Data Science and Machine Learning with Python hands on

**PROJECT ON MAMMOGRAPHY**

PREDICT IF MAMMOGRAPHY MASS IS BELIGNT OR NOT

horizontal line

# Introduction

We'll be using the "mammographic masses" public dataset from the UCI repository. This data contains 961 instances of masses detected in mammograms, and contains the following attributes:  
1. BI-RADS assessment: 1 to 5 (ordinal)

2. Age: patient's age in years (integer)

3. Shape: mass shape: round=1 oval=2 lobular=3 irregular=4 (nominal)

4. Margin: mass margin: circumscribed=1 microlobulated=2 obscured=3 ill-defined=4 spiculated=5 (nominal)

5. Density: mass density high=1 iso=2 low=3 fat-containing=4 (ordinal)

6. Severity: benign=0 or malignant=1 (binominal)

BI-RADS is an assessment of how confident the severity classification is; it is not a “predictive” attribute and so we will discard it. The age, shape, margin, and density attributes are the features that we will build our model with, and “severity” is the classification we will attempt to predict based on those attributes.

Although “shape” and “margin” are nominal data types, which sklearn typically doesn't deal with well, they are close enough to ordinal that we shouldn't just discard them. The “shape” for example is ordered increasingly from round to irregular.

A lot of unnecessary anguish and surgery arises from false positives arising from mammogram results. If we can build a better way to interpret them through supervised machine learning, it could improve a lot of lives.

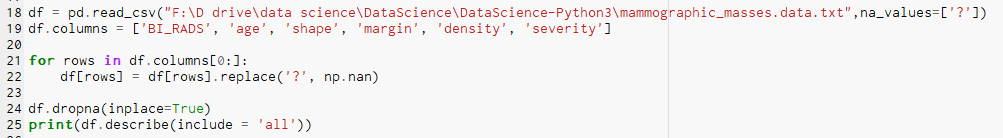
## Import data

Import data from the csv file and define the column headers

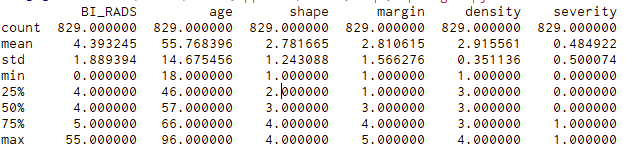


### Data Cleaning

Replace all the missing data into NaN. It was observed that the data were randomly missing. Hence, dropped all the rows with missing data



Output: We can see the stats for each and every column. The range looks convincing. For example, mean age is 55. Shape is ~2, which has range from 0-4



### Convert the Pandas dataframes into numpy arrays

Here we divide data into two parts, into numpy array. First part is age, shape, margin and density. Using these parameters we can predict severity.



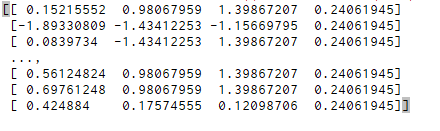
### Data Normalization

Standardize features by removing the mean and scaling to unit variance. It is with 0 mean and unit variance.

We have Standardize our inputs.

### 

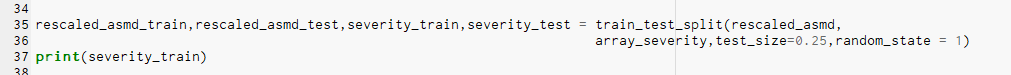
Output:



### Create Train and Test Data

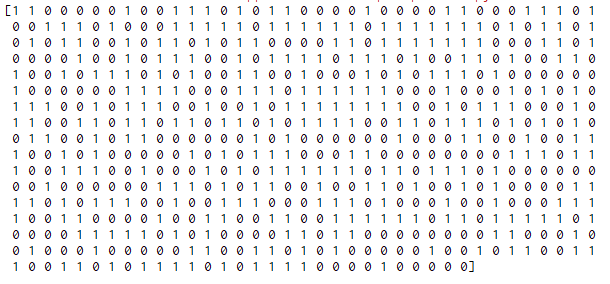
Split arrays or matrices into 75% train and 25% test subsets

Splitting input as well as predicting values.



Output:

Trained out of 621 rows



## Create a Decision Tree Classifier

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, decision tree algorithm can be used for solving regression and classification problems too.

#### Decision Tree Algorithm Pseudocode

1. Place the best attribute of the dataset at the **root** of the tree.
2. Split the training set into **subsets**. Subsets should be made in such a way that each subset contains data with the same value for an attribute.
3. Repeat step 1 and step 2 on each subset until you find **leaf nodes** in all the branches of the tree.

#### Python code

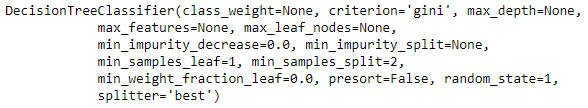
We use sklearn library



fit is used to train classifier on training set



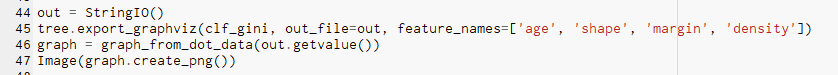
Output:



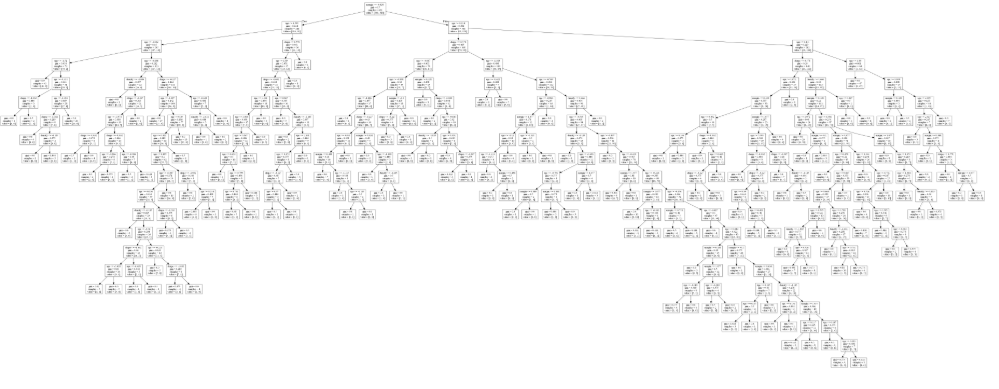
### Display the output

We need to install Graphviz in Windows system





Output:



### Measure the accuracy

We use K-Fold cross validation to get a better measure of your model's accuracy



Output:

The result is 0.734644851182. Which means it is able to predict 73.46% values from test data.

## Create a Random Forest Classifier

In general, the more trees in the forest the more robust the forest looks like. In the same way in the random forest classifier, the higher the number of trees in the forest gives the high accurate results.

We are creating more number of decision trees and we can create more number of decision trees. As all the calculation of nodes selection will be same for the same dataset.

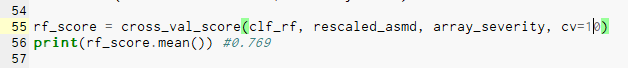
We will model more number of decision trees to create the forest. Unlike decision tree, we are not going to use the same apache of constructing the decision with information gain or gini index approach. Instead of using information gain or gini index for calculating the root node, the process of finding the root node and splitting the feature nodes will happen randomly.

#### Python code:





### Measure the accuracy

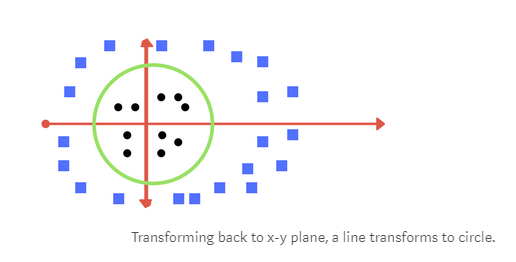


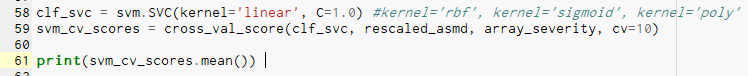
K-Fold cross validation to get a better measure of your model's accuracy. The output is 0.769

## Create a Standard Vector Classification - SVM

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

When we transform back this line to original plane, it maps to circular boundary as shown in image. These transformations are called kernels.





### Measure the accuracy

K-Fold cross validation to get a better measure of your model's accuracy. The output are :

* Linear Kernel - 0.796341113583 - The linear kernel is what you would expect, a linear model.
* Rbf - 0.801044596504 - RBF uses normal curves around the data points, and sums these so that the decision boundary can be defined by a type of topology condition such as curves where the sum is above a value of 0.5
* Sigmoid - 0.743252452318 - Similar to logistic regression as next topic
* Poly - 0.792610511733 - the boundary is of some defined but arbitrary order

## Create a KNN

In KNN, Predictions are made for a new instance by searching through the entire training set for the K most similar instances (the neighbors) and summarizing the output variable for those K instances. For regression this might be the mean output variable, in classification this might be the mode (or most common) class value.

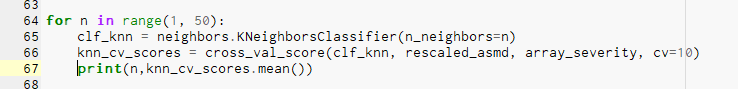
To determine which of the K instances in the training dataset are most similar to a new input, a distance measure is used. For real-valued input variables, the most popular distance measure is Euclidean distance.

The algorithm can be summarized as:

1. A positive integer k is specified, along with a new sample
2. We select the k entries in our database which are closest to the new sample
3. We find the most common classification of these entries
4. This is the classification we give to the new sample

#### Python Code:

We will calculate score from 1 to 50 neighbours, and try to find out the best K for our data set.



### Measure the accuracy

Output: The output seems to get better with with increasing number of K. For K=50, the cv score is 0.79

1 0.726020807971

2 0.695782432867

3 0.756214754488

4 0.733321788898

5 0.773302967969

6 0.76362068483

7 0.792639547738

8 0.780474161454

9 0.790127408589

10 0.786454528917

11 0.790156444594

12 0.781706967242

13 0.780531183969

14 0.775638792102

15 0.782868057596

16 0.779297328688

17 0.781692624155

18 0.774347214643

19 0.780371311028

20 0.781547094301

21 0.784029847613

22 0.782824678505

23 0.779224913592

24 0.784072876873

25 0.785321774904

26 0.786497908008

27 0.785293088731

28 0.788878510558

29 0.784044540531

30 0.79005429383

31 0.790068636917

32 0.787644305445

33 0.784073226705

34 0.785278045982

35 0.786482865259

36 0.786482865259

37 0.788921889649

38 0.786511901264

39 0.786497208346

40 0.785321425073

41 0.785306732155

42 0.781692274324

43 0.782897093601

44 0.782897093601

45 0.784116605796

46 0.784087569791

47 0.787717070372

48 0.790127058758

49 0.791346570953

## Create a Naive Bayes

In machine learning we are often interested in selecting the best hypothesis (h) given data (d).

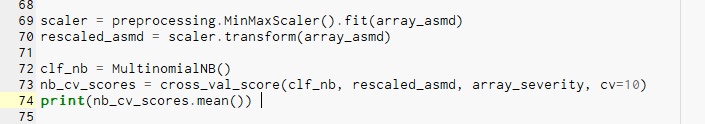
In a classification problem, our hypothesis (h) may be the class to assign for a new data instance (d).

One of the easiest ways of selecting the most probable hypothesis given the data that we have that we can use as our prior knowledge about the problem. Bayes’ Theorem provides a way that we can calculate the probability of a hypothesis given our prior knowledge.

Bayes’ Theorem is stated as:

P(h|d) = (P(d|h) \* P(h)) / P(d)

#### Python Code:



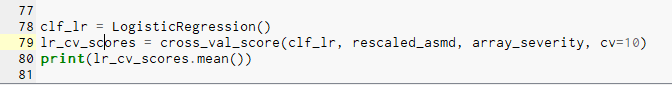
### Measure the accuracy

The output is 0.784205113135

## Create a Logistic Regression

Given data on time spent studying and exam scores. Linear Regression and logistic regression can predict different things:  
  
1) Linear Regression could help us predict the student’s test score on a scale of 0 - 100. Linear regression predictions are continuous (numbers in a range).  
2) Logistic Regression could help use predict whether the student passed or failed. Logistic regression predictions are discrete (only specific values or categories are allowed). We can also view probability scores underlying the model’s classifications.

#### Python Code:



### Measure the accuracy

The output is 0.803570728909. I think we have a winner here.

## Libraries imported

import pandas as pd

import numpy as np

from sklearn import preprocessing

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

from sklearn.externals.six import StringIO

from pydotplus import graph\_from\_dot\_data

from IPython.display import Image

from sklearn.model\_selection import cross\_val\_score

from sklearn.ensemble import RandomForestClassifier

from sklearn import svm

from sklearn import neighbors

from sklearn.naive\_bayes import MultinomialNB

from sklearn.linear\_model import LogisticRegression

## Machine learning quick reference

