**I. Pen-and-paper**



Same procedure as 1 a until calculation

**II. Programming and critical analysis**

|  |  |  |  |
| --- | --- | --- | --- |
| Actual\Predicted | Negative | Positive |  |
| Negative | 419 | 25 | 444 |
| Positive | 18 | 221 | 239 |
|  | 437 | 246 | 683 |

Table : Confusion Matrix Without Early Stopping

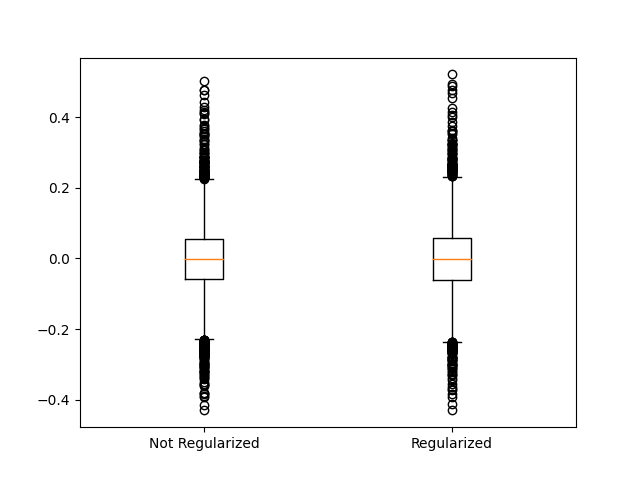
|  |  |  |  |
| --- | --- | --- | --- |
| Actual\Predicted | Negative | Positive |  |
| Negative | 389 | 55 | 444 |
| Positive | 5 | 234 | 239 |
|  | 394 | 289 | 683 |

Table : Confusion Matrix With Early Stopping

Two reasons for the only slight observed differences are:

1. The use of k-fold cross validation where the model is repeatedly refitted on parts of the dataset.
2. Early stopping being meant to stop a single model when it starts having increased generalized error.

These make them not very suited to be used together

1. 

Four strategies that can be used to minimize the observed error of the multi-layer perceptron regressor are:

1. Increase the training sample
2. Early stopping: prevent overfitting by stopping the training when the testing error rate starts increasing.
3. Change the complexity of the network structure and parameters by adding/removing nodes.
4. Regularization: Ensuring the weights keep small, since this indicates a less complex model and therefore more stable and less prone to error from outliers in the input.

**III. APPENDIX**

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from scipy.io import arff

from sklearn.neural\_network import MLPClassifier

from sklearn.neural\_network import MLPRegressor

from sklearn.model\_selection import KFold

from sklearn.model\_selection import StratifiedKFold

from sklearn.model\_selection import cross\_val\_predict

from sklearn.metrics import confusion\_matrix

GROUPN = 0

def quest2():

    # Extract Data

    D\_breast = pd.DataFrame( arff.loadarff( "breast.w.arff" )[0] )

    # Elements array

    X = D\_breast.drop(columns=D\_breast.columns[-1]).to\_numpy().astype(int)

    # Results array binarized

    Y = D\_breast[D\_breast.columns[-1]].replace(b'benign', 0).replace(b'malignant', 1)

    stratifiedk\_splits = StratifiedKFold(n\_splits=5, random\_state=GROUPN, shuffle=True)

    clf = MLPClassifier(hidden\_layer\_sizes=(3, 2), random\_state=GROUPN)

    Y\_pred = cross\_val\_predict(clf, X, Y, cv=stratifiedk\_splits)

    conf\_matrix = confusion\_matrix(Y, Y\_pred)

    clf\_es = MLPClassifier(hidden\_layer\_sizes=(3, 2), random\_state=GROUPN, early\_stopping=True)

    Y\_es\_pred = cross\_val\_predict(clf\_es, X, Y, cv=stratifiedk\_splits)

    conf\_matrix\_es = confusion\_matrix(Y, Y\_es\_pred)

    print("Confusion matrix")

    print(conf\_matrix)

    print("Confusion matrix - Early Stopping")

    print(conf\_matrix\_es)

def quest3():

    # Extract Data

    D\_kin = pd.DataFrame( arff.loadarff( "kin8nm.arff" )[0] )

    # Elements array

    X = D\_kin.drop(columns=D\_kin.columns[-1]).to\_numpy()

    Y = D\_kin[D\_kin.columns[-1]].to\_numpy()

    k\_splits = KFold(n\_splits=5, random\_state=GROUPN, shuffle=True)

    clf = MLPRegressor(alpha=0.1, random\_state=GROUPN)

    Y\_pred = cross\_val\_predict(clf, X, Y, cv=k\_splits)

    residuals = np.subtract(Y, Y\_pred)

    clf\_reg = MLPRegressor(alpha=0, random\_state=GROUPN)

    Y\_reg\_pred = cross\_val\_predict(clf\_reg, X, Y, cv=k\_splits)

    residuals\_reg = np.subtract(Y, Y\_reg\_pred)

    plt.boxplot([residuals, residuals\_reg], labels=("Not Regularized", "Regularized"))

    plt.savefig("graph\_ex3")

quest2()

quest3()

**END**