Meaningful predictive modeling

By Anahita Bilimoria

1 This projects consists of two datasets:

1.0.1 1.Dataset for predicting admissions for master's programs.(Regression problem, regularisation and MSE,)

1.0.2 2.Dataset for predicting presence of heart disease in the patient. (Classification problem, regularisation, Accuracy (TN, TP, FP, FN) and error (Balanced error rate))

1.1 Regression

1.1.1 Dataset for predicting admissions for master's programs.

The dataset contains several parameters which are considered important during the application for Masters Programs The parameters included are:

1. GRE Scores (out of 340)
2. TOEFL Scores (out of 120)
3. University Rating (out of 5)
4. Statement of Purpose and Letter of Recommendation Strength (out of 5)
5. Undergraduate GPA (out of 10)
6. Research Experience (either 0 or 1)
7. Chance of Admit (ranging from 0 to 1)

1.1.2 Length of dataset : 400

1.1.3 You can find the dataset here :

https://www.kaggle.com/adityadeshpande23/admissionpredictioncsv#Admission_Predict.csv (https://www.kaggle.com/adityadeshpande23/admissionpredictioncsv#Admission_Predict.csv)

1.1.4 Dataset Cleaning :

1.The feature "Research Experience" had values 1 or 0,so it needs to be converted into True or False. 2.The rest of the features namely "GRE Scores","TOEFL Scores","University Rating","Statement of Purpose and Letter of Recommendation Strength","Undergraduate GPA" need to be converted into float datatype.

In [1]:

```
# Importing python libraries
import csv
import numpy as np
from sklearn import linear_model
from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn.metrics import mean_squared_error
import random
```

In [2]:

```
#  Reading the csv file

path = "Admission_Predict.csv"
csv_file = open(path, mode='r')
reader = csv.reader(csv_file,delimiter=",")
headers = next(reader)
```

In [3]:

```
#  Dataset Cleaning

X = []
Y = []
data = []

for row in reader:
    x_values = row[1:]
    values = [float(value) for value in x_values]

    if values[6] == float(1):
        values[6] = True
    else:
        values[6] = False

    data.append(values)
```

In [4]:

```
#  Training Features and Training labels,Testing Features and Testing labels

random.shuffle(data)
length = len(data)

X = [x[:7] for x in data]
Y = [x[7] for x in data]

trainX = X[:length//2]
validX = X[length//2:3*length//4]
testX = X[3*length//4:]

trainY = Y[:length//2]
validY = Y[length//2:3*length//4]
testY = Y[3*length//4:]

trainX = np.array(trainX)
trainY = np.array(trainY)
testX = np.array(testX)
testY = np.array(testY)
validX = np.array(validX)
validY = np.array(validY)
```

In [5]:

```
#  Regression model using Ridge
# Regularising with a value of 1, 0.5, 1.5, 2.0. the values of the MSE increase belo

reg1 = linear_model.Ridge(1.0, fit_intercept = False)
reg = reg1.fit(trainX, trainY)
```

In [6]:

```
# Validation data against the regression model

validX1 = validX[0]
validY1 = validY[0]

predictedValidY1 = reg.predict([validX1])
```

In [7]:

```
#  Testing data against the regression model

testX1 = testX
testY1 = testY
predictedY1 = reg.predict(testX1)
```

In [8]:

```
# Calculating the MSE and FVU on the test set

difference = [(x-y)**2 for (x,y) in zip(predictedY1, testY1)]

MSE = sum(difference) / len(difference)

print("MSE : ", MSE)
FVU = MSE / np.var(testX1)
R2 = 1 - FVU

print("R2 : ", R2)
```

```
MSE :  0.006849927926739739
R2 :  0.999999429811163
```

In [9]:

```
# Printing key outputs

print()
print()
print("Features : ",headers[1:7])
print()
print()
print("Data sample : ",data[0])
print()
print()
print("length of dataset : ",length)
print()
print()
print("One of the Feature vectors for training : ",trainX[0])
print()
print()
print("Output for the above feature vector : ",trainY[0])
print()
print()
print("Regression parameters of the Regression model : ",reg.coef_)
print()
print()
print("One of the Feature vectors for testing : ",testX1[0])
print()
print()
print("Expected Output of the above feature vector : ",testY1[0])
print()
print()
print("Actual Output of the above feature vector using Regression model ( validatior
print()
print()
print("Actual Output of the above feature vector using Regression model ( testing da
print()
print()
```

```
Features :  ['GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
'LOR ', 'CGPA']


Data sample :  [326.0, 116.0, 3.0, 3.5, 4.0, 9.14, True, 0.81]


length of dataset :  400


One of the Feature vectors for training :  [326.   116.     3.     3.5
4.     9.14   1.  ]


Output for the above feature vector :  0.81


Regression parameters of the Regression model :  [-0.00298625  0.00396
134  0.03711721 -0.01846286  0.02106425  0.1279936
  0.04541322]
```

```
One of the Feature vectors for testing :  [306. 100.   2.   3.   3.
8.   0.]


Expected Output of the above feature vector :  0.48


Actual Output of the above feature vector using Regression model ( val
idation data ):  0.9137628173677201


Actual Output of the above feature vector using Regression model ( tes
ting data ):  0.5883284626896295
```

1.2 2.Classification

1.2.1 Dataset for predicting presence of heart disease in the patient.

The dataset contains several parameters which are considered important for the prediction of heart disease.They are as follows:

```
1. age
2. sex
3. chest pain type (cp) (4 values)
4. resting blood pressure (trestbps)
5. serum cholestoral in mg/dl (chol)
6. fasting blood sugar > 120 mg/dl (fbs)
7. resting electrocardiographic results (values 0,1,2) (restecg)
8. maximum heart rate achieved (thalach)
9. exercise induced angina (exang)
10. oldpeak = ST depression induced by exercise relative to rest (oldpeak)
11. the slope of the peak exercise ST segment (slope)
12. number of major vessels (0-3) colored by flourosopy (ca)
13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect (thal)
14. target(1 or 0) (target)
```

1.2.2 Length of dataset : 303

1.2.3 You can find the dataset here :

https://www.kaggle.com/ronitf/heart-disease-uci (https://www.kaggle.com/ronitf/heart-disease-uci)

1.2.4 Dataset Cleaning :

1.The feature ”sex”,”fbs”,”restecg”,”exang” had values 1 or 0,so it needs to be converted into True or False.
2.The rest of the features namely ”age”,”cp”,”trestbps”,”chol”,”thalach”,”oldpeak”,”slope”,”ca”, ”thal” need to be converted into float datatype.

In [10]:

```
#  Reading the csv file

path = "heart.csv"
csv_file = open(path, mode='r',encoding="utf-8-sig")
reader = csv.reader(csv_file,delimiter=",")
headers = next(reader)
```

In [11]:

```
# Dataset Cleaning

X = []
Y = []
data = []

for row in reader:
    values = [float(value) for value in row]
    if values[1] == float(1):
        values[1] = True
    else:
        values[1] = False

    if values[5] == float(1):
        values[5] = True
    else:
        values[5] = False
    if values[6] == float(1):
        values[6] = True
    else:
        values[6] = False
    if values[8] == float(1):
        values[8] = True
    else:
        values[8] = False
    data.append(values)
```

In [12]:

```
#  Training Features and Training labels,Testing Features and Testing labels, Valida
# We split the dataset into training, validation and test sets using the ratio 50:25

random.shuffle(data)
length = len(data)
X = [x[:13] for x in data]
Y = [x[13] for x in data]

trainX = X[:length//2]
validX = X[length//2:3*length//4]
testX = X[3*length//4:]

trainY = Y[:length//2]
validY = Y[length//2:3*length//4]
testY = Y[3*length//4:]

trainX = np.array(trainX)
trainY = np.array(trainY)
testX = np.array(testX)
testY = np.array(testY)
validX = np.array(validX)
validY = np.array(validY)
```

In [13]:

```
# Classification model with ridge

classi1 = linear_model.Ridge(1.0, fit_intercept = False)
classi = classi1.fit(trainX, trainY)
```

In [14]:

```
# Validation data against the model

validX1 = validX[0]
validY1 = validY[0]
predictedValidY1 = classi.predict([validX1])
```

In [15]:

```
# Testing data against the classification model

testX1 = testX[0]
testY1 = testY[0]
predictedY1 = classi.predict([testX1])
```

In [16]:

```
#  Calculating the list of ages and all ages > 35 as True
x_class = [x for x in data]
y_class = [(x[0] > 35) for x in data]

model1 = linear_model.LogisticRegression(max_iter = 200)
model = model1.fit(x_class, y_class)

predictions = model.predict(x_class)
correct = predictions == y_class

accuracy = sum(correct) / len(correct)
print("Accuracy : ",accuracy)
```

```
Accuracy :  0.9966996699669967
```

In [17]:

```
# Now calculating the True positives, True negatives, False positives and False nega

TP = sum([(p and l) for (p,l) in zip(predictions, y_class)])
FP = sum([(p and not l) for (p,l) in zip(predictions, y_class)])
TN = sum([(not p and not l) for (p,l) in zip(predictions, y_class)])
FN = sum([(not p and l) for (p,l) in zip(predictions, y_class)])

print("TP = ", TP)
print("FP = ", FP)
print("TN = ", TN)
print("FN = ", FN)
```

```
TP =  296
FP =  1
TN =  6
FN =  0
```

In [18]:

```
# Calculating the accuracy based on above counts

total_accuracy = (TP + TN) / (TP + FP + TN + FN)
print("Total accuracy : ", total_accuracy)
```

```
Total accuracy :  0.9966996699669967
```

In [19]:

```
# Now calculating the True positive rate and True negative rate

TPR = TP / (TP+ FN)
TNR = TN / (TN + FP)

print("True positive rate : ", TPR)
print("True negative rate : ", TNR)
```

```
True positive rate :  1.0
True negative rate :  0.8571428571428571
```

```
In [20]:
```

```python
# Calculating the balanced error rate

BER = 1 - 1/2 * (TPR + TNR)
print("Balanced error rate : ", BER)
```

```
Balanced error rate :  0.0714285714285714
```

In [21]:

```
# Printing key outputs

print()
print()
print("Features : ",headers[:13])
print()
print()
print("Data sample : ",data[0])
print()
print()
print("length of dataset : ",length)
print()
print()
print("One of the Feature vectors for training : ",trainX[0])
print()
print()
print("Output for the above feature vector : ",trainY[0])
print()
print()
print("One of the Feature vectors for testing : ",testX1)
print()
print()
print("Expected Output of the above feature vector : ",testY1)
print()
print()
print("Actual Output of the above feature vector using Classification model for Test
print()
print()
```

```


Features :  ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',
'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']


Data sample :  [56.0, True, 0.0, 130.0, 283.0, True, False, 103.0, Tru
e, 1.6, 0.0, 0.0, 3.0, 0.0]


length of dataset :  303


One of the Feature vectors for training :  [ 56.    1.    0.  130.  28
3.    1.    0.  103.    1.    1.6   0.    0.
   3. ]


Output for the above feature vector :  0.0


One of the Feature vectors for testing :  [ 65.    1.    3.  138.  28
2.    1.    0.  174.    0.    1.4   1.    1.
   2. ]


Expected Output of the above feature vector :  0.0


Actual Output of the above feature vector using Classification model f
```

```
or Testing data:  0.8668038635149697
```