In [4]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.rcParams["figure.figsize"] = (10, 20)
import mpld3
mpld3.enable_notebook()
```

In [5]:

```
df = pd.read_csv('diabetes.csv')
```

In [6]:

```
df.head()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

In [7]:

```
df.isnull().sum()
```

Out[7]:

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [9]:

```
df.shape
```

Out[9]:

```
(768, 9)
```

In [10]:

```
df['Pregnancies'].value_counts()
```

Out[10]:

```
1     135
0     111
2     103
3      75
4      68
5      57
6      50
7      45
8      38
9      28
10     24
11     11
13     10
12      9
14      2
15      1
17      1
Name: Pregnancies, dtype: int64
```

In [13]:

```
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

In [14]:

```
from sklearn.model_selection import train_test_split
```

In [15]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [16]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [17]:

```
clf = RandomForestClassifier(n_estimators = 10)
```

In [18]:

```
clf.fit(X_train, y_train)
```

Out[18]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=N
one,
                       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=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [27]:

```
clf.score(X_train, y_train)
```

Out[27]:

```
0.9869706840390879
```

In [28]:

```
predict = clf.predict(X_test)
```

In [29]:

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

In [30]:

```
print(classification_report(predict, y_test))
```

```
              precision    recall  f1-score   support

           0       0.84      0.82      0.83       110
           1       0.57      0.61      0.59        44

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

In [31]:

```
print(accuracy_score(predict, y_test)*100)
```

```
75.97402597402598
```

In [32]:

```
from xgboost import XGBClassifier
```

In [33]:

```
model = XGBClassifier()
```

In [34]:

```
model.fit(X_train, y_train)
```

Out[34]:

```
XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints=None,
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=0, num_parallel_tree=1,
              objective='binary:logistic', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method=Non
e,
              validate_parameters=False, verbosity=None)
```

In [35]:

```
model.score(X_train, y_train)
```

Out[35]:

```
1.0
```

In [36]:

```
pre = model.predict(X_test)
```

In [38]:

```
print(classification_report(pre, y_test))
```

```
              precision    recall  f1-score   support

           0       0.80      0.84      0.82       102
           1       0.66      0.60      0.63        52

    accuracy                           0.76       154
   macro avg       0.73      0.72      0.72       154
weighted avg       0.76      0.76      0.76       154
```

In [43]:

```
model.score(X_test, y_test)
```

Out[43]:

```
0.7597402597402597
```

In [44]:

```
print(accuracy_score(pre, y_test)*100)
```

```
75.97402597402598
```

In [ ]:

```

```