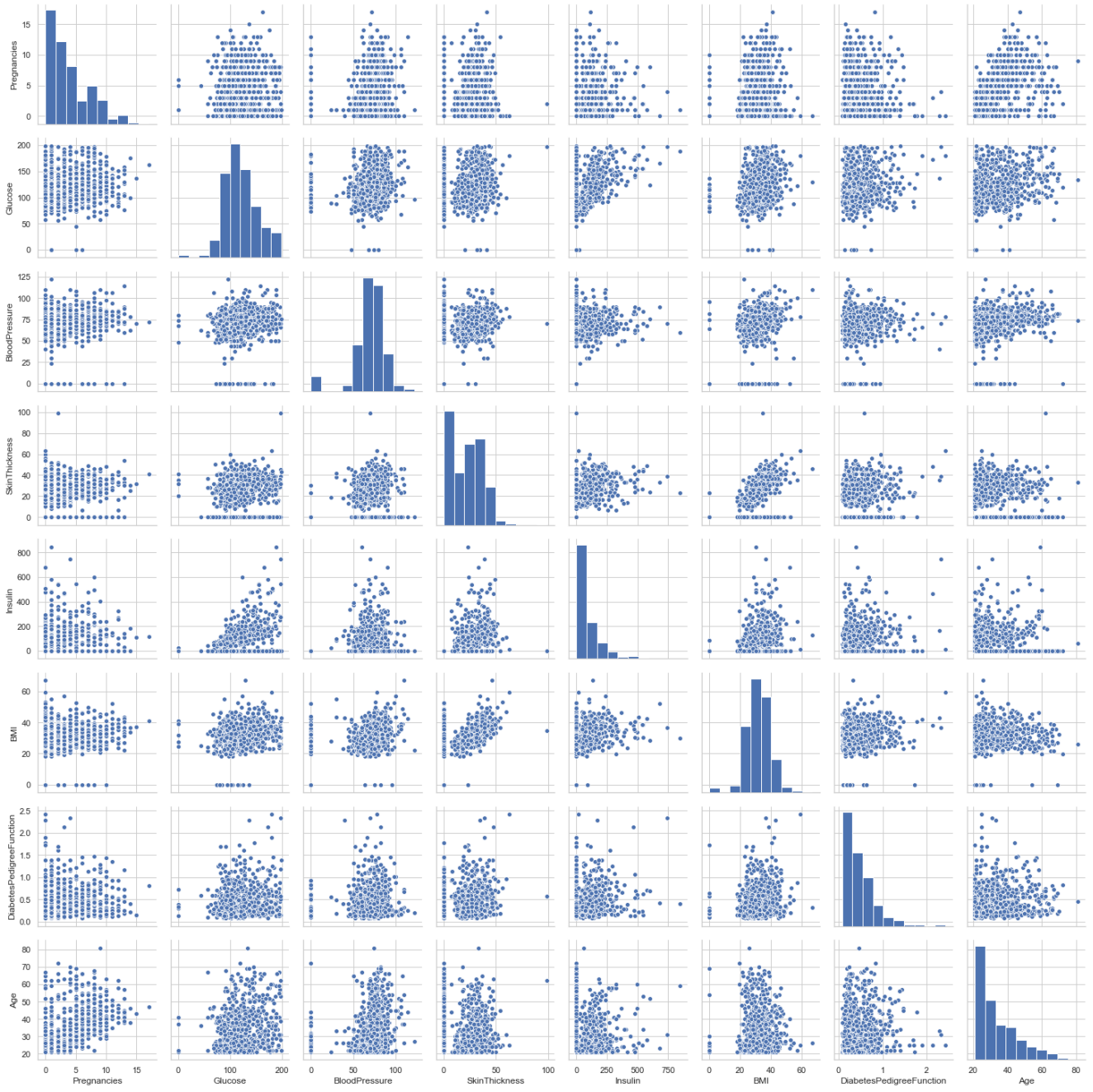
In [266]:
```
X=dataset.mask(dataset==0).fillna(dataset.mean())
X
print("number of rows missing glucose_conc: {0}".format(len(X.loc[X['Glucose'] == 0])))
print("number of rows missing diastolic_bp: {0}".format(len(X.loc[X['BloodPressure'] ==
print("number of rows missing insulin: {0}".format(len(df.loc[X['Insulin'] == 0])))
print("number of rows missing bmi: {0}".format(len(X.loc[X['BMI'] == 0])))
print("number of rows missing diab_pred: {0}".format(len(X.loc[X['DiabetesPedigreeFunct
print("number of rows missing age: {0}".format(len(X.loc[X['Age'] == 0])))
print("number of rows missing skin: {0}".format(len(X.loc[X['SkinThickness'] == 0])))
```

```
number of rows missing glucose_conc: 0
number of rows missing diastolic_bp: 0
number of rows missing insulin: 0
number of rows missing bmi: 0
number of rows missing diab_pred: 0
number of rows missing age: 0
number of rows missing skin: 0
```

In [267]:
```
X= dataset.iloc[:,[0,1,2,3,4,5,6,7]]
```

In [268]:
```
sns.pairplot(X)
```

Out[268]:
```
<seaborn.axisgrid.PairGrid at 0x1e6d8854400>
```

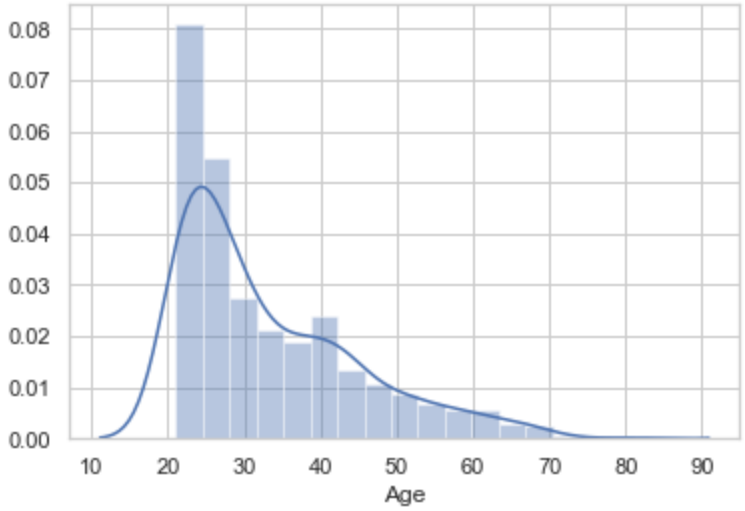


In [ ]:
```

```

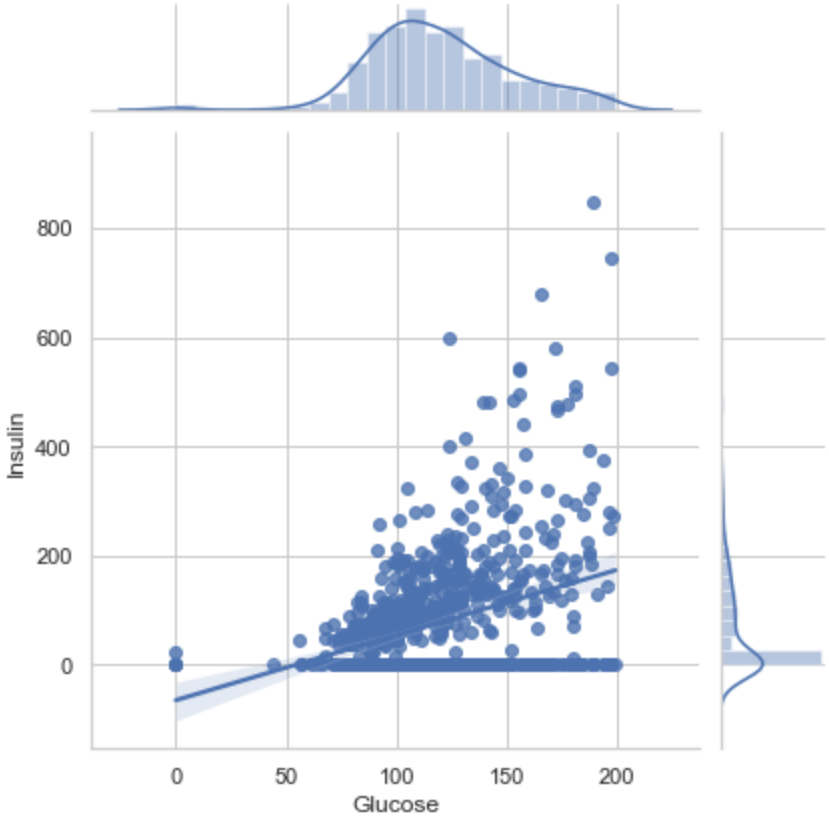
In [269]:

```
import seaborn as sns
sns.set(style="whitegrid")
ax = sns.distplot(X['Age'])
#Through the histogram we can see the distribution of the data taken,
#As we can see that data has more young people than the old people.
```



In [270]:
```
sns.jointplot(x='Glucose',y='Insulin',data=X,kind='reg')
```

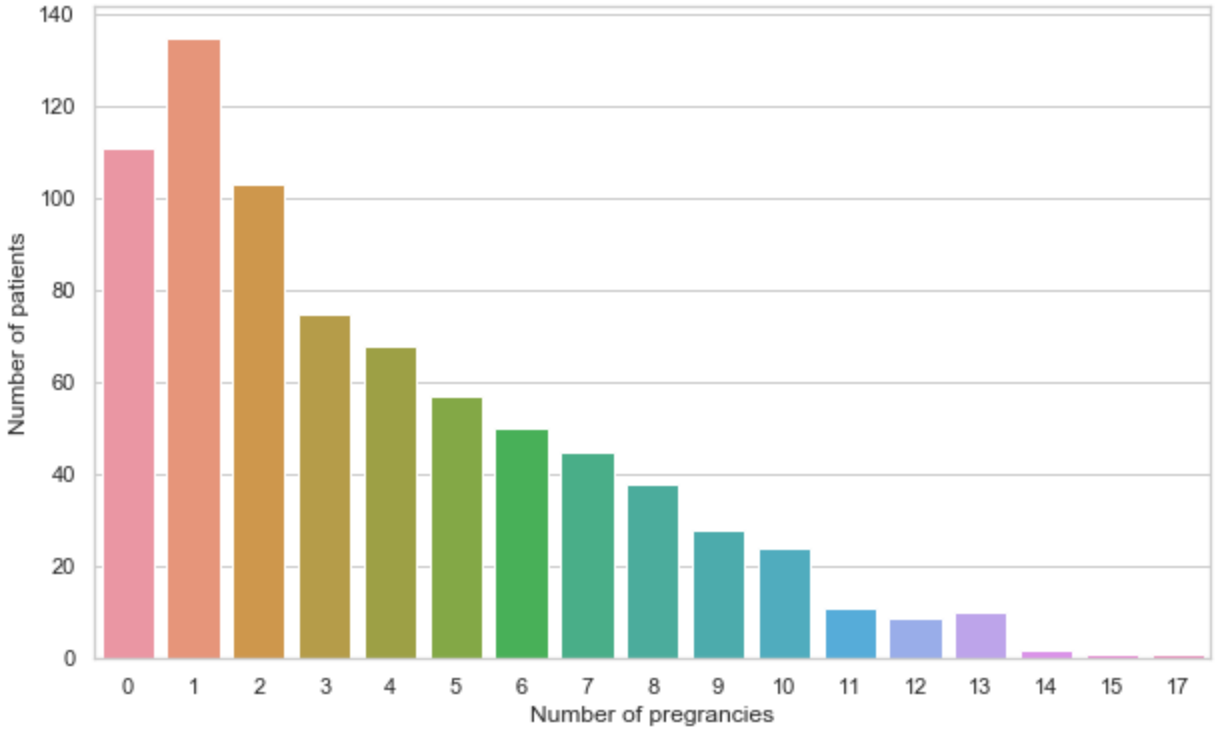
Out[270]:
```
<seaborn.axisgrid.JointGrid at 0x1e6d4c1e470>
```



In [271]:
```
Y
```

Out[271]:
```
array([[ 0.627, 50.   ],
       [ 0.351, 31.   ],
       [ 0.672, 32.   ],
       ...,
       [ 0.245, 30.   ],
       [ 0.349, 47.   ],
       [ 0.315, 23.   ]])
```

In [272]:
```
plt.figure(figsize=(10,6))
sns.countplot(x="Pregnancies", data=X)
plt.xlabel("Number of pregrancies")
plt.ylabel("Number of patients")
plt.show()
```



In [273]:
```
dataset.head()
```

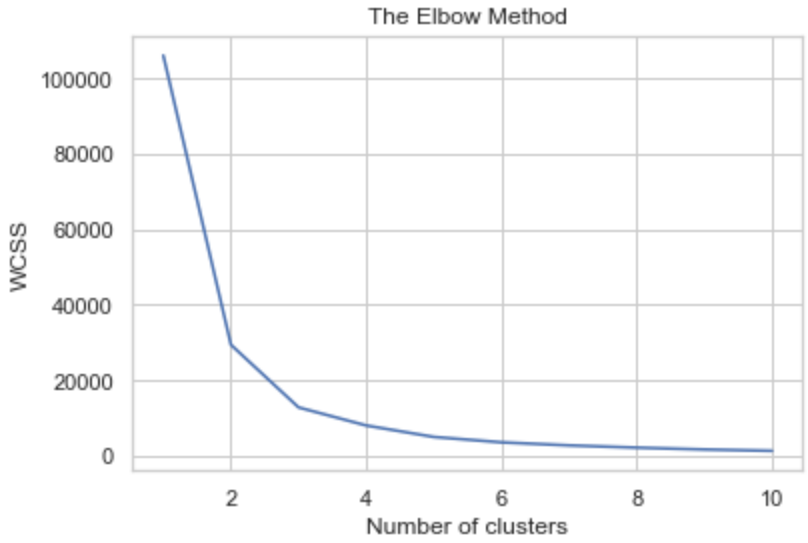
Out[273]:

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

In [307]:
```
Y=X.mask(dataset==0).fillna(dataset.mean())
Y=X.iloc[:,[6,7]].values
Y
```

Out[307]:
```
array([[ 0.627, 50.   ],
       [ 0.351, 31.   ],
       [ 0.672, 32.   ],
       ...,
       [ 0.245, 30.   ],
       [ 0.349, 47.   ],
       [ 0.315, 23.   ]])
```

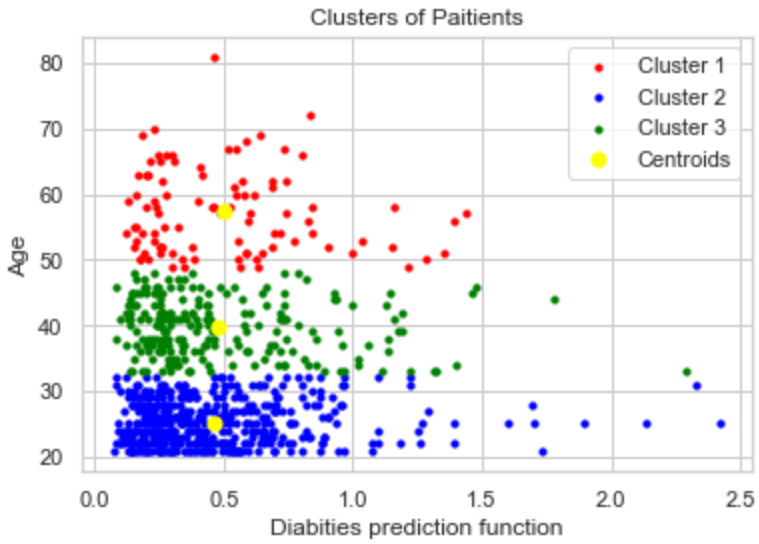
In [308]:
```
# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter=300, n_init=10)
    kmeans.fit(Y)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [309]:
```
# Fitting K-Means to the dataset
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter=300, n_init=10)
y_kmeans = kmeans.fit_predict(Y)
print(kmeans.cluster_centers_)
```

```
[[ 0.5047234  57.32978723]
 [ 0.46015098 25.19474836]
 [ 0.48234101 39.75115207]]
```

In [310]:
```
# Visualising the clusters (only for 2D clustering)
plt.scatter(Y[y_kmeans == 0, 0], Y[y_kmeans == 0, 1], s = 10, c = 'red', label = 'Clust
plt.scatter(Y[y_kmeans == 1, 0], Y[y_kmeans == 1, 1], s = 10, c = 'blue', label = 'Clus
plt.scatter(Y[y_kmeans == 2, 0], Y[y_kmeans == 2, 1], s = 10, c = 'green', label = 'Clu
#healthy BMI 18.5-24.5
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 50, c = '
plt.title('Clusters of Paitients')
plt.xlabel('Diabities prediction function')
plt.ylabel('Age')
plt.legend()
plt.show()
```



ASSESMENT

In [311]:
```
#lower db score is preferred
from sklearn.metrics import davies_bouldin_score as db
db(Y,y_kmeans)
```

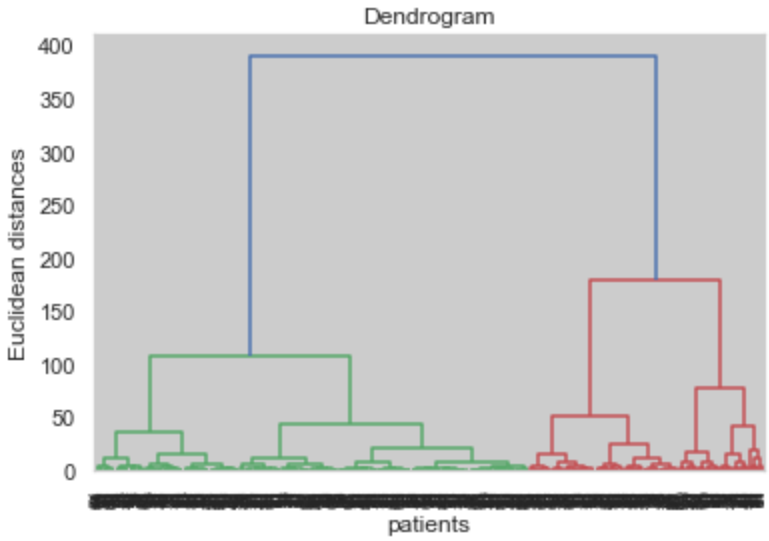
Out[311]: 0.489263054058259

In [312]:
```
from sklearn.metrics import silhouette_score as ss
ss(Y,y_kmeans)
```

Out[312]: 0.6476126010968893

Heirarchiacal Clustering

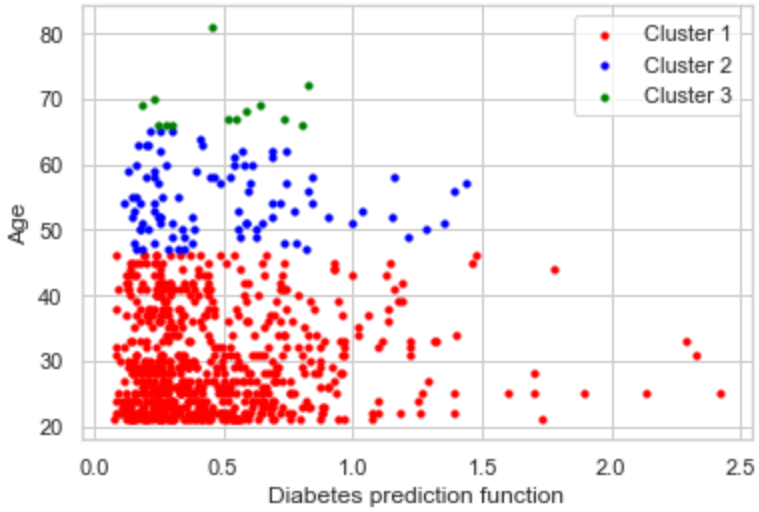
In [313]:
```
# Using the dendrogram to find the optimal number of clusters
import scipy.cluster.hierarchy as sch
dendrogram = sch.dendrogram(sch.linkage(Y, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('patients')
plt.ylabel('Euclidean distances')
plt.show()
```



complete linkage

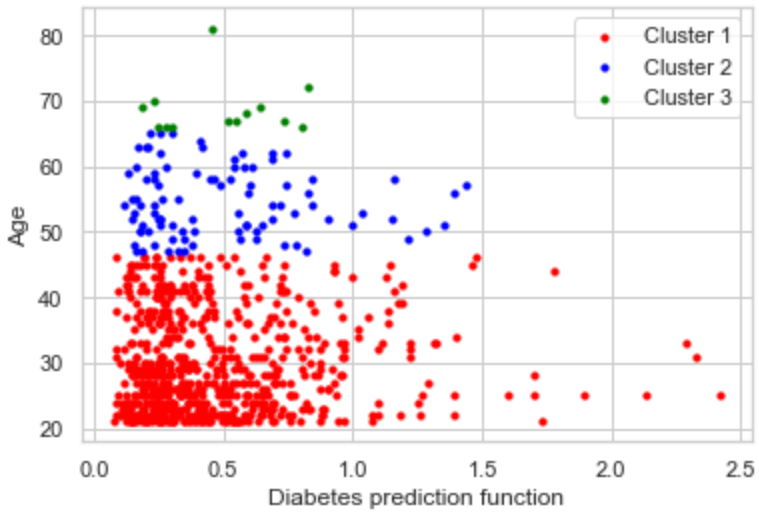
In [314]:
```
# Fitting Hierarchical Clustering to the dataset
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 3, affinity = 'euclidean', linkage = 'complet
y_hc = hc.fit_predict(Y)
```

In [318]:
```
# Visualising the clusters
plt.scatter(Y[y_hc == 0, 0], Y[y_hc == 0, 1], s = 10, c = 'red', label = 'Cluster 1')
plt.scatter(Y[y_hc == 1, 0], Y[y_hc == 1, 1], s = 10, c = 'blue', label = 'Cluster 2')
plt.scatter(Y[y_hc == 2, 0], Y[y_hc == 2, 1], s = 10, c = 'green', label = 'Cluster 3')
plt.xlabel('Diabetes prediction function')
plt.ylabel('Age')
plt.legend()
plt.show()
```



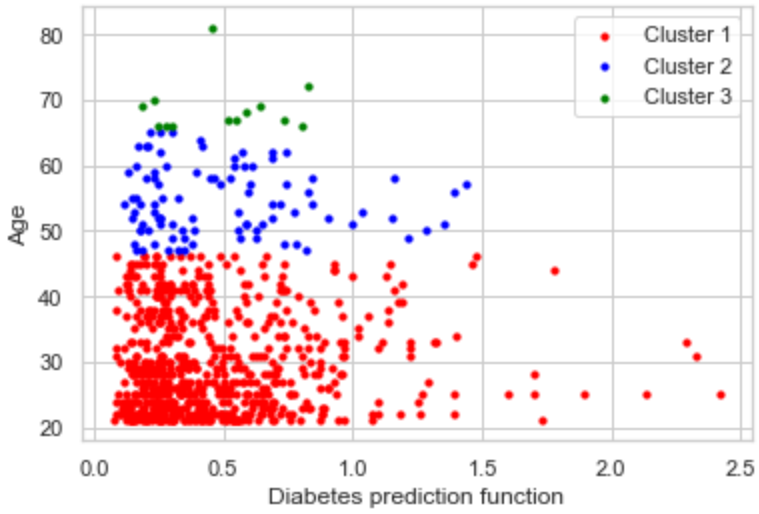
Single linkeage

In [319]:
```
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters=3, affinity = 'euclidean', linkage = 'single')
#dendrogram = sch.dendrogram(hc)
hc.fit_predict(Y)
# Visualising the clusters
plt.scatter(Y[y_hc == 0, 0], Y[y_hc == 0, 1], s = 10, c = 'red', label = 'Cluster 1')
plt.scatter(Y[y_hc == 1, 0], Y[y_hc == 1, 1], s = 10, c = 'blue', label = 'Cluster 2')
plt.scatter(Y[y_hc == 2, 0], Y[y_hc == 2, 1], s = 10, c = 'green', label = 'Cluster 3')
plt.xlabel('Diabetes prediction function')
plt.ylabel('Age')
plt.legend()
plt.show()
```



Average linkage

In [320]:
```
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters=3, affinity = 'euclidean', linkage = 'average')
#dendrogram = sch.dendrogram(hc)
hc.fit_predict(Y)
# Visualising the clusters
plt.scatter(Y[y_hc == 0, 0], Y[y_hc == 0, 1], s = 10, c = 'red', label = 'Cluster 1')
plt.scatter(Y[y_hc == 1, 0], Y[y_hc == 1, 1], s = 10, c = 'blue', label = 'Cluster 2')
plt.scatter(Y[y_hc == 2, 0], Y[y_hc == 2, 1], s = 10, c = 'green', label = 'Cluster 3')
plt.xlabel('Diabetes prediction function')
plt.ylabel('Age')
plt.legend()
plt.show()
```



In [ ]:

In [ ]:

In [ ]: