# Neural Networks Ex1- What Breast cancer will recur?

Authors: Shoval Tayro , Elad Havshush , Uriya Havshush

**Report:**

We have used Python(with several libraries such as NumPy pandas Seaborn Sklearn)

All of the codes are attached in py files.

1. Time it took to train the get the final results:

**Perceptron:** 0.06 sec

**Adaline:** 0.02 sec

**Backpropagation:**  0.15 sec

We can see that our implementation of adaline is faster than the others.

The reason for this is the fact that adaline algorithm is more simple then backpropogation(one neuron with simple algorithm vs complicated algorithm with NN) , and in perceptron we stoped the fit at the middle because the data is non linear so the results may vary.

**Parameters Adjustment:**

**Perceptron:**

In order to find the best parameters we have run the model each time with different values for the learning rate and for the amount of iterations.

*The best parameters:*

* **eta** = 0.01
* **n\_iter** =10

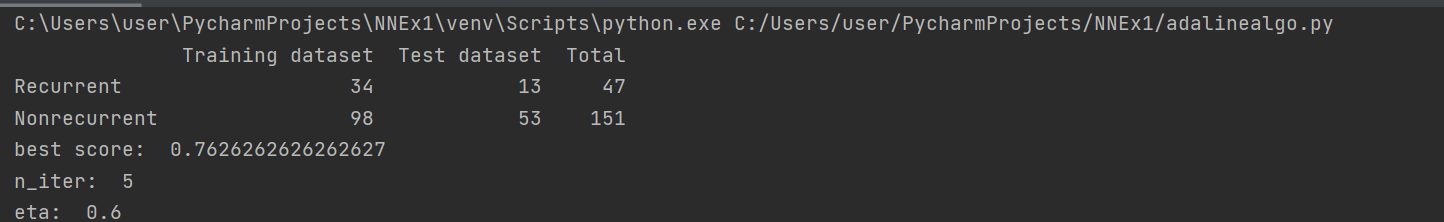
**Adaline:**

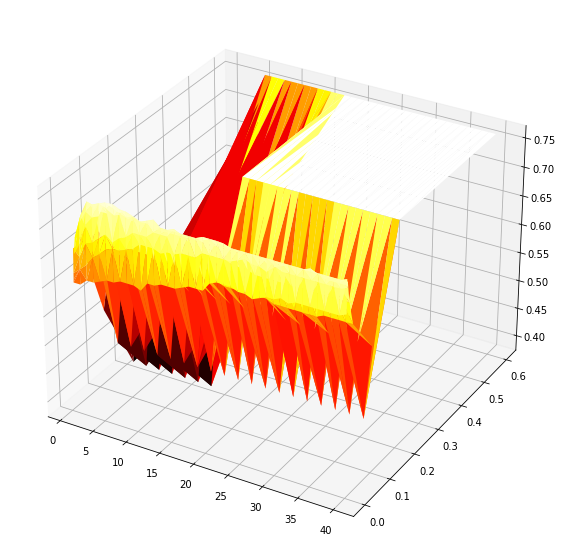
In order to find the best parameters we have run the model each time with different values for the learning rate and for the amount of iterations.

The same method was used also in the perceptron algorithm.

We made a graph shown the highest score and the best parameters.

**Code:**

##Finding best parameters  
i\_range = range(1, 41)  
l\_range = [0.01, 0.005, 0.001, 0.0005, 0.0001, 0.1, 0.025, 1.0, 0.003]  
zdata = []  
xdata = []  
ydata = []  
for i in i\_range:  
 #check every l\_rangechecking it in adaline and add cv score , iter\_n , eta to list  
 for l in l\_range:  
 ada = Adaline(n\_iter=i, eta=l, random\_state=random\_state)  
 ada.fit(X\_train, y\_train)  
 y\_predict = ada.predict(X\_test)  
 cv = cross\_val\_score(ada, X, y, cv=3, scoring='accuracy').mean()  
 zdata.append(cv)  
 xdata.append(i)  
 ydata.append(l)  
matplotlib  
##show it using 3D view  
fig = plt.figure(figsize=(10, 10))  
#https://matplotlib.org / mpl\_toolkits / mplot3d / tutorial.html  
ax = plt.axes(projection='3d')  
#we choose the color cmap  
#for more options see: https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html  
ax.plot\_trisurf(xdata, ydata, zdata, cmap='viridis');  
fig.show()  
#print best score  
x = 0  
for i in zdata:  
 if (max(zdata) == i):  
 break  
 x += 1  
print("best score: ", zdata[x])  
print("n\_iter: ", xdata[x])  
print("eta: ", ydata[x]) 

  
As we can see in the graph, the number of iteration grows the score is lower(unless eta is high you can see the color get lighter), and the eta is around the average.

*The best parameters:*

* **eta** = 0.6
* **n\_iter** =5

**Backpropagation:**

 In order to find the best parameters we run Grid whose gets some optional parameters and returns the best parameters for the highest accuracy.

GridSearch is a function from sklearn library :" https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html"

*The best parameters:*

* **hidden\_layer\_sizes** = (32, 16), two hidden layers with 32 neurons in the first and 16 neurons in the second.
* **for one layer size:** (45)
* **activation** (Activation function for the hidden layer= ' relu'
* **solver** (The solver for weight optimization) = 'adam' (-refers to stochastic gradient descent
* **max\_iter** (Maximum number of iterations. The solver iterates until convergence or this number of iterations) = 50, .

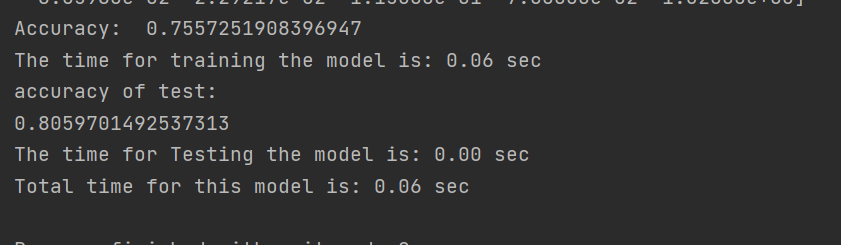
**Performance on the training set and on the testing set:**

Train-Test Split: we split the data into 2 classes : 'Train'-66%, 'verify'-33%.

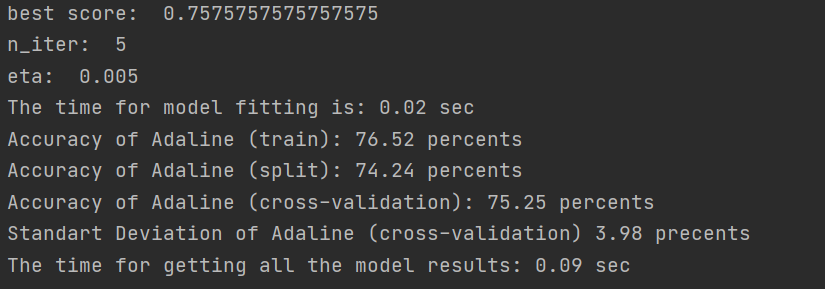
**Cross Validation:**

We run it 3 times and calculate the average and std of the 3 results.

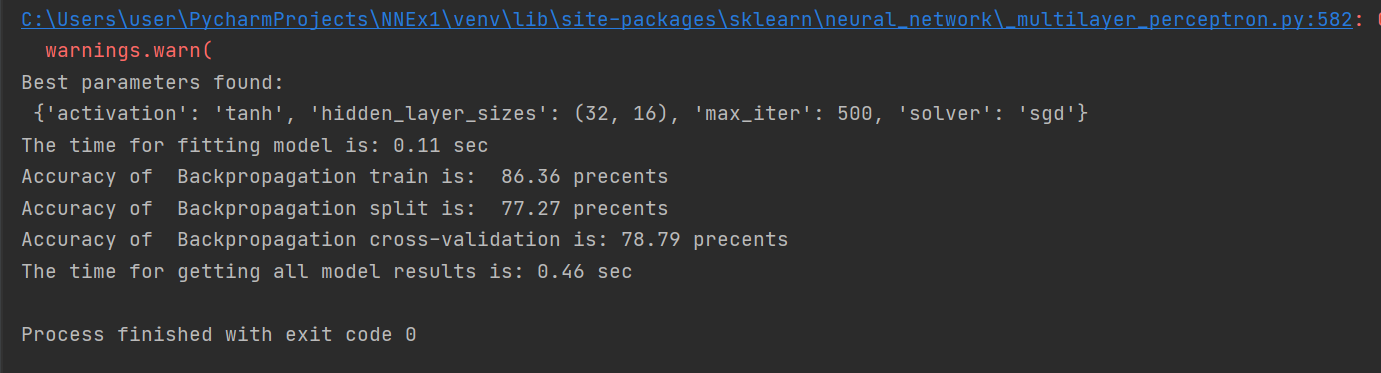
**Perceptron:**

****

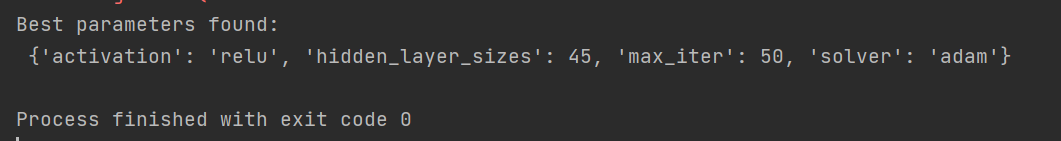
**Adaline:**

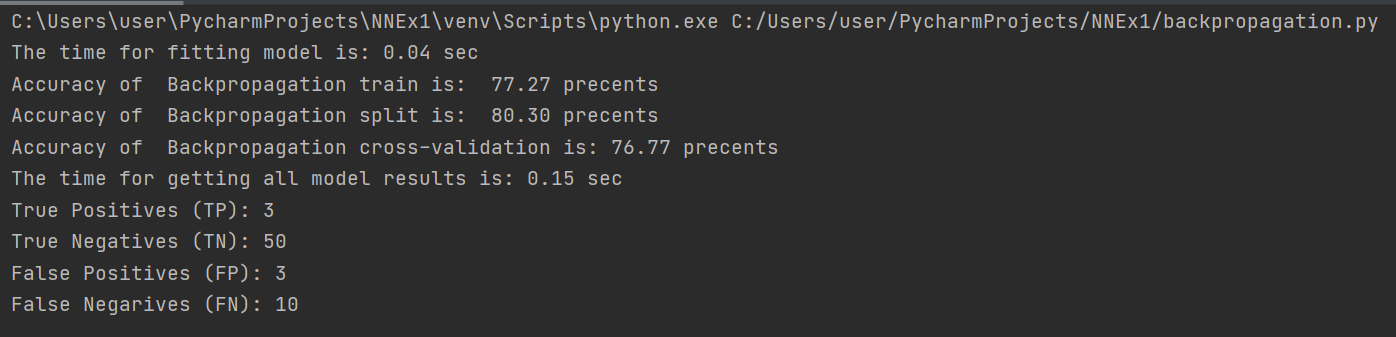
****

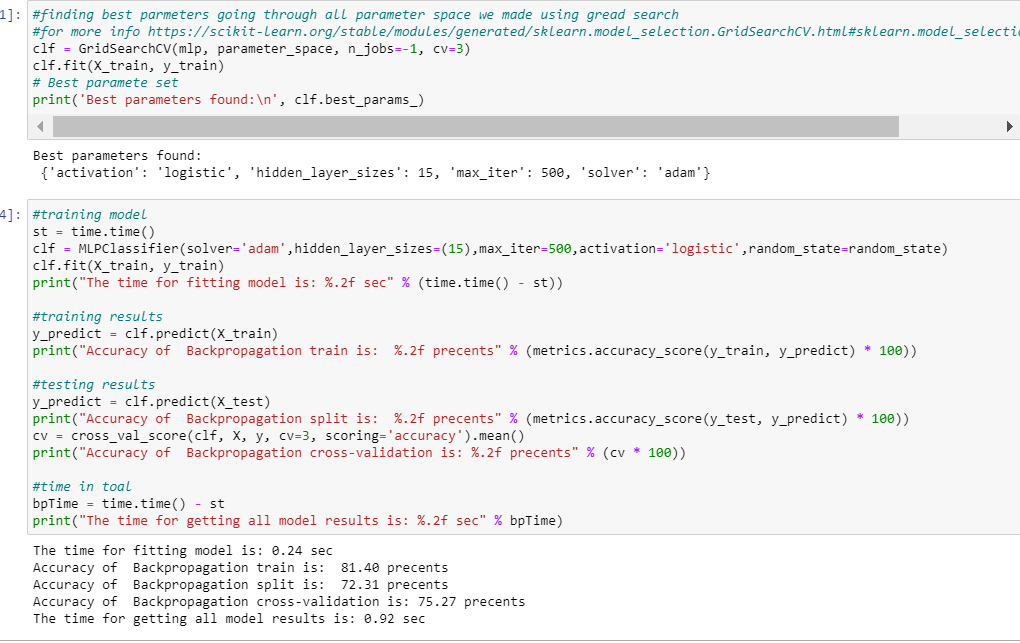
**Backpropagation:**

****

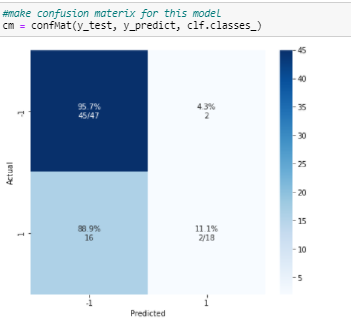
* One layer size:



Score:



* The accuracy of the training set implies that the model doesn't fit perfectly to the training data. We can refer, that the cause is that we have limited data.
* The cross-validation has higher results than the split to train-test. This is due to the fact that cross-validation is more reliable and also we train it few times then we get more accurate results.



Code & Explanations:

We have used Python libraries:

MLPClassifier implementes Backpropagation algorithm.

Import the data from 'wpbc.data' using pandas

### The description of the data:

1. Number of instances: 198
2. Number of attributes: 34 (ID, outcome, 32 real-valued input features)
3. Attribute information  
   1) ID number  
   2) Outcome (R = recur, N = nonrecur)  
   3) Time (recurrence time if field 2 = R, disease-free time if field 2 = N)  
   4-33) Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)

b) texture (standard deviation of gray-scale values)

c) perimeter

d) area

e) smoothness (local variation in radius lengths)

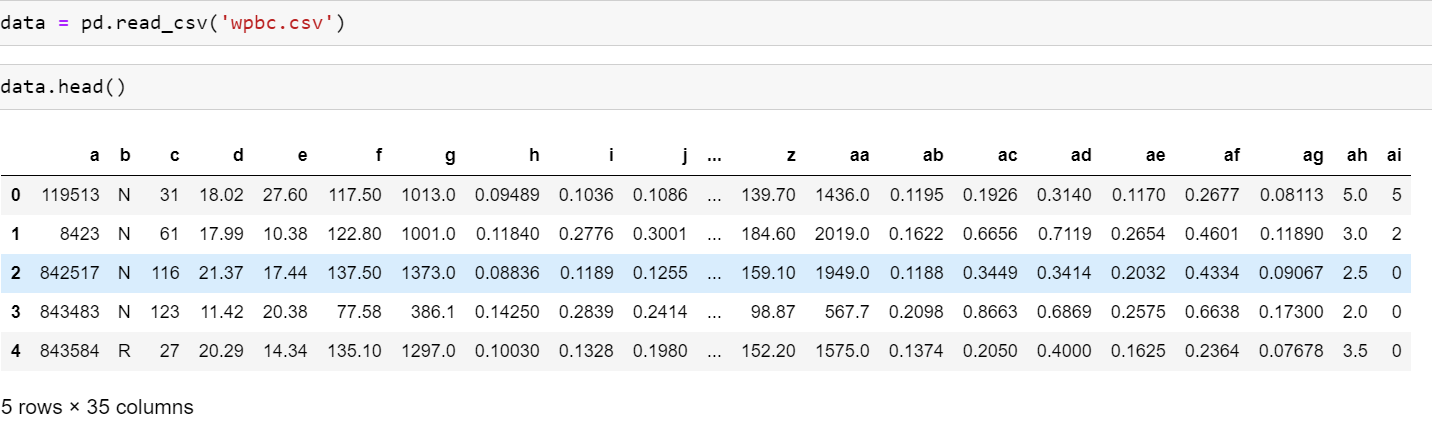
f) compactness (perimeter^2 / area - 1.0)

g) concavity (severity of concave portions of the contour)

h) concave points (number of concave portions of the contour)

i) symmetry

j) fractal dimension ("coastline approximation" - 1)

****

**explain about filtering data:**

we used panda to read the csv into dataframe.

After reading the data we filtered the data in a way that all of the features are presented with numbers and also we dropped few columns that are not needed for the training.

After that we splatted the data as mentioned above.

**Additionally to all above in Adaline and backpropagation we try to filter the data more.**

We made a correlation ,ap of all features given in'wpbc.csv' to try to find the best features for this algorithms.

**Code:** fig **=** plt.figure(figsize**=**(25,17))

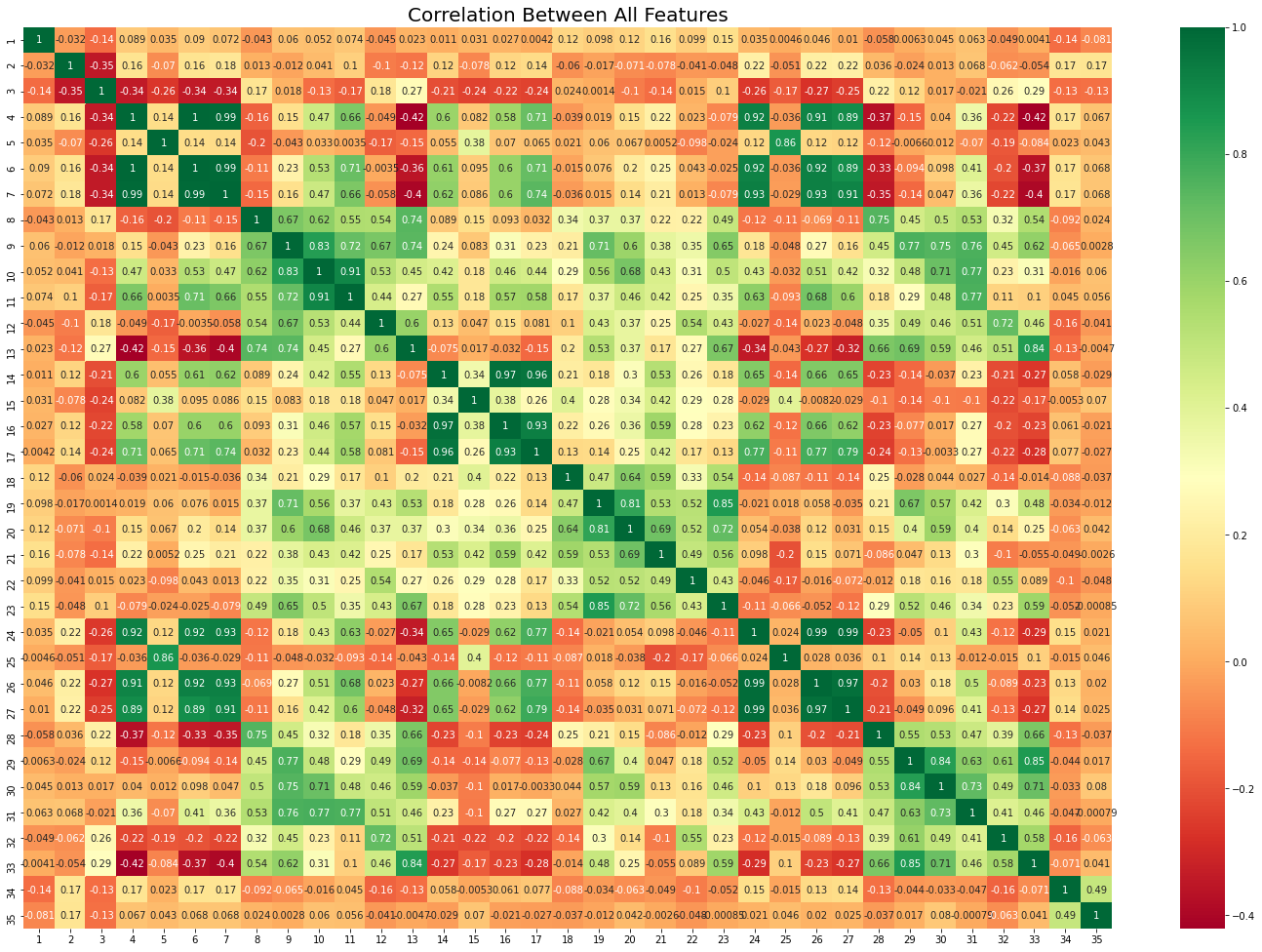
sns.heatmap(data.corr(),annot**=True**,cmap**=**'RdYlGn')

plt.title('Correlation Between All Features', size**=**20)

plt.show()

\*we use plt to show the cor map and use pandas correlation function

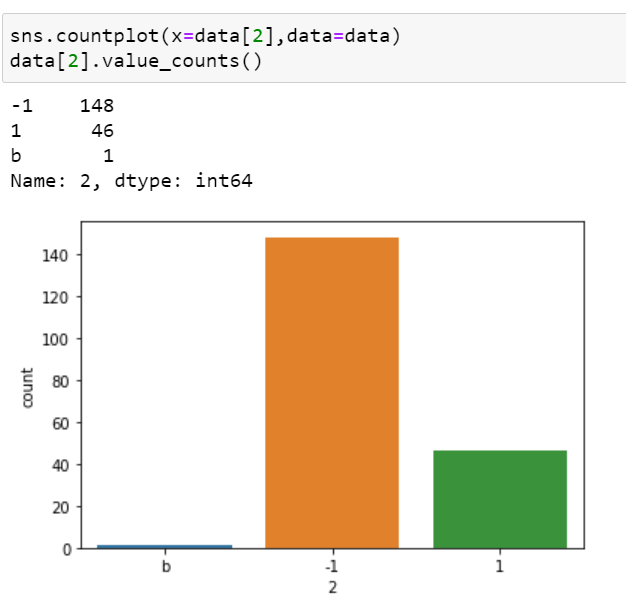
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html

****

The first coloumn is the patients ID and the third coloumn(lables)

According to the map we can see that 1,9,19 has no cor with target coloumn, 2,6,7,2,26 has good correlation with coloumn 4 , 3 ,17 has good correlation with 14 so we choose to give up all this columns at this algorithms.

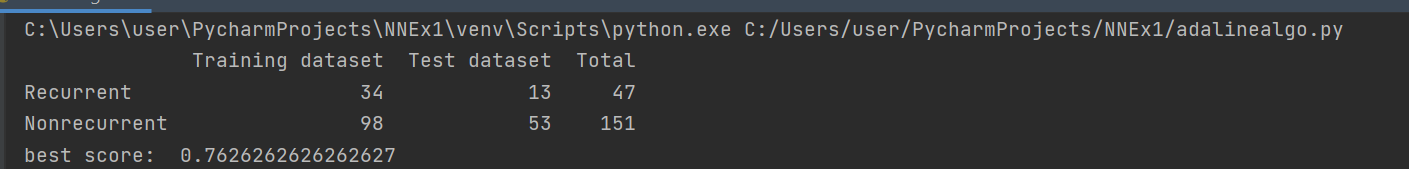
**Pics of R/N amount:**



From the pic above we see that we have 148 patients the are Non-recurrence and 46 the recurrence, we use this information in dividing to train test classes.

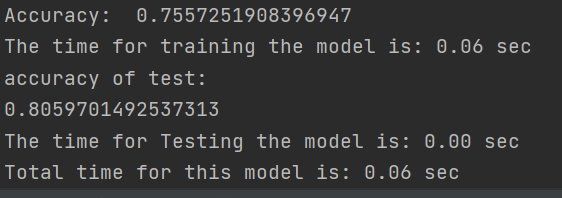
**Code:**

scaler = StandardScaler()  
X = scaler.fit\_transform(data.iloc[:, 1:].values)  
y = data.iloc[:, 0].values  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=random\_state)  
freqs = pd.DataFrame({"Training dataset": [(y\_train == 1).sum(), (y\_train == -1).sum()],  
 "Test dataset": [(y\_test == 1).sum(), (y\_test == -1).sum()],  
 "Total": [(y\_train == 1).sum() + (y\_test == 1).sum(),  
 (y\_train == -1).sum() + (y\_test == -1).sum()]},  
 index=["Recurrent", "Nonrecurrent"])  
freqs[["Training dataset", "Test dataset", "Total"]]

****

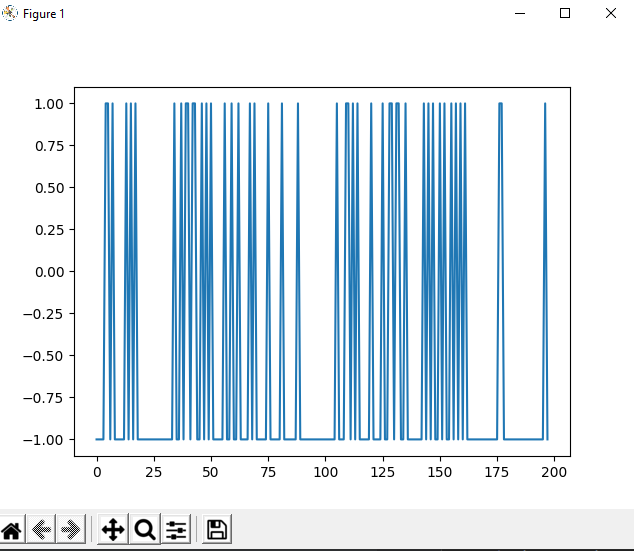
**Cormap pic**

**Perceptron Alorithm:**

****

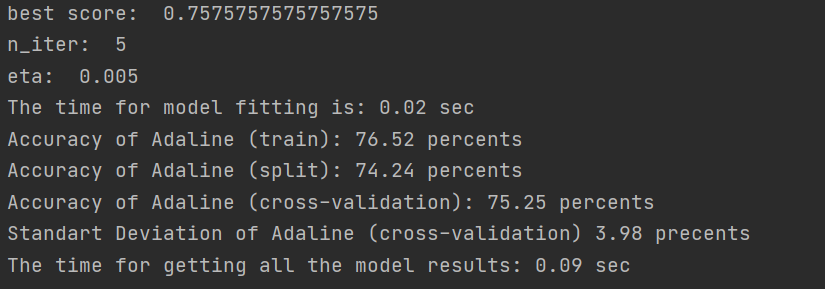
in the perceptron algorithm we noticed that because the data is non-linear the results may vary – some times the new weights improved the accuracy and in other cases it made it worse so we decided to stop the algorithm when it reaches 75% accuracy to avoid it from getting worse.

Also attached plot that shows that the data is non linear- the labels (1,-1) are scattered all over the dataset.



**Adaline algorithm:**

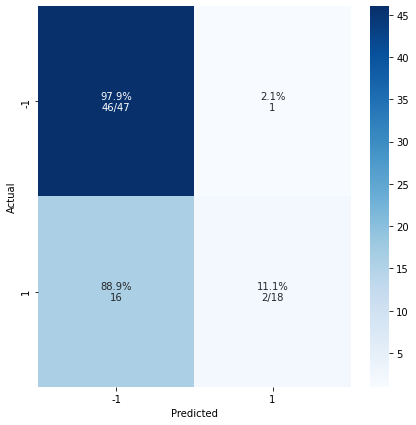
**Score:**



Confusion matrix:

**Code:**

def confMat(y\_true, y\_pred, labels, figsize=(7,7)):  
 #make conusion matrix usingg skl function  
 #for more info https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion\_matrix.html  
 cm = confusion\_matrix(y\_true, y\_pred, labels=labels)  
 cm\_sum = np.sum(cm, axis=1, keepdims=True)  
 cm\_perc = cm / cm\_sum.astype(float) \* 100  
 annot = np.empty\_like(cm).astype(str)  
 nrows, ncols = cm.shape  
 for i in range(nrows):  
 for j in range(ncols):  
 c = cm[i, j]  
 p = cm\_perc[i, j]  
 if i == j:  
 s = cm\_sum[i]  
 annot[i, j] = '%.1f%%\n%d/%d' % (p, c, s)  
 elif c == 0:  
 annot[i, j] = '0'  
 else:  
 annot[i, j] = '%.1f%%\n%d' % (p, c)  
 cm = pd.DataFrame(cm, index=labels, columns=labels)  
 cm.index.name = 'Actual'  
 cm.columns.name = 'Predicted'  
 fig, ax = plt.subplots(figsize=figsize)  
 sns.heatmap(cm, annot=annot, fmt='', ax=ax, cmap='Blues')  
 #plt.savefig(filename)  
 plt.show()  
 return cm  
#make confusion matrix for adaline run  
cm = confMat(y\_test, y\_predict, [-1,1])



**Backpropogation Algorithm:**

MLPClassifier implements a multi-layer perceptron (MLP) algorithm that trains using Backpropagation.

The advantages of Multi-layer Perceptron are:  
Capability to learn non-linear models.  
Capability to learn models in real-time (on-line learning) using partial\_fit.

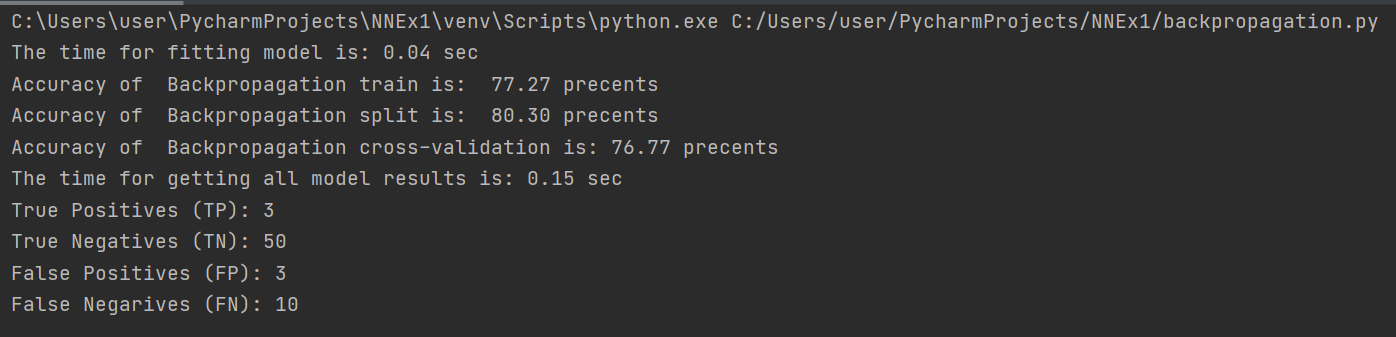
For more about mlp: "https://scikit-learn.org/stable/modules/neural\_networks\_supervised.html"

We use mlp classifier on our 'train test' data we explained above.

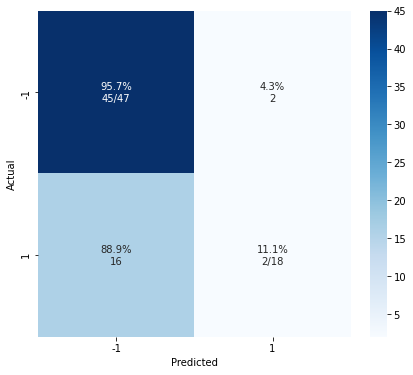
For more info about MLPClassifier**:** [**https://scikit-learn.org/stable/modules/generated**](https://scikit-learn.org/stable/modules/generated)

**/sklearn.neural\_network.MLPClassifier.html**

**BackPropogation results:**

****

**Pic corelation map**

****

**Summary and discussion of process/ problems:**

The goal of our model is to predict if a patient is recurrent or not.

The target column is "b" which hold the information that represents if the patient was recurrent or not. \*R\* =recurrent ,\*N\* = nonrecurrent.

We switched the R to 1 and N to -1.

All of the models are not accurate enough – about 75% and that is after we tried to choose the best data and features as possible in order to get the best results.

Whenever we had missing data in the dataset we added data to the patient by using the average value for the column in order that the missing data wont impact the results.

We have a lot of features in the dataset, so we built a correlation map that shows the correlation between the different features and between them to the labels in order to see if there are features that are most likely not going to change the results so we can remove them to minimize the dataset.

We chose to remove features that have no correlation with the labels and also features with no correlations at all.

We observed that if we had more data and in particular, more data of recurrent patients- the chances of getting higher accuracy of the model grows up because the process will reflect more data about them that is missing in the current dataset.

We found that the best result for this problem with this dataset was achieved using the

Cross-validation and BackPropogation model and was 76.77%