In [1]:
```python
# import the Libraries

import numpy as np        # used for multidimensional array
import pandas as pd       # used for import the dataset
import seaborn as sns
import matplotlib.pyplot as plt   # used for plotting the Graph

%matplotlib inline
```

In [48]:
```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, precision_recall_curve, auc, roc_curve
from sklearn.tree import DecisionTreeClassifier, export_graphviz

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics.pairwise import cosine_similarity
```

In [49]:
```python
#----Data Handelling-----
```

In [50]:
```python
# import the dataset
dataset= pd.read_csv('diabetes.csv')
```

In [51]:
```python
#Data understanding - data types
dataset.dtypes
```

Out[51]:
```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

In [52]:
```python
#print the attributes
dataset.keys()
```

Out[52]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [53]:
```python
#data shape
dataset.shape
```

Out[53]:
```
(768, 9)
```

In [54]:
```python
#Data understanding - first few rows of data in dataframe
dataset.head()
```

Out[54]:

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

In [55]:
```python
#Multiple 0 so we change them with Nan
```

In [56]:
```python
# Replacing zero values with NaN
dataset[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]] = dataset[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]].replac
```

In [57]:
```python
# Count of NaN
dataset.isnull().sum()
```

Out[57]:
```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

In [58]:
```python
# Replacing NaN with mean values
dataset["Glucose"].fillna(dataset["Glucose"].mean(), inplace = True)
dataset["BloodPressure"].fillna(dataset["BloodPressure"].mean(), inplace = True)
dataset["SkinThickness"].fillna(dataset["SkinThickness"].mean(), inplace = True)
dataset["Insulin"].fillna(dataset["Insulin"].mean(), inplace = True)
dataset["BMI"].fillna(dataset["BMI"].mean(), inplace = True)
```

In [59]:
```python
# Feature scaling using MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
dataset_scaled = sc.fit_transform(dataset)
```

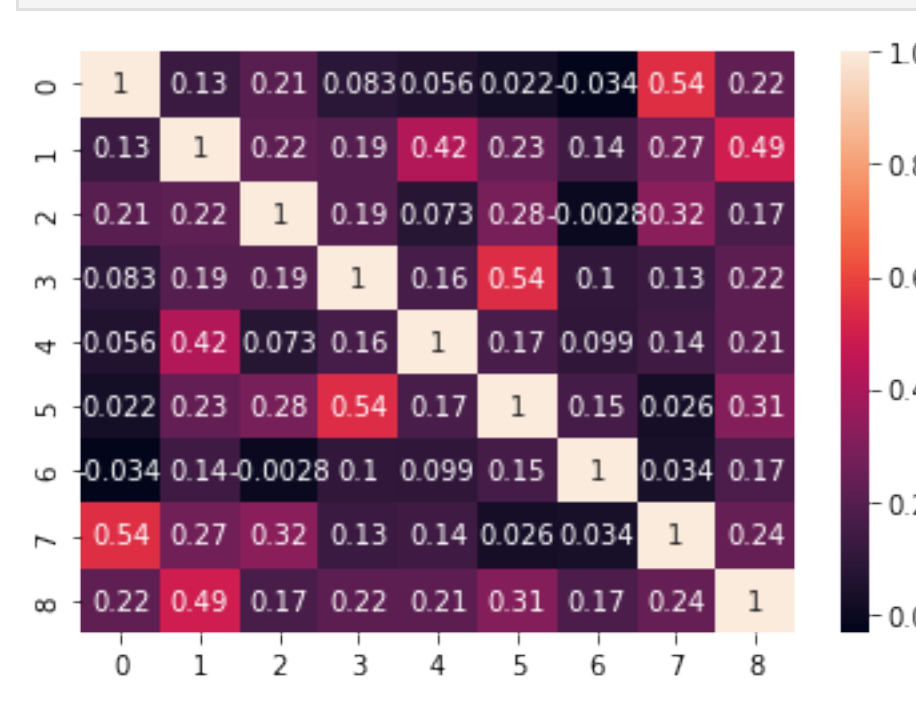
In [60]:
```python
dataset1 = pd.DataFrame(dataset_scaled)
dataset1
```

Out[60]:

	0	1	2	3	4	5	6	7	8
0	0.352941	0.670968	0.489796	0.304348	0.170130	0.314928	0.234415	0.483333	1.0
1	0.058824	0.264516	0.428571	0.239130	0.170130	0.171779	0.116567	0.166667	0.0
2	0.470588	0.896774	0.408163	0.240798	0.170130	0.104294	0.253629	0.183333	1.0
3	0.058824	0.290323	0.428571	0.173913	0.096154	0.202454	0.038002	0.000000	0.0
4	0.000000	0.600000	0.163265	0.304348	0.185096	0.509202	0.943638	0.200000	1.0
...	...	...	...	...	...	...	...	...	...
763	0.588235	0.367742	0.530612	0.445652	0.199519	0.300613	0.039710	0.700000	0.0
764	0.117647	0.503226	0.469388	0.217391	0.170130	0.380368	0.111870	0.100000	0.0
765	0.294118	0.496774	0.489796	0.173913	0.117788	0.163599	0.071307	0.150000	0.0
766	0.058824	0.529032	0.367347	0.240798	0.170130	0.243354	0.115713	0.433333	1.0
767	0.058824	0.316129	0.469388	0.260870	0.170130	0.249489	0.101196	0.033333	0.0

768 rows × 9 columns

In [61]:
```python
# Heatmap
sns.heatmap(dataset1.corr(), annot = True)
plt.show()
```



In [62]:
```python
#corr matrix shows the dataset columns that have the highest correlation to outcome = 8
#1 (Glucose), 5 (Insulin), 7 (Age)
```

In [63]:
```python
# Selecting features - [Glucose, Insulin, BMI]
X = dataset1.iloc[:, [1, 5, 7]].values
Y = dataset1.iloc[:, 8].values
```

In [64]:
```python
# Splitting X and Y
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, random_state = 42, stratify = dataset1[8] )
```

In [65]:
```python
# Checking dimensions
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("Y_train shape:", Y_train.shape)
print("Y_test shape:", Y_test.shape)
```

```
X_train shape: (614, 3)
X_test shape: (154, 3)
Y_train shape: (614,)
Y_test shape: (154,)
```

In [66]:
```python
import sklearn
from sklearn.cluster import KMeans
KMeans_Clustering = KMeans(n_clusters =2, random_state=0)
KMeans_Clustering.fit(X_train)
```

Out[66]:
```
KMeans(n_clusters=2, random_state=0)
```

In [67]:
```python
print(KMeans_Clustering.cluster_centers_)
```

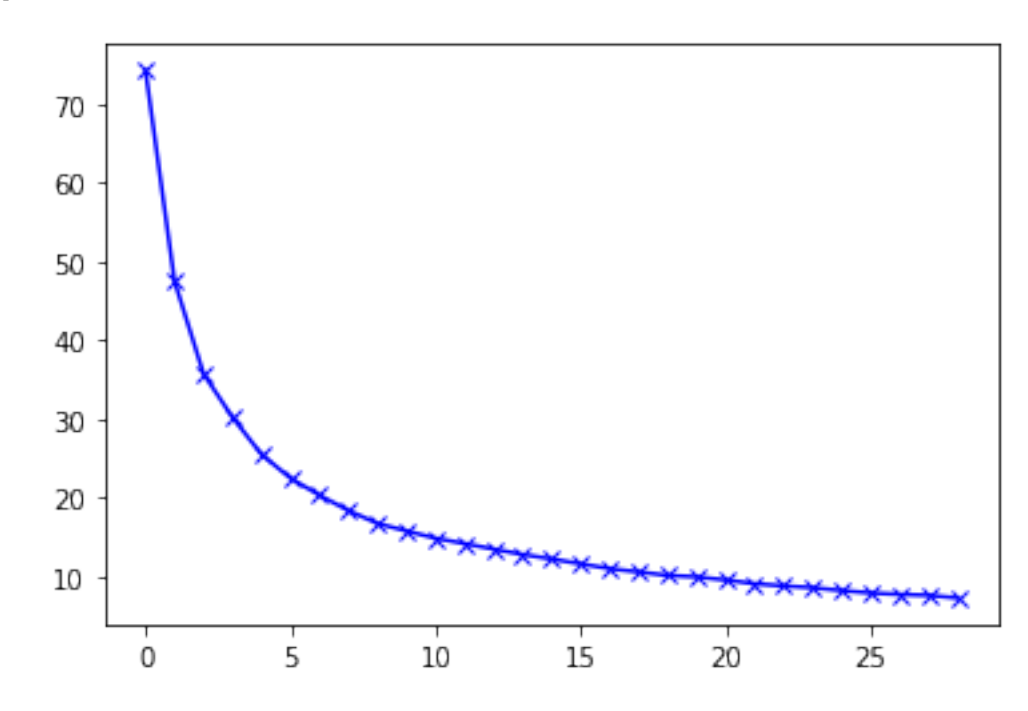
```
[[0.67796063 0.32827361 0.38242009]
 [0.40335441 0.27094314 0.10835443]]
```

In [88]:
```python
#----modelling k means cluster-------
```

In [89]:
```python
n_clusters=30
cost=[]
for i in range(1,n_clusters):
    kmean= KMeans(i)
    kmean.fit(X)
    cost.append(kmean.inertia_)
```

In [90]:
```python
plt.plot(cost, 'bx-')
```

Out[90]:
```
[<matplotlib.lines.Line2D at 0x7ff253b76d90>]
```



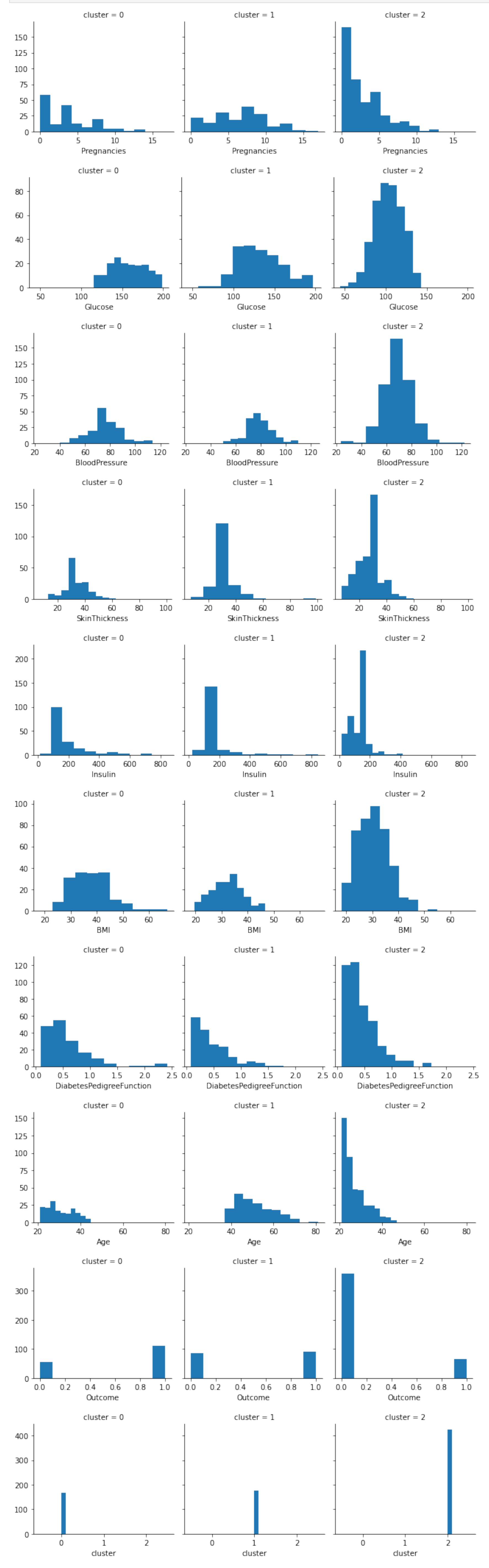
In [91]:
```python
#chosing 6 clusters

kmean= KMeans(3)
kmean.fit(X)
labels=kmean.labels_
```

In [92]:
```python
clusters=pd.concat([dataset, pd.DataFrame({'cluster':labels})], axis=1)
clusters.head()
```

Out[92]:

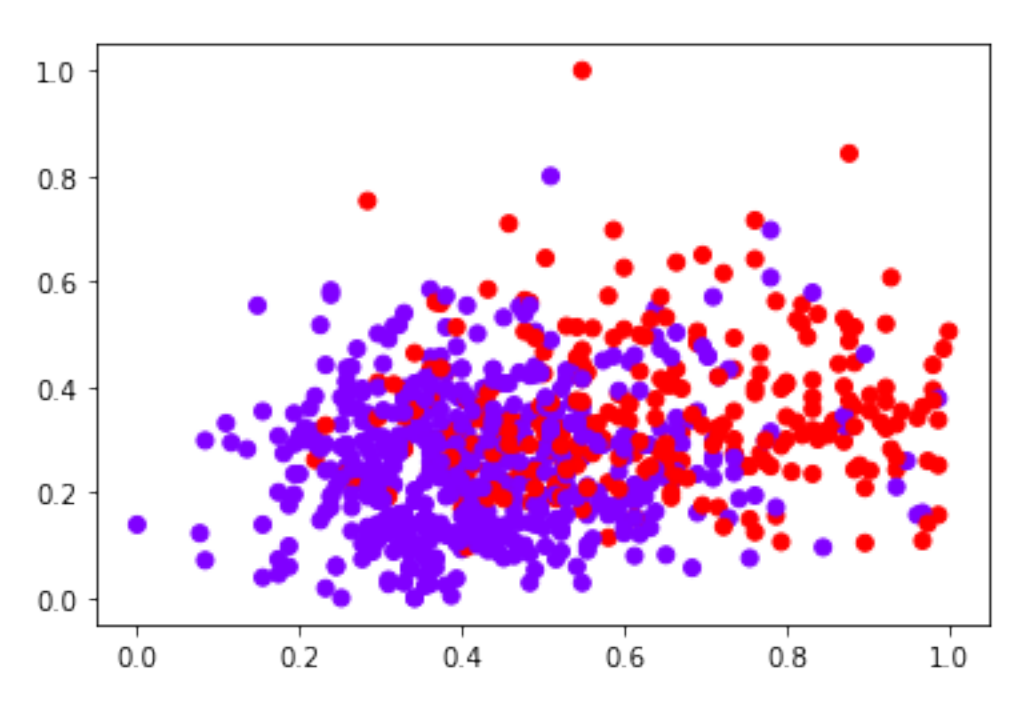
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	cluster
0	6	148.0	72.0	35.00000	155.548223	33.6	0.627	50	1	1
1	1	85.0	66.0	29.00000	155.548223	26.6	0.351	31	0	2
2	8	183.0	64.0	29.15342	155.548223	23.3	0.672	32	1	0
3	1	89.0	66.0	23.00000	94.000000	28.1	0.167	21	0	2
4	0	137.0	40.0	35.00000	168.000000	43.1	2.288	33	1	0

In [93]:
```python
#Interpretation of Clusters
for c in clusters:
    grid= sns.FacetGrid(clusters, col='cluster')
    grid.map(plt.hist, c)
```



In [68]:
```python
plt.scatter(dataset1.iloc[:, [1]].values,dataset1.iloc[:, [5]].values, c=dataset1[8], cmap='rainbow')
```

Out[68]:
```
<matplotlib.collections.PathCollection at 0x7ff253ab6f70>
```



In [69]:
```python
dataset['Outcome'].value_counts()
```

Out[69]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [81]:
```python
#0 --> Non-Diabetic

#1 --> Diabetic

from scipy.cluster.hierarchy import dendrogram, linkage
```

In [82]:
```python
dataset.groupby('Outcome').mean()
```

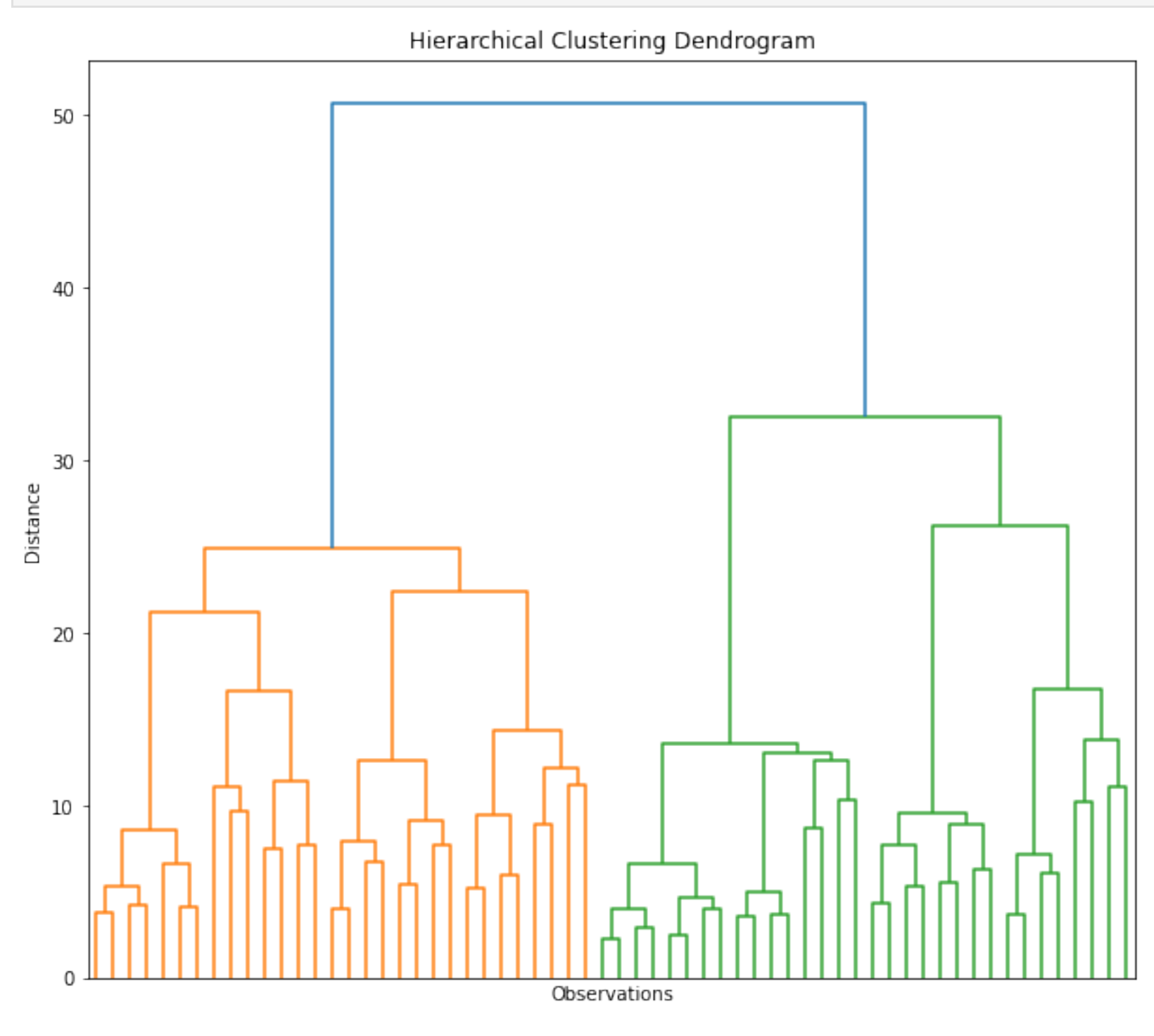
Out[82]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
Outcome								
0	3.298000	110.710121	70.935397	27.768651	142.210761	30.888434	0.429734	31.190000
1	4.865672	142.165573	75.147324	31.736944	180.431548	35.384757	0.550500	37.067164

In [83]:
```python
scaler = StandardScaler()
data_scaled = scaler.fit_transform(dataset)
```

In [84]:
```python
hier_cluster = linkage(data_scaled, method='ward')
```

In [85]:
```python
plt.figure(figsize=(10,9))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Observations')
plt.ylabel('Distance')
dendrogram(hier_cluster, truncate_mode='level', p = 5, show_leaf_counts=False, no_labels=True)
plt.show()
```



In [ ]:
```python
#number of optimal clusters is 2
```