lab:6

```
import pandas as pd
data= pd.read_csv ("diabetes.csv")
data.head()
  data.shape
  data.columns
  data.info
  data.value_counts()
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
sns.heatmap(data.head(20),(cmap='Reds',annot=True
                                    linewidth=.5)
    X=data[['Age']]
    Y= data['outcome']
from sklearn.linear_model import logistic.Regression
    lrm1 = logisticRegression()
    lrm2 .fit(X,Y)
    lrm1.coeff
  lrm1.intercept
  years-old = [[60]]
  lrm1.predict(year-old)
lrb = lrm1.coef_ * 60 + lrm1.intercept_
from scipy.special import expit
```

Lab6. Predictive Analytics for Hospitals

Objectives

In this lab, you will continue to work on Diabetes data from Lab4 and apply Predictive Analytics principles to recommend the course of action to be taken by the hospital.

Learning Outcomes

After completing this lab, you will be able to

- Perform prediction on single and multiple independent variables for the target
- Apply Forward Selection procedure to find the best features based on AUC scores
- Draw a line plot of AUC scores for variables and select them using cut-off
- Plot Gain curves and Life curves and interpret results

Business Use Case

As a Data Scientist, you are requested by the hospital to build a predictive model for them. The hospital will apply your predictive model to perform prediction on new patients whether they will become diabetic or not. Further, based on the prediction, the hospital will recommend action plan such as developing a diet plan, physical activities and others.

Step1. [Import dataset]

- Using Pandas, import **"diabetes.csv"** file and print properties such as head, shape, columns, dtype, info and value_counts.

Step2. [Identify relationships between feature]

- Create a Heatmap for the dataset and understand the data

Step3. [Prediction using one feature]

Will older people become diabetic?

- Create LogisticRegression model, train with "Age" as **X** and "Outcome" feature as **y**.
- Print model parameter values: coef_ and intercept_
- **Query:** A person is 60 years old. Will he be diabetic?
- Use model parameters and find function value. Your code will be as below.

```
lrf = logreg.coef_ * 60 + logreg.intercept_
from scipy.special import expit
expit(lrf)
```

If your output > 0.5, YES he will become diabetic. Otherwise, NO, he will not be diabetic.

Step4. [Prediction using many features]

Will Glucose, BMI and Age values make someone diabetic?

- Select the three features 'Glucose', 'BMI' and 'Age' from your dataset, call it as **X**
- Create a new LogisticRegression model, train with X and 'Outcome' as **y**.
- Query: For a person, glucose=150, bmi=30, age=40. Will he be diabetic?
- Find the value of expit() as before. Output will be: 0.5208271643241003

Note: You can also verify expit() output value as below

```
logreg.predict_proba([[150, 30, 40]])
```

Step5. [Build LoR model with all features]

- Create LoR model, train it with X_train and y_train values
- Now, compute and print its AUC value
- Can we get this AUC value with limited set of good features?. Yes, we are going to find with 'Forward Selection Procedure'.

```
if expit(lrf) > 0.5:
    print("YES, he will become diabetic")
else:
    print("No, he will not be diabetic")

X1 = data[["Age", "BMI", "Glucose"]]
lm2 = logistic.Regression()
lr m2.fit(X1, y)
l2 M2.predict([[150, 30, 40]])
l2M2.predict_proba[[[150, 30, 40]]]

import warnings
warnings.filterwarnings("ignore")
X3 = data.drop("Outcome", axis=1)
lrm3 = logistic Regression()

from sklearn.model_selection train_test_size_as20
X_train, X_test, Y_train, Y_test = train_test_split
                    (X3, Y, train_size=0.8, test_size=0.2
lrm3.fit(x_train, Y_train)
y_pred = lrm3.predict(x_test)
from sklearn.metrics import roc_auc_score.
print("LOR Ave", roc_auc_score(y_test, y_pred)
type(data.columns)
def get_auc(var, tar, df):
```

Step6. [Forward Selection Procedure]

Now, you have to find and select a good set of features with the best AUC score. The algorithm is

Forward stepwise variable selection procedure

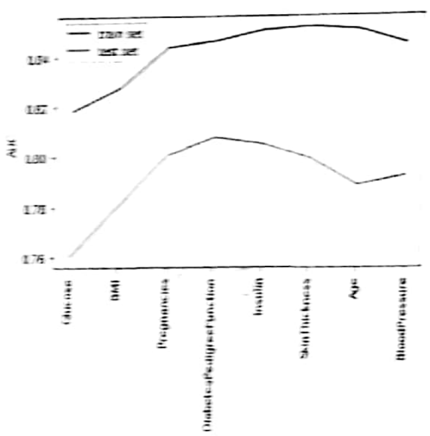
1. Empty set
2. Find best variable v
3. Find best variable v in combination with v
4. Find best variable v in combination with v, v
5. ...
6. (Until all variables are added or until predefined number of variables is added)

Implementation Steps of the forward stepwise procedure

1. Define a function get_auc() that calculates AUC given a certain set of variables
2. Define a function best_next() that returns next best variable in combination with current variables
3. Loop until desired number of variables

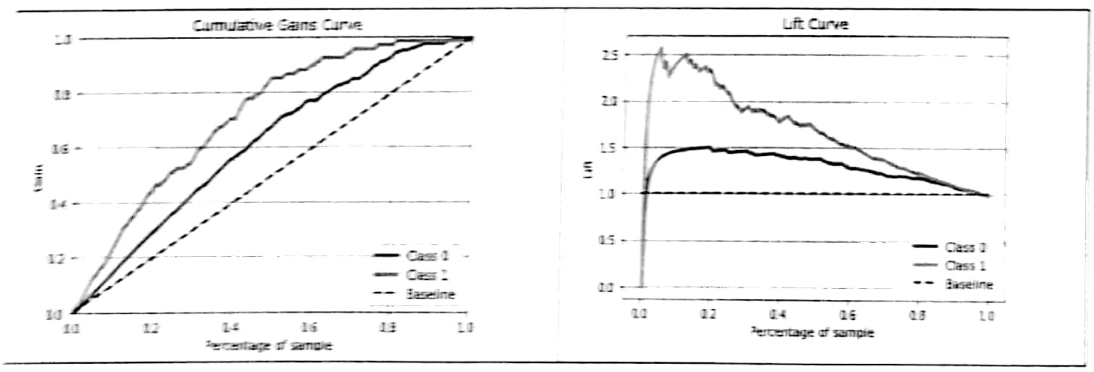
Step7. [Plot Line graph of AUC values and select cut-off]

- Split your dataset equally for training and testing
- Plot AUC values for each variable. The graph will appear as below.



The cut-off line can be drawn at the 4th variable. It gives the best AUC score around 0.81. Therefore, it will be sufficient to use only the first 4 features for training and testing. Otherwise, AUC is going down gradually.

Step8. [Draw Cumulative Gain Chart and Lift Chart]



Interpret these curves.

```
    dx = df[[var]]
    dy = df[tar]
  logreg = logistic Regression()
from sklearn.metrics import roc_auc_score
print ["Log Auc", roc_auc_score(y_test, y_pred)]
      type(data.columns)
  def get_auc(var, tar, df):
      dx = df[var]
      dy = df[tar]
    logreg = Logistic Regression[]
    log.reg.fit(dx, dy)
  pred = log.reg.predict_proba(dx)[:, 1]
  auc_val = roc_auc_score(dy, pred())
  return auc_val)
get_auc(["BHI", "Glucose"], ["Outcome"], data)
get.auc(['pregancies', 'Blood Pressure', 'Skin thickness'],
                        ["Outcome"], data)
  def next_best(current_card, tar, df):
      best_auc = -1
      best_var = None
      for in in card:
          auc_v = get_auc(current + [i], tar, df)
            if auc_v > best_auc:
            best_var = None
```

NOTES

```
        best_var = i
        return best_var
    tar = ["Outcome"]
Current = ['Insulin', 'BMI', 'Diabetes pedigreefunction',
                                                'Age']
Card = ['Pregnancies', 'Blood pressure', 'Skin Thickness']
next_var = next_best current, Card, tar, data)
print (next_var)
tar = ["Outcome"]
Current = [ ]
Card = ["Pregnancies', 'Glucose', 'Blood pressure', 'Skin',
        'Thickness', 'Insulin', 'BMI', 'Diabetes Pedigree
        function', 'Age']
max_num = 5
num_it = min (max_num, len Card))
for i in range (0, num_it):
    next_var = next_best (Current, Card, tar, data)
    Current = Current + [next_var]
    Card.remove (next_var)
    print ('variable added in step' + str(i+1) + "is" +
                                        next_var + ".")
print (current)
X_train, X_test, Y_train, Y_test = train_test_split
                        (X, Y, test_size = 0.5, stratify=Y)
```

```
pred 2 = lrm 3. predict = proba ( X_test)

train = pd. concat [X_train, Y_train], axis = 1)
test = pd. concat ([[X_test, Y_test], axis = 1)
def auc_train_test (variables, target, train, test):
        X_train = train [variables]
        X_test = train [variables]
        Y_train = train [target]
        Y_test = test [target]
        logreg = LogisticRegression ( )
    logreg.fit (X_train, Y_train].
    predictions_train = logreg . predict_proba ( X_train)
                                                    [ :, 1])

    predictions test = logreg_predict_proba (X_test) [:, 1]
auc_train = roc_auc_score ( Y_train, predictions_train)
auctest = roc_auc_score ( Y_test ; prediction stex)
return ( auc_train, auc_test)
  import matplotlib . pyplot as plt
  import numpy as np
 X = np . array (range (0, len ( auc_values_train)))
my_train = np . array ( auc_values_train)
my_test = np . array ( auc_values_test)
 plt . xticks (X, X_3.Columns, rotation = 90)
      plt . plot ( X, my train)
      plt . plot (x, my_test)
      plt . ylim (0.6, 1.0))   plt. show( )
```

NOTES

```
Import scikitplot as skplt.
skplt.metrics.plot_cumulative_gain, (y_test, pred_a)
plt.show()
plt.figure(figsize=(7,7))
Skplt.metrics.plot_lift_curve(y_test, pred_a)
plt.show()
```