# ARTIFICIAL INTELLIGENCE

## ASSIGNMENT 2+FINAL


**Submitted by**
**Adil Hussain Mughal (12084)**

**BS (SE-5th) MORNING**

**Date:** 02 July 2021

**Submitted to:** Sir Mr. Faiq Ahmed

**DEPARTMENT OF ENGINEERING NATIONAL UNIVERSTIY OF MODERN LANGUAGES, ISLAMABAD**

# Assignment#0

## Breast Cancer Coimbra Data Set

**Abstract**: Clinical features were observed or measured for 64 patients with breast cancer and 52 healthy controls.

| | | | | | |
|---|---|---|---|---|---|
| **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 116 | **Area:** | Life |
| **Attribute Characteristics:** | Integer | **Number of Attributes:** | 10 | **Date Donated** | 2018-03-06 |
| **Associated Tasks:** | Classification | **Missing Values?** | N/A | **Number of Web Hits:** | 98971 |

### Source:

Miguel Patrício(miguelpatricio '@' gmail.com), José Pereira (jafcpereira '@' gmail.com), Joana Crisóstomo (joanacrisostomo '@' hotmail.com), Paulo Matafome (paulomatafome '@' gmail.com), Raquel Seiça (rmfseica '@' gmail.com), Francisco Caramelo (fcaramelo '@' fmed.uc.pt), all from the Faculty of Medicine of the University of Coimbra and also Manuel Gomes (manuelmgomes '@' gmail.com) from the University Hospital Centre of Coimbra

### Data Set Information:

There are 10 predictors, all quantitative, and a binary dependent variable, indicating the presence or absence of breast cancer.
The predictors are anthropometric data and parameters which can be gathered in routine blood analysis. Prediction models based on these predictors, if accurate, can potentially be used as a biomarker of breast cancer.

### Features Information:

Quantitative Attributes:
Age (years)
BMI (kg/m2)
Glucose (mg/dL)
Insulin (µU/mL)
HOMA
Leptin (ng/mL)
Adiponectin (µg/mL)
Resistin (ng/mL)
MCP-1(pg/dL)

**Labels:**
1=Healthy controls
2=Patients

**TASK 1**

**Implement the decision Tree classifier to detect breast cancer.**

import pandas as pd

data = pd.read_csv('dataR2 (1).csv')

data.head()

## standardization

from numpy import asarray

from sklearn.preprocessing import StandardScaler

# define data

print(data)

# define standard scaler

scaler = StandardScaler()

# transform data

scaled = scaler.fit_transform(data)

print("Scaled Data")

print(scaled)

```
import pandas as pd
data = pd.read_csv('dataR2 (1).csv')
data.head()
# example of a standardization
from numpy import asarray
from sklearn.preprocessing import StandardScaler
# define data
print(data)
# define standard scaler
scaler = StandardScaler()
# transform data
scaled = scaler.fit_transform(data)
print("Scaled Data")
print(scaled)
```

```
     Age        BMI  Glucose  Insulin      HOMA   Leptin  Adiponectin  \
0     48  23.500000       70    2.707  0.467409   8.8071     9.702400
1     83  20.690495       92    3.115  0.706897   8.8438     5.429285
2     82  23.124670       91    4.498  1.009651  17.9393    22.432040
3     68  21.367521       77    3.226  0.612725   9.8827     7.169560
4     86  21.111111       92    3.549  0.805386   6.6994     4.819240
..   ...        ...      ...      ...       ...      ...          ...
111   45  26.850000       92    3.330  0.755688  54.6800    12.100000
112   62  26.840000      100    4.530  1.117400  12.4500    21.420000
113   65  32.050000       97    5.730  1.370998  61.4800    22.540000
114   72  25.590000       82    2.820  0.570392  24.9600    33.750000
115   86  27.180000      138   19.910  6.777364  90.2800    14.110000

     Resistin     MCP.1  Classification
0     7.99585   417.114               1
1     4.06405   468.786               1
2     9.27715   554.697               1
3    12.76600   928.220               1
4    10.57635   773.920               1
..        ...       ...             ...
```

```
     Resistin    MCP.1  Classification
0     7.99585  417.114               1
1     4.06405  468.786               1
2     9.27715  554.697               1
3    12.76600  928.220               1
4    10.57635  773.920               1
..        ...      ...             ...
111  10.96000  268.230               2
112   7.32000  330.160               2
113  10.33000  314.050               2
114   3.27000  392.460               2
115   4.35000   90.090               2

[116 rows x 10 columns]
Scaled Data
[[-0.57979363 -0.81667527 -1.23922225 ... -0.54551749 -0.34125061
  -1.10940039]
 [ 1.60182096 -1.37875056 -0.25829943 ... -0.86421418 -0.1912238
  -1.10940039]
 [ 1.53948912 -0.89176446 -0.30288683 ... -0.4416602   0.05821407
  -1.10940039]
 ...
 [ 0.47984774  0.89385486 -0.03536242 ... -0.3563202  -0.64049127
   0.90138782]
 [ 0.91617066 -0.39854568 -0.70417344 ... -0.92857684 -0.41283214
   0.90138782]
 [ 1.7888165  -0.0804471   1.79272102 ... -0.84103616 -1.29074683
   0.90138782]]
```

**Implement the decision Tree classifier to detect breast cancer**

```
import pandas as pd

data = pd.read_csv('dataset_stan.csv')

data.head()

inputs = data.drop('Classification', axis='columns')

target = data['Classification']

print(target)
```

In [25]:
```
import pandas as pd
data = pd.read_csv('dataset_stan.csv')
data.head()
```

Out[25]:

| BMI | Glucose | Insulin | HOMA | Leptin | Adiponectin | Resistin | MCP.1 | Classification | Age_st | BMI_st | Glucose_st | Insulin_st | HOMA_st | Leptin_st | Adipon |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| )00 | 70 | 2.707 | 0.467409 | 8.8071 | 9.702400 | 7.99585 | 417.114 | 1 | -0.579794 | -0.816675 | -1.239222 | -0.728739 | -0.614282 | -0.932334 | -0 |
| 195 | 92 | 3.115 | 0.706897 | 8.8438 | 5.429285 | 4.06405 | 468.786 | 1 | 1.601821 | -1.378751 | -0.258299 | -0.688038 | -0.548240 | -0.930413 | -0 |
| 570 | 91 | 4.498 | 1.009651 | 17.9393 | 22.432040 | 9.27715 | 554.697 | 1 | 1.539489 | -0.891764 | -0.302887 | -0.550073 | -0.464752 | -0.454219 | 1 |
| 521 | 77 | 3.226 | 0.612725 | 9.8827 | 7.169560 | 12.76600 | 928.220 | 1 | 0.666843 | -1.243303 | -0.927110 | -0.676965 | -0.574210 | -0.876021 | -0 |
| 111 | 92 | 3.549 | 0.805386 | 6.6994 | 4.819240 | 10.57635 | 773.920 | 1 | 1.788816 | -1.294601 | -0.258299 | -0.644743 | -0.521081 | -1.042682 | -0 |

In [26]:
```
inputs = data.drop('Classification', axis='columns')
target = data['Classification']
print(target)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
111    2
112    2
113    2
114    2
115    2
Name: Classification, Length: 116, dtype: int64
```

inputs_ext = inputs.drop(['Age', 'BMI', 'Glucose', 'Insulin', 'HOMA', 'Leptin', 'Adiponectin', 'Resistin', 'MCP.1', 'Classification_st'], axis = 'columns')

print(inputs_ext)

In [26]:
```
inputs = data.drop('Classification', axis='columns')
target = data['Classification']
print(target)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
111    2
112    2
113    2
114    2
115    2
Name: Classification, Length: 116, dtype: int64
```

In [28]:
```
inputs_ext = inputs.drop(['Age', 'BMI', 'Glucose', 'Insulin', 'HOMA', 'Leptin', 'Adiponectin', 'Resistin', 'MCP.1', 'Classificat
print(inputs_ext)
```

```
       Age_st    BMI_st  Glucose_st  Insulin_st   HOMA_st  Leptin_st  \
0   -0.579794 -0.816675   -1.239222   -0.728739 -0.614282  -0.932334
1    1.601821 -1.378751   -0.258299   -0.688038 -0.548240  -0.930413
2    1.539489 -0.891764   -0.302887   -0.550073 -0.464752  -0.454219
3    0.666843 -1.243303   -0.927110   -0.676965 -0.574210  -0.876021
4    1.788816 -1.294601   -0.258299   -0.644743 -0.521081  -1.042682
..        ...       ...         ...         ...       ...        ...
111 -0.766789 -0.146468   -0.258299   -0.666590 -0.534786   1.469335
112  0.292852 -0.148468    0.098400   -0.546881 -0.435039  -0.741611
113  0.479848  0.893855   -0.035362   -0.427172 -0.365106   1.825348
114  0.916171 -0.398546   -0.704173   -0.717467 -0.585883  -0.086651
115  1.788816 -0.080447    1.792721    0.987394  1.125766   3.333167
```

from sklearn import tree

```
model = tree.DecisionTreeClassifier()

model.fit(inputs_ext, target)

value = model.score(inputs_ext, target)

print(value)

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split

from sklearn import metrics

tree = DecisionTreeClassifier()

X_train, X_test, y_train, y_test = train_test_split(inputs, target, test_size=0.75,
random_state=0)


tree = tree.fit(X_train,y_train)


y_predict = tree.predict(X_test)


print(metrics.confusion_matrix(y_test, y_predict))


print("Accuracy:", metrics.accuracy_score(y_test, y_predict))
```

In [30]:
```
from sklearn import tree
model = tree.DecisionTreeClassifier()
model.fit(inputs_ext, target)
value = model.score(inputs_ext, target)
print(value)
```

```
1.0
```

In [31]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

In [32]:
```
tree = DecisionTreeClassifier()
X_train, X_test, y_train, y_test = train_test_split(inputs, target, test_size=0.75, random_state=0)

tree = tree.fit(X_train,y_train)

y_predict = tree.predict(X_test)
```

In [33]:
```
print(metrics.confusion_matrix(y_test, y_predict))

print("Accuracy:", metrics.accuracy_score(y_test, y_predict))
```

```
[[41  0]
 [ 0 46]]
Accuracy: 1.0
```

In [34]:
```
del.predict([[1.788816, -0.080447, 1.792721018, 0.987394168, 1.125765748, 3.333167101, 0.576644063, -0.841036164, -1.290746828]])
```

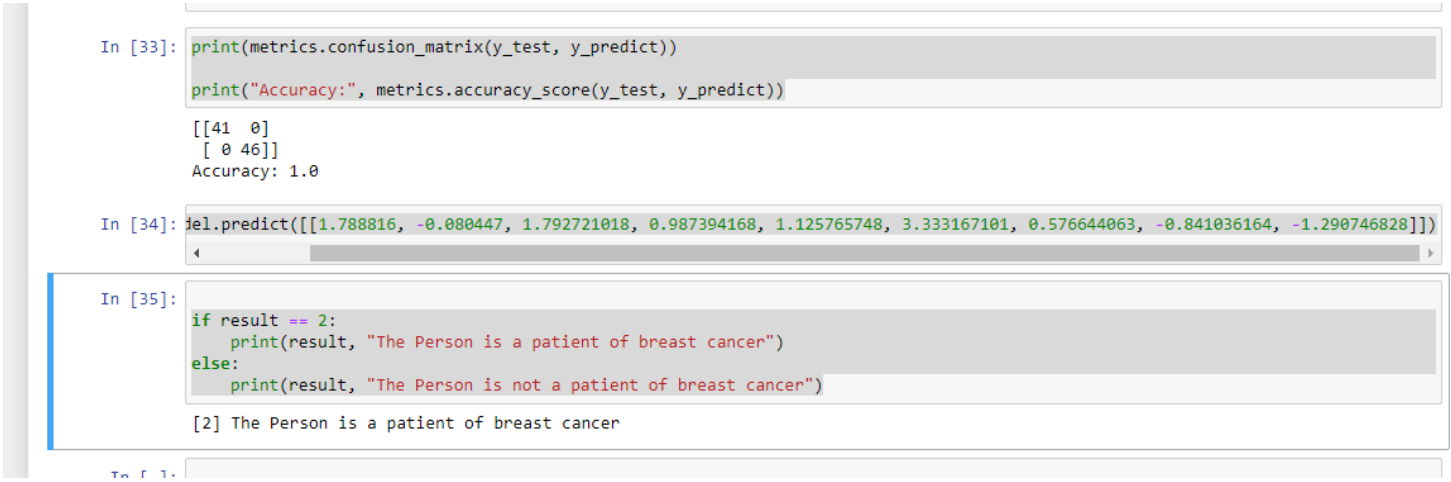
result = model.predict([[1.788816, -0.080447, 1.792721018, 0.987394168, 1.125765748, 3.333167101, 0.576644063, -0.841036164, -1.290746828]])

if result == 2:

    print(result, "The Person is a patient of breast cancer")

else:

    print(result, "The Person is not a patient of breast cancer")



**TASK 2**

**Implement the logistic regression to detect breast cancer.**

import pandas as pd

data = pd.read_csv('dataR2 (1).csv')

data.head()

# example of a standardization

from numpy import asarray

from sklearn.preprocessing import StandardScaler

# define data

print(data)

# define standard scaler

scaler = StandardScaler()

# transform data

scaled = scaler.fit_transform(data)

print("Scaled Data")

print(scaled)

```
import pandas as pd
data = pd.read_csv('dataR2 (1).csv')
data.head()
# example of a standardization
from numpy import asarray
from sklearn.preprocessing import StandardScaler
# define data
print(data)
# define standard scaler
scaler = StandardScaler()
# transform data
scaled = scaler.fit_transform(data)
print("Scaled Data")
print(scaled)
```

```
     Age        BMI  Glucose  Insulin      HOMA   Leptin  Adiponectin  \
0     48  23.500000       70    2.707  0.467409   8.8071     9.702400
1     83  20.690495       92    3.115  0.706897   8.8438     5.429285
2     82  23.124670       91    4.498  1.009651  17.9393    22.432040
3     68  21.367521       77    3.226  0.612725   9.8827     7.169560
4     86  21.111111       92    3.549  0.805386   6.6994     4.819240
..   ...        ...      ...      ...       ...      ...          ...
111   45  26.850000       92    3.330  0.755688  54.6800    12.100000
112   62  26.840000      100    4.530  1.117400  12.4500    21.420000
113   65  32.050000       97    5.730  1.370998  61.4800    22.540000
114   72  25.590000       82    2.820  0.570392  24.9600    33.750000
115   86  27.180000      138   19.910  6.777364  90.2800    14.110000

     Resistin     MCP.1  Classification
0     7.99585   417.114               1
1     4.06405   468.786               1
2     9.27715   554.697               1
3    12.76600   928.220               1
4    10.57635   773.920               1
..        ...       ...             ...
```

```
     Resistin    MCP.1  Classification
0     7.99585  417.114               1
1     4.06405  468.786               1
2     9.27715  554.697               1
3    12.76600  928.220               1
4    10.57635  773.920               1
..        ...      ...             ...
111  10.96000  268.230               2
112   7.32000  330.160               2
113  10.33000  314.050               2
114   3.27000  392.460               2
115   4.35000   90.090               2

[116 rows x 10 columns]
Scaled Data
[[-0.57979363 -0.81667527 -1.23922225 ... -0.54551749 -0.34125061
  -1.10940039]
 [ 1.60182096 -1.37875056 -0.25829943 ... -0.86421418 -0.1912238
  -1.10940039]
 [ 1.53948912 -0.89176446 -0.30288683 ... -0.4416602   0.05821407
  -1.10940039]
 ...
 [ 0.47984774  0.89385486 -0.03536242 ... -0.3563202  -0.64049127
   0.90138782]
 [ 0.91617066 -0.39854568 -0.70417344 ... -0.92857684 -0.41283214
   0.90138782]
 [ 1.7888165  -0.0804471   1.79272102 ... -0.84103616 -1.29074683
   0.90138782]]
```

```
import pandas as pd

from sklearn import metrics

import seaborn as sn

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression


data = pd.read_csv("dataset_stan.csv")

Raw_data = data.head()

print(Raw_data)
```

In [2]:
```
data = pd.read_csv("dataset_stan.csv")

Raw_data = data.head()
print(Raw_data)
```

```
   Age        BMI  Glucose  Insulin      HOMA   Leptin  Adiponectin  Resistin  \
0   48  23.500000       70    2.707  0.467409   8.8071     9.702400   7.99585
1   83  20.690495       92    3.115  0.706897   8.8438     5.429285   4.06405
2   82  23.124670       91    4.498  1.009651  17.9393    22.432040   9.27715
3   68  21.367521       77    3.226  0.612725   9.8827     7.169560  12.76600
4   86  21.111111       92    3.549  0.805386   6.6994     4.819240  10.57635

     MCP.1  Classification    Age_st    BMI_st  Glucose_st  Insulin_st  \
0  417.114               1 -0.579794 -0.816675   -1.239222   -0.728739
1  468.786               1  1.601821 -1.378751   -0.258299   -0.688038
2  554.697               1  1.539489 -0.891764   -0.302887   -0.550073
3  928.220               1  0.666843 -1.243303   -0.927110   -0.676965
4  773.920               1  1.788816 -1.294601   -0.258299   -0.644743

    HOMA_st  Leptin_st  Adiponectin_st  Resistin_st  MCP.1_st  \
0 -0.614282  -0.932334       -0.070222    -0.545517 -0.341251
1 -0.548240  -0.930413       -0.697350    -0.864214 -0.191224
2 -0.464752  -0.454219        1.797998    -0.441660  0.058214
3 -0.574210  -0.876021       -0.441945    -0.158867  1.142718
4 -0.521081  -1.042682       -0.786881    -0.336352  0.694716

   Classification_st
0            -1.1094
1            -1.1094
2            -1.1094
3            -1.1094
4            -1.1094
```

# Now, set the independent variables (represented as X) and the dependent variable (represented as y):

# Here X (Independent variable) = inputs_ext

# And  Y (dependent variable)   = target

nputs = data.drop('Classification', axis='columns')

target = data['Classification']

print(target)

In [4]:
```
# Now, set the independent variables (represented as X) and the dependent variable (represented as y):
# Here X (Independent variable) = inputs_ext
# And  Y (dependent variable)   = target
nputs = data.drop('Classification', axis='columns')
target = data['Classification']
print(target)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
111    2
112    2
113    2
114    2
115    2
Name: Classification, Length: 116, dtype: int64
```

```
inputs_ext = inputs.drop(['Age', 'BMI', 'Glucose', 'Insulin', 'HOMA', 'Leptin', 'Adiponectin', 'Resistin',
'MCP.1', 'Classification_st'], axis = 'columns')

print(inputs_ext)
```

In [5]:
```
inputs_ext = inputs.drop(['Age', 'BMI', 'Glucose', 'Insulin', 'HOMA', 'Leptin', 'Adiponectin', 'Resistin', 'MCP.1', 'Classificat
print(inputs_ext)
```

```
       Age_st    BMI_st  Glucose_st  Insulin_st   HOMA_st  Leptin_st  \
0   -0.579794 -0.816675   -1.239222   -0.728739 -0.614282  -0.932334
1    1.601821 -1.378751   -0.258299   -0.688038 -0.548240  -0.930413
2    1.539489 -0.891764   -0.302887   -0.550073 -0.464752  -0.454219
3    0.666843 -1.243303   -0.927110   -0.676965 -0.574210  -0.876021
4    1.788816 -1.294601   -0.258299   -0.644743 -0.521081  -1.042682
..        ...       ...         ...         ...       ...        ...
111 -0.766789 -0.146468   -0.258299   -0.666590 -0.534786   1.469335
112  0.292852 -0.148468    0.098400   -0.546881 -0.435039  -0.741611
113  0.479848  0.893855   -0.035362   -0.427172 -0.365106   1.825348
114  0.916171 -0.398546   -0.704173   -0.717467 -0.585883  -0.086651
115  1.788816 -0.080447    1.792721    0.987394  1.125766   3.333167

     Adiponectin_st  Resistin_st  MCP.1_st
0         -0.070222    -0.545517 -0.341251
1         -0.697350    -0.864214 -0.191224
2          1.797998    -0.441660  0.058214
3         -0.441945    -0.158867  1.142718
4         -0.786881    -0.336352  0.694716
..              ...          ...       ...
111        0.281654    -0.305255 -0.773527
112        1.649470    -0.600299 -0.593717
113        1.813843    -0.356320 -0.640491
114        3.459038    -0.928577 -0.412832
115        0.576644    -0.841036 -1.290747

[116 rows x 9 columns]
```

```
x_train, x_test, y_train, y_test = train_test_split(inputs_ext, target, test_size=0.25, random_state=0)

logistic_regression = LogisticRegression()

logistic_regression.fit(x_train, y_train)

y_pred = logistic_regression.predict(x_test)
```

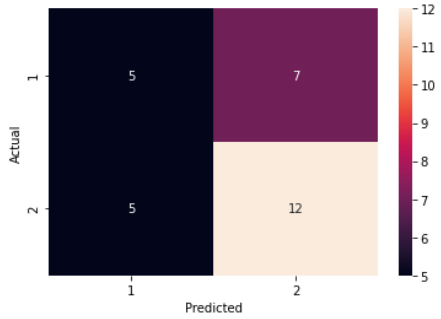
```
confusion_matrix = pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'])

sn.heatmap(confusion_matrix, annot=True)

print('Accuracy: ', metrics.accuracy_score(y_test, y_pred))
```

In [6]:
```
x_train, x_test, y_train, y_test = train_test_split(inputs_ext, target, test_size=0.25, random_state=0)
```

In [10]:
```
logistic_regression = LogisticRegression()
logistic_regression.fit(x_train, y_train)
y_pred = logistic_regression.predict(x_test)
```

In [9]:
```
confusion_matrix = pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'])
sn.heatmap(confusion_matrix, annot=True)
```

Out[9]: <AxesSubplot:xlabel='Predicted', ylabel='Actual'>




In [11]:
```
print('Accuracy: ', metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy:  0.5862068965517241
```

**For viewing code and output , please  open the link below**

**https://github.com/Enggadil/-AI-LAB-_BSSE-5-M-**