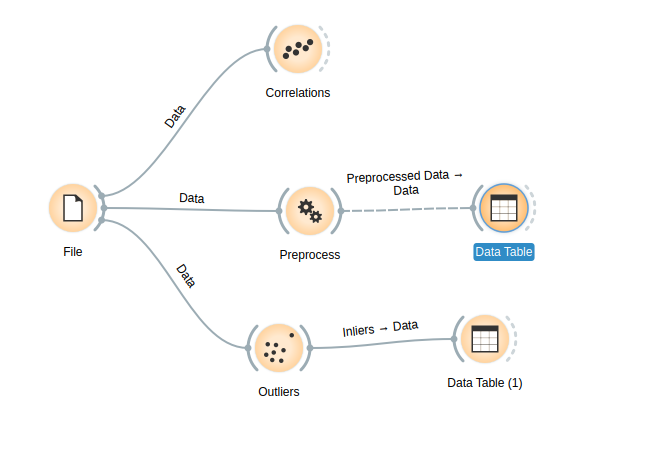
EXPERIMENT -5

**Aim:** Given a case study of Diabetes with data set. You are expected to perform data preprocessing using Orange Tool & Python/R/Java.

Quote your observations after the preprocessing.

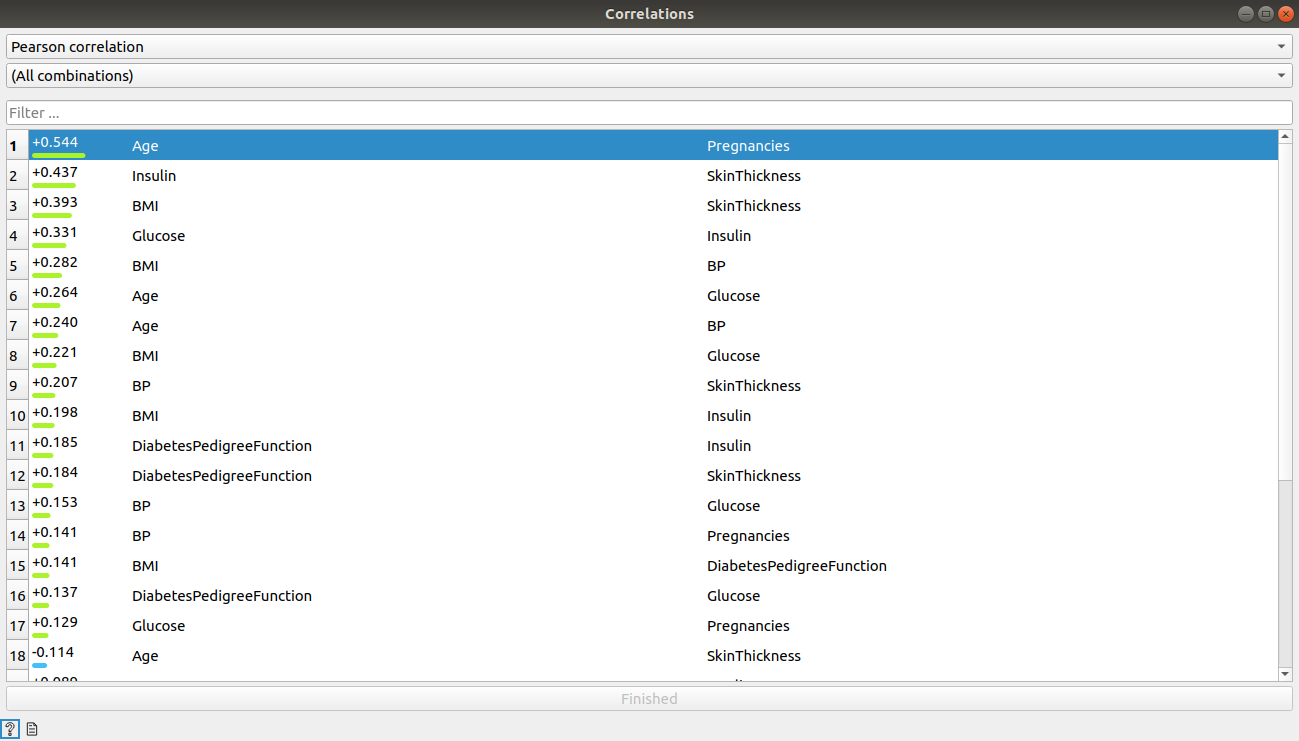
**Observations:**

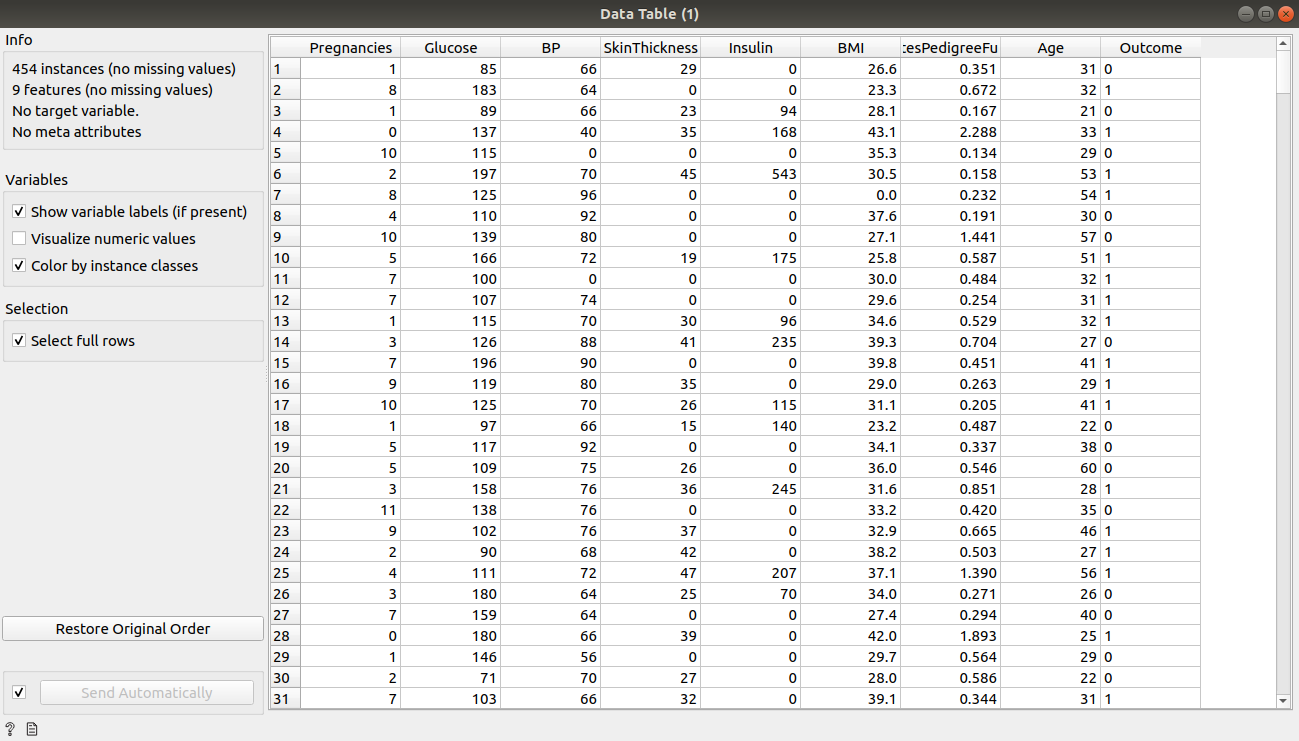
**Using Orange Tool O/p:**



**Using**

We use orange to see the correlations, preprocess the data using pca retaining only 4 components. We also detect and remove outliers

Correlation between the various features



Data after outliers being removed

**Using Python:**

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

file = '/home/lavina/Desktop/pima-indians-diabetes-database/diabetes.csv'

# load data set into frame

dataFrame = pd.read\_csv(file, names=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'])

# 2 groups

group0 = dataFrame[dataFrame['Outcome'] == 0]

group1 = dataFrame[dataFrame['Outcome'] == 1]

plt.boxplot(group0['Pregnancies'])

plt.xlabel('pregnanices for outcome 0')

plt.show()

# # 0utlier >=12 for pregnancies

plt.boxplot(group0['Glucose'])

plt.xlabel('glucose for outcome 0')

plt.show()

# # glucose <50 && >175

plt.boxplot(group0['BloodPressure'])

plt.xlabel('bp for outcome 0')

plt.show()

# # bp <40 && >100

plt.boxplot(group0['SkinThickness'])

plt.xlabel('skinthickness for outcome 0')

plt.show()

plt.boxplot(group0['Insulin'])

plt.xlabel('insulin for outcome 0')

plt.show()

# # insulin >250

plt.boxplot(group0['BMI'])

plt.xlabel('bmi for outcome 0')

plt.show()

# # bmi <10 && >50

plt.boxplot(group0['DiabetesPedigreeFunction'])

plt.xlabel('dpf for outcome 0')

plt.show()

# # pedigree fun >1

plt.boxplot(group0['Age'])

plt.xlabel('age for outcome 0')

plt.show()

# # age >57

for index, row in group0.iterrows():

if row['Pregnancies'] >= 12:

row['Pregnancies'] = np.median(group0['Pregnancies'])

if row['Glucose'] <= 12 or row['Glucose'] > 175:

row['Glucose'] = np.median(group0['Glucose'])

if row['BloodPressure'] <= 40 or row['BloodPressure'] > 100:

row['BloodPressure'] = np.median(group0['BloodPressure'])

if row['Insulin'] > 250:

row['Insulin'] = np.median(group0['Insulin'])

if row['BMI'] <= 10 or row['BMI'] > 50:

row['BMI'] = np.median(group0['BMI'])

if row['DiabetesPedigreeFunction'] > 1:

row['DiabetesPedigreeFunction'] = np.median(group0['DiabetesPedigreeFunction'])

if row['Age'] > 57:

row['Age'] = np.median(group0['Age'])

print(group0)

plt.boxplot(group1['Pregnancies'])

plt.xlabel('pregnanices for outcome 1')

plt.show()

plt.boxplot(group1['Glucose'])

plt.xlabel('glucose for outcome 1')

plt.show()

# # glucose <50

# plt.boxplot(group1['BloodPressure'])

plt.xlabel('bp for outcome 1')

plt.show()

# # bp <40 && >100

plt.boxplot(group1['SkinThickness'])

plt.xlabel('skinthickness for outcome 1')

plt.show()

# # skinThickness >75

plt.boxplot(group1['Insulin'])

plt.xlabel('insulin for outcome 1')

plt.show()

# # insulin >400

plt.boxplot(group1['BMI'])

plt.xlabel('bmi for outcome 1')

plt.show()

# # bmi <20 && >50

plt.boxplot(group1['DiabetesPedigreeFunction'])

plt.xlabel('dpf for outcome 1')

plt.show()

# # pedigree fun >1.5

plt.boxplot(group1['Age'])

plt.xlabel('age for outcome 1')

plt.show()

# # age >60

for index, row in group1.iterrows():

if row['Glucose'] <= 50:

row['Glucose'] = np.median(group1['Glucose'])

if row['BloodPressure'] <= 40 or row['BloodPressure'] > 100:

row['BloodPressure'] = np.median(group1['BloodPressure'])

if row['SkinThickness'] > 75:

row['SkinThickness'] = np.median(group1['SkinThickness'])

if row['Insulin'] > 400:

row['Insulin'] = np.median(group1['Insulin'])

if row['BMI'] <= 20 or row['BMI'] > 50:

row['BMI'] = np.median(group1['BMI'])

if row['DiabetesPedigreeFunction'] > 1.5:

row['DiabetesPedigreeFunction'] = np.median(group1['DiabetesPedigreeFunction'])

if row['Age'] > 60:

row['Age'] = np.median(group1['Age'])

print(group1)

dataFrame = group0.append(group1)

print(dataFrame)

#normalizing data

features = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']

# Separating out the features

x = dataFrame.loc[:, features].values

xplot =x

# Separating out the target

y = dataFrame.loc[:, ['Outcome']].values

# Standardizing the features mean=0 variance =1

x = StandardScaler().fit\_transform(x)

pca = PCA(n\_components=8)

Components = pca.fit\_transform(x)

plt.plot(np.cumsum(pca.explained\_variance\_ratio\_))

plt.xlabel('no of components')

plt.ylabel('cum explained variance')

plt.show()

# so 4 components show 80% of the data

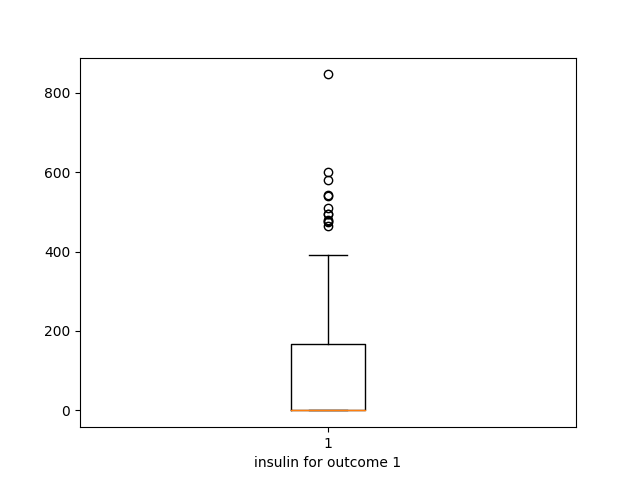
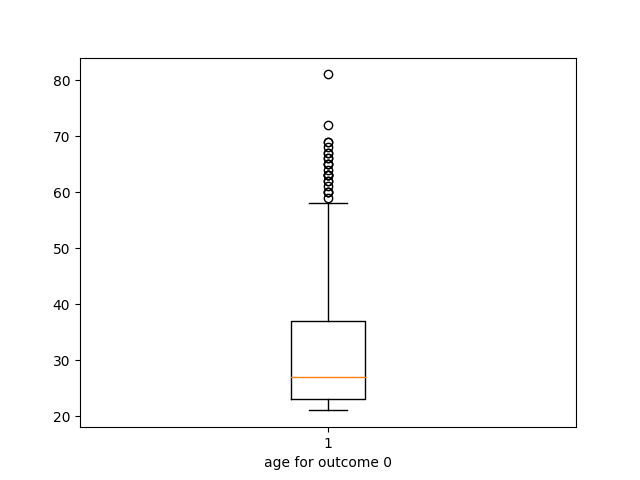
pca1 = PCA(n\_components=4)

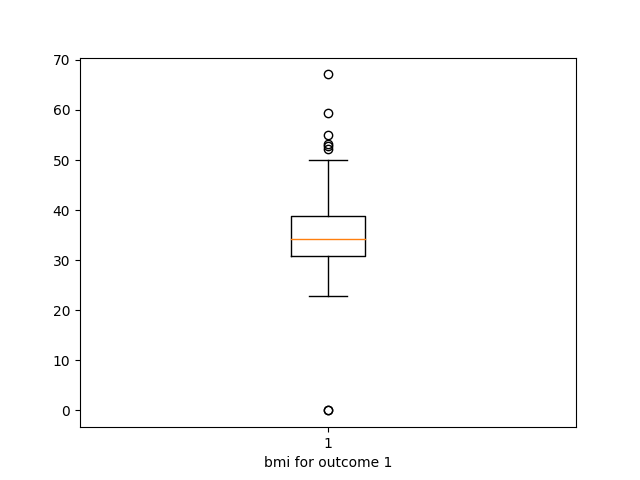
principalComponents = pca1.fit\_transform(x)

principalDf = pd.DataFrame(data=principalComponents, columns=['pc1', 'pc2', 'pc3', 'pc4'])

finalDf = pd.concat([principalDf, dataFrame[['Outcome']]], axis=1)

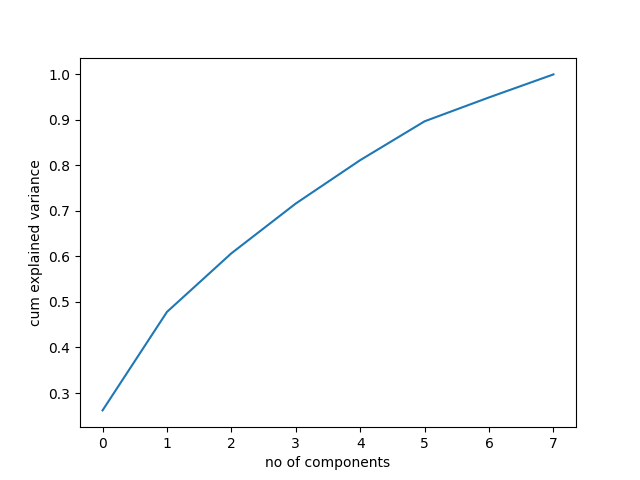
print(finalDf)



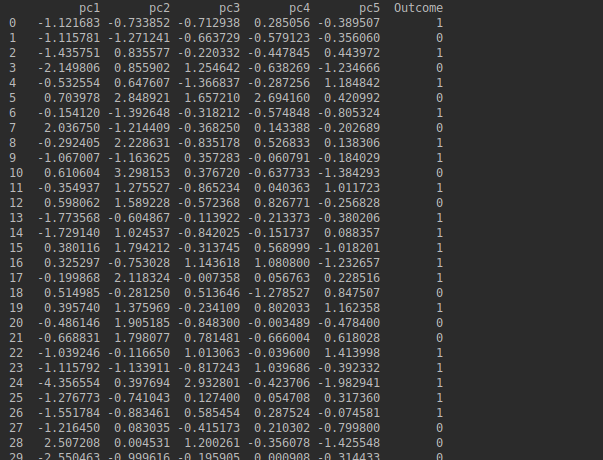


Detecting outliers group label wise . The outliers are replaced by the median of those specific groups.

Performing PCA on the data after outliers being removed. We see that if we use 5 components 90% of the data is represented.



Output after pca. We have retained 5 components.



**Conclusion**: Hence we have successfully preprocessed the data by removing outliers and performing pca.