1. import pandas as pd
2. import numpy as np
3. import matplotlib.pyplot as plt
4. from sklearn.cross\_validation import train\_test\_split

**#import CSV File into Dataframe**

datatry = pd.read\_csv('D:\Data Science\Study\python\Datasets\Bloodset.csv',sep=',')

**# Display first 5 Rows from the dataset**

datatry.head()

**# Display columns from the dataset**

datatry.columns

**# correcting column name from imported dataset**

datatry = datatry.rename(columns={'Donated\_March\_2007 ': 'Donated\_March\_2007'})

**#setup feature columns name in the X**

feature\_cols = ['Recency\_months','Frequency\_times','Monetary\_c.c.\_blood','Time\_months']

X = datatry[feature\_cols]

# **Returns rows and columns from dataset**

X.shape

**# y will be predicted label from dataset datatry**

y = datatry.Donated\_March\_2007

#returns rows and columns of y dataframe

y.shape

**# the dimension of y should be equal to the first dimension of matrix (2\*2 array) X**

**# dividing dataset into testing and training dataset (by default 25%-testing and 75%- training)**

from sklearn.cross\_validation import train\_test\_split

x\_train,x\_test,y\_train,y\_test = train\_test\_split(X,y,random\_state= 0)

x\_train.shape

(561, 4)

x\_test.shape

(187, 4)

y\_train.shape

(561, )

y\_test.shape

(187, )

from sklearn.linear\_model import LogisticRegression

logreg = LogisticRegression()

logreg.fit(x\_train,y\_train)

**# This is called the model fitting process in which model**

**will learn about input & those coefficient will useful to predict for the each observation in the test dataset.**

y\_pred\_class = logreg.predict(x\_test)

**# predict from the testing dataset**

from sklearn import metrics

print(metrics.accuracy\_score(y\_test,y\_pred\_class))

0.716577540107

#**classification accuracy against true response value (y\_test) vs predicted value (y\_pred\_class**)

# **Calculate null accuracy**

y\_test.value\_counts()

y\_test.mean()

1 - y\_test.mean()

or max(y\_test.mean(),1-y\_test.mean())

0.7058823529411764

# null accuracy is 70 % less then 71 % is the model accuracy so this model does not looks good

**confusion Matrix**

print(metrics.confusion\_matrix(y\_test,y\_pred\_class))

[[129 3]

[ 50 5]]

# **Accuracy**

print(metrics.accuracy\_score(y\_test,y\_pred\_class))

0.72192513369

#**classification error**

1 - metrics.accuracy\_score(y\_test,y\_pred\_class)

0.27807486631016043

# when actual value is positive how often model able to

predict corectly

(metrics.recall\_score(y\_test,y\_pred\_class))

0.690909090909

# There is no library to calculate specificity using pandas

so we need to calculate manually

#specificity means when actual value is negative then how often

model is able to predict negative values

# false positive rate

# when actual value is negative how often model is incorrect

#1 - specificity

# 1- 97.7%

# = 2.3%

Precision

print(metrics.precision\_score(y\_test,y\_pred\_class))

0.520547945205

0.520547945205

# 62.5 % is the precision of this model it mean that when positive value is presicted how often model predicted correctly.

# How consistently we predict positive values.

logreg.predict(x\_test)[0:10]

**# Returns predicted probabilities for each of the class**

logreg.predict\_proba(x\_test)[0:10,:]

array([[ 0.74945002, 0.25054998],

[ 0.56865375, 0.43134625],

[ 0.56535678, 0.43464322],

[ 0.57565611, 0.42434389],

[ 0.13648817, 0.86351183],

[ 0.61541565, 0.38458435],

[ 0.88432533, 0.11567467],

[ 0.61541565, 0.38458435],

[ 0.96469428, 0.03530572],

[ 0.55292212, 0.44707788]])

# **above is the predicted probabilities of class1 (0-column left side) and class2 (1)(column on right hand side).**

**Visualize the predicted probabilities from the class**

predicted\_probabilities = logreg.predict\_proba(x\_test)[:,1]

plt.hist(predicted\_probabilities,bins=8)

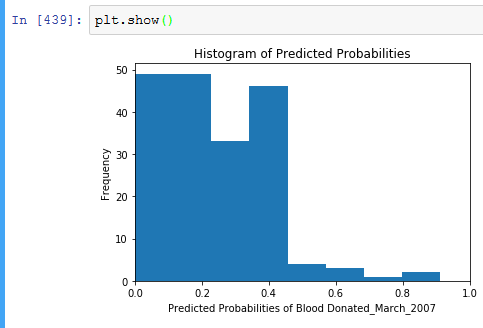
plt.xlim(0,1)

plt.title("Histogram of Predicted Probabilities")

plt.xlabel("Predicted Probabilities of Blood Donated\_March\_2007")

plt.ylabel("Frequency")

plt.show()



from sklearn.preprocessing import binarize

#Setting up threshold = .3 (so any actual whose probabilities

is .3 the it will be predicted probabilities =1 i.e modelwill become

sensitive for threshold =0.3)

y\_pred\_class1 = binarize(predicted\_probabilities,0.3)[0]

print(predicted\_probabilities[1:10])

print(y\_pred\_class1[1:10])

metrics.confusion\_matrix(y\_test,y\_pred\_class1)

array([[97, 35],

[17, 38]])

# if you compare with previous confusion matrix then realize that

left hand side column values moved to right hand side

print(metrics.accuracy\_score(y\_test,y\_pred\_class1))

y\_pred\_class[1:10]

print(metrics.recall\_score(y\_test,y\_pred\_class))

0.690909090909

# **There is no change in sensitive for this model for adjusting threshold =.3**

**so you may try with different threshold values to get maximum of sensitive.**

**# ROC Area under the curve**

t1,t2,threshold = metrics.roc\_curve(y\_test,predicted\_probabilities)

plt.plot(t1,t2)

plt.xlim([0.0,1.0])

plt.ylim([0.0,1.0])

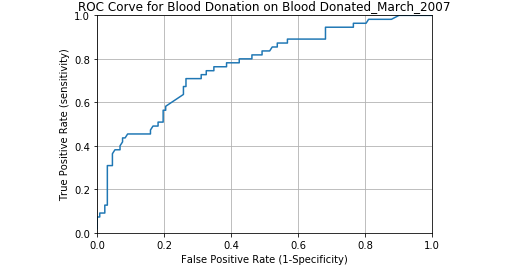
plt.title("ROC Corve for Blood Donation on Blood Donated\_March\_2007")

plt.xlabel("False Positive Rate (1-Specificity)")

plt.ylabel("True Positive Rate (sensitivity)")

plt.grid(True)

plt.show()



**#How much ROC Score for this model**

print(metrics.roc\_auc\_score(y\_test,predicted\_probabilities))

0.767906336088 (% of Area under the curve)

**# AUC is performing for this classification mode**

from sklearn.cross\_validation import cross\_val\_score

cross\_val\_score(logreg,X,y,cv=10,scoring = 'roc\_auc').mean()

0.94583476665520005