In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from IPython.display import display
from sklearn.preprocessing import LabelEncoder, MinMaxScaler, StandardScaler
import pickle
```

In [2]:
```
df = pd.read_csv('diabetes.csv')
print('NO OF ROWS AND COLUMN IN DATASET ',df.shape,'\n')
display(df.head(),"\n",df.dtypes)
```

```
NO OF ROWS AND COLUMN IN DATASET  (768, 9)
```

	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

```
'\n'

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

In [3]:
```
print('COUNT NO OF NULL IN EACH COLUMN','\n',df.isnull().sum())
```

```
COUNT NO OF NULL IN EACH COLUMN
 Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [4]:
```
df.describe().T
```

Out[4]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

In [5]:
```
data = df.copy(deep = True)
data[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = data[['Pregnancies','Glucose','BloodPressure',

## showing the count of Nans
print(data.isnull().sum())
```

```
Pregnancies                 111
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

In [6]:
```
data['Pregnancies'].fillna(data['Pregnancies'].mean(), inplace = True)

data['Glucose'].fillna(data['Glucose'].mean(), inplace = True)

data['BloodPressure'].fillna(data['BloodPressure'].mean(), inplace = True)

data['SkinThickness'].fillna(data['SkinThickness'].median(), inplace = True)

data['Insulin'].fillna(data['Insulin'].median(), inplace = True)

data['BMI'].fillna(data['BMI'].median(), inplace = True)
```

In [7]:
```
corr=data.corr()
corr.nlargest(12,'Outcome')['Outcome']
```

Out[7]:
```
Outcome                     1.000000
Glucose                     0.492928
BMI                         0.312038
Pregnancies                 0.248263
Age                         0.238356
SkinThickness               0.214873
Insulin                     0.203790
DiabetesPedigreeFunction    0.173844
BloodPressure               0.166074
Name: Outcome, dtype: float64
```

In [8]:
```
x = data.drop('Outcome', axis = 1)
y = data['Outcome']
x.head()
```

Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6.000000	148.0	72.0	35.0	125.0	33.6	0.627	50
1	1.000000	85.0	66.0	29.0	125.0	26.6	0.351	31
2	8.000000	183.0	64.0	29.0	125.0	23.3	0.672	32
3	1.000000	89.0	66.0	23.0	94.0	28.1	0.167	21
4	4.494673	137.0	40.0	35.0	168.0	43.1	2.288	33

Random Forest Using

In [9]:
```
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x,y,test_size=.2, random_state=1)

print('X TRAIN DATA ', xtrain.shape)
print('X TEST DATA ', xtest.shape)
print('Y TRAIN DATA ', ytrain.shape)
print('Y TEST DATA ', ytest.shape)
```

```
X TRAIN DATA  (614, 8)
X TEST DATA  (154, 8)
Y TRAIN DATA  (614,)
Y TEST DATA  (154,)
```

In [10]:
```
#from sklearn.preprocessing import StandardScaler

#sc = StandardScaler()
#x_train = sc.fit_transform(xtrain)
#x_test = sc.transform(xtest)
```

In [11]:
```
# Creating Random Forest Model
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=20)
classifier.fit(xtrain, ytrain)
```

Out[11]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [12]:
```
filename = open('diabetespredictmodel.pkl', 'wb')
pickle.dump(classifier,filename )
```

In [13]:
```
filename.close()
```

In [14]:
```
y_pred = classifier.predict(xtest)
```

In [15]:
```
from sklearn.metrics import confusion_matrix

print(confusion_matrix(ytest,y_pred))
```

```
[[89 10]
 [26 29]]
```

In [16]:
```
from sklearn.metrics import classification_report

print('CLASSIFICATION REPORT','\n',classification_report(ytest, y_pred))
```

```
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           0       0.77      0.90      0.83        99
           1       0.74      0.53      0.62        55

    accuracy                           0.77       154
   macro avg       0.76      0.71      0.72       154
weighted avg       0.76      0.77      0.76       154
```