project_course2

March 1, 2020

1 This projects consists of two datasets:

1.0.1 1.Dataset for predicting admissions for master's programs.(Regression problem)

1.0.2 2.Dataset for predicting presence of heart disease in the patient.(Classification problem)

1.1 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

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.

Importing python libraries

[1]:
```
import csv
import numpy as np
from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn.metrics import mean_squared_error
import random
```

Reading the csv file

[2]:
```
path = "Admission_Predict.csv"
csv_file = open(path, mode='r')
reader = csv.reader(csv_file,delimiter=",")
headers = next(reader)
```

Dataset Cleaning

[3]:
```
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)
```

Training Features and Training labels,Testing Features and Testing labels

[4]:
```
random.shuffle(data)

length = len(data)

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

trainX = X[:length//2]
testX = X[length//2:]
trainY = Y[:length//2]
testY = Y[length//2:]

trainX = np.array(trainX)
trainY = np.array(trainY)
testX = np.array(testX)
```

```
testY = np.array(testY)
```

Regression model

[5]:
```
reg = LinearRegression().fit(trainX, trainY)
```

Testing data against the regression model

[6]:
```
testX1 = testX[0]
testY1 = testY[0]
predictedY1 = reg.predict([testX1])
```

Printing key outputs

[7]:
```
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)
print()
print()
print("Expected Output of the above feature vector : ",testY1)
print()
print()
print("Actual Output of the above feature vector using Regression model :␣
↪",predictedY1[0])
print()
print()
```

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


Data sample :  [315.0, 105.0, 3.0, 2.0, 2.5, 8.34, False, 0.7]


length of dataset :  400


One of the Feature vectors for training :  [315.   105.     3.     2.     2.5
8.34   0.  ]


Output for the above feature vector :  0.7


Regression parameters of the Regression model :  [ 0.00130888  0.00276263
0.01257453 -0.01063467  0.02724681  0.11054891
  0.03614574]


One of the Feature vectors for testing :  [307.  101.    3.    4.    3.    8.2
0. ]


Expected Output of the above feature vector :  0.47


Actual Output of the above feature vector using Regression model :
0.610613019361951
```

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

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.

Reading the csv file

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

Dataset Cleaning

[9]:
```
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)
```

Training Features and Training labels,Testing Features and Testing labels

[10]:
```
random.shuffle(data)

length = len(data)

X = [x[:13] for x in data]
Y = [x[13] for x in data]

trainX = X[:length//2]
testX = X[length//2:]
trainY = Y[:length//2]
testY = Y[length//2:]

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

Classification model

[11]:
```
classi = LogisticRegression(solver="liblinear").fit(trainX, trainY)
```

Testing data against the classification model

[12]:
```
testX1 = testX[0]
testY1 = testY[0]
predictedY1 = classi.predict([testX1])
```

Printing key outputs

[13]:
```
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 :␣
↪",predictedY1[0])
print()
print()
```

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


Data sample :  [57.0, False, 0.0, 140.0, 241.0, False, True, 123.0, True, 0.2,
1.0, 0.0, 3.0, 0.0]


length of dataset :  303


One of the Feature vectors for training :  [5.70e+01 0.00e+00 0.00e+00 1.40e+02
2.41e+02 0.00e+00 1.00e+00 1.23e+02
 1.00e+00 2.00e-01 1.00e+00 0.00e+00 3.00e+00]


Output for the above feature vector :  0.0
```

```
One of the Feature vectors for testing :  [ 54.   0.   2. 135. 304.   1.   1.
170.   0.   0.   2.   0.   2.]


Expected Output of the above feature vector :  1.0


Actual Output of the above feature vector using Classification model :  1.0
```