Analytics using Python

Learning outcomes

1. You will learn Python , a useful language

2. Use programming for problem solving

Great Lakes Institute of Management

A guide to learn python for analytics

P. V. Subramanian

**A workbook on Analytics using Python**

**P. V. SUBRAMANIAN**

**Visiting faculty – Great Lakes Institute of Management**

*Copyright @ 2019 Great Lakes Institute of Management : All contents in this book, including (but not limited to) its contents are proprietary to Great Lakes Institute of Management, Chennai. Any effort at re-using, modifying and circulating both through Digital or Physical means, without the express written permission of the author, will amount to infringement of copyrights, and will be liable for suitable action.*

**Chapter 8. Continuation of Regression and Decision Trees and Ensemble Techniques basics**

Table of Contents

[**1)** **Regression continued** 5](#_Toc10978154)

[**1.1** **regularization techniques** 5](#_Toc10978155)

[**1.1.1** **Ordinary Least Square Regression** 13](#_Toc10978156)

[**1.1.2** **ridge Regression – l2 regularization techniques** 15](#_Toc10978157)

[**1.1.3** **lasso Regression – l1 regularization techniques** 17](#_Toc10978158)

[**1.1.4** **Elastic Net regression** 18](#_Toc10978159)

[**1.2** **non-linear models overview** 20](#_Toc10978160)

[**1.2.1** **Decision Trees (CART)** 20](#_Toc10978163)

[**1.2.2** **Random Forest Regression** 21](#_Toc10978164)

[**1.2.3** **Artificial Neural Network (ANN) Regression** 21](#_Toc10978165)

[**1.2.4** **Gradient Boosting Regression** 22](#_Toc10978166)

[**1.2.5** **SVR Regression** 22](#_Toc10978167)

[**1.2.6** **KNN Regression** 22](#_Toc10978168)

[**1.3** **build and test non-linear models** 23](#_Toc10978169)

[**2)** **decision trees overview** 31](#_Toc10978170)

[**1.1.** **Introduction** 31](#_Toc10978171)

[**1.2.** **Types of Decision Trees** 32](#_Toc10978172)

[**1.3.** **Terminology explained** 32](#_Toc10978173)

[**2.3.1** **Entrophy** 32](#_Toc10978178)

[**2.3.2** **Information Gain** 32](#_Toc10978179)

[**2.3.3** **Root Node** 32](#_Toc10978180)

[**2.3.4** **Splitting** 32](#_Toc10978181)

[**2.3.5** **Decision node** 32](#_Toc10978182)

[**2.3.6** **Terminal node: (aka leaf node)** 33](#_Toc10978183)

[**2.3.7** **Pruning** 33](#_Toc10978184)

[**2.3.8** **Branch: (aka sub-tree)** 33](#_Toc10978185)

[**2.3.9** **Parent and Child node:** 33](#_Toc10978186)

[**1.4.** **Different decision tree algorithms** 33](#_Toc10978187)

[**2.4.1** **id3** 33](#_Toc10978189)

[**2.4.2** **C4.5** 34](#_Toc10978190)

[**2.4.3** **CART** 34](#_Toc10978191)

[**2.4.4** **Chi-square Automatic Interaction Detection (CHAID)** 35](#_Toc10978192)

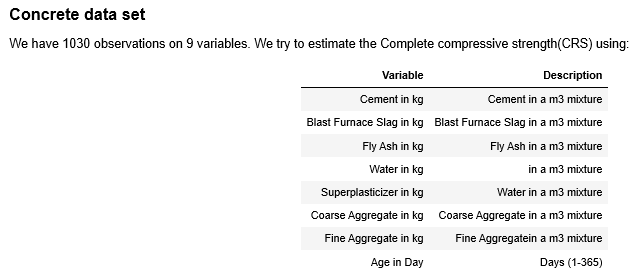
[**3)** **cart model** 36](#_Toc10978193)

[**3.1.** **Example 1** 36](#_Toc10978197)

# **Regression continued**

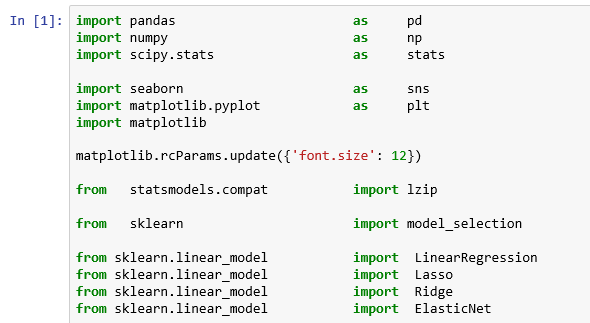
## **regularization techniques**

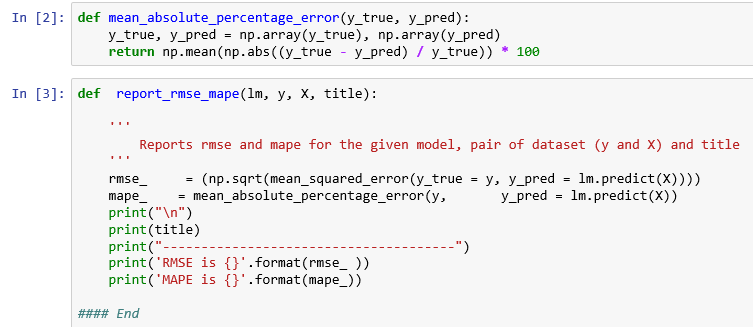
In the previous chapter 7, we had checked the linear regression assumptions on the concrete data

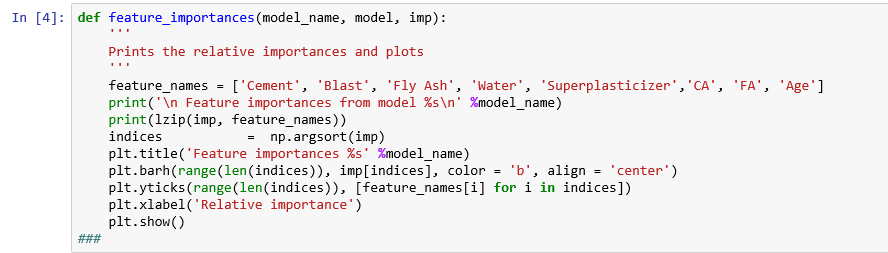


We also created a cleaned data set for both training and testing. We have 9 predictor variables and 823 observations in the training data set. we have 9 predictor variables and 206 observations in the testing data set. We shall apply various regularization techniques such as Ridge Regression, Lasso Regression and Elastic Net Regression on the training data set and test the model performance using the test data.

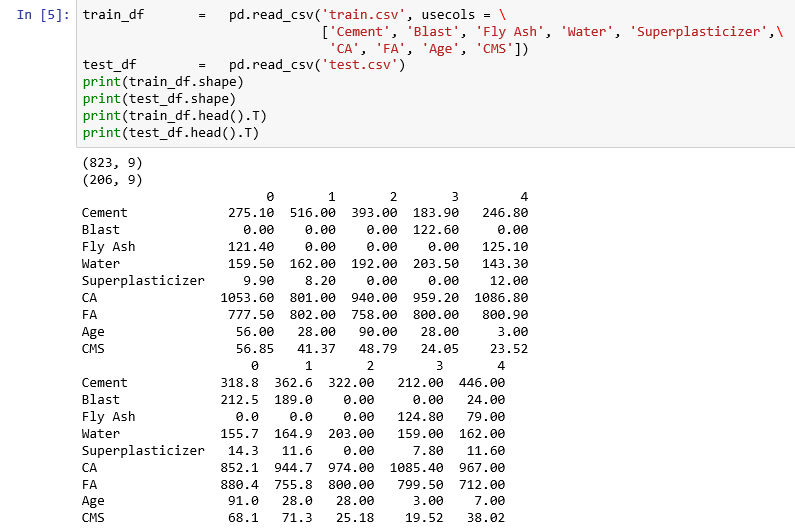
**Step 1: Import the required modules and define useful functions**

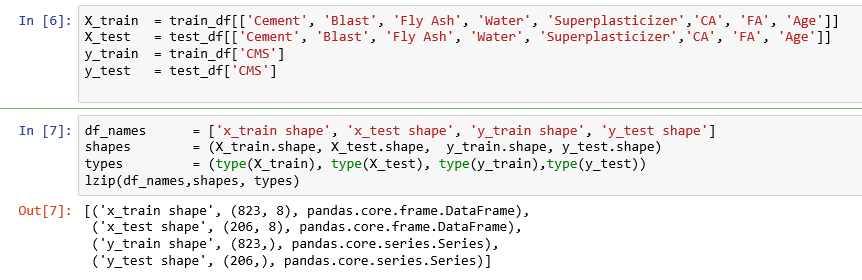






**Step 2: Read the csv files, train.csv and test.csv**





We have successfully read the data file into python memory as the counts match.

We need to build the model using the training data set and check the model performance using the testing dataset.

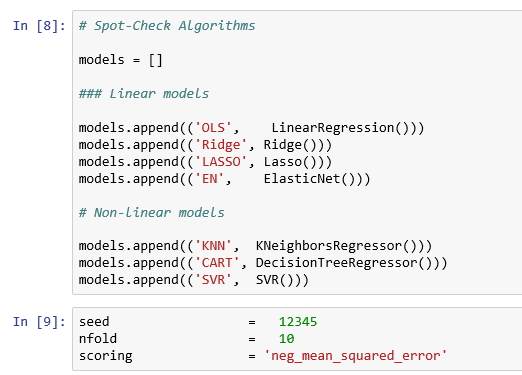
Prediction error or residuals is the difference between the predicted target variable values and the actual target variable values. Most popular measure to evaluate the model performance is Root Mean Square Error (RMSE) which is the arithmetic mean of the sum of the residuals. The model with low RMSE is the best model among many other models. This is implemented as neg\_mean\_squared\_error scoring method forcing us to multiply a negative sign.

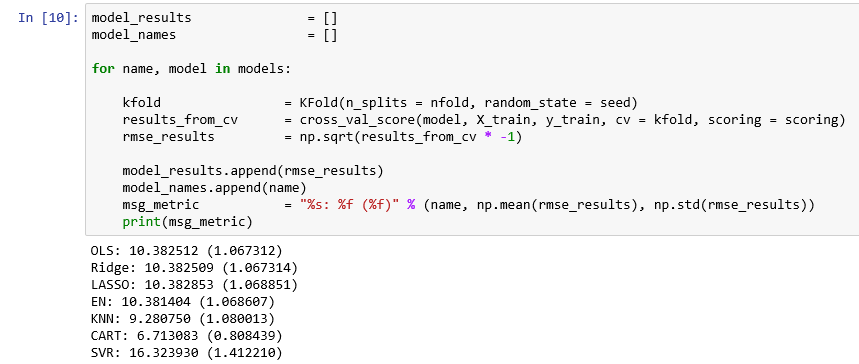
**Evaluate algorithms - spot check**

Let us use 10-fold cross-validation. We shall use RMSE to evaluate algorithms. The purpose of the K-fold is to prevent the classifier from over fitting the training data.

In K-Folds Cross Validation, we split our data into k different subsets (or folds). We use k-1 subsets to train our data and leave the last subset (or the last fold) as test data, which the classifier has not seen. We then average the model against each of the folds and then finalize our model.

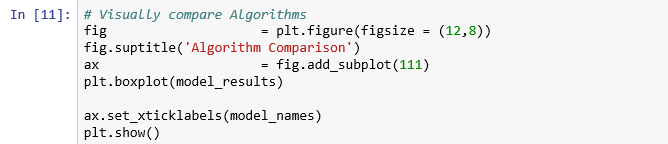
We shall use mean s

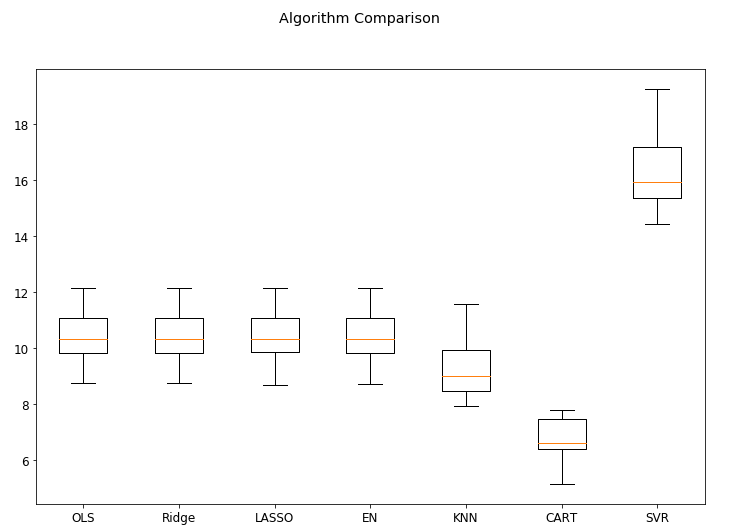
****

****

**We observe that the CART model gives the best accuracy at 6.71 as RMSE.**

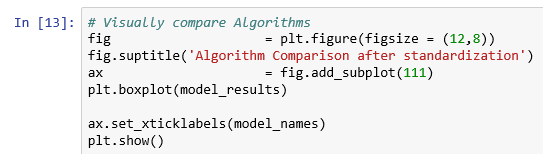
**Look at the distribution of scores across all cross-validation folds by algorithm.**

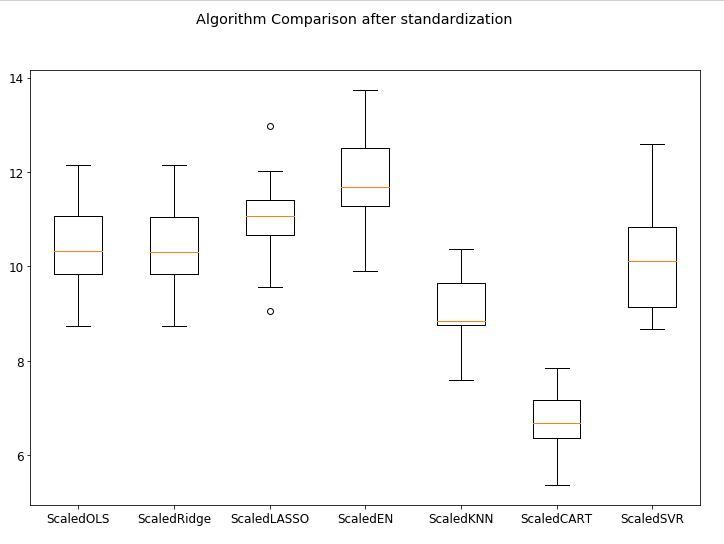
****

****

The diﬀering scales of the data is probably hurting the skill of all the algorithms and perhaps more so for SVR and KNN. Use a standardized data and repeat the above.



****

****

**Observation**: Scaling did have an eﬀect on SVR, driving the error lower than the other models such as OLS, Ridge, LASSO and Elastic Net.

**Build the model for each of the algorithms mentioned below:**

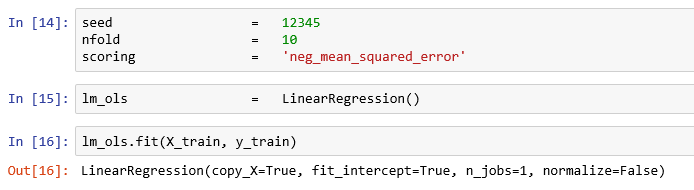
* Ordinary Least Square Regression
* Ridge Regression
* Lasso Regression
* Elastic Net regression

### **Ordinary Least Square Regression**

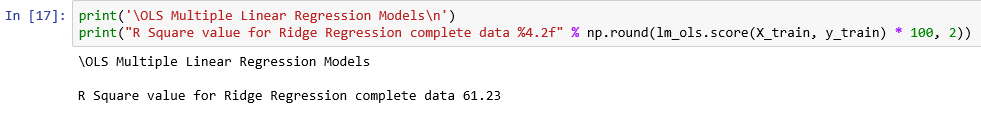
* The Ordinary Least Squares (OLS) method is a statistical procedure for using sample data to find the value of the estimated regression equation. The OLS Method was proposed by Carl Friedrich Gauss in 1809.
* The OLS estimator is consistent when
* the regressors are exogenous (Independent variable that affects a model without being affected by it (ref: Read more: http://www.businessdictionary.com/definition/exogenous-variable.html)
* optimal in the class of linear unbiased estimators when the errors are homoscedastic
* there is no auto correlation or serially uncorrelated

Serial correlation is the relationship between a variable and a lagged version of itself over various time intervals. Refer: https://www.investopedia.com/terms/s/serial-correlation.asp

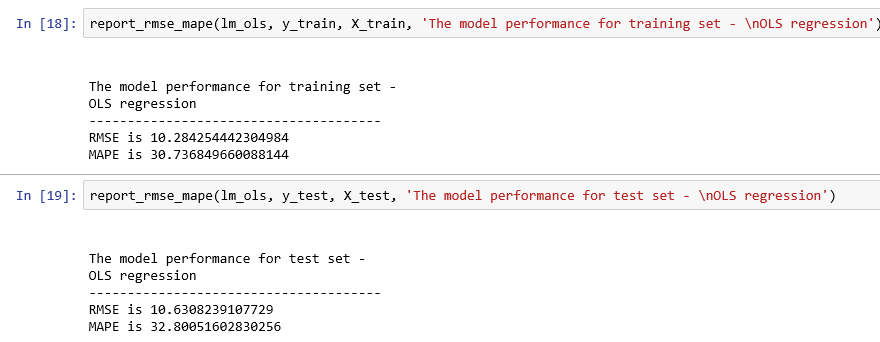
* Under these conditions, OLS method provides minimum-variance mean-unbiased estimation when the errors have finite variances.
* Under the additional assumption that the errors are normally distributed, OLS is the maximum likelihood estimator.
* OLS chooses the parameters of a linear function of a set of explanatory variables by the principle of least squares: minimizing the sum of the squares of the differences between the observed dependent variable (values of the variable being predicted) in the given dataset and those predicted by the linear function. Refer: https://en.wikipedia.org/wiki/Ordinary\_least\_squares
* We shall use LinearRegression function available in sklearn.linear\_model module.
* To ensure repeatability, we shall set the random seed.
* We shall use 10-fold cross validation and scoring as negative mean squared error.



* We shall calculate the R square which explains the percentage of the variation in the target field, CMS by the model.



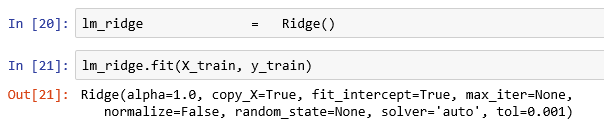
* This is not very great and let us check the accuracy measures for both training and testing data sets.



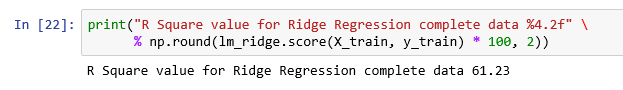
* This is no big difference between measures like RMSE, MAPE in both training and testing data sets.

### **ridge Regression – l2 regularization techniques**

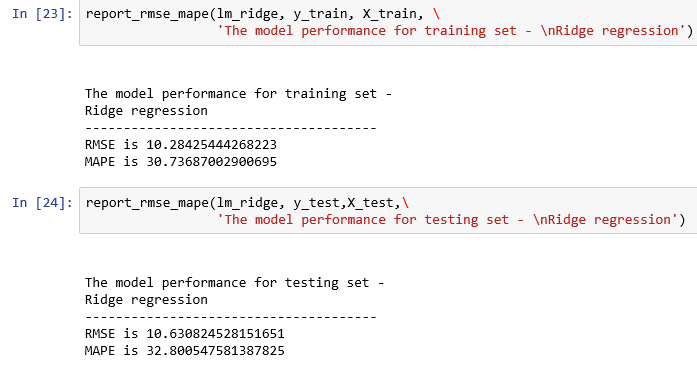
* Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity.
* Here, we minimize a quantity (alpha) which is the sum of the squared residuals, plus a term usually proportional to the sum (or often a weighted sum) of the squared parameters. Essentially, we “penalize” large values of the parameters in the quantity we’re seeking to minimize.
* Unlike least squares method, ridge regression produces a set of coefficient estimates for different values of the tuning parameter.
* By changing the values of alpha, we are basically controlling the penalty term. Higher the values of alpha, bigger is the penalty and therefore the magnitude of coefficients are reduced.
* For more information, please refer to
  + https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/
  + https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Ridge\_Regression.pdf
* We shall use Ridge function available in sklearn.linear\_model module.



* We shall calculate the R square which explains the percentage of the variation in the target field, CMS by the model.



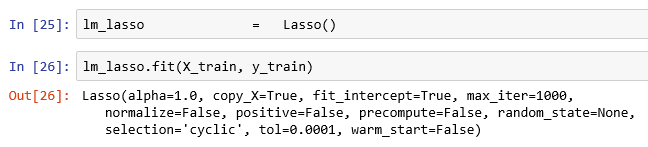
This is not very great and let us check the accuracy measures for both training and testing data sets.



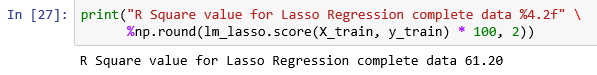
* This is no big difference between measures like RMSE, MAPE in both training and testing data sets.

### **lasso Regression – l1 regularization techniques**

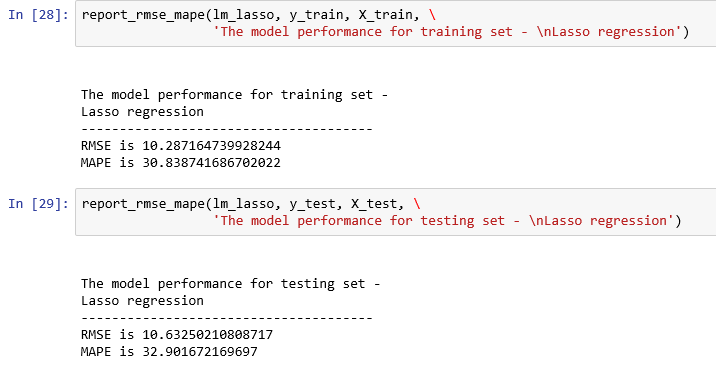
* LASSO (Least Absolute Shrinkage Selector Operator) is also a regularization method like Ridge that tries to avoid overfitting penalizing large coefficients.
* LASSO also adds a penalty for non-zero coefficients, but unlike ridge regression which penalizes sum of squared coefficients (L2 penalty), LASSO penalizes the sum of their absolute values (L1 penalty).
* This method has as great advantage the fact that it can shrink some of the coefficients to exactly zero, performing thus a selection of attributes with the regularization.
* We shall use Lasso function available in sklearn.linear\_model module.



* We shall calculate the R square which explains the percentage of the variation in the target field, CMS by the model.



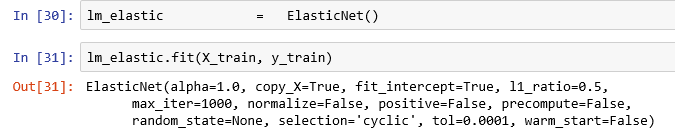
* This is not very great and let us check the accuracy measures for both training and testing data sets.



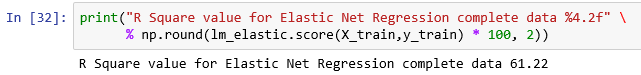
* This is no big difference between measures like RMSE, MAPE in both training and testing data sets.

### **Elastic Net regression**

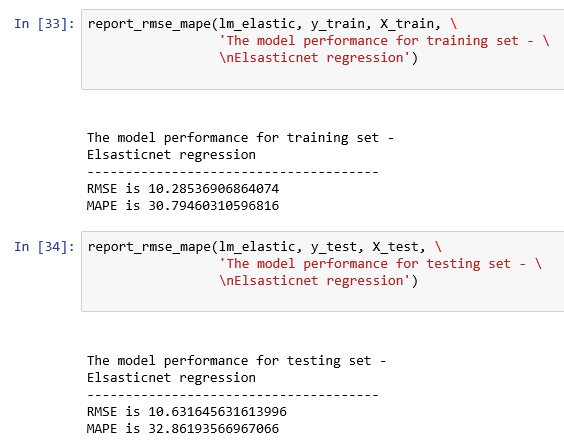
* Elastic Net combines the penalties of ridge regression and LASSO regression.
* The Elastic Net addresses criticism of LASSO of “over-regularization” by balancing between LASSO and ridge penalties.
* A hyper-parameter, namely λ, would be used to regularize the model such that the model would become a LASSO in case of λ = 1 and a ridge in case of λ = 0.
* λ can be tuned easily by the cross-validation.
* We shall use Lasso function available in sklearn.linear\_model module.



* We shall calculate the R square which explains the percentage of the variation in the target field, CMS by the model.



* This is not very great and let us check the accuracy measures for both training and testing data sets.



* This is no big difference between measures like RMSE, MAPE in both training and testing data sets.
* We have tried most popular linear regression models and observed that the R square is not improving beyond 60% (approximately). We need to try non-linear models.

## **non-linear models overview**

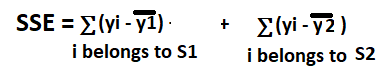
* We can try non-linear models as mentioned below:
* CART Regression
* Random Forest Regression
* Artificial Neural Network Regression
* Gradient Boosting Regression
* SVR Regression
* KNN Regression

KFold will provide train/test indices to split data in train and test sets. It will split dataset into k consecutive folds (without shuffling by default).Each fold is then used a validation set once while the k - 1 remaining folds form the training set. For more details, refer to http://scikit-learn.org/stable/modules/generated/sklearn.cross\_validation.KFold.html

* For more details, refer https://machinelearningmastery.com/spot-check-regression-machine-learning-algorithms-python-scikit-learn/

### **Decision Trees (CART)**

* Decision trees or the Classification and Regression Trees (CART) use the training data to select the best points to split the data in order to minimize a cost metric. The default cost metric for regression decision trees is the mean squared error, as specified in the criterion parameter.
* In Decision Tree for Regression, each root node represents a single input variable (x) and a split point on that variable (assuming the variable is numeric). The leaf nodes of the tree contain an output variable (y) which is used to make a prediction.
* The most common method for constructing regression tree is CART (Classification and Regression Tree) methodology, which is also known as recursive partitioning.
* The method starts by searching for every distinct values of all its predictors, and splitting the value of a predictor that minimizes the following statistic (other regression tree models have different optimization criteria):



* where are the average values of the dependent variable in groups S1 and S2.
* For groups S1 and S2, the method will recursively split the predictor values within groups. In practice, the method stops when stopping criteria is met (i.e., the sample size of the split group falls below certain threshold, e.g., 20).
* Unlike linear regression models that calculate the coefficients of predictors, tree regression models calculate the relative importance of predictors. The relative importance of predictors can be computed by summing up the overall reduction of optimization criteria like SSE.
* We create a CART model for regression using the DecisionTreeRegressor class.

### **Random Forest Regression**

* Random Forest builds several decision trees and merges them together rather than relying on individual decision trees to get a more accurate and stable prediction.
* Regression forests are used for nonlinear multiple regression. Each leaf contains a distribution for the continuous output variables(s).
* Steps involved:

1. Sample multiple subsamples with replacement from the training data
2. Train a decision tree for regression (splitting e.g. by maximizing reduction in variance ) on each subsample, where each leave node outputs the mean of all label values in the node.
3. Predict by averaging over the predictions of all decision trees.

* We create a Random Forest model for regression using the RandomForestRegressor class imported from sklearn.ensemble module.

### **Artificial Neural Network (ANN) Regression**

* An Artificial Neural Network (ANN) is useful if the relationship between the predictors and target is non-linear.
* A neural network is a flexible model that adapts itself to the shape of the data by selecting the best type of regression (linear, polynomial etc.).
* To improve the predictive power or to handle more complex models, add hidden neuron layers.

Refer:

* *https://missinglink.ai/guides/neural-network-concepts/neural-networks-regression-part-1-overkill-opportunity/*
* [*https://www.quora.com/How-do-I-use-neural-network-to-do-the-regression-problem-rather-than-classification*](https://www.quora.com/How-do-I-use-neural-network-to-do-the-regression-problem-rather-than-classification)
* We create a ANN model for regression using the MLPRegressor class imported from sklearn.neural\_network module.
* MLPRegressor trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters.
* It can also have a regularization term added to the loss function that shrinks model parameters to prevent overfitting.
* This implementation works with data represented as dense and sparse numpy arrays of floating point values. To get more details, refer:

*https://scikit-learn.org/stable/modules/generated/sklearn.neural\_network.MLPRegressor.html*

### **Gradient Boosting Regression**

* Gradient Boosting for regression builds an additive model in forward stage wise regression; it allows for the optimization of arbitrary differentiable loss functions such squared error.
* In each stage, a regression tree is fit on the negative gradient of the given loss function.
* We construct a Gradient Boosting model for regression using the GradientBoostingRegressor class.

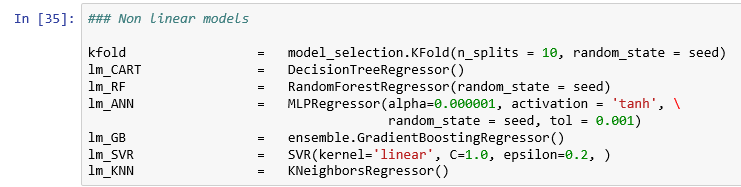
### **SVR Regression**

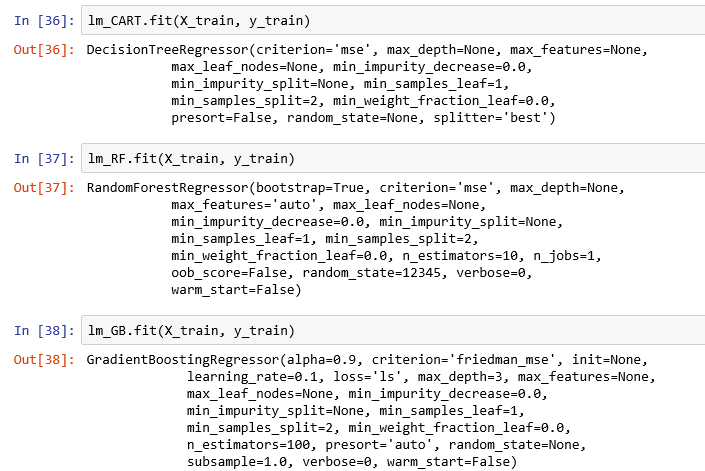
* Support Vector Machines, developed for binary classification is extended for the prediction real-valued problems. This is known as Support Vector Regression (SVR).
* Like the classification example, SVR is built upon the LIBSVM library.
* You can create an SVM model for regression using the SVR class.

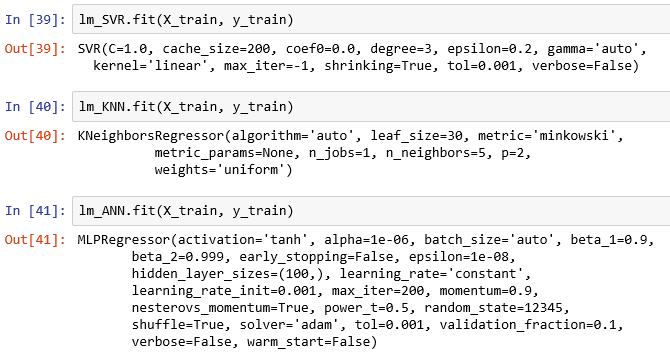
### **KNN Regression**

* KNN locates the K most similar instances in the training dataset for a new data instance.
* From the K neighbors, a mean or median output variable is taken as the prediction.
* Of note is the distance metric used (the metric argument).
* The Minkowski distance is used by default, which is a generalization of both the Euclidean distance (used when all inputs have the same scale) and Manhattan distance (for when the scales of the input variables differ).
* We construct a KNN model for regression using the KNeighborsRegressor class.

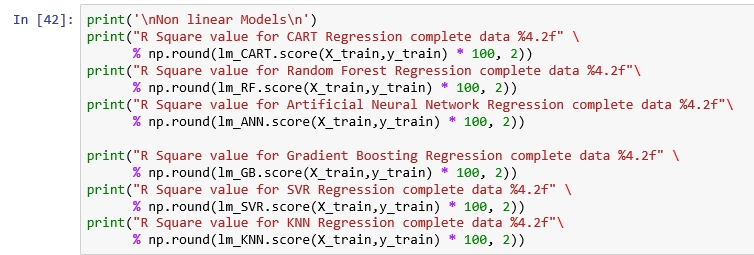
## **build and test non-linear models**

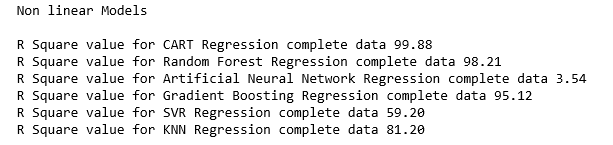






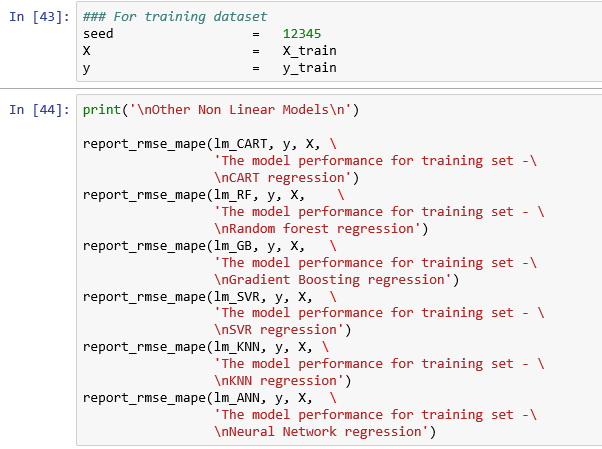
**Get R squared value for all these models.**

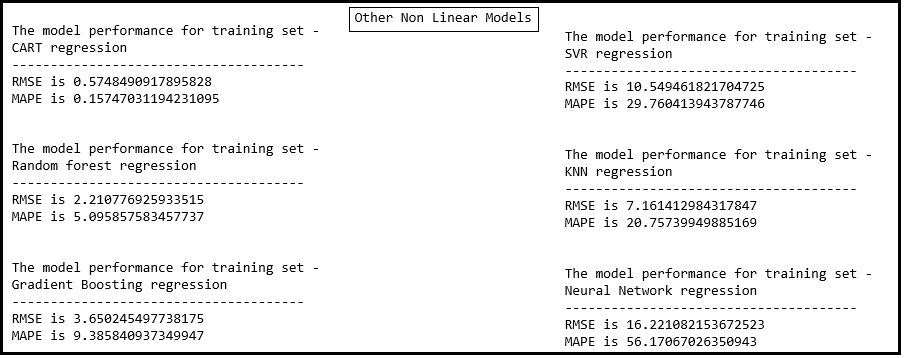




We observe that CART, Random Forest and Gradient Boosting models report a very high R square of above 95%.

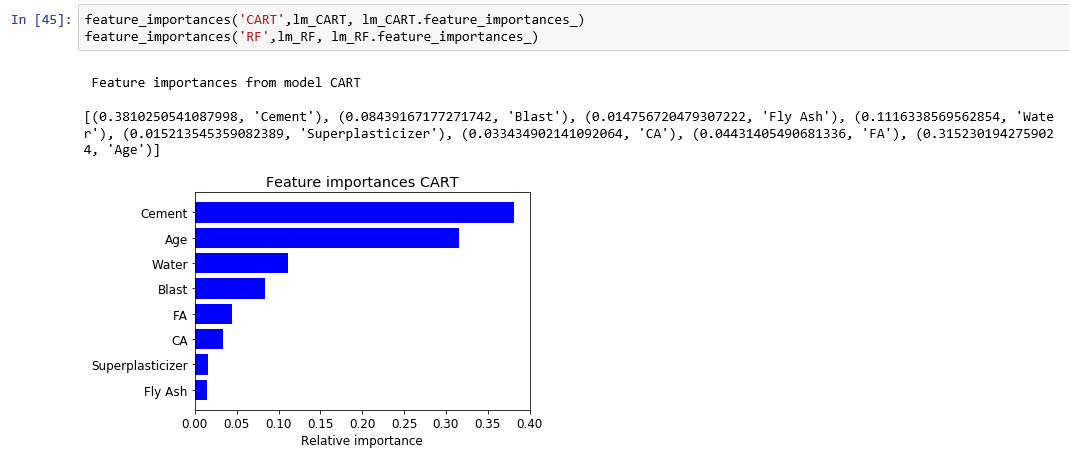
**We report the RMSE and MAPE for the training data sets.**

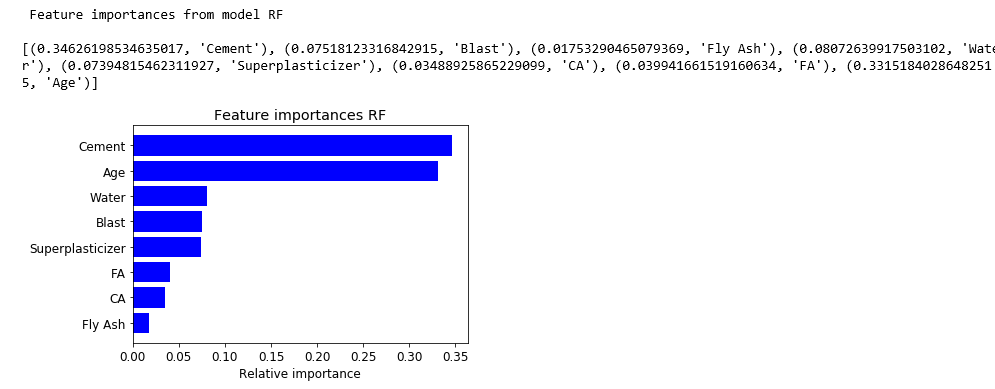




**We report the variable importance revealed by various models.**

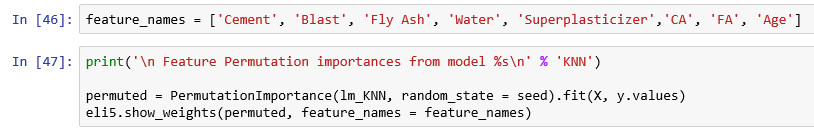
***Here, the importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the Gini importance.***

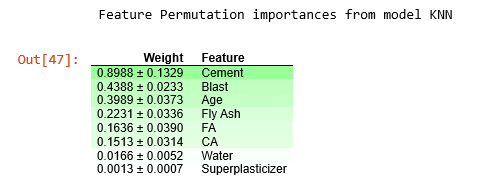
****

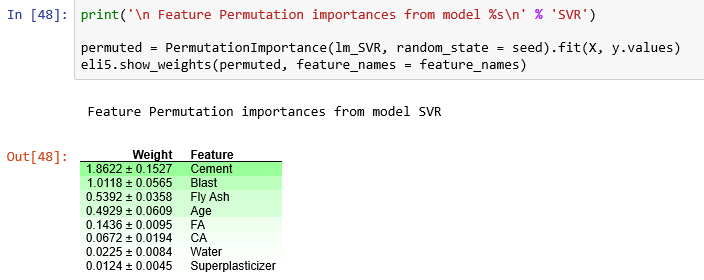
****

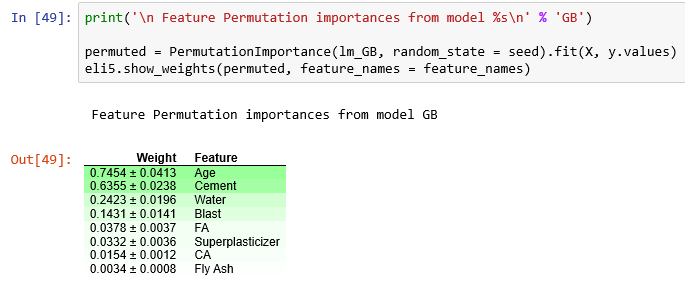
**We observe that Cement, Age and Water are the three most important variables as per CART and Random Forest Models even though there is a difference in importance.**

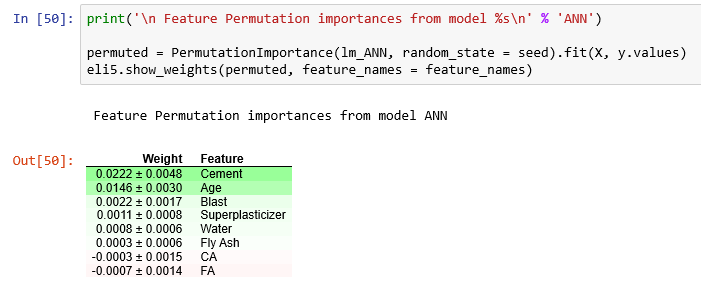
**For other non-linear models, we will try permutation importance method. Permutation Importance works for many scikit-learn estimators. It shuffles the data and the removes different input variables to see what relative change results in the calculating the training model. It measures how much the outcome goes up or down given the input variable, thus calculating their impact on the results. Refer** [**https://www.bmc.com/blogs/scikit-learn-permutation-importance/**](https://www.bmc.com/blogs/scikit-learn-permutation-importance/)

****

****

****

****

****

**Top three variables of importance from various models:**

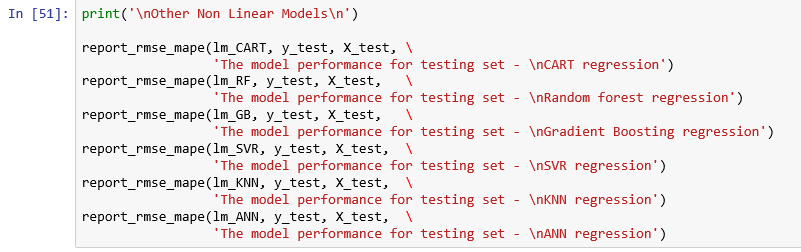
**KNN: Cement, Blast and Age**

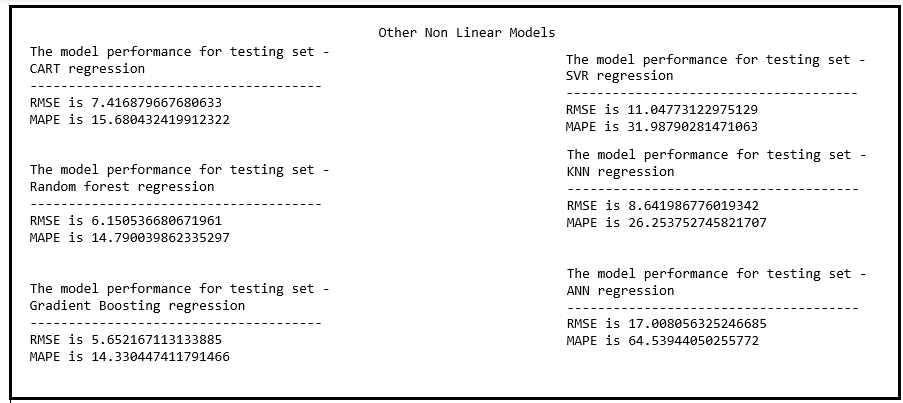
**SVR: Cement, Blast and Fly Ash**

**GB: Age, Cement and Water**

**ANN: Cement, Age, Blast**

**We report the RMSE and MAPE for the test data sets.**





**From the above table, we observe that the best model is Gradient Boosting with lowest RMSE for test data at 5.62 with R squared at 95% while the RMSE for test data for CART model is 7.42 (inferior) even though R squared value is 99.88%.**

For more details, refer:

* http://www.science.smith.edu/~jcrouser/SDS293/labs/lab10-py.html
* <https://theprofessionalspoint.blogspot.com/2019/02/implement-decision-tree-algorithm-in_22.html>
* https://www.datavedas.com/regression-problems-in-python/

# **decision trees overview**

## **Introduction**

Decision Trees (DT), Supervised Classification algorithms are very favorite tool as they are easy to

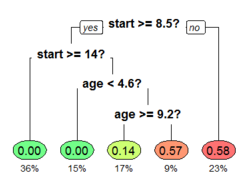
comprehend the model's logic in the form of tree-rules and easy to implement.

They extract predictive information in the form of Tree rules.

A decision Tree is a flow-chart like structure, where each non-leaf (internal) node denotes a test on an attribute, each branch represents the outcome of a test, and each leaf or terminal node holds a class label. The topmost node in a tree is the root node.

An example from <https://en.wikipedia.org/wiki/Decision_tree_learning>

Decision Tree which estimates the probability of kyphosis after surgery, given the age of the patient and the vertebra at which surgery was started.



## **Types of Decision Trees**

1. Classification tree - When the predictive outcome is the class to which the data belongs such as a customer will churn or not.
2. Regression tree - When the predicted outcome can be considered a real number such as predicting the price of a house, length of stay of a patient in the hospital

## **Terminology explained**



### **Entrophy**

Entrophy is a measure of unpredictability.

**Formula for calculating entrophy**

Entrophy = -p(E1) \* log2(p(E1)) - -p(E2) \* log2(p(E2)) -p(E3) \* log2(p(E3)) - ... -p(En) \* log2(p(En))

where E1, E2, ..., En are discrete events and p(E1), p(E2), ..., P(En) are their respective probabilities.

### **Information Gain**

Information Gain is a decrease in entrophy.

### **Root Node**

It represents entire population or sample and this further is splitted into two or more homogeneous sets.

### **Splitting**

It is a process of dividing a node into two or more sub-nodes.

### **Decision node**

In this node a decision rule is applied to split a sub-node into further sub-nodes.

### **Terminal node: (aka leaf node)**

It is the node that do not split further.

### **Pruning**

It is the process of removing sub-nodes of a decision node.

### **Branch: (aka sub-tree)**

It is a section of entire tree.

### **Parent and Child node:**

Parent node of the sub-nodes is a node, which is divided into sub-nodes

Those sub-nodes are the child of the parent node.

## **Different decision tree algorithms**



### **id3**

Iterative dichotomiser 3 (ID3) was invented by Ross Quinian.

For more details, refer https://nulpointerexception.com/2017/12/16/a-tutorial-to-understand-decision-tree-id3-learning-algorithm/#more-253

#### **Construction of the id 3 tree**

* 1. Create an attribute list
  2. Find the maximum information gain among all the features. Assign it to the root node.
  3. Remove the attribute assigned in the root node from the attribute list and again find the maximum increase in information gain for each branch. Assign the attribute as a child node of each branch and remove that attribute from attribute list for that branch.
  4. Repeat step c until you get branches with only pure leaf.

### **C4.5**

C4.5, an extension of ID3 is an algorithm used to generate a decision tree developed by Ross Quinlan.

It addresses the issues not solved by ID3. Some of the issues are:

a. Choosing an appropriate attribute selection measure

b. Handling training data with missing attribute values, different costs

c. Avoid overfitting the data by employing Single Pass Pruning Process

d. Work with both Discrete and Continuous data

C 4.5 generates a decision tree where each node splits the classes based on the gain of information.

The attribute with the highest normalized information gain is used as the splitting criteria.

For more details, refer:

<http://www2.cs.uregina.ca/~dbd/cs831/notes/ml/dtrees/c4.5/tutorial.html>

<https://en.wikipedia.org/wiki/C4.5_algorithm>

<https://towardsdatascience.com/what-is-the-c4-5-algorithm-and-how-does-it-work-2b971a9e7db0>

https://www.quora.com/What-is-the-C4-5-algorithm-and-how-does-it-work

### **CART**

The term Classification and Regression Tree (CART) analysis is an umbrella term used to refer to both of the Regression and Classification types. This is introduced by by Leo Breiman, Jerome Friedman, Richard Olshen and Charles Stone in 1984.

The CART decision tree is a Binary Decision Tree. At each level of decision tree, the algorithm identifies a condition, which variable and level to be used for splitting input node (data sample) into two child nodes. This process is repeated until we only have terminal nodes at which point there is no more condition to be tested.

#### **Construction of the CART tree**

1. Use training data with a target variable, which is and other predictor variables.
2. Find the best split for each of the predictor variables.
3. Select the best variable for the split
4. Split the input data into Left and Right Nodes based on the value of one variable
5. Repeat steps 2- 4 on each of the nodes until stopping criteria (when data cannot be split any more) is met.
6. Predict the class for the target variable in each terminal node.
7. Prune the tree as appropriate.

**For more details, refer to**

1. http://dni-institute.in/blogs/cart-algorithm-for-decision-tree/
2. https://www.datasciencecentral.com/profiles/blogs/introduction-to-classification-regression-trees-cart

### **Chi-square Automatic Interaction Detection (CHAID)**

CHAID was a technique created by Gordon V Kass in 1980 for discovering relationships between a categorical target variable and other categorical predictor variables. CHAID, a predictive model to help determine how variables best merge to explain the outcome in the given target variable.

CHAID is non-parametric model.

CHAID is useful when the data set has too many categorical variables and we want to discover the relationship between response variable and the predictor variables.

In practice, CHAID is often used in the context of direct marketing to select groups of consumers and predict how their responses to some variables affect other variables, while other applications were in the field of medical and psychiatric research.

#### **Construction of the CHAID tree**

CHAID algorithm subdivides the data set into a series of subgroups that

1. Share similar characteristics towards a specific target variable and that

2. Maximizes the ability to predict the value of the target variable.

Chi-square tests are applied at each of the stages in building the CHAID tree to ensure that each branch is associated with a statistically significant predictor of the response variable.

For more details, refer to the following links:

1. https://www.statisticssolutions.com/non-parametric-analysis-chaid/
2. https://www.fieldglobal.com/index.php/chaid-analysis
3. https://select-statistics.co.uk/blog/chaid-chi-square-automatic-interaction-detector/
4. <https://en.wikipedia.org/wiki/Chi-square_automatic_interaction_detection>

# **cart model**



## **Example 1**

Dataset pima-indians-diabetes.data is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.

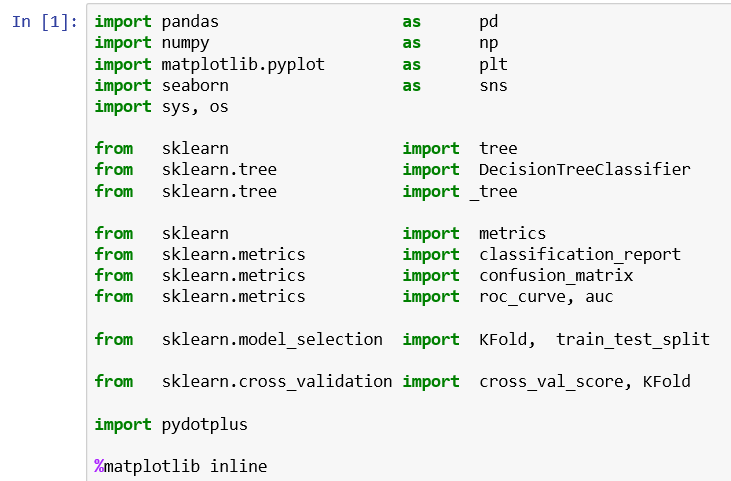
The objective is to predict based on diagnostic measurements whether a patient has diabetes. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Source: Kaggle https://www.kaggle.com/uciml/pima-indians-diabetes-database

|  |  |  |
| --- | --- | --- |
| **SlNo** | **Variable** | **Description** |
| 1 | Pregnancies | Number of times pregnant |
| 2 | Glucose | Plasma glucose concentration - 2 hours in an oral glucose tolerance test |
| 3 | BloodPressure | Diastolic blood pressure (mm Hg) |
| 4 | SkinThickness | Triceps skin fold thickness (mm) |
| 5 | Insulin | 2-Hour serum insulin (mu U/ml) |
| 6 | BMI | Body mass index (weight in kg/(height in m)^2) |
| 7 | DiabetesPedigreeFunction | Diabetes pedigree function |
| 8 | Age | Age (years) |
| 9 | label | Class variable (0 or 1) |

**Solution**

**1) Load required modules**

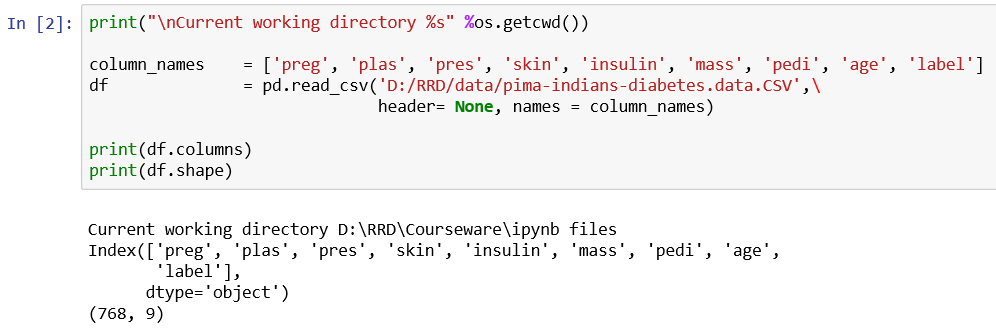
****

**We require the following modules:**

* + 1. pandas for reading a csv file, perform data manipulations etc.
    2. numpy for performing mathematical computations
    3. matplotlib.pyplot and seaborn for visualization
    4. os and sys to get system information
    5. sklearn for statistical analysis
    6. sklearn.tree for building CART model
    7. sklearn.metrics for calculation of metrics such as confusion matrix, aurox etc. for the model
    8. sklearn.model\_selection for splitting the data into train and test and also for K fold splitting
    9. pydotplus for showing the CART tree

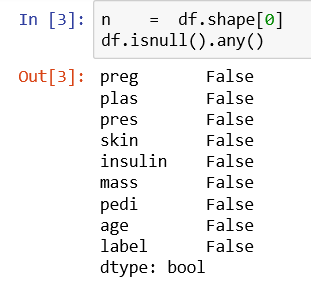
**Note: We use %matplotlib inline to display the images in the note book itself.**

**2) Read data**

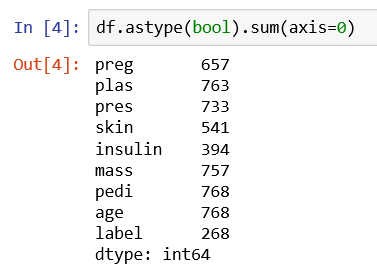
****

We have 768 observations and nine variables.

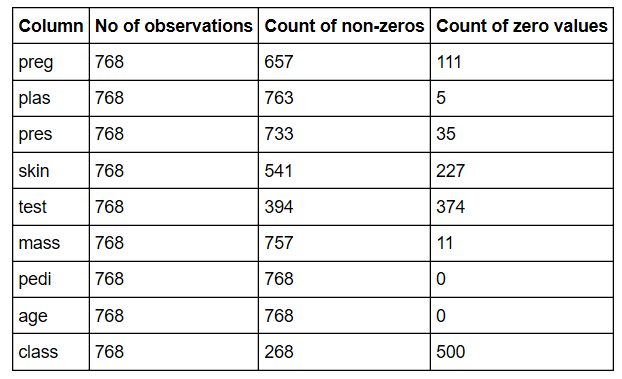
**3) Check for null or zero values**



**We observe that there are no missing values in the data. Let us get count of zero values in the data frame.**

****

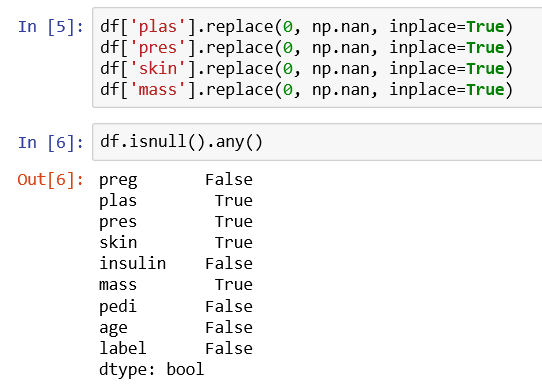
**We have found there are 0 values for the above columns. To get the value of zero counts, simply subtract these numbers from 768.**

****

**Let us analyze each column.**

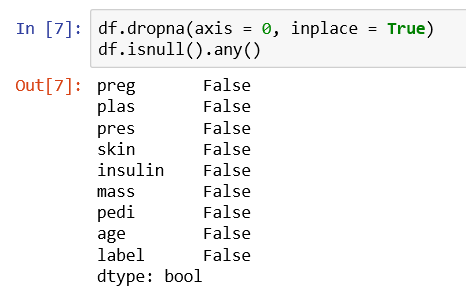
1. There are 111 zero values for the column, preg c Number of times pregnant. These zero values make sense since the patient was not pregnant at all at any point in time.
2. There are 5 zero values for the column, plas - which represents Plasma glucose concentration - 2 hours in an oral glucose tolerance test. No living person can have zero plasma. So we need to remove them or impute.
3. There are 35 zero values for the column, pres - which represents Diastolic blood pressure (mm Hg). No living person can have zero blood pressure. So, we need to remove them or impute.
4. There are 227 zero values for the column, skin - which represents Triceps skin fold thickness (mm)
5. No living person can have zero skin thickness. So, we need to remove them or impute.
6. There are 374 zero values for the column, test - which represents 2-Hour serum insulin (mu U/ml)
7. No living person can have zero serum insulin blood sugar. So we need to remove them or impute.
8. There are 11 zero values for the column, mass - which represents Body mass index (weight in kg/(height in m)^2) No living person can have zero BMI. So we need to remove them or impute.
9. There are 500 zero values for the column, class, which is a target variable. There are 268 diabetic people and 500 non-diabetic people. No need for any imputation.

**Convert zero as NA**

****

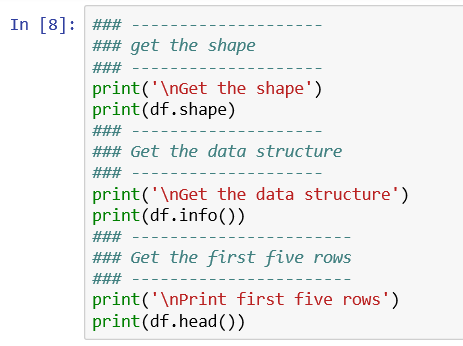
**We observe that NAs appear in the columns: plas, pres, skin and mass.**

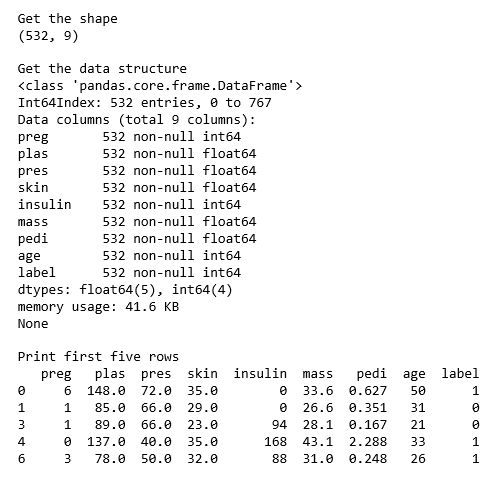
**We shall drop the rows with missing values instead of imputing them.**

****

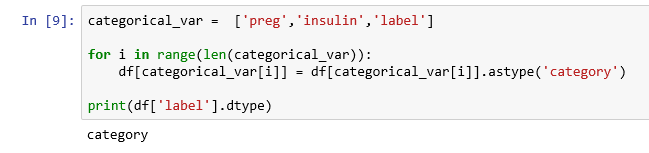
Now there are no missing values in the data frame, data. Data is ready for model building.

**4) Understand the data structure**

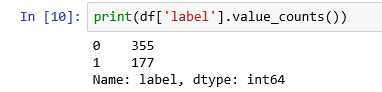
****

****

We have 532 observations and 9 numerical variables and only three variables ( pregnant, insulin and class) need to be converted to a categorical variables.

****

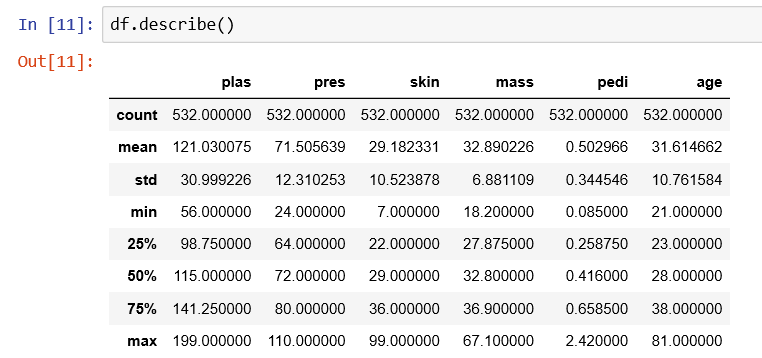
**Get target variable, class distribution**

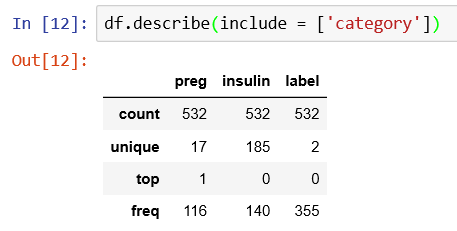
****

We have got only 532 observations with arguably valid values even though the original data has 768 observations.

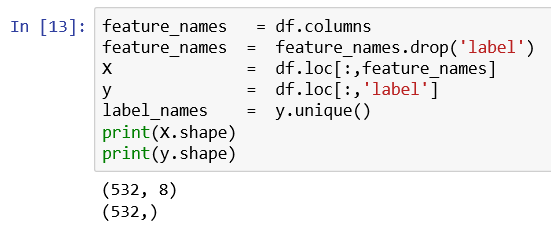
Class distribution in the target variable, class is not balanced. There are 177 observations for class 1 (diabetic) and 355 observations for class 0 (non-diabetic).

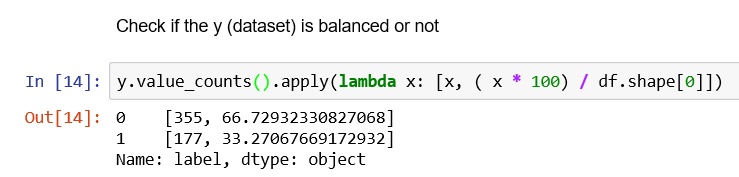
**Get the basic summary statistics**



****

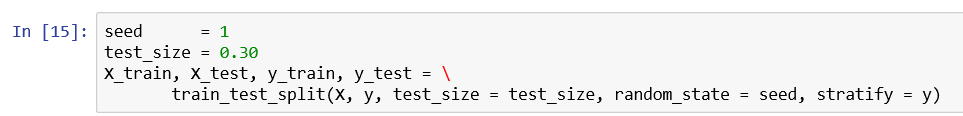
**Create X, independent variable and y dependent variable**

****

****

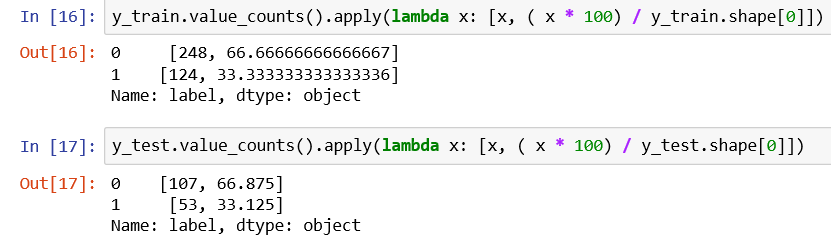
**The data set, y is not balanced and is having the class proportion of 67% : 33% .**

**Split randomly the data into training and test data set in the same proportion in the original data**

****

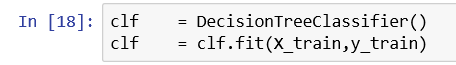
**Note:** We had set the seed to enforce repeatability and set stratify parameter as y (the column name of the target variable) to split the data using stratified random sampling method.

**Check the class proportion in both training dataset and test dataset.**



Same class proportion of 67%:33% is maintained in both training data and test data set.

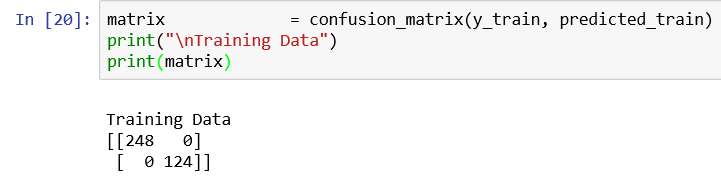
**Build a CART model (no pruning) using the training data set**

****

**Predict the response for the training data set**

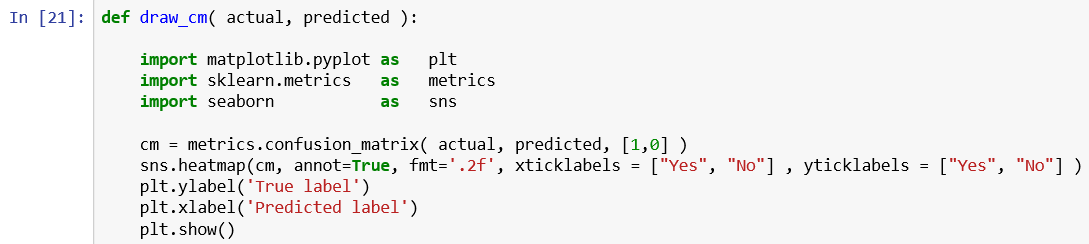
****

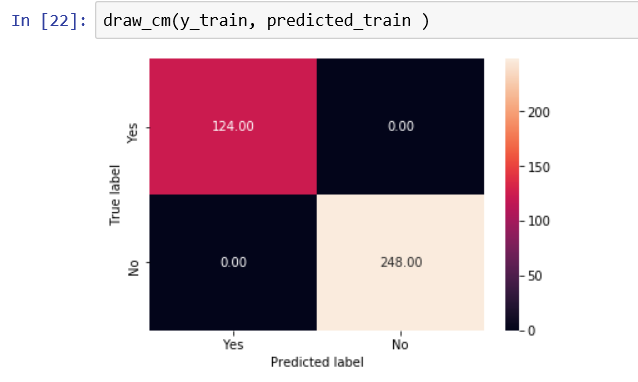
**Create the confusion matrix for the training data**

****

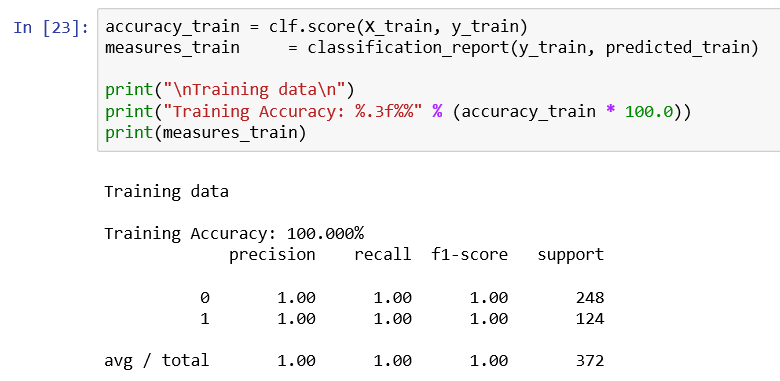
**Show the Confusion Matrix visually**

*First define a function to draw the confusion matrix image*

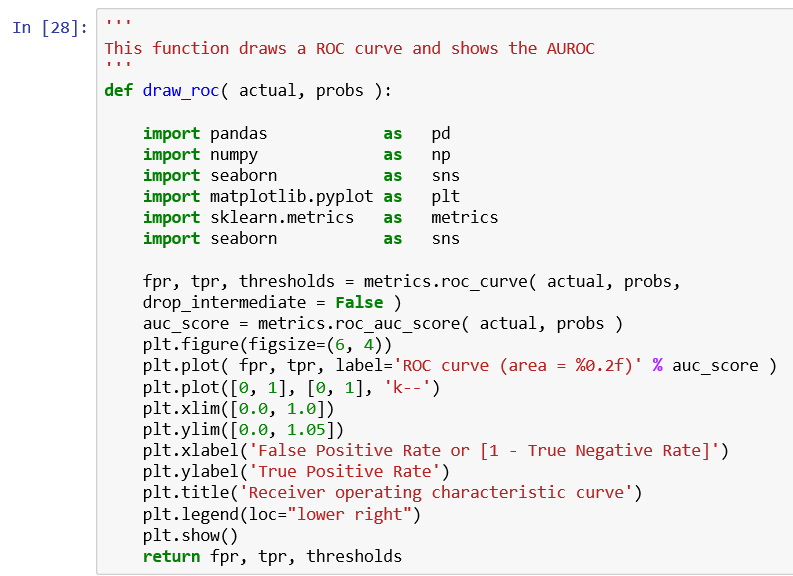
****

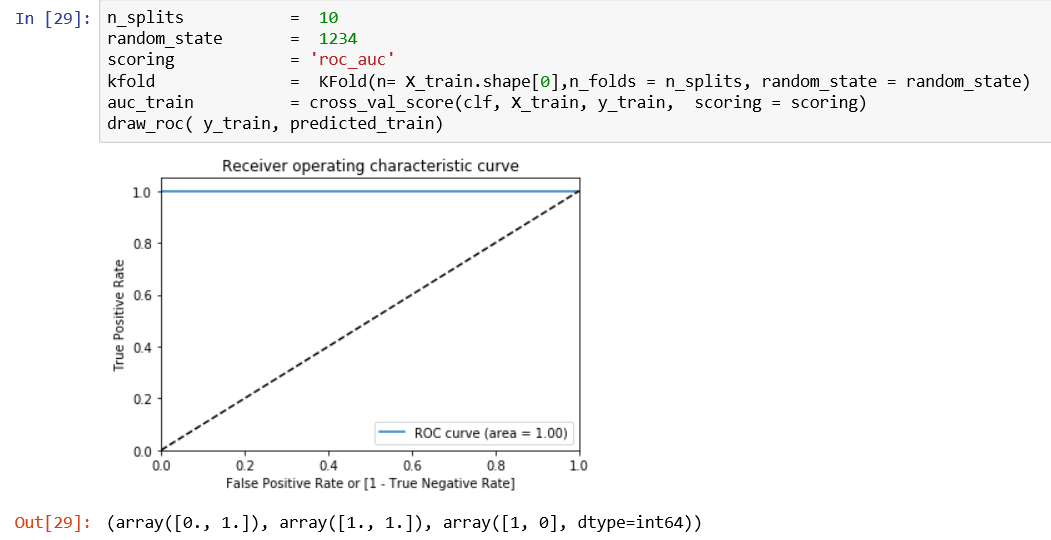
****

**Get metrics for the training data**

****

**Define a function for drawing ROC curve**

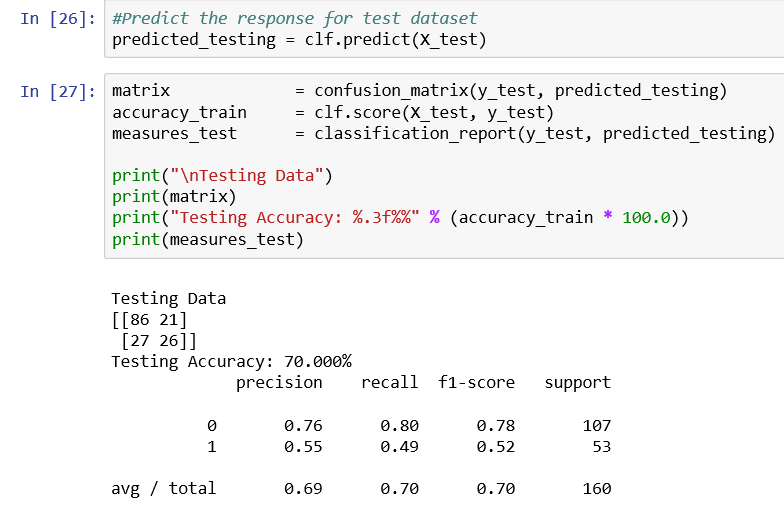
****

****

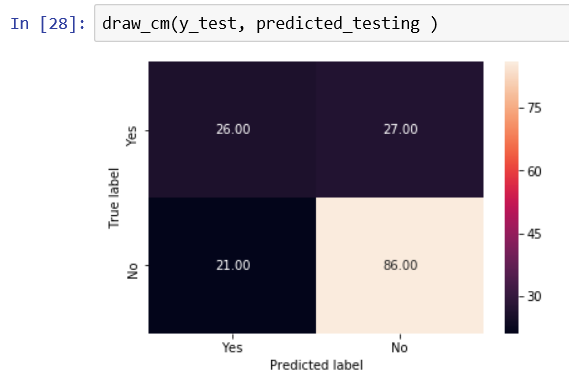
*We have got too good metrics for the training dataset (possibly, overfit) as below:*

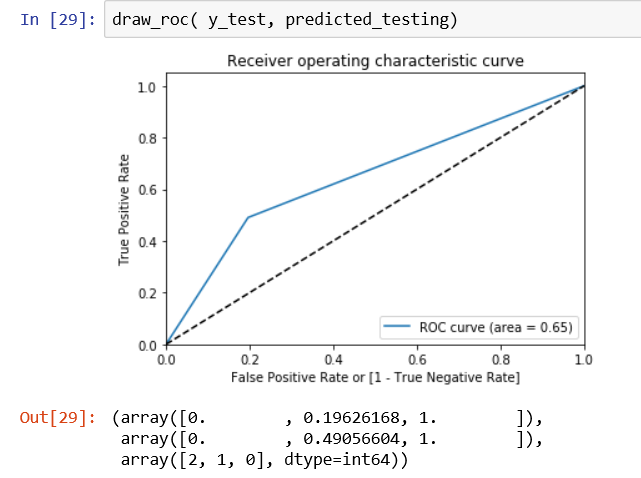
*Precision and Recall as 1 for both classes, 100% as accuracy ratio and auroc as 100%.*

**Performance metrics for test data**

****

**Visual representation of Confusion Matrix and ROC**

****

****

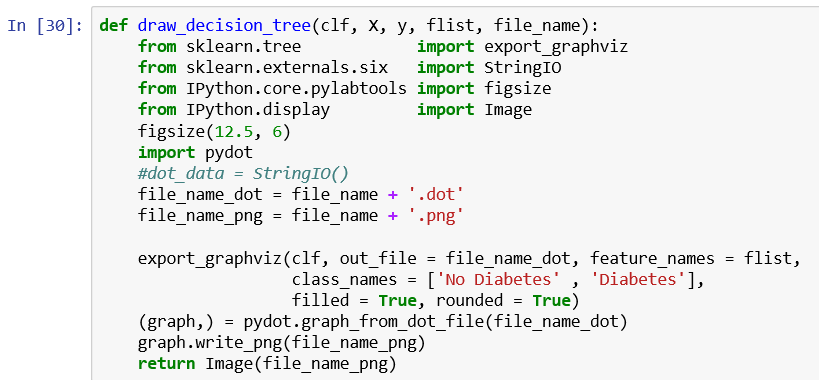
**Compare performance metrics of training data with test data**

|  |  |  |
| --- | --- | --- |
| **Metrics** | **Training dataset** | **Testing dataset** |
| **Precision for class 1** | **1** | **0.55** |
| **Recall for class 1** | **1** | **0.49** |
| **Accuracy Ratio** | **100%** | **70%** |
| **AUROC** | **1** | **0.65** |

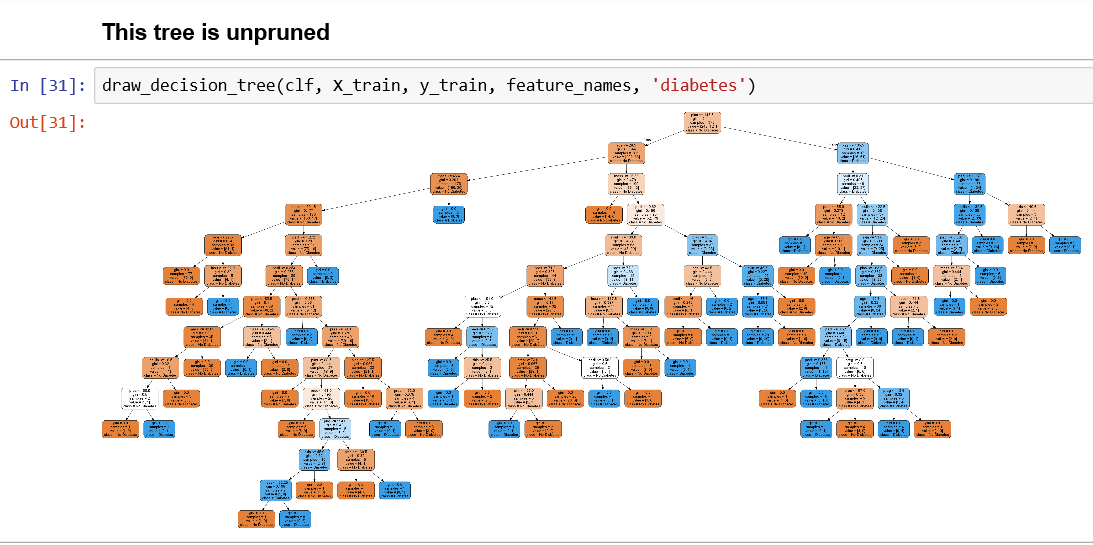
**There is a sharp decline in the metrics for the testing data when compared to training dataset indicating the model is a overfitting.**

**Let us show the CART tree structure of the model built**

Define a function to draw the Tree

****

*Since we have already built a tree model without pruning, let us use the same.*

****

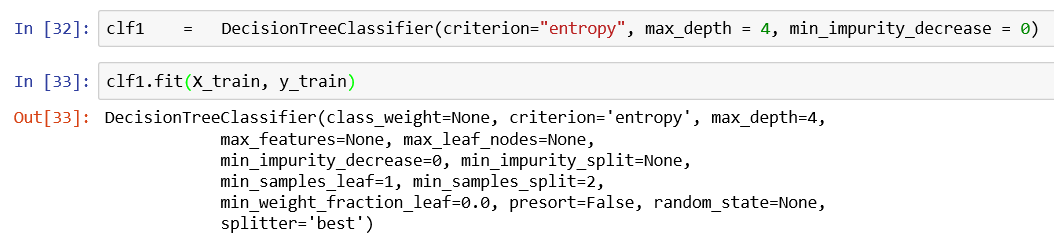
*We observe that the above tree is unexplainable and difficult to comprehend. Let us improve the graph by pruning.*

**Prune the CART Tree**

* Criterion parameter allows us to use the different-different attribute selection measure. Allowed criteria are “gini” (default) for the Gini index and “entropy” for the information gain.
* Splitter parameter allows us to choose the split strategy. Allowed strategies are “best” to choose the best split and “random” to choose the best random split.
* Max\_depth parameter allows us to state the maximum depth of the tree. If None, then nodes are expanded until all the leaves contain less than min\_samples\_split samples. The higher value of maximum depth causes overfitting, and a lower value causes underfitting.

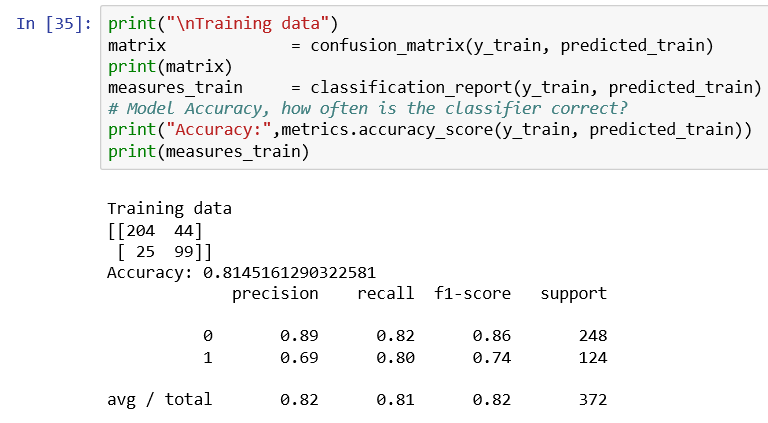


**Fit and predict responses for training data**

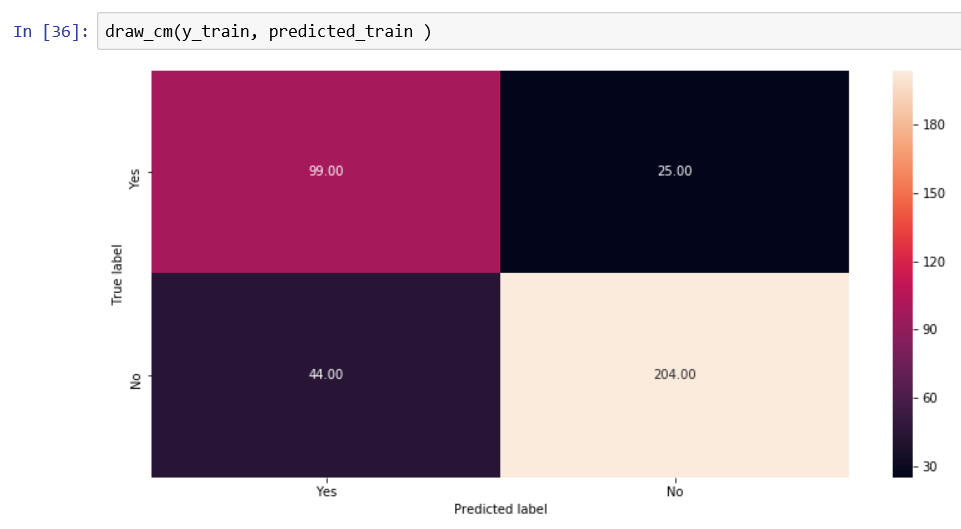
****

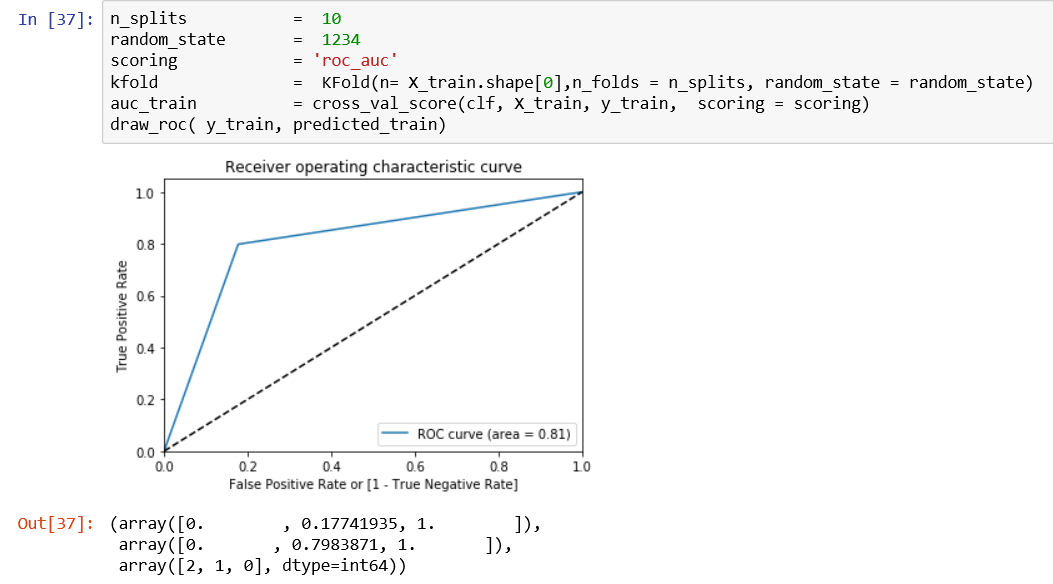
****

**Performance metrics for training data**

****

**Visual Confusion Matrix**

****

****

**We observe that pruning has improved the model performance because of the following metrics:**

1) Precision for Class 1 : 0.69

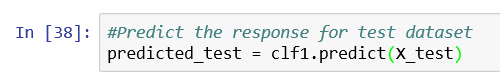
2) Recall for Class 1: 0.80

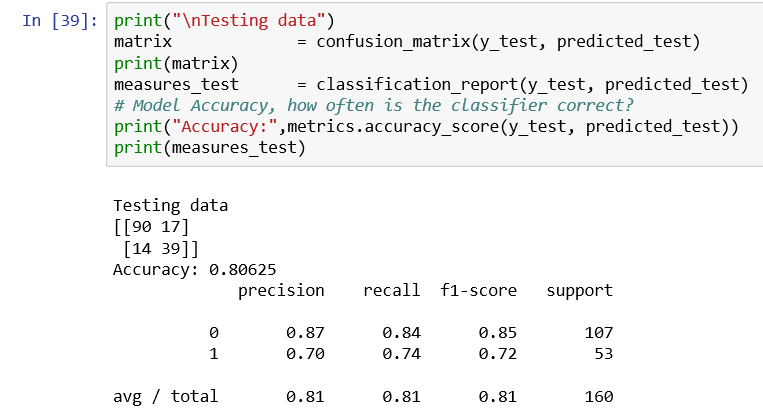
3) Accuracy Ratio: 81.45%

4) AUROC: 0.81

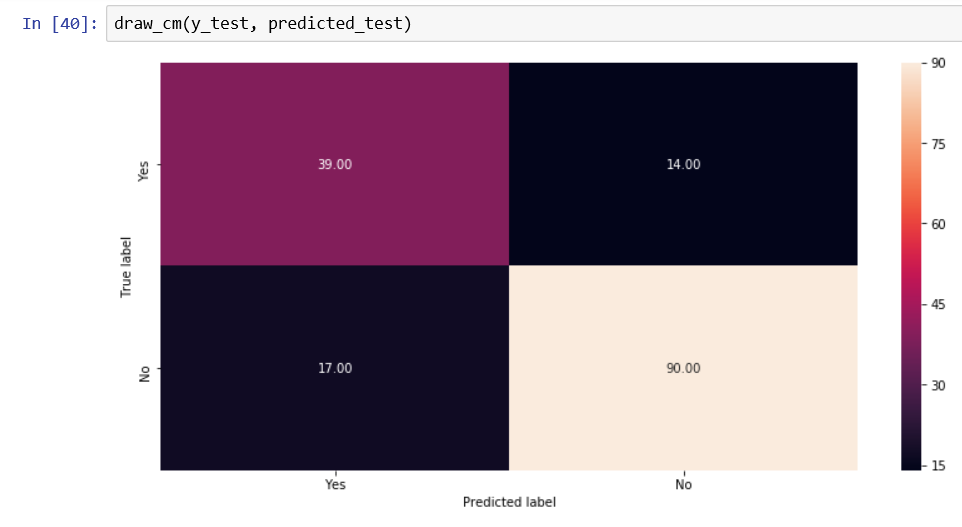
**Performance metrics for testing data**

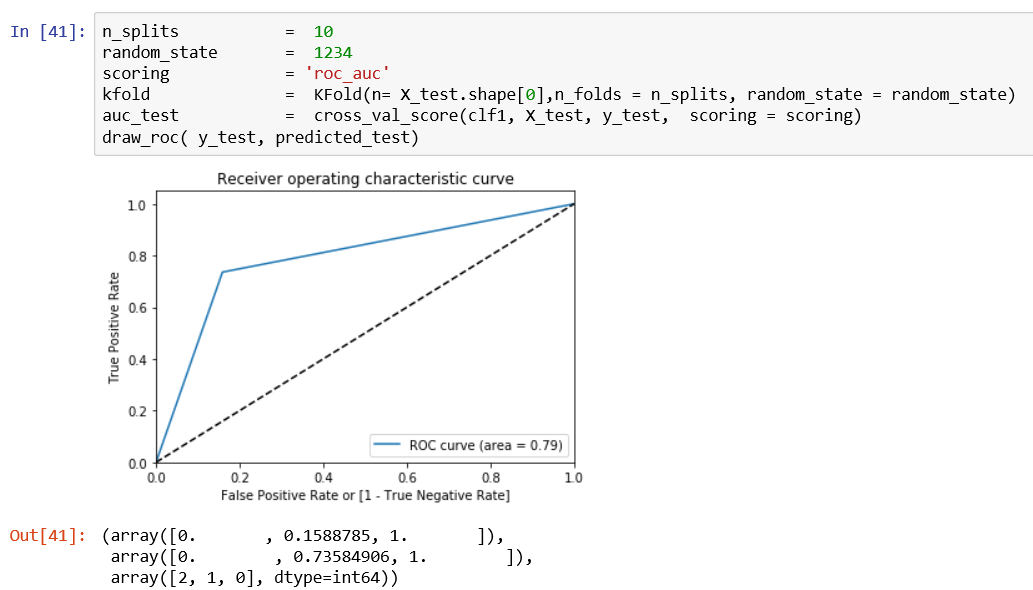
Predict the response for the test data

****

****

**Visual Confusion Matrix**

****

****

**Compare performance metrics of training data with test data**

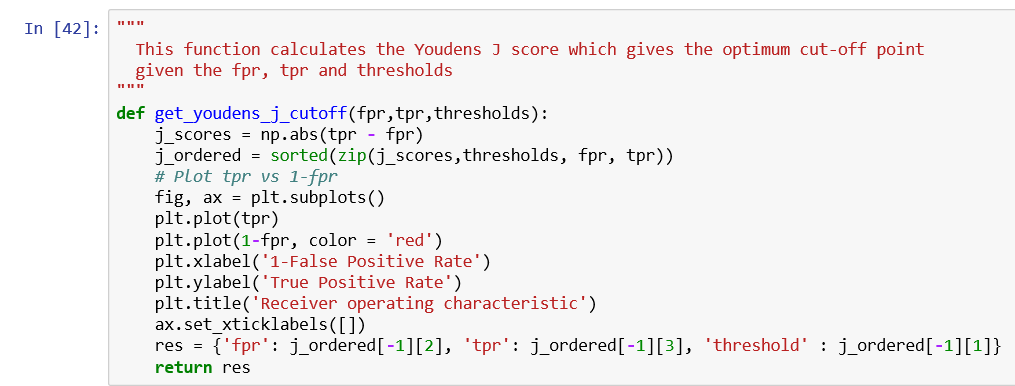
|  |  |  |
| --- | --- | --- |
| **Metrics** | **Training dataset** | **Testing dataset** |
| **Precision for class 1** | **0.69** | **0.70** |
| **Recall for class 1** | **0.80** | **0.74** |
| **Accuracy Ratio** | **81.45%** | **81.45%** |
| **AUROC** | **0.81** | **0.8063** |

**There is no great difference between the metrics for the testing data when compared to training dataset.**

**Find the optimum cut-off**

We know the data set is not balanced and hence the standard cut-off of 50% for creating the confusion matrix will not hold good. We need to find the optimum cut-off to divide the data set into two classes.

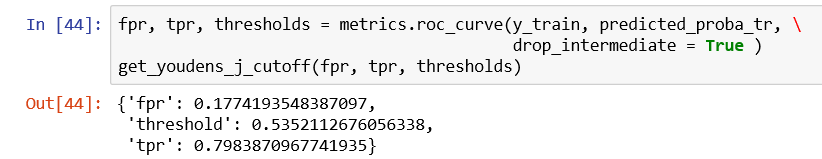
**Write a function to calculate Youden’s J score which gives the optimum cut-off.**



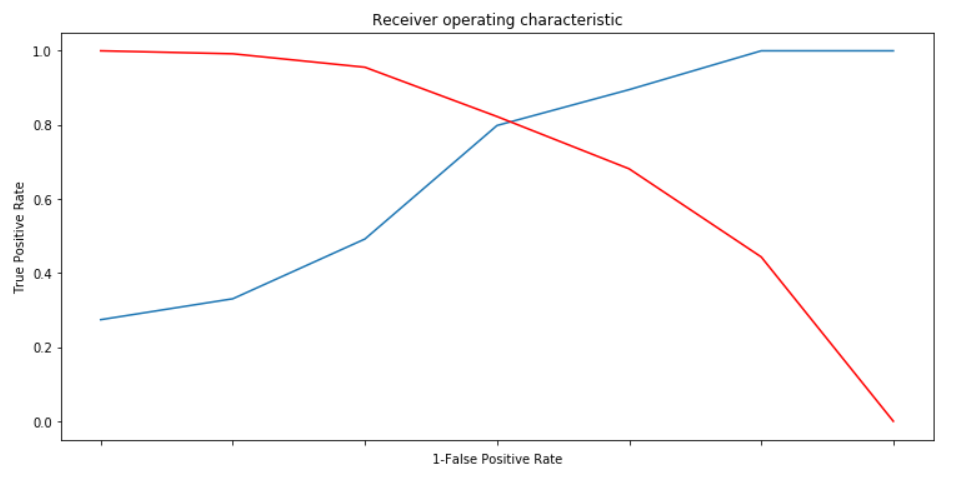
**Get the predicted probabilities for the target variable.**

****

**Apply Youden’s J score logic**

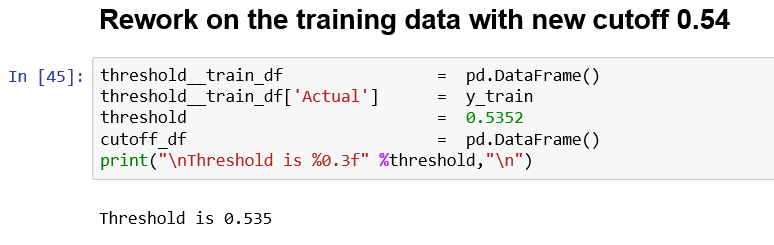
****

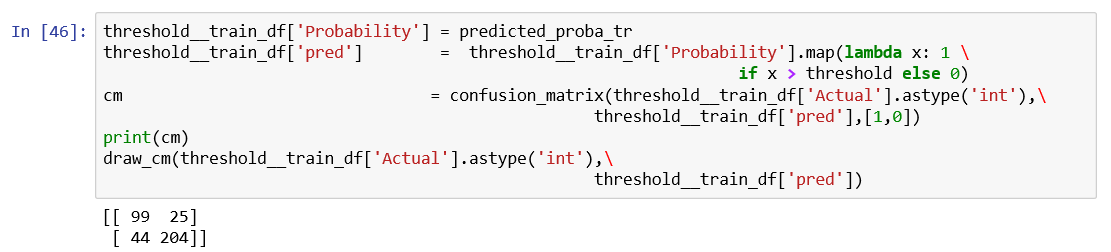
**Visual representation of True Positive Rate and False Positive Rate at various cutoff**

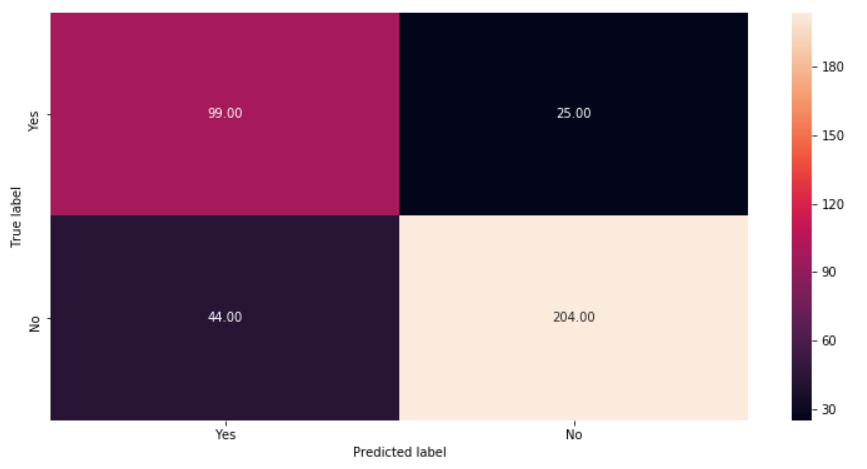
****

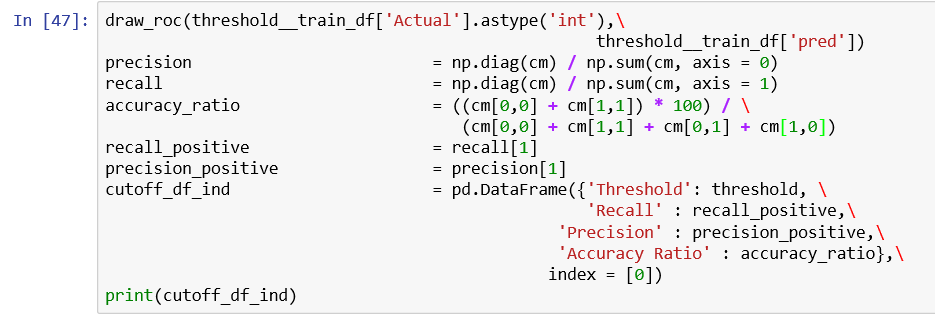
**So, the optimum cut-off is 0.54 instead of standard 0.50.**

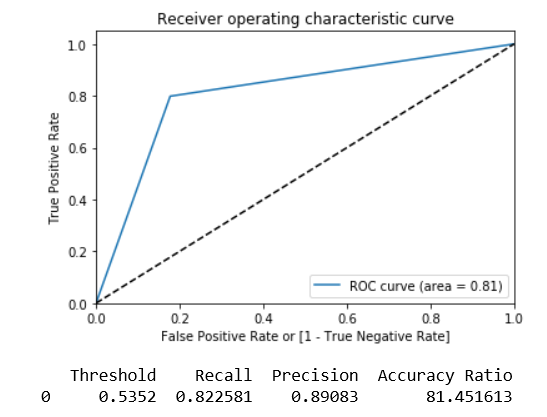
**Rework using the optimum cut-off on training data**

****

****

****

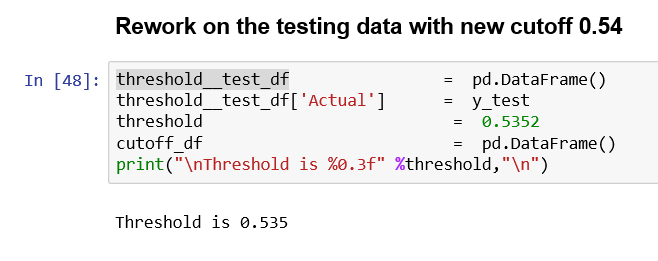
****

****

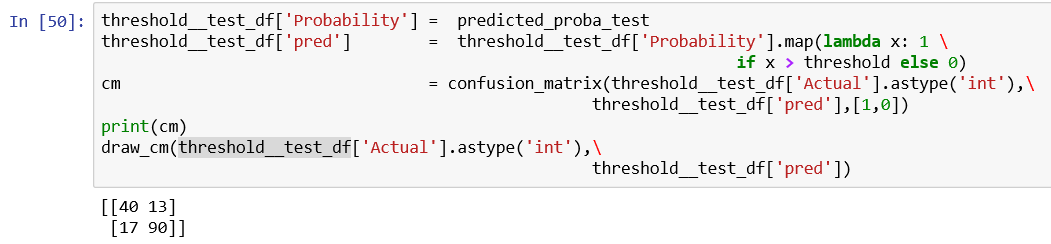
**We notice a marginal increase in the metrics for training data set after optimum cut-off point is used for classification as shown below:**

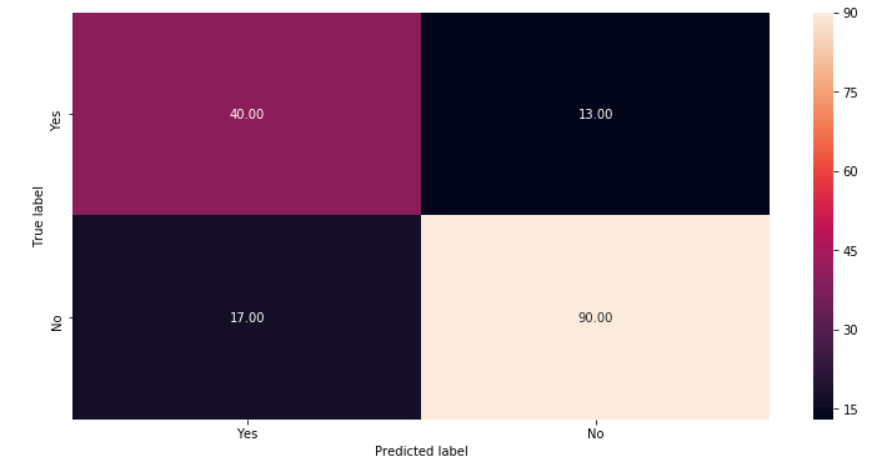
1. Precision is increased to 0.8908 from 0.69
2. Recall is increased to 0.8226 from 0.80
3. Accuracy Ratio is increased to 81.45 from 81.45
4. AUROC remains same at 0.81

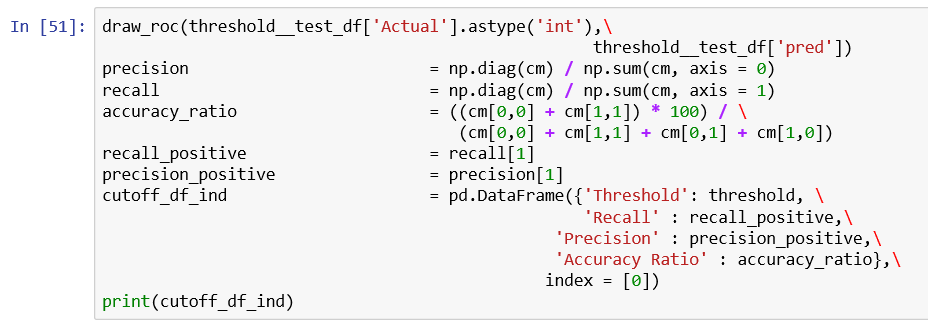
**Rework using the optimum cut-off on testing data**

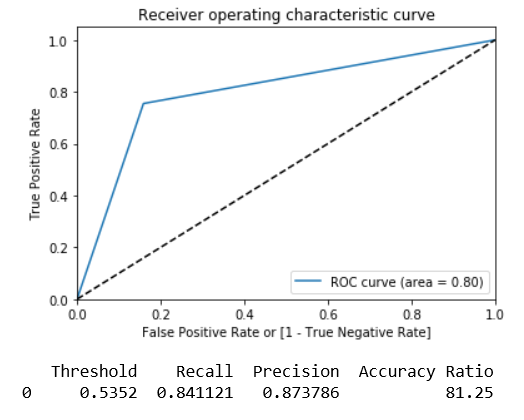
****

****





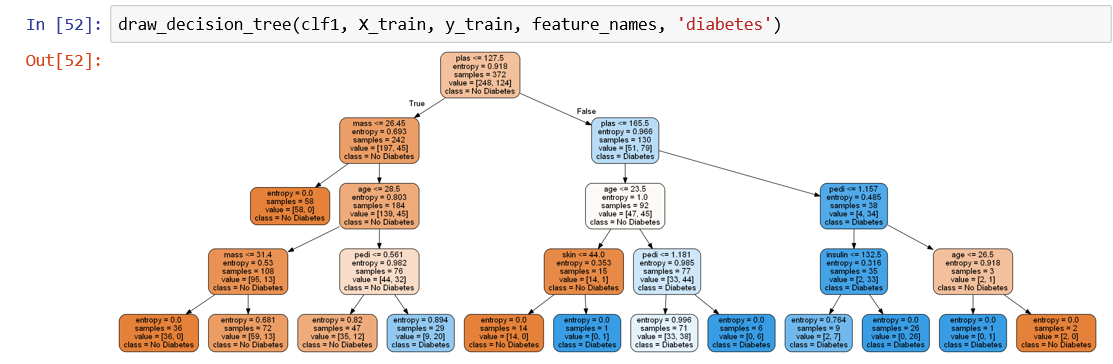
****

****

**We notice a good increase in the metrics for training data set after optimum cut-off point is used for classification as shown below:**

1. Precision is increased to 0.8737 from 0.70
2. Recall is increased to 0.8411 from 0.74
3. Accuracy Ratio is reduced slightly to 81.25 from 81.45
4. AUROC is reduced slightly to 0.80 from 0.8063

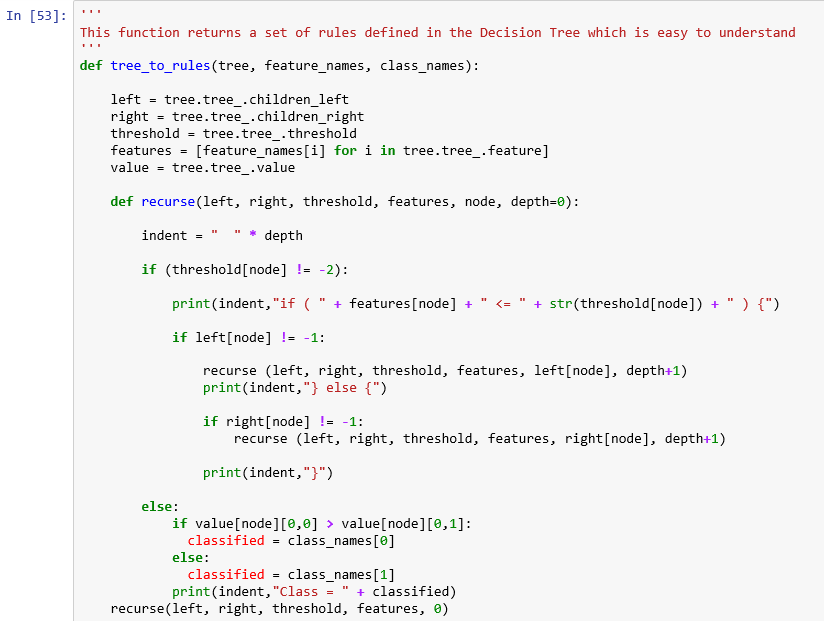
**Draw the CART tree**



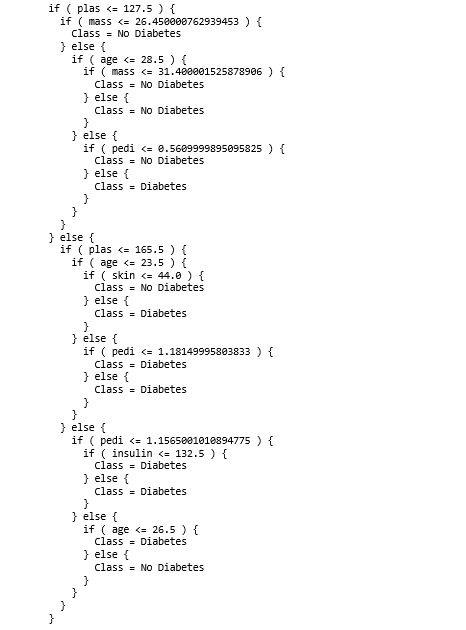
**Get the rules from the above tree**

**How do you interpret the tree or understand the rules from the above picture?**

**Define a function to extract the rules from the tree**

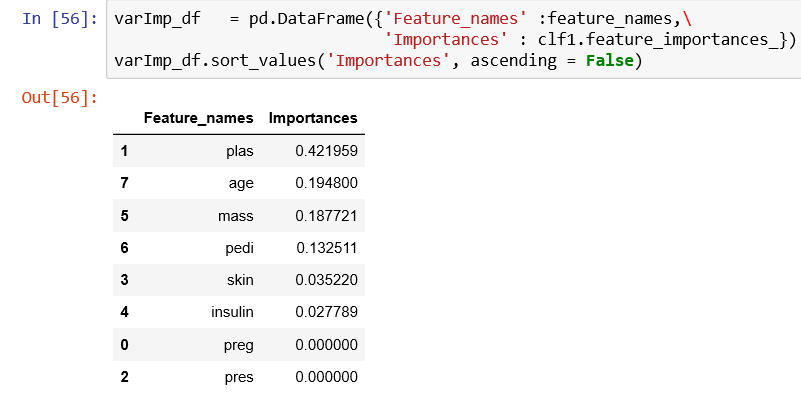


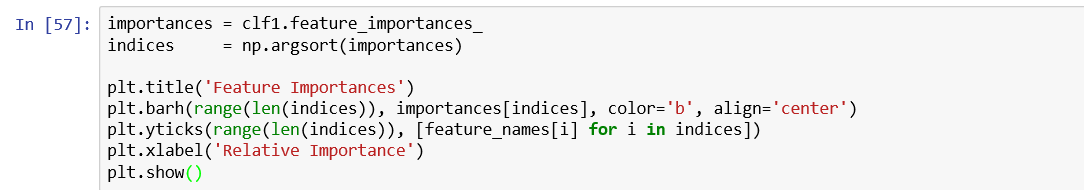


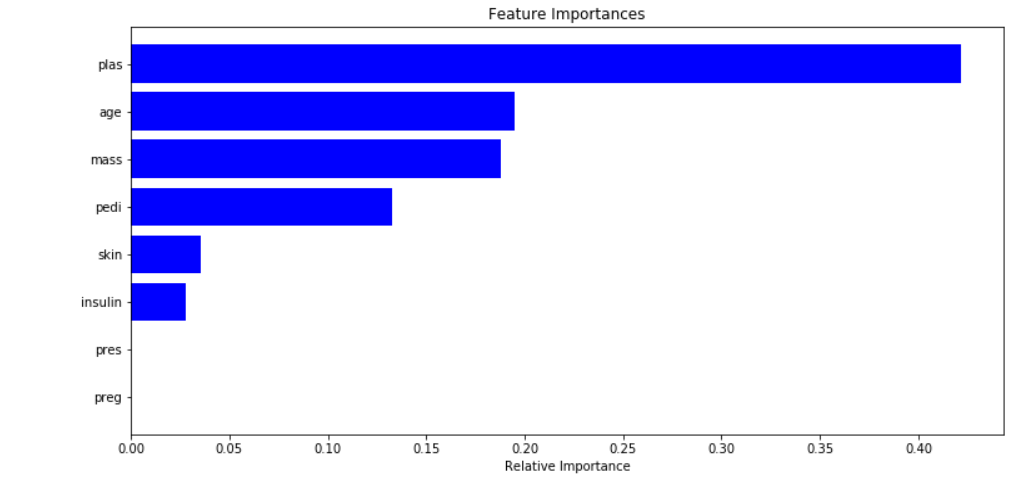
****

**Get important variables**

**Variable importance plot**

****

****

****

**According to the above variable importance plot, the top five variables,**

* **plas (Plasma glucose concentration - 2 hours in an oral glucose tolerance test),**
* **age,**
* **mass (Body Mass Index) ,**
* **pedi (Diabetes pedigree function) and**
* **skin (Triceps skin fold thickness (mm))**

**are important in predicting the target variable, class (diabetic or not).**