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 7. Regression models basics**

Table of Contents

[**1)** **Regression** 4](#_Toc10441337)

[**1.1.** **Introduction** 4](#_Toc10441338)

[**1.2.** **Application of regression** 5](#_Toc10441339)

[**1.3.** **Identification of problem** 5](#_Toc10441340)

[**1.4.** **Regression types** 6](#_Toc10441341)

[**1.4.1** **Linear Regression** 6](#_Toc10441355)

[**1.4.2** **Logistic regression** 10](#_Toc10441356)

[**1.4.2.1.** **Assumptions for logistic regression models** 11](#_Toc10441364)

[**1.4.2.2.** **Why linear regression is appropriate for binary target variable?** 11](#_Toc10441365)

[**2)** **Example 1 – Linear regression** 12](#_Toc10441367)

[**2.1** **Test the assumptions for linear regression** 18](#_Toc10441370)

[**2.1.1**  **No Outliers** 18](#_Toc10441374)

[**2.1.2**  **normality of the residuals** 20](#_Toc10441375)

[**2.1.3**  **No multi-collinearity** 21](#_Toc10441376)

[**2.1.4**  **No auto correlation** 24](#_Toc10441377)

[**2.1.5**  **absence of heteroscedasticity** 24](#_Toc10441378)

[**2.1.6** L**inearity** 25](#_Toc10441379)

[**2.2** **Build the model** 26](#_Toc10441380)

[**2.3** **Evaluate model performance** 28](#_Toc10441385)

[**2.3.1.** **Prediction Accuracy** 28](#_Toc10441387)

[**3)** **Example 2 – Logistic regression** 30](#_Toc10441388)

[**3.1** **Logistic Regression Assumptions** 32](#_Toc10441389)

[**3.2**  **Model Building** 36](#_Toc10441390)

[**4)** **Appendix** 42](#_Toc10441391)

# **regression**

## **Introduction**

|  |  |
| --- | --- |
| * Linear regression is a mathematical technique for finding the straight line that best fits the values of a linear function, plotted on a scatter graph as data points. - Business Dictionary\* * Response variable (aka dependent or outcome or target) is the variable of focus in a research study. * Predictor variable (aka independent or explanatory) is the variable that explains the variation in the response variable, and it might affect the response variable. |  |

* For example, the response variable is volume of Sales in thousands on a given day in an online store and the predictor variable is the advertisement expenses.
* Focus of the regression analysis is on the relationship between a response variable and one or more predictor variables. To be more specific, this helps one to understand how the typical value of the response variable changes when any one of the predictor variables is varies, keeping other predictor variables constant.

**Reference\*:**

* <http://www.businessdictionary.com/definition/linear-regression.html>
* <https://en.wikipedia.org/wiki/Regression_analysis>

## **Application of regression**

**Primary uses for regression in business:**

1. **Forecasting:** This will predict events that have yet to occur. For example, number of people who will subscribe to a cell phone services, number of people who will watch a TV program.
2. **Optimization of business process:** For example, it is important for a cell phone company to know the relationship between the number of calls made to the call center by customers and number of customers churned.

## **Identification of problem**

* Before doing the regression analysis, one must review the literature to develop a deep understanding of the relevant variables, their relationships, and the expected coefficient signs and effect magnitude. Refer http://statisticsbyjim.com/regression/regression-analysis-tips/ for more details.
* To determine the response variable, Pedhazur\* suggests that the variable have acceptable measurement qualities (i.e., reliability and validity).
* Goal of the model selection is to minimize the number of predictors which account for the maximum variance in the response variable.

Refer https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2845248/#B15 for more details.

**Reference:**

\* Pedhazur EJ. Multiple Regression in Behavioral Research. 3rd. ed. Fort Worth, TX: Harcourt Brace College Publishers; 1997.

## **Regression types**

Based on the type of dependent variable, we have two types of regressions.



### **Linear Regression**

* When the dependent variable is continuous, independent variable(s) can be continuous or discrete, nature of regression line is linear.
* Linear Regression establishes a relationship between dependent variable (Y) and one or more independent variables (X) using a best fit straight line (also known as regression line).
* 
* Let Y denote the dependent variable, X1, X2, ... Xk are independent variables.
* The value of Y at time t, in the sample data is determined by the linear equation:
* β0, β1,β2, ...βk are constants and εs are independent and identically distributed normal random variables with mean 0.
* β0 is called the intercept of the model.
* β0, β1,β2, ...βk are regression coefficients.
* The betas together with the mean and standard deviation of the epsilons are the parameters of the model.

The equation for predicting Yt from the corresponding X values is given by:



where the bs are the estimates of the coefficients of betas.

Example: Regression equation between hours studied and marks scored

Marks = 38.9988127 + 0.876757168 \* Hours Studied.

#### **Assumptions for standard linear regression models**

1. **Linearity**

This implies that the mean of the response variable is a linear combination of the parameters (regression coefficients) and the predictor variables. Because the predictor variables are treated as fixed values, linearity is really only a restriction on the parameters. Polynomial regression uses linear regression to fit the target variable as an arbitrary polynomial function of the predictor variables and it is too powerful to overfit the data.

1. **Constant variance (a.k.a. homoscedasticity)**

This means that different values of the response variable have the same variance in their errors regardless of the values of the predictor variables.

1. **Independence of errors (No auto correlation)**

This assumes that the errors of the target variable are uncorrelated with each other. Auto correlation occurs when the residuals (difference between observed and estimated values for the target variable) are not independent from each other.

1. **Normality of the residuals**

The residual errors are assumed to be normally distributed.

1. **Lack of perfect multi-collinearity in predictor variables**

When we have two or more perfectly correlated predictor variables, we have the problem of multi-collinearity. This problem is relevant to Ordinary Least Squares method and taken care of regularization methods such as Ridge or Lasso.

1. **Lack of outliers in the data**

Linear Regression is extremely sensitive to outliers as It affects the regression line and eventually the forecasted values.

For more details, refer to the following links:

* https://en.wikipedia.org/wiki/Linear\_regression#Assumptions
* <https://machinelearningmastery.com/linear-regression-for-machine-learning/>
* <https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/>
* https://www.statisticssolutions.com/assumptions-of-linear-regression/

#### **types of linear regression models**

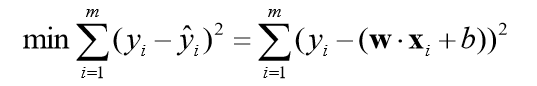
Assume the relationship between X and y is linear,

Let

* w represents the coefficients
* b an intercept
* ε the error term.
*  be the estimate of yi

1. **Ordinary Least Squares**

This method seeks to minimize the sum of the squared residual errors.



<https://en.wikiversity.org/wiki/Least-Squares_Method>

https://cs.adelaide.edu.au/~chhshen/teaching/ML\_SVR.pdf

1. **Gradient Descent**

This method iteratively minimizes the residual error of the model on the training data. This method known as Gradient Descent works by starting with random values for each coefficient.

The sum of squared errors is calculated for each pair of input and output values. A learning rate is used as scale factor and the coefficients are updated in the direction towards minimizing the error. This process is repeated until a mi mum sum squared error is achieved or no further improvement is possible.

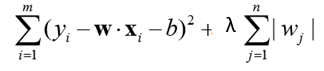
https://machinelearningmastery.com/linear-regression-for-machine-learning/

1. **Regularization**

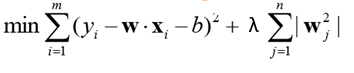
These methods not only minimize the sum of squared error of the model but also reduces the complexity of the model.

**A few regularization methods:**

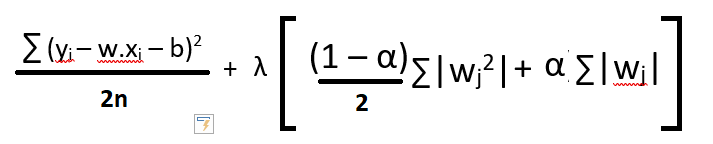
* Assume there is a relationship between x and y. Let w represents coefficients and b an intercept. Let λ be the parameter that adds penalty for number of features.
* **Lasso Regression** (L1 regularization): Here OLS is modified to minimize the absolute sum of the coefficients



* **Ridge Regression** (L2 regularization): Here OLS is modified to minimize the squared absolute sum of coefficients.

****

* The main difference between the above techniques is that Lasso shrinks the less important feature’s coefficient to zero. This results in removal of some of the features altogether. This works well for feature selection when we have many features.
* **ElasticNet Regression:** Here OLS is modified to minimize the complexity of the regression model (magnitude and number of regression coefficients) by penalizing the model using the sum squared coefficient values (L2 norm) and sum absolute coefficient values (L1 norm).
* Elastic net is basically a combination of both L1 and L2 regularization. It uses both L1 and L2 penalty term, its equation looks like:



* Here, i ranges from 1 to n and j ranges from 1 to m
* In addition to setting a λ, we can tune α between 0 and 1

Refer:

* https://hackernoon.com/an-introduction-to-ridge-lasso-and-elastic-net-regression-cca60b4b934f
* https://datafai.com/2018/03/18/lasso-ridge-and-elastic-net-regularization/

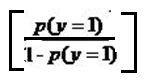
1. **Other models for regression**

* **K Nearest Neighbors:** This 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 by using the Minkwoski or Manhattan distance.
* **CART** uses the training data to select the best points to split the data in order to minimize a cost metric such as Mean squared error**.**
* **Support Vector Regression (SVR)**
* SVR finds a function f(x) with maximum deviation from the target, y
* f(x) = min ((1/2)|w2 |)subject to the constraints
* yi - wxi - b ≤ ε
* wxi + b -yi ≤ ε
* Refer to https://cs.adelaide.edu.au/~chhshen/teaching/ML\_SVR.pdf

**Page 90**

**file:///E:/Jason/ML/machine\_learning\_mastery\_with\_python.pdf**

### **Logistic regression**

* When the dependent variable is binary (0/ 1, True/ False, Yes/ No), nature of regression is binary logistic.
* When the dependent variable is multi-class, nature of regression is binary logistic.

Logit(p) = log

=

https://www.statisticssolutions.com/what-is-logistic-regression/

http://people.duke.edu/~rnau/Notes\_on\_linear\_regression\_analysis--Robert\_Nau.pdf

Example: What genre of movie such as Romantic, Comedy, Thriller, most of the movie goers in the age group 21- 15 prefer?



### **Assumptions for logistic regression models**

1. Binary logistic regression requires that the dependent variable is a binary variable that is measured on a dichotomous scale. Ordinal logistic regression requires that the dependent variable to be ordinal.
2. The logistic regression requires there to be little or no multi-collinearity among the independent variables.
3. The logistic regression requires the observations to be independent of each other (not from repeated measurements or matched data).
4. Logistic regression assumes the relationship between independent variables and log odds to be linear.
5. Logistic regression requires a large sample size, at least 10 cases with the least frequent outcome for each independent variable in the model.

For more details, refer to https://www.statisticssolutions.com/assumptions-of-logistic-regression/

### **Why linear regression is appropriate for binary target variable?**

Logistic regression predicts the probability of an outcome that can only have two values.

1. Linear regression model will predict values outside the acceptable range of (0 - 1).
2. The data will have only one of the two possibilities for the target variable and the residuals will not be normally distributed about the fitted line.

For more details, refer to <http://www.appstate.edu/~whiteheadjc/service/logit/intro.htm>



# **Example 1 – Linear regression**

Concrete is the most important material in civil engineering. The concrete compressive strength is a highly nonlinear function of age and ingredients. We shall estimate the Complete compressive strength(CRS) using:

**Data Description**

|  |  |
| --- | --- |
| Variable | **Description** |
| Cement in kg | Cement in a m3 mixture |
| Blast Furnace Slag in kg | Blast Furnace Slag in a m3 mixture |
| Fly Ash in kg | Fly Ash in a m3 mixture |
| Water in kg | Water in kg in a m3 mixture |
| Superplasticizer in kg | Superplasticizer in in a m3 mixture |
| Coarse Aggregate in kg | Coarse Aggregate in a m3 mixture |
| Fine Aggregate in kg | Fine Aggregate in a m3 mixture |
| Age in Day | Days (1-365) |

**Source:**

**Original Owner and Donor: Prof. I-Cheng Yeh**

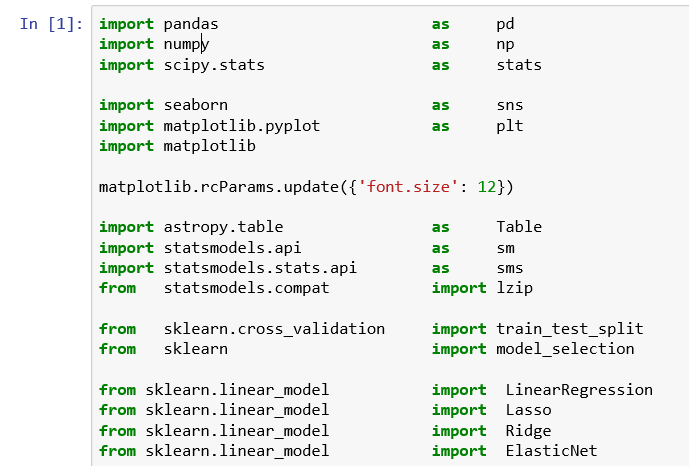
**Department of Information Management**

**Chung-Hua University,**

**Hsin Chu, Taiwan 30067, R.O.C.**

**https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength**

1. **Import the required libraries**

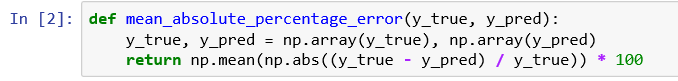


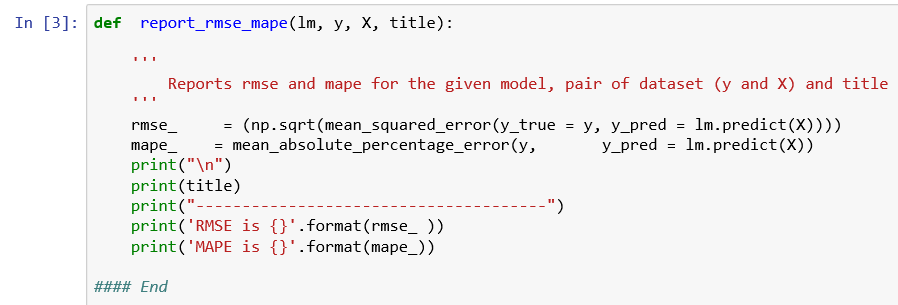


**We shall use**

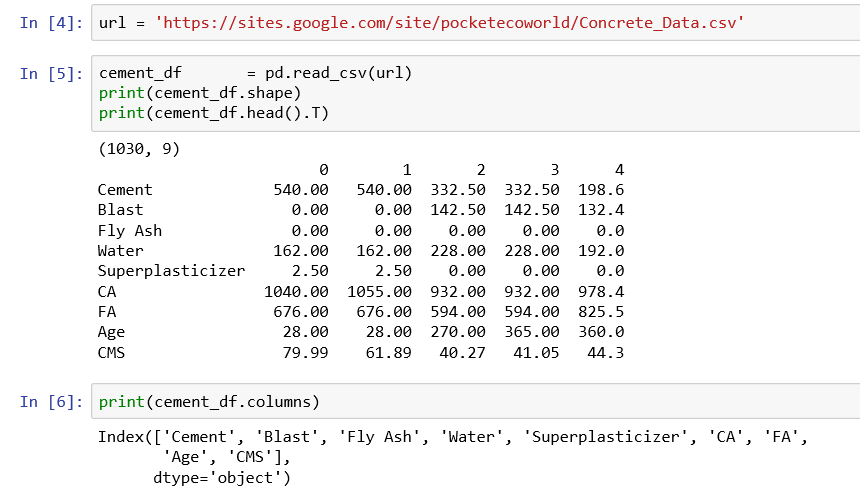
1. pandas for accessing data file and performing manipulations on data frames.
2. numpy for performing mathematical operations
3. matplotlib.pyplot and seaborn for visualization
4. statsmodel and sklearn for using statistical and machine learning techniques

**Define a function for calculating RMSE and MAPE.**

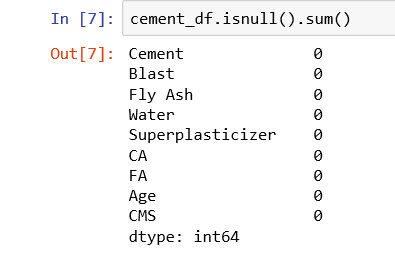




1. **Load the data**

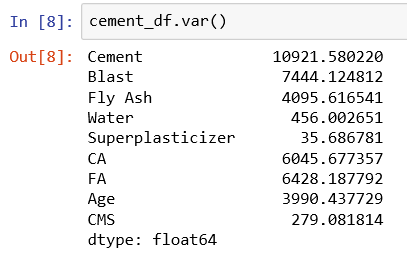
****

1. **Check for missing values**

****

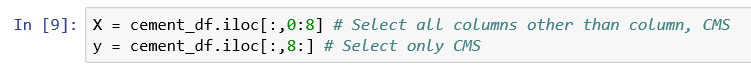
We conclude that there are no missing values.

1. **Identify and remove variables of near zero variance**

****

**We observe that there are no variables having zero or near zero variance..**

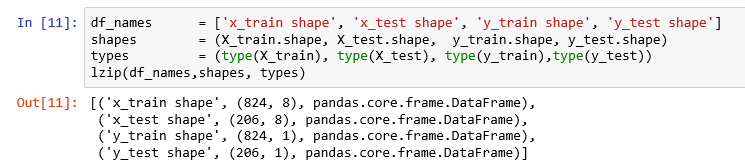
1. **Split the data into predictor and response variable, X and y**

****

1. **Split the data into training and test data**

* Slit the data into 80:20 ratio to create train and test data
* Set a random seed to ensure repeatability of the results
* There are 824 observations and 8 variables in the training dataset for X (predictor dataset)
* There are 206 observations and 8 variables in the testing dataset for X (predictor dataset).

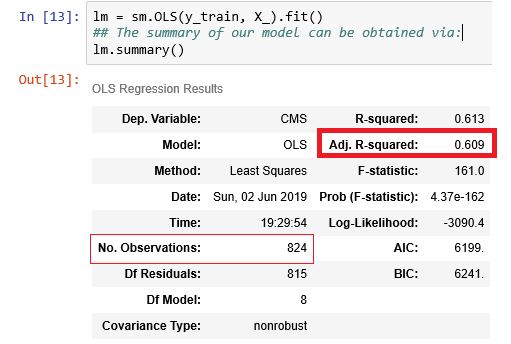
****

****

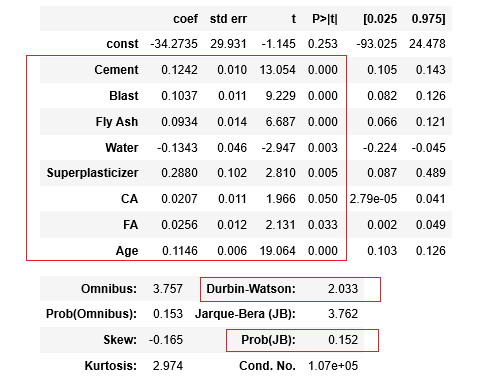
1. **To check the assumptions, build the model using the training data**

To test the assumptions, run a linear regression using statsmodels.

Note that statsmodels does not add intercept term automatically and hence we need to create an intercept to our model.

****

We find the Adjusted R square is very modest at 0.609 and that means 60.9% of the variation in the response variable, y is explained by the model. The number of observations in the training dataset is 824.

****

From the above coefficients table, we find that the variables 'Cement', 'Blast', 'Fly Ash', 'Water', 'Superplasticizer', 'CA', 'FA', 'Age', are significant at 5% level of significance.



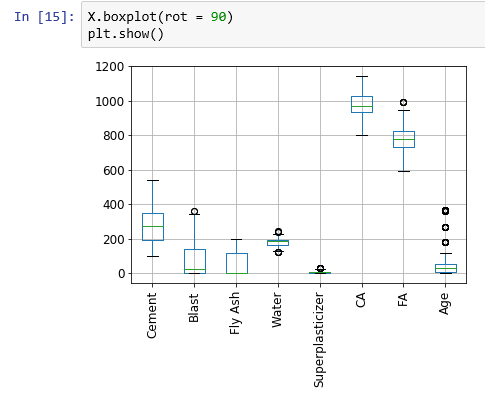


## **Test the assumptions for linear regression**

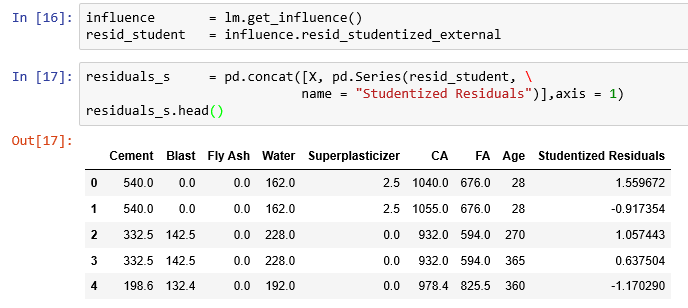


### **2.1.1 No Outliers**

**Detecting Outliers**

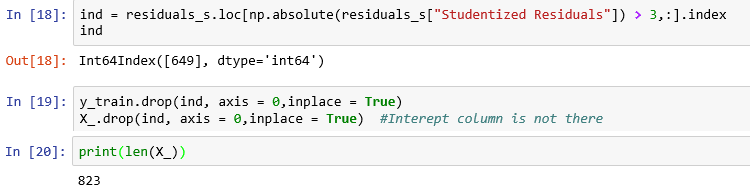


We observe a few outliers. Let us check formally also. Get the studentized residuals using get\_influence( ) and calculate studentized residuals. If the absolute value of studentized residuals is more than 3 then that observation is considered as an outlier.



Remove any observation, if the absolute value of studentized residuals is more than 3.

We try to create a logical vector for the absolute studentized residuals more than 3

****

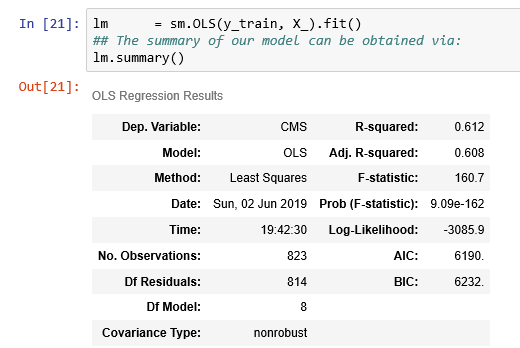
We observe that the number of observations has decreased from 824 to 823 after removing one outlier.

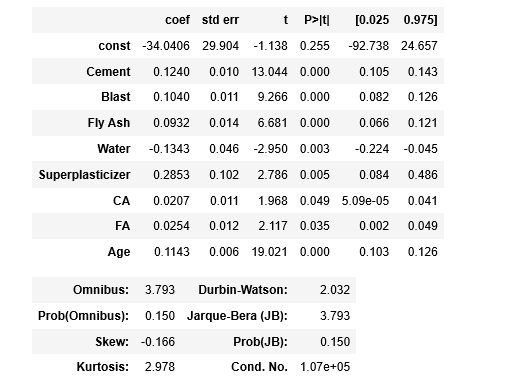
**There are no outliers to worry about.**

### **2.1.2 normality of the residuals**

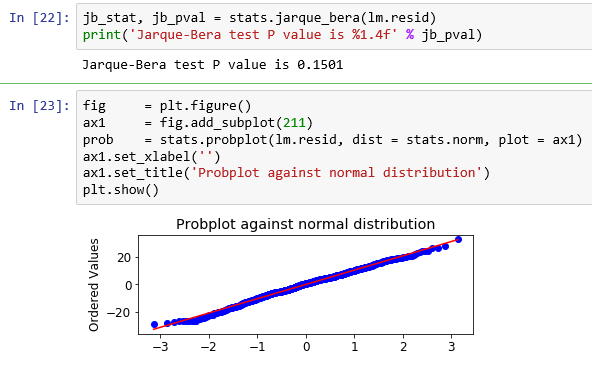
We use Jarque-Bera test from scipy library to check the normality of residuals.

* Null Hypothesis: The residuals are normally distributed.
* Alternative Hypothesis: The residuals are not normally distributed.





From the summary table, we observe that the probability (JB) is 0.15 and it is > 0.05. So, we don’t have evidence to reject the null hypothesis and conclude that the residuals are normally distributed.



From the above graph, we observe that the residuals are almost normally distributed.

### **2.1.3 No multi-collinearity**

**https://www.listendata.com/2018/01/linear-regression-in-python.html**

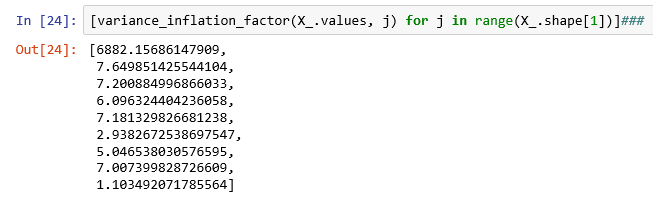
* Multi-collinearity increases the estimate of standard error of regression coefficients which makes some variables statistically insignificant when they should be significant.
* We can detect multi-collinearity by:
* By plotting scatter plots between predictor variables to have a visual description of their relationship.
* By calculating the correlation coefficients between the variables, we learn the extent of multi-collinearity in the data.
* By calculating the Variable Inflation Factor (VIF) for each variable. VIF measures how much the variance of an estimated regression coefficients increases if your predictors are correlated. The higher the value of VIF for the regressor, the more it is highly correlated to other variables.



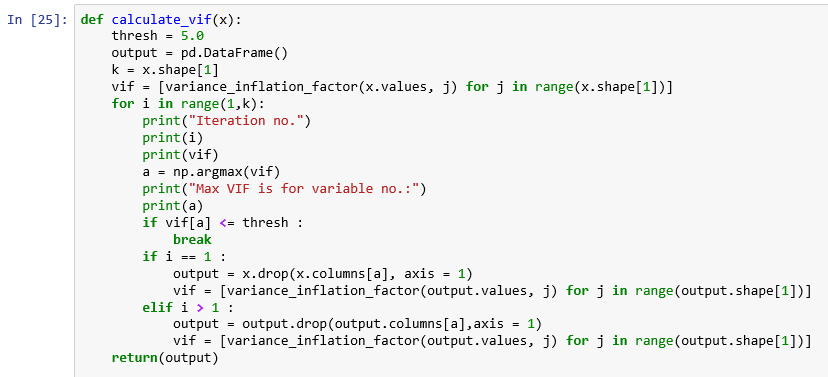
* Here we take one of the explanatory variables as the target variable and all others as independent variables. So, we run a regression between one of those independent variables with remaining independent variables.

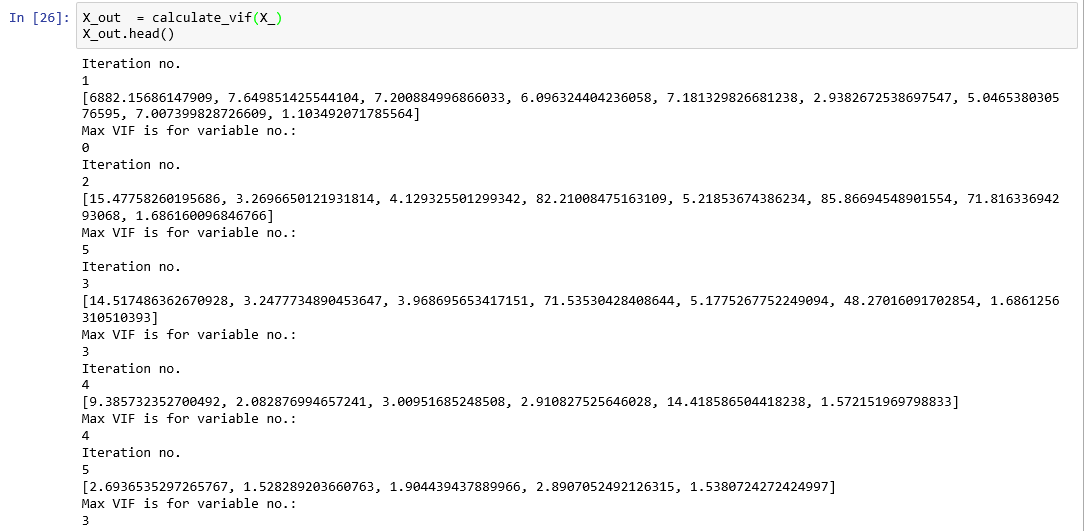
**Detecting and Removing Multicollinearity**

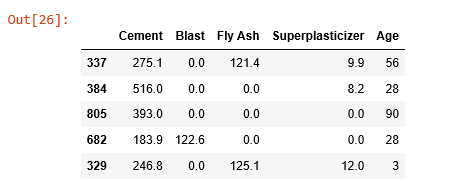
* We use the statsmodels library to calculate VIF
* Write a function to detect and remove VIF

****

* We create a function to remove the collinear variables.
* We choose a threshold of 5 which means if VIF is more than 5 for a particular variable then that variable will be removed.







**Now we have only five predictor variables and no multi-collinearity.**

### **2.1.4 No auto correlation**

**The Durbin Watson test is a measure of auto-correlation in residuals from linear regression.**

Hypothesis for the Durbin-Watson test are:

* H0: No first order auto correlation
* Ha: First order correlation exists

Assumptions are: The errors are normally distributed with a mean 0 and the errors are stationary.

The Durbin Watson test reports a test statistic with a value from 0 to 4

* 2 indicates there is No auto correlation
* Value between 0 and 2, indicates the presence of positive autocorrelation
* Value between 2 to 4, indicates the presence of negative autocorrelation

In our case it is 1.975 which is approximately 2 and hence, there is no first order auto correlation.

**We can confirm the absence of autocorrelation we use Ljungbox test.**

* Null Hypothesis: Autocorrelation is absent.
* Alternative Hypothesis: Autocorrelation is present.

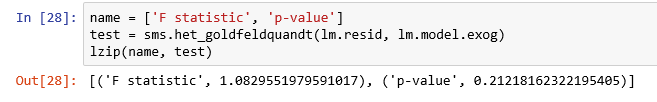


Since p value of 0.5515 > 0.05, we do not have sufficient evidence to reject the null hypothesis and conclude that auto-correlation is absent.

### **2.1.5 absence of heteroscedasticity**

**Use Goldfeld Quandt to test for heteroscedasticity.**

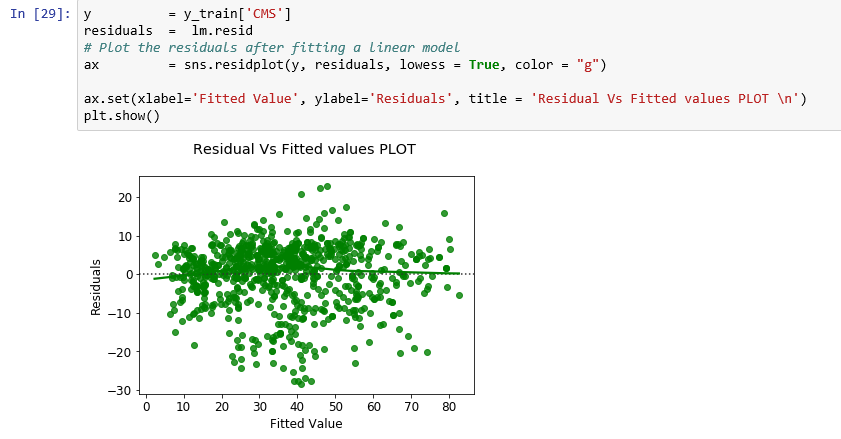
* Null Hypothesis: Error terms are homoscedastic
* Alternative Hypothesis: Error terms are heteroscedastic.



The p-value is 0.2122; So, the assumption of homoscedasticity is satisfied; hence, we can say that the residuals have constant variance.

### **2.1.6 linearity**

The residual vs fitted values plot is used to check for constant variance and linearity, and to identify potential outliers in the data.



The residual plot indicates that the model’s residuals are mostly restricting to mean of zero to a great extent exhibiting linearity.

**All the assumptions of our linear regression model are satisfied.**

**Now data is ready for model building.**

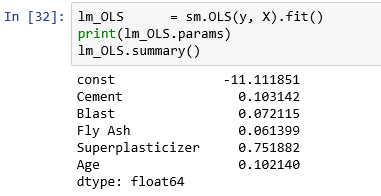
## **build the model**

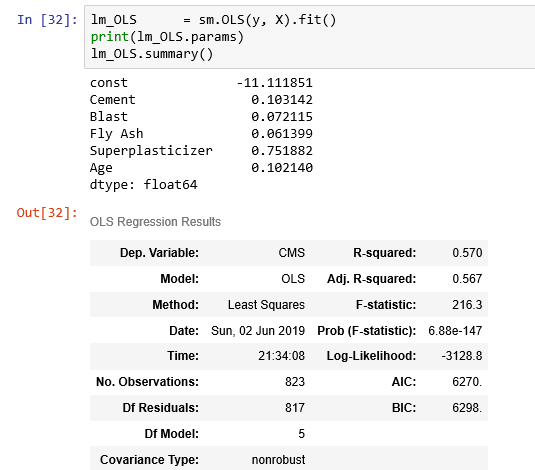
* We have already used the train data to build a model.
* Now we use the test data to evaluate the model performance.



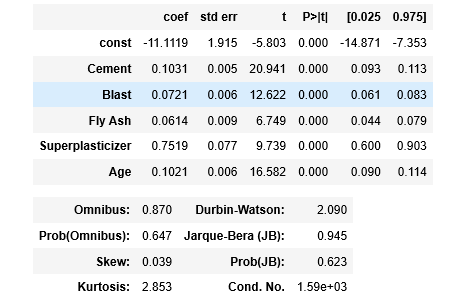
We have 179 observations for the training data and 77 observations for the test data.







We observe that the adjusted R square is 56.7% only after we tested and satisfied the assumptions.



From the above coefficients table, we observe five predictor variables are significant at 5% level of significance.

The regression equation is given below:

CMS = - 11.111851 + 0.103142 \* Cement + 0.072115 \* Blast + 0.061399 \* Fly Ash +

0.751882 \* Superplasticizer + 0.102140 \* Age

**Interpretation**

1. Holding all the other variables constant, one unit (Kg) increase in Cement will increase CMS by 0.103142 units.
2. Holding all the other variables constant, one unit (Kg) increase in Blast will increase CMS by 0.072115 units.
3. Holding all the other variables constant, one unit (Kg) increase in Fly Ash will increase CMS by 0.061399 units.
4. Holding all the other variables constant, one unit (Kg) increase in Superplasticizer will increase CMS by 0.072115 units.
5. Holding all the other variables constant, one unit (day) increase in Age will increase CMS by 0.102140 units.

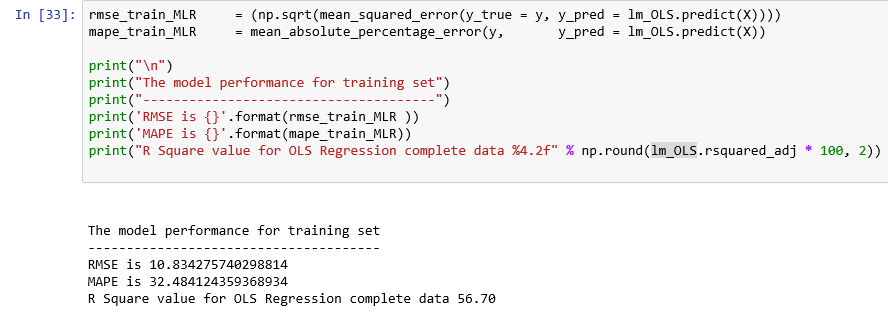
## **Evaluate model performance**

We will evaluate our model using RMSE, MAPE and R2-score.



### **Prediction Accuracy**

* 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.

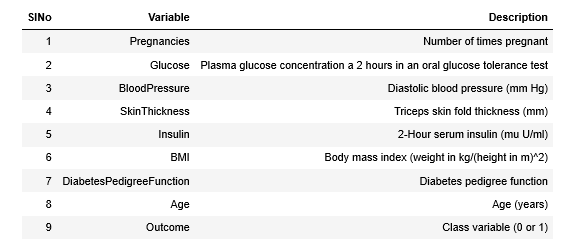
****

****

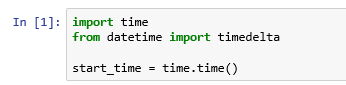
**We observe that between training and testing data, that there is no great degradation of model performance.**

# **Example 2 – Logistic regression**

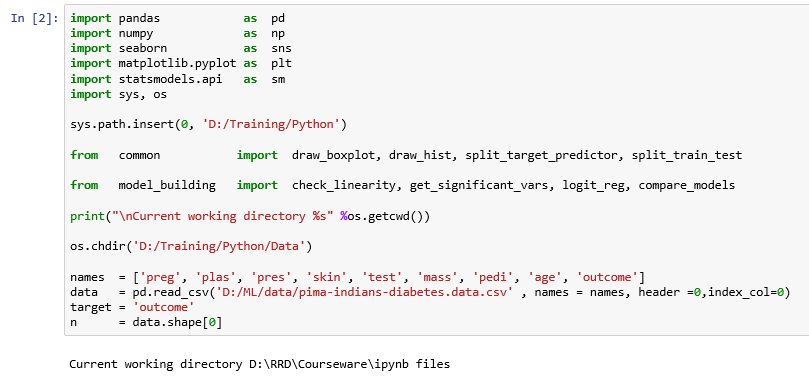
* We shall use the dataset: pima-indians-diabetes.data.
* It is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.
* The objective is to predict based on diagnostic measurements whether a patient has diabetes. Several constraints were placed on the selection of these 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>



**To print the elapsed time, first get the start time**

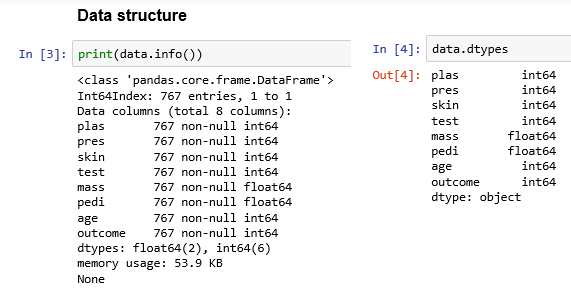


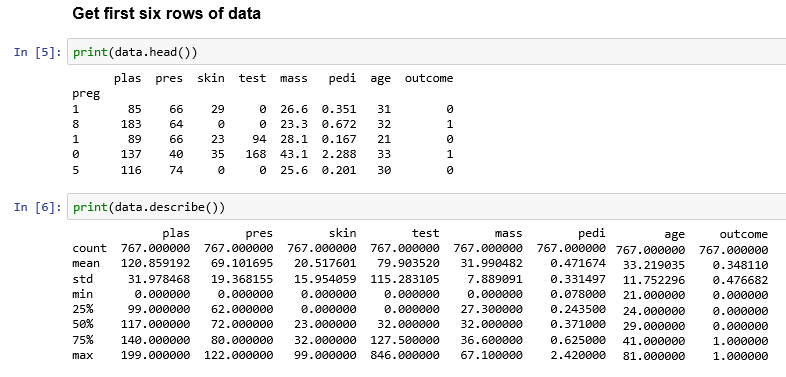
**Import the required libraries**



You would have noticed the use of two user defined libraries namely, common and model\_building. They are defined in the files, common.py and model\_building.py located in the directory, D:/Training/Python. See the use of sys.path.insert. Contents of these two files are given in the Appendix.

**Understand the data**

****

****

## **3.1 Logistic Regression Assumptions**

1. ***Binary logistic regression requires the target / dependent variable to be binary.***

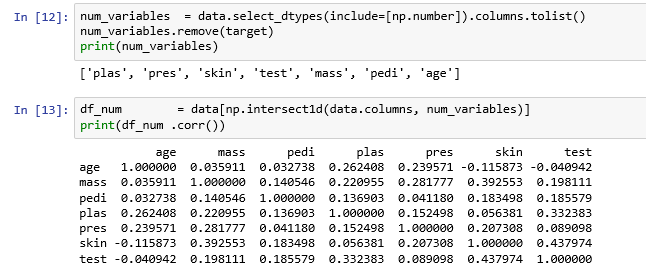
For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome. In our case it is class 1.

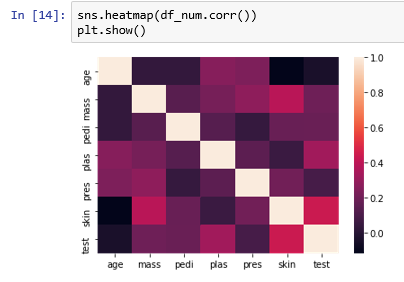
1. ***Only the meaningful variables should be included.***

We have ensured that there are no unwanted variables selected for model building.

1. ***Absence of multicollinearity***

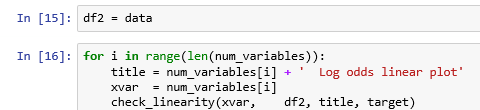
Let us find the correlation among numerical variables.

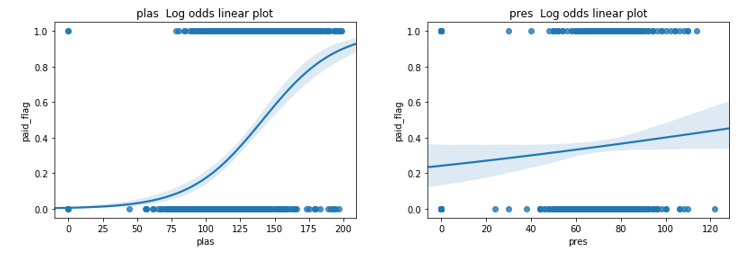


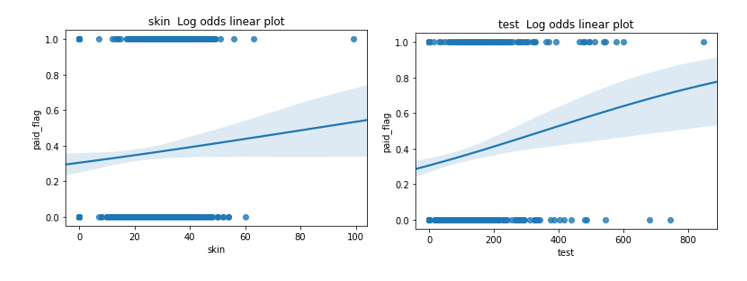


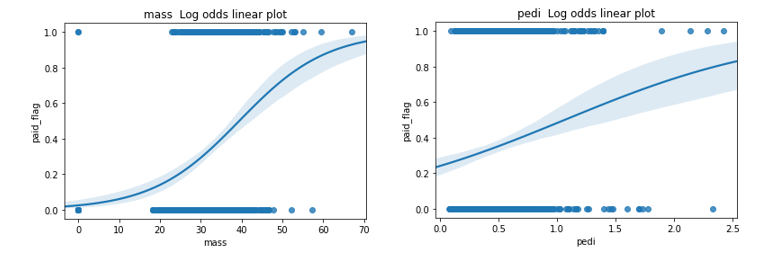
1. ***The independent variables are linearly related to the log odds***

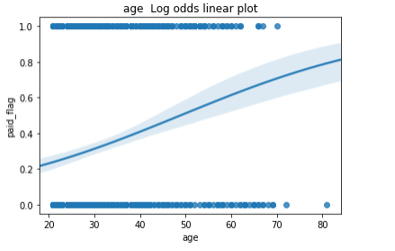
* We need to check the assumption of Independent variables are linearly related to the log odds.
* One way to checking this is to plot the Independent variables in question and look for an S-shaped curve.











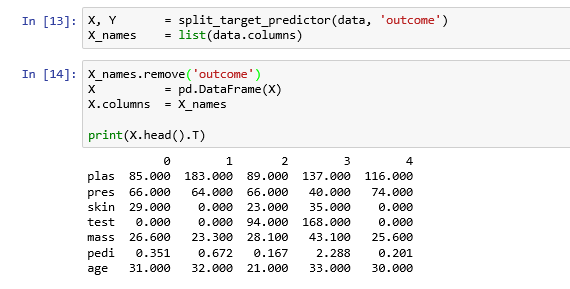
The above graphs show the resemblance of the S curve.

1. ***Logistic regression requires quite a large number of observations.***

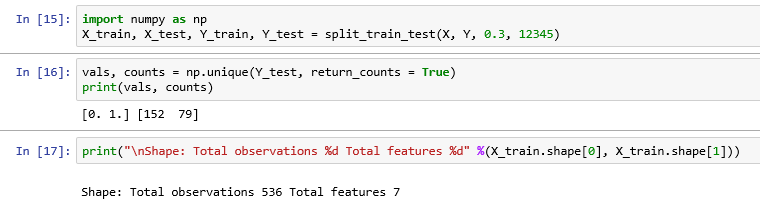
* A general rule of thumb is that at least 10 per variable.
* We have 768 observations and 9 variables; so, we have approximately 85 observations per variable.

## **3.2 Model Building**

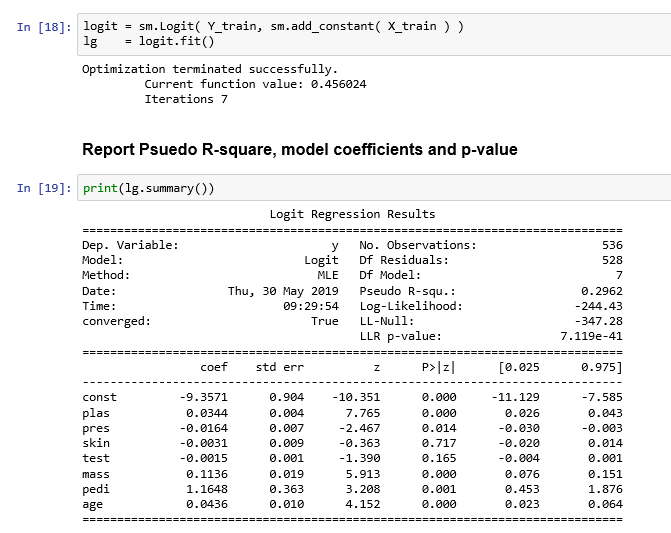
* Logistic Regression is a classification algorithm that is used to predict the probability of a target variable which is categorical. Here, the target variable is a dichotomous variable that contains data coded as 1 (desired outcome like success) or 0 (Example: failure).
* We have two popular options for building a logistic regression model; they are scikit-learn and StatsModels.
* Good thing about statsmodels is the summary output it produces. Students with R back ground will like it as it looks familiar.



**Split the data into training and test data**



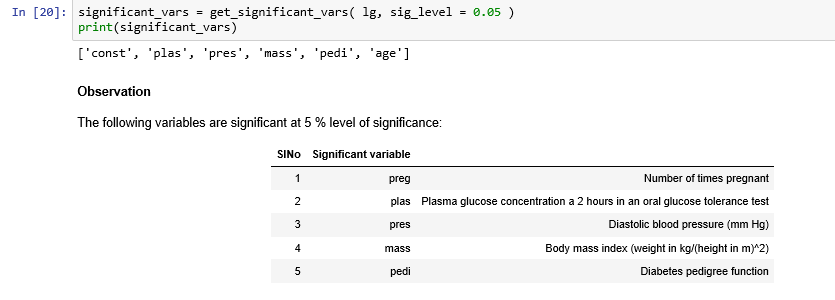
**Build the model using training data**

****

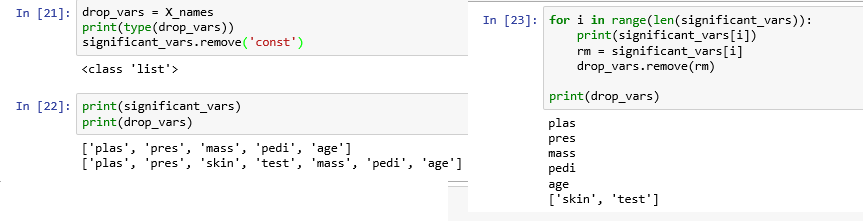
**Observation**

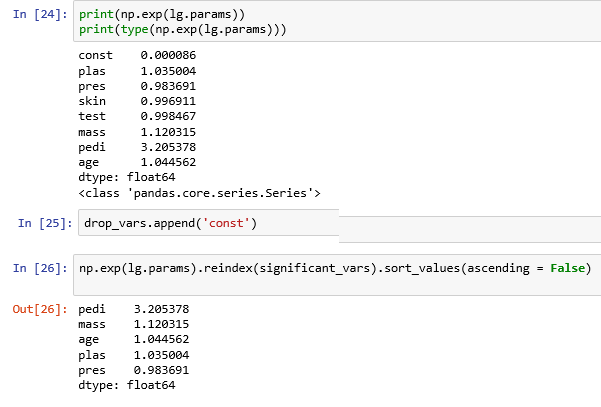
* We observe that the McFadden R square (Pseudo R square) is 29.62 % and the model fitness is good. This McFadden approach is one minus the ratio of two log likelihoods. The numerator is the log likelihood of the logit model selected and the denominator is the log likelihood if the model just had an intercept.
* A goodness of fit using McFadden‟s pseudo r square (ρ^2) is used for fitting the overall model. McFadden suggested ρ^2 values of between 0.2 and 0.4 should be taken to represent a very good fit of the model (Louviere et al.,2000).
* Refer http://www.lifesciencesite.com/lsj/life1002/286\_B01288life1002\_2028\_2036.pdf

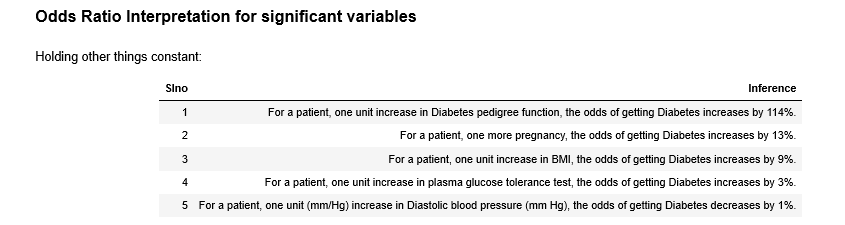
**List the significant variables at 5% level of significance**

****

**Get Odds ratio**

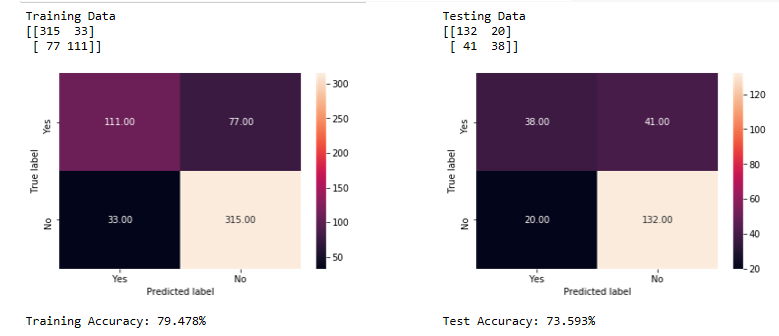
****

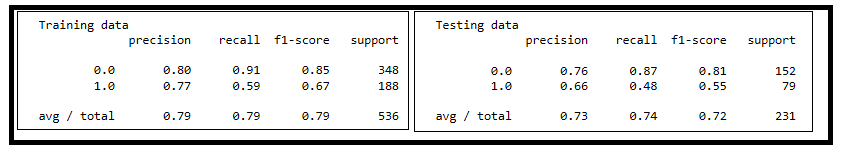
****

****

**Report the accuracy measures such as Precision, recall, accuracy ratio, auroc by using the function, logit\_reg().**

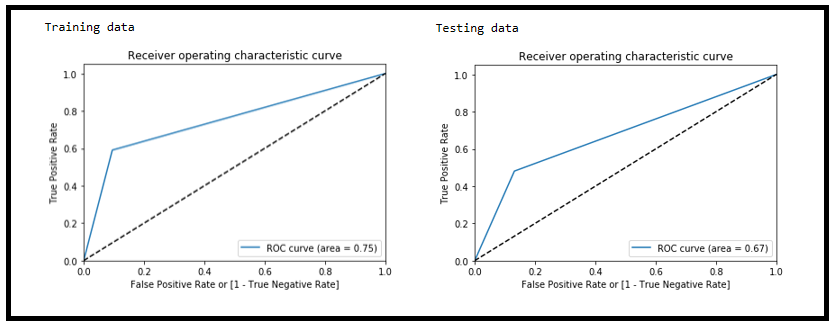
****

****

****

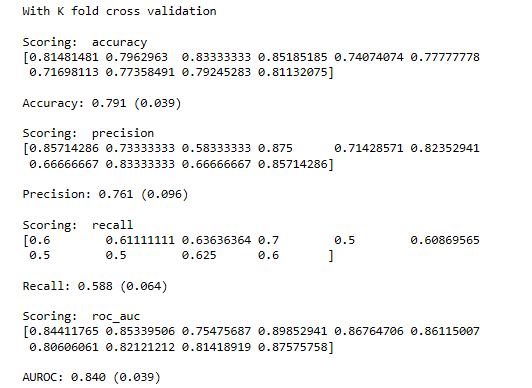
**We are interested in the minority class,1 since the data set is unbalanced. Precision, Recall for class 1 is important for us. Recall for class 1 in training data is 0.59 which is ok but in the test data, it is only 0.48, which is bad.**

**Let us compare the AUROC values.**

****

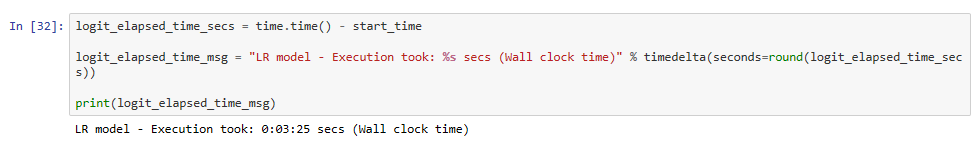
**AUROC values are OK and there is much variation between training and test data.**

**We have done k Fold (10) cross validation and let us compare each of the above metrics in each fold.**

****

**We observe the accuracy measures in each of the folds do not vary much. For example, recall average is 0.588 and the standard deviation is 0.064 only.**

**Note the time it took for processing. To get the elapsed time, get the end time of processing.**

****

**Logistic Regression model took 3 minutes and 25 seconds to process around 700 observations with 8 variables.**

# Appendix

1. **Contents of the file, common.py**

# -\*- coding: utf-8 -\*-

"""

Created on Sat Aug 4 14:55:04 2018

@author: PVS

@description: This file holds common functions used by many analytics Python programs

"""

import pandas as pd

import numpy as np

"""

1) Function Name: split\_target\_predictor

Description: This \*\*function\*\* Split data into target and predictor variables.

Input: 1) Data frame 2) Target variable name

Output: 1) Array of predictor variables 2) Array of target variables

"""

def split\_target\_predictor(dataframe, target\_name):

import pandas as pd

df = pd.DataFrame(dataframe)

predictor\_names = list(df.columns)

array = df.values

#target\_name = input('Target variable name: ')

predictor\_names.remove(target\_name)

idx = [df.columns.get\_loc(c) for c in df.columns if c in predictor\_names]

X = array[:,idx]

idy = df.columns.get\_loc(target\_name)

Y = array[:,idy]

return X, Y

### --------------------------------------------------------------------------------------------

"""

2) Function Name: split\_train\_test

Description: This function splits the data into training data and test data in the specified

proportion.

Input: 1) Array for Predictor variables

2) Array for Target variable

3) Test data proportion

4) Seed for random number used in split

Output: 1) Array of Training data for predictor variables

2) Array of Training data for target variable

3) Array of Testing data for predictor variables

4) Array of Testing data for target variable

"""

def split\_train\_test(X, Y, test\_size = 0.3, seed = 123):

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

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = test\_size, random\_state = seed)

return X\_train, X\_test, Y\_train, Y\_test

### --------------------------------------------------------------------------------------------

"""

3) Function: draw\_barplot()

Description: Description: This program does the following:

i) prints the cross tab with count and row percentages

ii) plots grouped bar chart (grouped by the target variable, AHD

Input:

1) Data Frame

2) Column name of the predictor variable selected

3) Label for the predictor variable

4) Title for the grouped bar chart

Output:

1) Cross tab with count and row percentages

2) Grouped Bar chart

"""

def draw\_barplot(df, X, xlab, title, Y):

x = df[X]

ct = pd.crosstab(df[X], Y)

print("\n Cross tab with count\n")

print(ct)

### Cross tab with row percentages

print("\n Cross tab with row % \n")

ct1 = pd.crosstab(df[X], Y, normalize = 'index').round(4)\*100

print(ct1)

# now stack and reset

stacked = ct.stack().reset\_index().rename(columns={0:'value'})

# plot grouped bar chart

ax = sns.barplot(x = stacked[X], y = stacked.value, hue = stacked.AHD)

ax.set\_title(title)

plt.show()

### -----------------------------------------------------------------------------------------------------------------------------

"""

4) Function name: draw\_boxplot()

Description: This program does the following:

i) prints the summary statistics grouped by target variable

ii) plots box chart (grouped by the target variable

Input:

1) Data Frame containing the data for analysis

2) Column name of the predictor variable selected

3) Label for the predictor variable

4) Title for the grouped bar chart

Output:

1) Cross tab with count and row percentages

2) Grouped Bar chart

"""

def draw\_boxplot(df, X, xlab, title, Y):

import seaborn as sns

import matplotlib.pyplot as plt

x = df[X]

print("\n Summary statistics for " + xlab + "\n")

print(df[X].groupby(df[Y]).describe().T)

sns.set\_style("whitegrid")

ax = sns.boxplot(x = df[X] ,y = df[Y])

ax.set\_title(title)

plt.show()

### -----------------------------------------------------------------------------------------------------------------------------

"""

5) Function name: draw\_hist

Description: This function plots a histogram for the specified variable

Input:

1) Data Frame containing numerical variable

Output:

1) A \*\*histogram\*\* graph

"""

def draw\_hist(df, X, title):

import seaborn as sns

import matplotlib.pyplot as plt

sns.distplot(df[X]).set\_title(title);

plt.show();

### -----------------------------------------------------------------------------------------------------------------------------

"""

6) Function name: percentage\_missing

Description: This function calculates and reports percentage of missing values in each column of a data frame

Input:

1) Data Frame to be inspected

Output:

1) Column-wise percentage of missing values

"""

def percentage\_missing(Dataset):

if isinstance(Dataset, pd.DataFrame):

adict={} #a dictionary conatin keys columns names and values percentage of missin value in the columns

for col in Dataset.columns:

adict[col] = (np.count\_nonzero(Dataset[col].isnull())\*100) / len(Dataset[col])

return pd.DataFrame(adict, index = ['PercMissing'], columns = adict.keys())

else:

raise TypeError("can only be used with panda dataframe")

## Source: https://stackoverflow.com/questions/26266362/how-to-count-the-nan-values-in-a-column-in-pandas-dataframe

### -----------------------------------------------------------------------------------------------------------------------------

"""

7) Function name: var\_in\_column

Description: This function calculates and reports variance in each column of a data frame

Input:

1) Data Frame to be inspected

Output:

1) Column-wise variance

"""

def var\_in\_column(Dataset):

if isinstance(Dataset, pd.DataFrame):

adict={} #a dictionary conatin keys columns names and values percentage of missin value in the columns

for col in Dataset.columns:

adict[col] = Dataset[col].var()

return pd.DataFrame(adict, index = ['Variance'], columns = adict.keys())

else:

raise TypeError("can only be used with panda dataframe")

### -----------------------------------------------------------------------------------------------------------------------------

1. **Contents of the file, model\_building.py**

# -\*- coding: utf-8 -\*-

"""

Created on Sat Aug 4 16:06:29 2018

@author: PVS

@description: This file holds functions for building models

"""

"""

1) Function name: draw\_cm

Description: This function does the following:

Plots a heat graph for confusion matrix

Input:

1) Data Frame containing actual values

2) Data frame containing predicted values

Output:

1) a nice confusion matrix graph

"""

def draw\_cm( actual, predicted ):

import matplotlib.pyplot as plt

import sklearn.metrics as metrics

import seaborn as sns

cm = metrics.confusion\_matrix( actual, predicted, [1,0] )

sns.heatmap(cm, annot=True, fmt='.2f', xticklabels = ["Yes", "No"] , yticklabels = ["Yes", "No"] )

plt.ylabel('True label')

plt.xlabel('Predicted label')

plt.show()

### -----------------------------------------------------------------------------------------------------------------------------

"""

2) Function name: draw\_roc

Description: This function does the following:

i) plots a ROC and reports the area under ROC for the specified input

Input:

1) Data Frame containing actual values for the target variable

2) Data frame containing predicted probabilities

Output:

1) a nice ROC graph with AUROC mentioned

"""

def draw\_roc( actual, probs ):

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import sklearn.metrics as metrics

import seaborn as sns

fpr, tpr, thresholds = metrics.roc\_curve( actual, probs,

drop\_intermediate = False )

auc\_score = metrics.roc\_auc\_score( actual, probs )

plt.figure(figsize=(6, 4))

plt.plot( fpr, tpr, label='ROC curve (area = %0.2f)' % auc\_score )

plt.plot([0, 1], [0, 1], 'k--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate or [1 - True Negative Rate]')

plt.ylabel('True Positive Rate')

plt.title('Receiver operating characteristic curve')

plt.legend(loc="lower right")

plt.show()

return fpr, tpr, thresholds

### -----------------------------------------------------------------------------------------------------------------------------

"""

3) Function name: get\_youdens\_j\_cutoff

Description: This function calculates the Youdens J score which gives the optimum cut-off point given

the fpr, tpr and thresholds

Input:

1) List of fpr values

2) List of tpr values

3) thresholds

Output:

1) Youdens J score

"""

def get\_youdens\_j\_cutoff(fpr,tpr,thresholds):

j\_scores = tpr-fpr

j\_ordered = sorted(zip(j\_scores,thresholds))

return j\_ordered[-1][1]

"""

4) Function name: get\_cost\_cutoff

Description: This function calculates the optimum cut-off point based on lowest cost of misclassification

Input:

1) Data Frame containing actual values for the target variable

2) Data frame containing predicted values

3) Cost of False positive - misclassification

4) Cost of False Negative - misclassification

Output:

1) Total cost of misclassification

"""

def get\_cost\_cutoff(actual, predicted, cost\_fp, cost\_fn):

cm = metrics.confusion\_matrix( actual, predicted, [1,0] )

cm\_matrix = np.array( cm )

est\_cost = cm\_matrix[0,1] \* cost\_fn + cm\_matrix[1, 0] \* cost\_fp

return est\_cost

### -----------------------------------------------------------------------------------------------------------------------------

"""

5) Function name: check\_linearity()

Description: This function checks whether continuous Independent Variable is linearly related to the log odds of the IV by plotting the IV in question and look for an S-shaped curve.

Input:

1) Data Frame containing the independent variable

2) Data frame containing the complete data

3) Title for the graph

Output:

1) a plot showing the linearity of the independent variable

"""

def check\_linearity(x1, df, title, y1):

import matplotlib.pyplot as plt

import seaborn as sns

sns.regplot(x = x1, y= y1, data= df, logistic= True).set\_title(title)

plt.show()

### -----------------------------------------------------------------------------------------------------------------------------

"""

6) Function name: get\_significant\_vars

Description: This function lists the significant variables at the specified significance level from the coefficients table

Input:

1) Fitted model

2) Desired Level of significance

Output:

1) List of significant variables at the desired level of significance

"""

def get\_significant\_vars( lm, sig\_level ):

import pandas as pd

var\_p\_vals\_df = pd.DataFrame( lm.pvalues )

var\_p\_vals\_df['vars'] = var\_p\_vals\_df.index

var\_p\_vals\_df.columns = ['pvals', 'vars']

return list( var\_p\_vals\_df[var\_p\_vals\_df.pvals <= sig\_level]['vars'] )

### -----------------------------------------------------------------------------------------------------------------------------

"""

7) Function Name: logit\_reg

Description: This \*\*function\*\* builds the logistic regression model.

Input: 1) splits for k fold

2) random seed number

3) Training data for predictor variables

4) Testin data for predictor variables

5) Training data for target variable

6) Testing data for target variable

Output: 1) AUROC 2) Metrics - Precision, Recall, F1

"""

def logit\_reg(n\_splits, random\_state, X\_train, X\_test, Y\_train, Y\_test ):

import statsmodels.api as sm

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

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

from sklearn.model\_selection import KFold

model = LogisticRegression()

model.fit(X\_train, Y\_train)

predicted\_train = model.predict(X\_train)

matrix = confusion\_matrix(Y\_train, predicted\_train)

print("\nTraining Data")

print(matrix)

draw\_cm(Y\_train, predicted\_train )

accuracy\_train = model.score(X\_train, Y\_train)

print("Training Accuracy: %.3f%%" % (accuracy\_train \* 100.0))

print("\nTesting Data")

predicted\_testing = model.predict(X\_test)

matrix = confusion\_matrix(Y\_test, predicted\_testing)

print(matrix)

draw\_cm(Y\_test, predicted\_testing)

accuracy\_test = model.score(X\_test, Y\_test)

print("Test Accuracy: %.3f%%" % (accuracy\_test \* 100.0))

measures\_train = classification\_report(Y\_train, predicted\_train)

print("\nTraining data")

print(measures\_train)

measures\_test = classification\_report(Y\_test, predicted\_testing)

print("\nTesting data")

print(measures\_test)

kfold = KFold(n\_splits = n\_splits, random\_state = random\_state)

scoring = 'roc\_auc'

auc\_train = cross\_val\_score(model, X\_train, Y\_train, scoring=scoring)

print("\nTraining data")

draw\_roc( Y\_train, predicted\_train)

auc\_test = cross\_val\_score(model, X\_test, Y\_test, scoring=scoring)

print("\nTesting data")

draw\_roc( Y\_test, predicted\_testing)

print("\nWith K fold cross validation")

scoring = 'accuracy'

print("\nScoring: %s" %scoring)

cv\_accuracy\_train = cross\_val\_score(model, X\_train, Y\_train, cv = kfold, scoring=scoring)

print(cv\_accuracy\_train)

print("\nAccuracy: %.3f (%.3f)" % (cv\_accuracy\_train.mean(), cv\_accuracy\_train.std()))

scoring = 'precision'

print("\nScoring: %s" %scoring)

cv\_precision\_train = cross\_val\_score(model, X\_train, Y\_train, cv = kfold, scoring=scoring)

print(cv\_precision\_train)

print("\nPrecision: %.3f (%.3f)" % (cv\_precision\_train.mean(), cv\_precision\_train.std()))

scoring = 'recall'

print("\nScoring: %s" %scoring)

cv\_recall\_train = cross\_val\_score(model, X\_train, Y\_train, cv = kfold, scoring=scoring)

print(cv\_recall\_train)

print("\nRecall: %.3f (%.3f)" % (cv\_recall\_train.mean(), cv\_recall\_train.std()))

scoring = 'roc\_auc'

print("\nScoring: %s" %scoring)

cv\_roc\_auc\_train = cross\_val\_score(model, X\_train, Y\_train, cv = kfold, scoring=scoring)

print(cv\_roc\_auc\_train)

print("\nAUROC: %.3f (%.3f)" % (cv\_roc\_auc\_train.mean(), cv\_roc\_auc\_train.std()))

"""

['accuracy', 'adjusted\_rand\_score', 'average\_precision', 'f1', 'f1\_macro',

'f1\_micro', 'f1\_samples', 'f1\_weighted', 'log\_loss', 'mean\_absolute\_error',

'mean\_squared\_error', 'median\_absolute\_error', 'precision',

'precision\_macro', 'precision\_micro', 'precision\_samples',

'precision\_weighted', 'r2', 'recall', 'recall\_macro', 'recall\_micro',

'recall\_samples', 'recall\_weighted', 'roc\_auc']

"""

### ------------------------------------------------------------------------------------------

"""

8) Function Name: CART

Description: This \*\*function\*\* builds the CART decision tree model.

Input: 1) splits for k fold

2) random seed number

3) Training data for predictor variables

4) Testing data for predictor variables

5) Training data for target variable

6) Testing data for target variable

Output: 1) AUROC 2) Metrics - Precision, Recall, F1

"""

def CART(n\_splits, random\_state, X\_train, X\_test, Y\_train, Y\_test ):

import statsmodels.api as sm

from sklearn.ensemble import BaggingClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

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

from sklearn.model\_selection import KFold

from sklearn.externals.six import StringIO

from IPython.display import Image

from sklearn import tree

from sklearn.tree import export\_graphviz

import pydotplus

import graphviz

cart = DecisionTreeClassifier()

kfold = KFold(n\_splits = n\_splits, random\_state = random\_state)

num\_trees = 100

model = BaggingClassifier(base\_estimator = cart, n\_estimators = num\_trees, random\_state = random\_state)

# results = cross\_val\_score(model, X, Y, cv = kfold)

# print(results)

model.fit(X\_train, Y\_train)

predicted\_train = model.predict(X\_train)

matrix = confusion\_matrix(Y\_train, predicted\_train)

print("\nTraining Data")

print(matrix)

draw\_cm(Y\_train, predicted\_train )

accuracy\_train = model.score(X\_train, Y\_train)

print("Training Accuracy: %.3f%%" % (accuracy\_train \* 100.0))

print("\nTesting Data")

predicted\_testing = model.predict(X\_test)

matrix = confusion\_matrix(Y\_test, predicted\_testing)

print(matrix)

draw\_cm(Y\_test, predicted\_testing)

accuracy\_test = model.score(X\_test, Y\_test)

print("Test Accuracy: %.3f%%" % (accuracy\_test \* 100.0))

measures\_train = classification\_report(Y\_train, predicted\_train)

print("\nTraining data")

print(measures\_train)

measures\_test = classification\_report(Y\_test, predicted\_testing)

print("\nTesting data")

print(measures\_test)

kfold = KFold(n\_splits = n\_splits, random\_state = random\_state)

scoring = 'roc\_auc'

auc\_train = cross\_val\_score(model, X\_train, Y\_train, scoring=scoring)

print("\nTraining data")

draw\_roc( Y\_train, predicted\_train)

auc\_test = cross\_val\_score(model, X\_test, Y\_test, scoring=scoring)

print("\nTesting data")

draw\_roc( Y\_test, predicted\_testing)

print("\nWith K fold cross validation")

scoring = 'accuracy'

print("\nScoring: %s" %scoring)

cv\_accuracy\_train = cross\_val\_score(model, X\_train, Y\_train, cv = kfold, scoring=scoring)

print(cv\_accuracy\_train)

print("\nAccuracy: %.3f (%.3f)" % (cv\_accuracy\_train.mean(), cv\_accuracy\_train.std()))

scoring = 'precision'

print("\nScoring: %s" %scoring)

cv\_precision\_train = cross\_val\_score(model, X\_train, Y\_train, cv = kfold, scoring=scoring)

print(cv\_precision\_train)

print("\nPrecision: %.3f (%.3f)" % (cv\_precision\_train.mean(), cv\_precision\_train.std()))

scoring = 'recall'

print("\nScoring: %s" %scoring)

cv\_recall\_train = cross\_val\_score(model, X\_train, Y\_train, cv = kfold, scoring=scoring)

print(cv\_recall\_train)

print("\nRecall: %.3f (%.3f)" % (cv\_recall\_train.mean(), cv\_recall\_train.std()))

scoring = 'roc\_auc'

print("\nScoring: %s" %scoring)

cv\_roc\_auc\_train = cross\_val\_score(model, X\_train, Y\_train, cv = kfold, scoring=scoring)

print(cv\_roc\_auc\_train)

print("\nAUROC: %.3f (%.3f)" % (cv\_roc\_auc\_train.mean(), cv\_roc\_auc\_train.std()))

clf = tree.DecisionTreeClassifier()

clf = clf.fit(X\_train, Y\_train)

dot\_data = tree.export\_graphviz(clf,

out\_file = None,

filled = True,

rounded = True,

special\_characters = True)

# graph = pydotplus.graph\_from\_dot\_data(dot\_data)

#

# graph.write\_png('tree.png')

graph = graphviz.Source(dot\_data)

graph

"""

['accuracy', 'adjusted\_rand\_score', 'average\_precision', 'f1', 'f1\_macro',

'f1\_micro', 'f1\_samples', 'f1\_weighted', 'log\_loss', 'mean\_absolute\_error',

'mean\_squared\_error', 'median\_absolute\_error', 'precision',

'precision\_macro', 'precision\_micro', 'precision\_samples',

'precision\_weighted', 'r2', 'recall', 'recall\_macro', 'recall\_micro',

'recall\_samples', 'recall\_weighted', 'roc\_auc']

"""

### ------------------------------------------------------------------------------------------

"""

9) Function Name: compare\_models

Description: This \*\*function\*\* compares various algorithms on

1) AUROC 2) Precision, 3) Recall

Input: 1) splits for k fold

2) random seed number

3) Training data for predictor variables

4) Training data for target variable

Output: Model comparison on these metrics 1) AUROC 2) Metrics - Precision, Recall

"""

def compare\_models(n\_splits, random\_state, X\_train, Y\_train):

### To compare algorithms

from matplotlib import pyplot

from sklearn.model\_selection import KFold

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

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

### Prepare models

models = []

models.append(('LR', LogisticRegression()))

models.append(('KNN', KNeighborsClassifier()))

models.append(('CART', DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('RF', RandomForestClassifier()))

### Evaluate model in turn

results = []

names = []

scores\_req = ['roc\_auc', 'precision', 'recall']

for i in range(len(scores\_req)):

scoring = scores\_req[i]

print(scoring)

for name, model in models:

print("\n n\_splits %d random\_state %d" % (n\_splits, random\_state))

# kfold = KFold(n\_splits = n\_splits, random\_state = random\_state)

kfold = KFold(n\_splits = 10, random\_state = 12345)

# print("\n")

# print(model)

# print(X\_train[0:2,])

# print(Y\_train[0:2,])

# print(scoring)

cv\_results = cross\_val\_score(model, X\_train, Y\_train, cv = kfold, scoring = scoring)

results.append(cv\_results)

names.append(name)

msg = "%s: %f (%f)" % (name, cv\_results.mean(), cv\_results.std())

print(msg)

### Box plot algorithm comparison

sub\_title = 'Algorithm Comparison using ' + scoring

fig = pyplot.figure()

fig.suptitle(sub\_title)

ax = fig.add\_subplot(111)

pyplot.boxplot(results)

ax.set\_xticklabels(names)

pyplot.show()

### ------------------------------------------------------------------------------------------