**MLP over diabetic dataset**

In [1]:
```
# importing required libraries
import numpy as np #nd-arrays
import pandas as pd #read data from datasources
from sklearn.preprocessing import LabelEncoder #Encode numerical varible into categorical variables
from sklearn.preprocessing import StandardScaler #Standardize the data using mean and std
from sklearn.model_selection import train_test_split #split the data into train and test
from sklearn.metrics import accuracy_score,confusion_matrix #evaluate a model
from sklearn.neural_network import MLPClassifier #build a model using mlp
from sklearn.model_selection import GridSearchCV #tune the hyperparamters
from sklearn.metrics import average_precision_score, make_scorer, recall_score #design custom scoring functions
```

In [2]:
```
#Reading the data into dataframe
diabetes_df=pd.read_excel('data_file.xlsx')
diabetes_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2703 entries, 0 to 2702
Data columns (total 9 columns):
GENDER      2703 non-null int64
AGE         2703 non-null int64
Height      2703 non-null int64
Weight      2703 non-null float64
BMI         2703 non-null float64
BAI         2703 non-null float64
HBA1C1      2703 non-null float64
OGTT1FBS    2703 non-null int64
NDD         2703 non-null int64
dtypes: float64(4), int64(5)
memory usage: 190.2 KB
```

In [3]:
```
#work on the copy of the dataset
clean_df=diabetes_df.copy()
#Pruning the data
clean_df.drop_duplicates(keep = 'first',inplace=True)
print(clean_df.shape)
```

```
(1065, 9)
```

In [4]:
```
clean_df.head(1)
```

Out[4]:

| | GENDER | AGE | Height | Weight | BMI | BAI | HBA1C1 | OGTT1FBS | NDD |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 37 | 156 | 88.0 | 36.160421 | 41.53 | 5.1 | 102 | 0 |

In [5]:
```
#Assigning labels with appropriate numerics
nondia=-1
diabetic=1
#create a dataframe and assign a column to indicate the diabetic status
Ynew = pd.DataFrame(nondia, index=clean_df.index, columns=['diabetic'])
```

In [6]:
```
#Identify the diabetic status of each record using the blood test results of FBS or HBA1C1
Ynew.iloc[list(np.where((clean_df.OGTT1FBS>=126) | (clean_df.HBA1C1>=6.5))[0])]=diabetic
#Concatenate the diabetic status with the anthroprometric features of the dataset
data_df=pd.concat([clean_df.iloc[:,:6],Ynew],axis=1)
```

In [7]:
```
data_df.head(1)
```

Out[7]:

| | GENDER | AGE | Height | Weight | BMI | BAI | diabetic |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 37 | 156 | 88.0 | 36.160421 | 41.53 | -1 |

In [8]:
```
#Find the diabetic and non-diabetic patients
diabetic_yes=data_df.iloc[list(np.where(data_df.diabetic==diabetic)[0])]
diabetic_no=data_df.iloc[list(np.where(data_df.diabetic==nondia)[0])]
```

In [9]:
```
diabetic_yes.describe()
```

Out[9]:

| | GENDER | AGE | Height | Weight | BMI | BAI | diabetic |
|---|---|---|---|---|---|---|---|
| count | 528.000000 | 528.000000 | 528.000000 | 528.000000 | 528.000000 | 528.000000 | 528.0 |
| mean | 0.571970 | 51.812500 | 160.280303 | 68.404356 | 26.640045 | 29.618864 | 1.0 |
| std | 0.495262 | 10.921528 | 7.562330 | 12.759664 | 4.809005 | 7.753363 | 0.0 |
| min | 0.000000 | 25.000000 | 138.000000 | 36.500000 | 15.390454 | 8.300000 | 1.0 |
| 25% | 0.000000 | 43.750000 | 156.000000 | 59.000000 | 23.290154 | 24.790000 | 1.0 |
| 50% | 1.000000 | 50.000000 | 159.500000 | 68.000000 | 26.527004 | 28.145000 | 1.0 |
| 75% | 1.000000 | 59.000000 | 165.000000 | 76.000000 | 29.585799 | 33.847500 | 1.0 |
| max | 1.000000 | 80.000000 | 186.000000 | 102.000000 | 41.207076 | 58.250000 | 1.0 |

In [10]:
```
diabetic_no.describe()
```

Out[10]:

| | GENDER | AGE | Height | Weight | BMI | BAI | diabetic |
|---|---|---|---|---|---|---|---|
| count | 537.000000 | 537.000000 | 537.000000 | 537.000000 | 537.000000 | 537.000000 | 537.0 |
| mean | 0.450652 | 44.463687 | 158.217877 | 68.325885 | 27.261189 | 31.687058 | -1.0 |
| std | 0.498023 | 11.890164 | 7.930377 | 13.118822 | 4.770046 | 8.276824 | 0.0 |
| min | 0.000000 | 20.000000 | 139.000000 | 33.500000 | 13.671875 | 14.080000 | -1.0 |
| 25% | 0.000000 | 35.000000 | 153.000000 | 61.000000 | 24.508946 | 25.650000 | -1.0 |
| 50% | 0.000000 | 44.000000 | 158.000000 | 68.000000 | 26.959840 | 30.140000 | -1.0 |
| 75% | 1.000000 | 52.000000 | 162.000000 | 76.000000 | 29.757785 | 38.260000 | -1.0 |
| max | 1.000000 | 84.000000 | 186.000000 | 110.000000 | 50.219138 | 59.760000 | -1.0 |

### Data split into training and test data

In [11]:
```
train_x, test_x, train_y, test_y = train_test_split(data_df.iloc[:,:6],
                                                    data_df.diabetic,
                                                    test_size=0.3, random_state=43)
```

### Normalizing the data using StandardScaler

In [12]:
```
sc=StandardScaler() # creating an instance for StandardScaler class
train_x=sc.fit_transform(train_x) # estimate mu and sigma for train set and transform
test_x=sc.transform(test_x) # transform the test set
```

In [13]:
```
# fetch the diabetic records from train set
diabetic_yes_train=train_x[list(np.where(train_y==diabetic)[0])]

# fetch the non-diabetic records from train set
diabetic_no_train=train_x[list(np.where(train_y==nondia)[0])]

# display the counts for each class
print('non-diabetic=',diabetic_no_train.shape,'diabetic=',diabetic_yes_train.shape)
```

```
non-diabetic= (369, 6) diabetic= (376, 6)
```

In [14]:
```
# fetch the diabetic records from test set
diabetic_yes_test=test_x[list(np.where(test_y==diabetic)[0])]

# fetch the non-diabetic records from test set
diabetic_no_test=test_x[list(np.where(test_y==nondia)[0])]

# display the counts for each class from test set
print('non-diabetic=',diabetic_no_test.shape,'diabetic=',diabetic_yes_test.shape)
```

```
non-diabetic= (168, 6) diabetic= (152, 6)
```

### Evaluation metrics

In [15]:
```
#evaluate a model using confusion matrix and accuracy score between true and actual
def evaluate(yt,yp):
    cf=confusion_matrix(yt,yp) #estimate confusion matrix
    acc=accuracy_score(yt,yp)  # estimate accuracy of the model
    return cf,acc
```

In [16]:
```
# Display metrics
def display(yt,yp,model):
    cf,acc = evaluate(yt,yp)
    print('Model=',model,'\ncf=',cf,'\n')#,'\nacc=',acc,'\n')
```

#### Multi-Layer Perceptron Classifier

In [17]:
```
#Perform Classification using MLP Classifier
mlpc = MLPClassifier(hidden_layer_sizes=1,activation='tanh',
                     learning_rate='invscaling',max_iter=10000,
                     solver='sgd',
                     random_state=0,
                     early_stopping=True) # create a MLPClassifier instance

mlpc.fit(train_x, train_y) # fit the model for trainset
train_yp=mlpc.predict(train_x) # predict the y for train set
test_yp=mlpc.predict(test_x) # predict the y for test set
```

In [18]:
```
# display the results
display(train_y,train_yp,'MLP: Training')
display(test_y,test_yp,'MLP: Testing')
```

```
Model= MLP: Training
cf= [[  0 369]
 [  0 376]]

Model= MLP: Testing
cf= [[  0 168]
 [  0 152]]
```

In [19]:
```
mlpc.classes_
```

Out[19]:
```
array([-1,  1], dtype=int64)
```

In [20]:
```
mlpc.loss_
```

Out[20]:
```
0.8712554959856844
```

In [21]:
```
mlpc.coefs_
```

Out[21]:
```
[array([[ 0.09068052],
        [ 0.39944042],
        [ 0.1902262 ],
        [ 0.08273035],
        [-0.14168853],
        [ 0.26935191]]), array([[1.35782622]])]
```

In [22]:
```
mlpc.intercepts_
```

Out[22]:
```
[array([-0.11940647]), array([1.60265222])]
```

In [23]:
```
mlpc.n_layers_
```

Out[23]:
```
3
```

In [24]:
```
mlpc.n_iter_
```

Out[24]:
```
12
```

In [25]:
```
mlpc.n_outputs_
```

Out[25]:
```
1
```

In [26]:
```
mlpc.out_activation_
```

Out[26]:
```
'logistic'
```

In [27]:
```
mlpc.n_iter_
```

Out[27]:
```
12
```

**Hyperparamter tuning using Gridsearchcv**

In [28]:
```
# recall = tp / (tp + fn) = Sensitivity or True Positive Rate / True Negative Rate
# precision = tp / (tp + fp) = Positive predictive value
custom_scorer = {'recall':make_scorer(recall_score, pos_label=diabetic),
                 'precision':make_scorer(average_precision_score, pos_label=diabetic)}
```

In [29]:
```
gscv = GridSearchCV(MLPClassifier(max_iter=10000,random_state=0),
                    {'activation':('tanh','logistic','relu'),
                     'hidden_layer_sizes':range(1,4,1),'solver':['adam','sgd']},
                    cv=5,verbose=False,
                    scoring=custom_scorer,refit='recall')
gscv.fit(train_x,train_y)
gscv.best_params_
```

```
c:\users\gitaa\spyder\venv\lib\site-packages\sklearn\model_selection\_search.py:814: DeprecationWarning: The default of the `iid` parameter will change from True to False in version 0.22 and will be removed in 0.24. This will change numeric results when test-set sizes are unequal.
  DeprecationWarning)
```

Out[29]:
```
{'activation': 'logistic', 'hidden_layer_sizes': 1, 'solver': 'adam'}
```

In [30]:
```python
#Perform Classification using MLP Classifier
mlpc = MLPClassifier(hidden_layer_sizes=(1),activation='logistic',
                     max_iter=10000,
                     solver='adam',
                     random_state=0) # create a MLPClassifier instance

mlpc.fit(train_x, train_y) # fit the model for trainset
train_yp=mlpc.predict(train_x) # predict the y for train set
test_yp=mlpc.predict(test_x) # predict the y for test set

# display the results
display(train_y,train_yp,'MLP with "sgd" solver and 1,4 hidden nodes')
display(test_y,test_yp,'For Testing')
```

```
Model= MLP with "sgd" solver and 1,4 hidden nodes
cf= [[202 167]
 [ 96 280]]

Model= For Testing
cf= [[ 92  76]
 [ 42 110]]
```

In [31]:
```python
mlpc.coefs_
```

Out[31]:
```
[array([[ 0.41181678],
        [ 1.50426546],
        [ 0.54746517],
        [-0.06032402],
        [-0.33931276],
        [-0.38891654]]), array([[0.75813057]])]
```

In [32]:
```python
mlpc.intercepts_
```

Out[32]:
```
[array([-0.55865635]), array([-0.1876706])]
```

In [33]:
```python
mlpc.score(test_x,test_y)
```

Out[33]:
```
0.63125
```

END OF SCRIPT