In [1]:

```
#importing packages
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

#Filtering Warnings
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
#Importing Diabetes Data
DiabetesData = pd.read_csv("pima-indians-diabetes.csv")
DiabetesData.head()
```

Out[2]:

	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 [3]:

```
X=DiabetesData.iloc[:,0:8]
y=DiabetesData.iloc[:,8]
```

In [4]:

```
#Dividing Data in test and train
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1)
```

Creating pipelines

Creating pipelines for Logistic regression, Decision Tree and Random Forest models

Pipeline steps will include

---

1. Data Preprocessing using MinMax Scaler

2. Reducing Dimensionality using PCA

3. Training respective models

In [5]:

```
#Logistic Regression Pipeline
LogisticRegressionPipeline=Pipeline([('myscaler',MinMaxScaler()),
                     ('mypca',PCA(n_components=3)),
                     ('logistic_classifier',LogisticRegression())])
```

In [6]:

```
#Decision tree Pipeline
DecisionTreePipeline=Pipeline([('myscaler',MinMaxScaler()),
                     ('mypca',PCA(n_components=3)),
                     ('decisiontree_classifier',DecisionTreeClassifier())])
```

In [7]:

```
#Random Forest Pipeline
RandomForestPipeline=Pipeline([('myscaler',MinMaxScaler()),
                     ('mypca',PCA(n_components=3)),
                     ('randomforest_classifier',RandomForestClassifier())])
```

Modelling and Evalution

In [8]:

```
## Defining the pipelines in a list
mypipeline = [LogisticRegressionPipeline, DecisionTreePipeline, RandomForestPipeline]
```

In [9]:

```
# Fit the pipelines
for mypipe in mypipeline:
    mypipe.fit(X_train, y_train)
```

In [10]:

```
#Defining variables for choosing best model
accuracy=0.0
classifier=0
pipeline=""
```

In [11]:

```
# Creating dictionary of pipelines and training models
PipelineDict = {0: 'Logistic Regression', 1: 'Decision Tree', 2: 'Random Forest'}
```

In [12]:

```
#getting test accuracy for all classifiers
for i,model in enumerate(mypipeline):
    print("{} Test Accuracy: {}".format(PipelineDict[i],model.score(X_test,y_test)))
```

```
Logistic Regression Test Accuracy: 0.7597402597402597
Decision Tree Test Accuracy: 0.6883116883116883
Random Forest Test Accuracy: 0.7727272727272727
```

In [13]:

```
#Choosing best model for the given data
for i,model in enumerate(mypipeline):
    if model.score(X_test,y_test)>accuracy:
        accuracy=model.score(X_test,y_test)
        pipeline=model
        classifier=i
print('Classifier with best accuracy:{}'.format(PipelineDict[classifier]))
```

```
Classifier with best accuracy:Random Forest
```