In [5]:

```
## importing all requirde libraries ##

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings("ignore")
from sklearn.preprocessing import LabelEncoder
```

In [7]:

```
# installing the dataset
df = pd.read_csv(r"C:\Users\kr200\Downloads\diabetes.csv")
df.head()
```

Out[7]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunc |
|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2 |

In [8]:

```
# Dataset types
df.dtypes
```

Out[8]:

```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

In [10]:

```
print(np.unique( df['Outcome']))
```

```
[0 1]
```

In [11]:

```
df.columns
```

Out[11]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insul
in',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [12]:

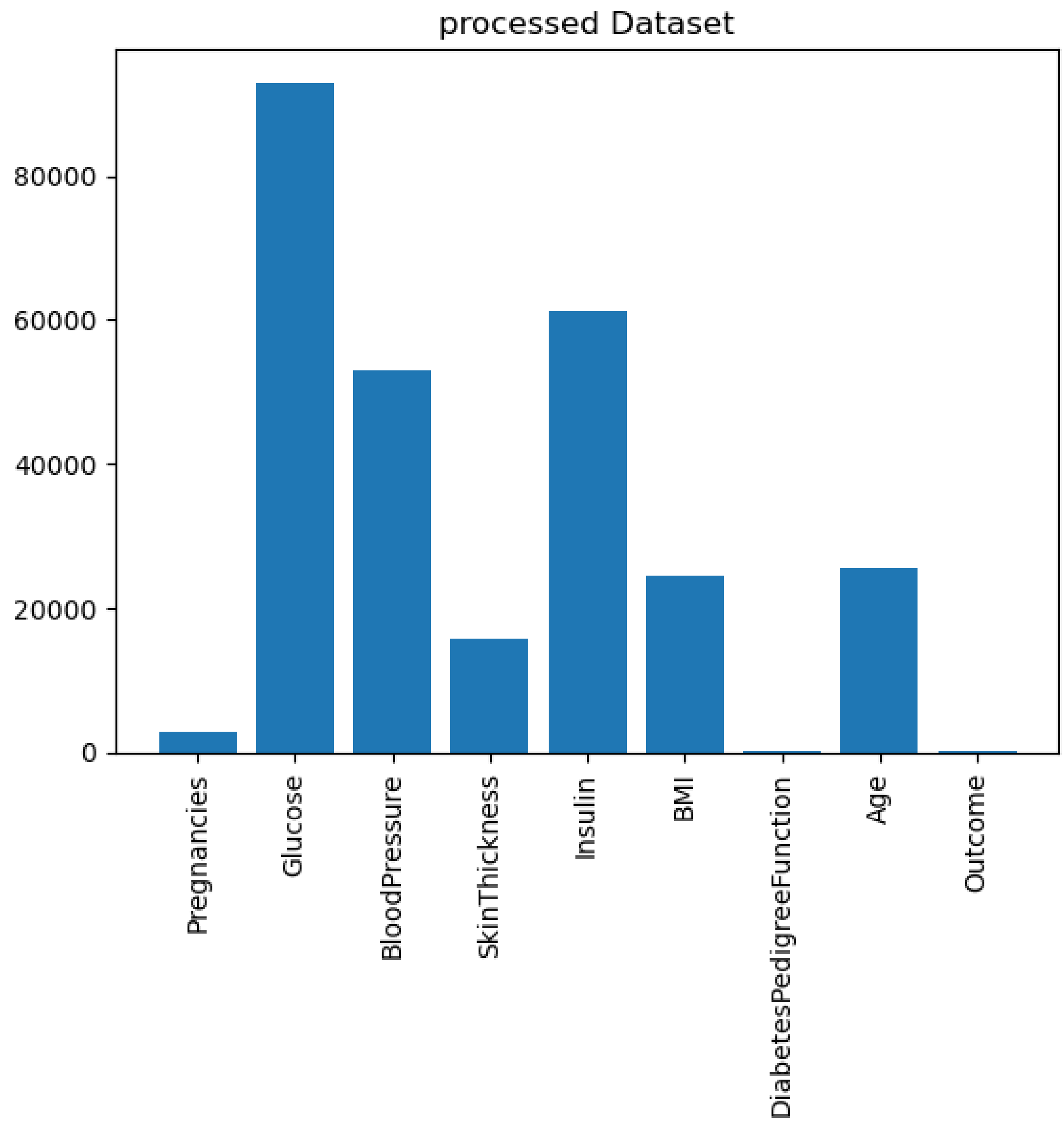
```
df.sum()
```

Out[12]:

```
Pregnancies                  2953.000
Glucose                     92847.000
BloodPressure               53073.000
SkinThickness               15772.000
Insulin                     61286.000
BMI                         24570.300
DiabetesPedigreeFunction      362.401
Age                         25529.000
Outcome                       268.000
dtype: float64
```

In [13]:

```
# Lord Iros dataset
categories = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BM
scores = [2953.000, 92847.000,53073.000 , 15772.000, 61286.000, 24570.300, 362.401, 2552

# Creating a numpy array of indices for the categories
x = np.arange(len(categories))

# Plotting the bar graph
plt.bar(x, scores)

# Adding category labels on the x-axis
plt.xticks(x, categories, rotation='vertical')

# Adding a title
plt.title('processed Dataset')

# Displaying the graph
plt.show()
```



In [17]:

```
df.columns
selected_df=df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age',]]

#independent var(Data)
X=np.asarray(selected_df)

#dependent var(target)
y=np.asarray(df['Outcome'])

print(X.shape)
print(y.shape)
```

```
(768, 8)
(768,)
```

In [18]:

```
# Training the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
X_train.shape
X_test.shape
y_train.shape
y_test.shape
```

Out[18]:

```
(154,)
```

In [19]:

```
# importing all required libraries and modules
from sklearn.linear_model import Perceptron
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB


# Giving Parameters to the fumctions
clf1=KNeighborsClassifier(n_neighbors=5,metric="minkowski")
clf2=SVC(C=1.0,kernel="rbf")
clf3=GaussianNB(priors=None)
clf4=Perceptron(alpha=0,l1_ratio=0.15,max_iter=100)
clf5=DecisionTreeClassifier(criterion="gini",splitter="best", max_depth=5)


clf=[clf1,clf2,clf3,clf4,clf5]
clf_name=["kneighbors","svc","gaussionNB","perceptron","decisiontree"]

from sklearn.metrics import accuracy_score
accuracy={}
import time
acc={}
t={}
for model,model_name in zip(clf,clf_name):
    st=time.time()
    model.fit(X_train,y_train)
    pred=model.predict(X_test)
    et=time.time()
    acc[model_name]=accuracy_score(y_test,pred)
    t[model_name]=et-st

for i,j in acc.items():
    print(i,":-",j)
```

```
kneighbors :- 0.6623376623376623
svc :- 0.7662337662337663
gaussionNB :- 0.7662337662337663
perceptron :- 0.5714285714285714
decisiontree :- 0.7922077922077922
```

In [15]:

```
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.mixture import GaussianMixture
from sklearn.cluster import Birch

clustering_algorithms = {
    'K-Means': KMeans(n_clusters=5),
    'Agglomerative': AgglomerativeClustering(n_clusters=5),
    'DBSCAN': DBSCAN(eps=0.5, min_samples=5),
    'GMM': GaussianMixture(n_components=5),
    'BIRCH': Birch(n_clusters=5)
}

data = df

for algorithm_name, algorithm in clustering_algorithms.items():
    labels = algorithm.fit_predict(data)
    print(f"Algorithm: {algorithm_name}")
    print("Cluster Labels:")
    print(labels)
    print("------")
```

```
Algorithm: K-Means
Cluster Labels:
[3 3 3 0 1 3 0 3 2 3 3 3 3 2 1 3 1 3 0 0 1 3 3 3 1 0 3 1 0 3 3 4 0 3 3
1 3
 3 3 1 0 3 3 4 3 3 3 3 3 3 0 3 3 4 4 3 4 0 3 1 3 3 3 1 3 3 3 3 0 0 0 1
3 4
 3 3 3 3 3 3 3 3 0 3 3 0 3 0 0 3 3 1 0 3 0 1 3 0 0 1 3 3 3 0 3 1 3 1 3
0 1
 2 0 3 1 3 3 3 3 0 0 3 0 3 3 0 1 0 1 3 1 3 1 3 0 1 0 0 3 4 3 3 0 3 4 3
3 0
 3 3 1 3 1 2 3 3 0 1 0 1 3 0 4 3 3 1 3 3 3 0 3 1 3 0 0 1 3 1 3 3 3 0 0
3 3
 3 2 0 0 1 3 0 3 3 3 1 3 0 0 4 3 3 3 0 1 3 4 3 0 3 3 3 3 1 1 4 1 0 3 3
2 3
 3 1 0 3 3 3 2 0 3 4 0 3 0 3 1 3 3 3 3 0 3 1 1 3 3 2 4 3 3 3 0 3 4 3 3
3 4
 1 1 3 3 3 3 0 3 3 3 3 3 0 3 0 3 0 3 0 3 4 3 1 1 3 3 1 2 1 0 0 0 0 1 1
3 1
 4 1 1 3 3 1 0 3 3 0 1 1 1 1 3 1 0 0 3 0 0 3 1 3 1 3 3 3 3 1 1 3 0 0 3
```

In [ ]:

In [ ]: