# Random Forests - To predict Whether Patients has diabetics or not .

**Random forest is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.**

In [1]:

```
#Setup
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, precision_recall_curve, roc
from sklearn.ensemble import RandomForestClassifier
```

In [6]:

```
names = ["times_pregnant", "glucose_tolerance_test", "blood_pressure", "s
         "bmi", "pedigree_function", "age", "has_diabetes"]
data = pd.read_csv('data/pima-indians-diabetes.csv', names=names, header=
```

In [7]:

```
print(data.shape)
```

```
(767, 9)
```

In [8]:

```
#Print no of integers, floats and strings
data.dtypes.value_counts()
```

Out[8]:

```
int64      7
float64    2
dtype: int64
```

In [9]:

```
#Data should be numerical
data.head()
```

Out[9]:

| | times_pregnant | glucose_tolerance_test | blood_pressure | skin_thickness | insulin |
|---|---|---|---|---|---|
| **0** | 1 | 85 | 66 | 29 | 0 |
| **1** | 8 | 183 | 64 | 0 | 0 |
| **2** | 1 | 89 | 66 | 23 | 94 |
| **3** | 0 | 137 | 40 | 35 | 168 |
| **4** | 5 | 116 | 74 | 0 | 0 |

In [10]:

```
#Print no of entries for each color
data.has_diabetes.value_counts()
```

Out[10]:

```
0    500
1    267
Name: has_diabetes, dtype: int64
```

In [11]:

```
#Print % of each colors
data.has_diabetes.value_counts(normalize=True)
```

Out[11]:

```
0    0.65189
1    0.34811
Name: has_diabetes, dtype: float64
```

# # Preprocessing Steps

### 1. Select Features and Split to X and y .

In [12]:

```
X = data.iloc[:, :-1].values
y = data["has_diabetes"].values
```

### 2. Split data to train and test .

In [13]:

```
# Split the data to Train, and Test (75%, 25%)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
```

## Modelling with Random Forest

### 1. Train a Random Forest model with 200 trees on the training data.

In [14]:

```
## Train the RandomForest Model
rf_model = RandomForestClassifier(n_estimators=200)
rf_model.fit(X_train, y_train)
```

Out[14]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_
weight=None,
                       criterion='gini', max_depth=None, max
_features='auto',
                       max_leaf_nodes=None, max_samples=Non
e,
                       min_impurity_decrease=0.0, min_impuri
ty_split=None,
                       min_samples_leaf=1, min_samples_split
=2,
                       min_weight_fraction_leaf=0.0, n_estim
ators=200,
                       n_jobs=None, oob_score=False, random_
state=None,
                       verbose=0, warm_start=False)
```

## 2.Calculate the accuracy and roc_auc_score of the predictions.

In [15]:

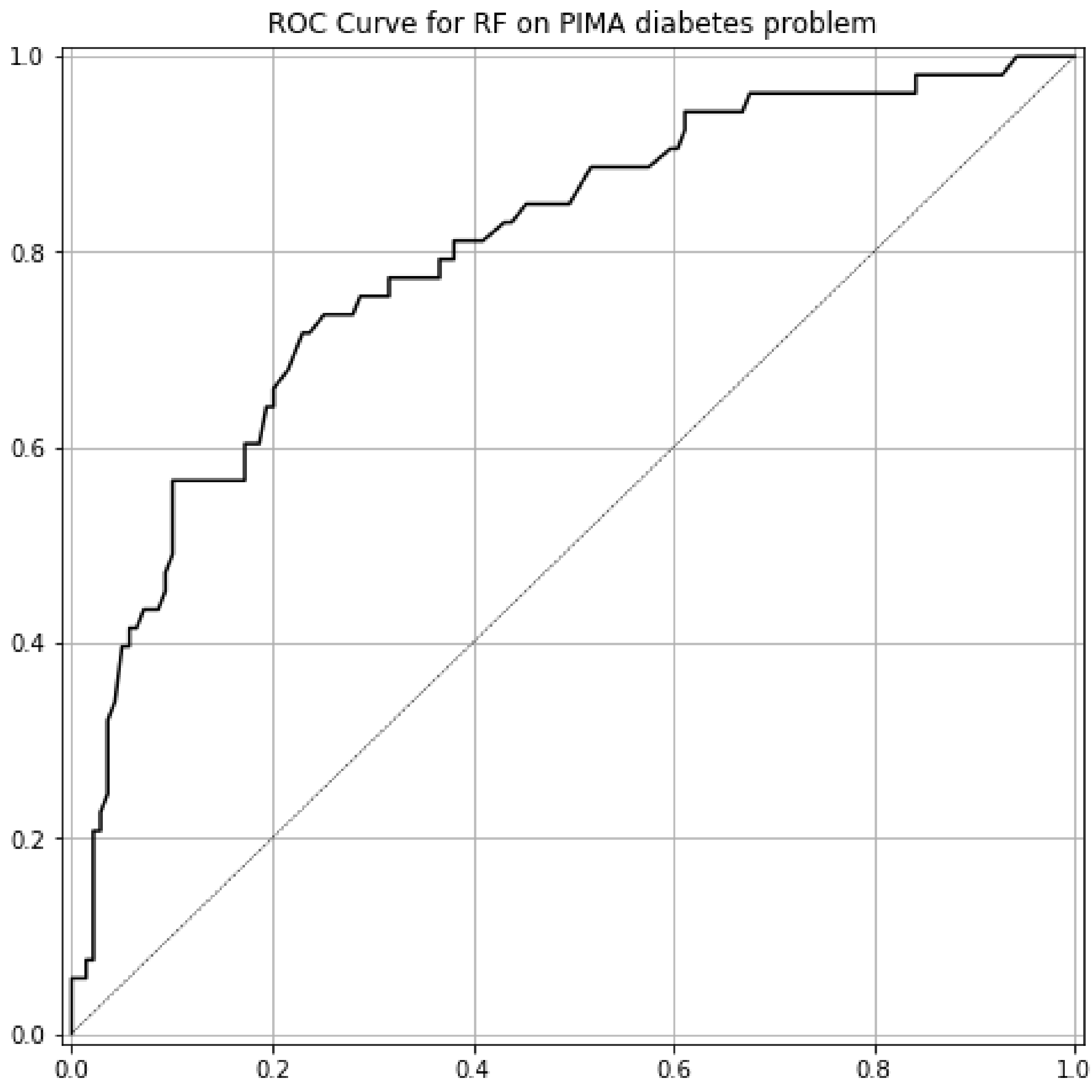
```
# Make predictions on the test set
y_pred_class_rf = rf_model.predict(X_test)
y_pred_prob_rf = rf_model.predict_proba(X_test)


print('accuracy is {:.3f}'.format(accuracy_score(y_test,y_pred_class_rf))
print('roc-auc is {:.3f}'.format(roc_auc_score(y_test,y_pred_prob_rf[:,1]
```

```
accuracy is 0.766
roc-auc is 0.799
```

In [16]:

```
def plot_roc(y_test, y_pred, model_name):
    fpr, tpr, thr = roc_curve(y_test, y_pred)
    fig, ax = plt.subplots(figsize=(8, 8))
    ax.plot(fpr, tpr, 'k-')
    ax.plot([0, 1], [0, 1], 'k--', linewidth=.5)  # roc curve for random
    ax.grid(True)
    ax.set(title='ROC Curve for {} on PIMA diabetes problem'.format(model
           xlim=[-0.01, 1.01], ylim=[-0.01, 1.01])
plot_roc(y_test, y_pred_prob_rf[:, 1], 'RF')
```



## Receiver Operating Characteristic(ROC) curve is a plot of the true positive rate against the false positive rate. It shows the tradeoff between sensitivity and specificity.

In [ ]:

```
```