Aprendizagem 2021/22

Homework I – Group 053

I. Pen-and-paper

1) a) Given the tanh activation function, firstly we must derive it.

$$\frac{\partial \tanh(x)}{\partial x} = 1 - tanh^2(x)$$

We are now ready to do forward propagation: $net^{[L]} = W^{[L]} \cdot a^{[L-1]} + b^{[L]}$ and $a^{[L]} = tanh(net^{[L]})$.

From the given dataset, we already know that:

$a^{[0]} = (1 \quad 1 \quad 1 \quad 1 \quad 1)^T$

And from here, we can start computing the forward propagation:

$$net^{[1]} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 6 \\ 1 \\ 6 \end{pmatrix} ; \; a^{[1]} = tanh(\begin{pmatrix} 6 \\ 1 \\ 6 \end{pmatrix}) = \begin{pmatrix} 0.99998771165 \\ 0.76159415595 \\ 0.99998771165 \end{pmatrix}$$

Similarly, we compute the *net* and *a* matrices of the following layers:

$$net^{[2]} = \begin{pmatrix} 3.76156957926 \\ 3.76156957926 \end{pmatrix} ; \; a^{[2]} = \begin{pmatrix} 0.99891971703 \\ 0.99891971703 \end{pmatrix}$$

$$net^{[3]} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} ; \; a^{[3]} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Recalling the squared error loss function:

$$E(y, a^{[L]}) = \frac{1}{2}\sum_{i=1}^{2}(a_i^{[L]} - y_i)^2 = \frac{1}{2}(a^{[L]} - y)^2$$

Now we want to do the backward phase. In general, we will need to know how to derive all functions in our network, so let us compute them beforehand:

$$\frac{\partial E}{\partial x^{[L]}}(t, a^{[L]}) = \frac{\partial E}{\partial (a^{[L]}-y)^2} \cdot \frac{\partial (a^{[L]}-y)^2}{\partial (a^{[L]}-y)} \cdot \frac{\partial (a^{[L]}-y)}{\partial a^{[L]}} = \frac{1}{2} \cdot 2(a^{[L]} - y) \cdot 1 = a^{[L]} - y$$

$$\frac{\partial a^{[L]}}{\partial net^{[L]}}(net^{[L]}) = 1 - tanh^2(net^{[L]}) \qquad \frac{\partial net^{[L]}}{\partial W^{[L]}}(W^{[L]}, b^{[L]}, a^{[L-1]}) = a^{[L-1]}$$

$$\frac{\partial net^{[L]}}{\partial b^{[L]}}(W^{[L]}, b^{[L]}, a^{[L-1]}) = 1 \qquad \frac{\partial net^{[L]}}{\partial a^{[L-1]}}(W^{[L]}, b^{[L]}, a^{[L-1]}) = W^{[L]}$$

To start the recursion, we need the delta from the last layer:

$$\delta^{[3]} = \frac{\partial E}{\partial a^{[3]}} \circ \frac{\partial a^{[3]}}{\partial net^{[3]}} = (a^{[3]} - y) \circ (1 - \tanh^2(net^{[3]}))$$

$$= \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 \\ -1 \end{pmatrix}\right) \circ \left(\begin{pmatrix} 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \end{pmatrix}^2\right)$$

$$= \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

Now we can start the recursion to obtain other layers' deltas:

$$\delta^{[2]} = \frac{\partial net^{[3]}}{\partial a^{[2]}}^T \cdot \delta^{[3]} \circ \frac{\partial a^{[2]}}{\partial net^{[2]}} = (W^{[3]})^T \cdot \delta^{[3]} \circ \left(1 - \tanh^2(net^{[2]})\right) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\delta^{[1]} = \frac{\partial \mathrm{net}^{[2]^{\mathrm{T}}}}{\partial a^{[1]}} \cdot \delta^{[2]} \circ \frac{\partial a^{[1]}}{\partial \mathrm{net}^{[1]}} = \left(\mathrm{W}^{[2]}\right)^{\mathrm{T}} \cdot \delta^{[2]} \circ \left(1 - \tanh^2\left(\mathrm{net}^{[1]}\right)\right) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Finally, we can go to the last phase and perform the updates. We start with the first layer:

$$\frac{\partial \mathrm{E}}{\partial W^{[1]}} = \delta^{[1]} \cdot \frac{\partial net^{[1]^T}}{\partial \mathrm{W}^{[1]}} = \delta^{[1]} \cdot (a^{[0]})^T = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\frac{\partial \mathrm{E}}{\partial b^{[1]}} = \delta^{[1]} \cdot \frac{\partial net^{[1]^T}}{\partial \mathrm{b}^{[1]}} = \delta^{[1]} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Therefore, we have:

$$W_{new}^{[1]} = W_{old}^{[1]} - \eta \frac{\partial \mathrm{E}}{\partial W^{[1]}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad b_{new}^{[1]} = b_{old}^{[1]} - \eta \frac{\partial \mathrm{E}}{\partial b^{[1]}} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Similarly, we update the second layer:

$$\frac{\partial \mathrm{E}}{\partial W^{[2]}} = \delta^{[2]} \cdot \frac{\partial net^{[2]^T}}{\partial \mathrm{W}^{[2]}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \frac{\partial \mathrm{E}}{\partial b^{[2]}} = \delta^{[2]} \cdot \frac{\partial net^{[2]^T}}{\partial \mathrm{b}^{[2]}} = \delta^{[2]} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$W_{new}^{[2]} = W_{old}^{[2]} - \eta \frac{\partial \mathrm{E}}{\partial W^{[2]}} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad b_{new}^{[2]} = b_{old}^{[2]} - \eta \frac{\partial \mathrm{E}}{\partial b^{[2]}} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

And lastly, all that is left to do is to update the parameters for the output layer:

$$\frac{\partial \mathrm{E}}{\partial W^{[3]}} = \delta^{[3]} \cdot \frac{\partial net^{[3]^T}}{\partial \mathrm{W}^{[3]}} = \delta^{[3]} \cdot (a^{[2]})^T = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0.99891971703 & 0.99891971703 \end{pmatrix}$$

$$= \begin{pmatrix} -0.99891971703 & -0.99891971703 \\ 0.99891971703 & 0.99891971703 \end{pmatrix}$$

$$\frac{\partial \mathrm{E}}{\partial b^{[3]}} = \delta^{[3]} \cdot \frac{\partial net^{[3]^T}}{\partial \mathrm{b}^{[3]}} = \delta^{[3]} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

And hence we have:

$$W_{new}^{[3]} = W_{old}^{[3]} - \eta \frac{\partial \mathrm{E}}{\partial W^{[3]}} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} -0.99891971703 & -0.99891971703 \\ 0.99891971703 & 0.99891971703 \end{pmatrix}$$

$$= \begin{pmatrix} 0.099891971703 & 0.099891971703 \\ -0.099891971703 & -0.099891971703 \end{pmatrix}, \text{ and:}$$

$$b_{new}^{[3]} = b_{old}^{[3]} - \eta \frac{\partial \mathrm{E}}{\partial b^{[3]}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.1 \\ -0.1 \end{pmatrix}$$

b) Recalling the *softmax* function:

$$softmax([net_1 \quad net_2 \quad \ldots \quad net_D]^T) = [a_1 \quad a_2 \quad \ldots \quad a_D]^T, \text{ where } a_i = \frac{exp(net_i)}{\sum_{k=1}^{D} exp(net_k)}$$

Before deriving this activation function, we must notice that each a_i depend on every net_i. Therefore, we must find a general derivative with respect to net_i.

$$\frac{\partial a_i}{\partial net_j} = \frac{\partial}{\partial net_j} \frac{exp(net_i)}{\sum_{k=1}^{D} exp(net_k)}$$

If we have $i = j$:

$\frac{\partial a_i}{\partial net_j} = a_i(1 - a_i)$

If we have $i \neq j$:

$\frac{\partial a_i}{\partial net_j} = -a_i \cdot a_j$

We can now do the forward propagation:

$$net^{[L]} = W^{[L]} \cdot a^{[L-1]} + b^{[L]} \text{ and } a^{[L]} = \begin{cases} tanh(net^{[L]}), if\ L < 3 \\ softmax(net^{[L]}), if\ L = 3 \end{cases}$$

$a^{[0]} = (1 \quad 1 \quad 1 \quad 1 \quad 1)^T$

Since we know that the *softmax* activation function is only applied at the output layer, and everything else is as it was in 1a), we have that:

$$net^{[1]} = \begin{pmatrix} 6 \\ 1 \\ 6 \end{pmatrix} \qquad net^{[2]} = \begin{pmatrix} 3.76156957926 \\ 3.76156957926 \end{pmatrix} \qquad net^{[3]} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$a^{[1]} = \begin{pmatrix} 0.99998771165 \\ 0.76159415595 \\ 0.99998771165 \end{pmatrix} \qquad a^{[2]} = \begin{pmatrix} 0.99891971703 \\ 0.99891971703 \end{pmatrix}$$

And now, we have that:

$$a^{[3]} = softmax(net^{[3]}) = softmax\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}\right) = \begin{pmatrix} \frac{e^0}{e^0 + e^0} \\ \frac{e^0}{e^0 + e^0} \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$$

Noticing that the error loss function has also changed, we recall the cross-entropy loss function:

$$E(y, a^{[3]}) = -\sum_{i=1}^{2} y_i log(a_i^{[3]})$$

We can now start backward propagation. What changes in this case is the delta of the last layer, whereas in the remaining ones we'll use the same method to calculate its deltas as we did in 1a). In order to get $\delta^{[3]}$ we must derive our error loss function with respect to net_i.

$\delta_i^{[3]} = \frac{\partial E(y, a^{[3]})}{\partial\, net_i} = a_i^{[3]} - y_i$, whereby definition we obtain that $\delta^{[3]} = \frac{\partial E(y, a^{[3]})}{\partial\, net_i} = a^{[3]} - y$

The remaining derivatives will be computed as before:

$\frac{\partial a^{[L]}}{\partial \mathrm{net}^{[L]}}(\mathrm{net}^{[L]}) = 1 - tanh^2(\mathrm{net}^{[L]})$

$\frac{\partial net^{[L]}}{\partial \mathrm{W}^{[L]}}(\mathrm{W}^{[L]}, \mathrm{b}^{[L]}, a^{[L-1]}) = a^{[L-1]}$

$\frac{\partial net^{[L]}}{\partial \mathrm{b}^{[L]}}(\mathrm{W}^{[L]}, \mathrm{b}^{[L]}, a^{[L-1]}) = 1$

$\frac{\partial net^{[L]}}{\partial \mathrm{a}^{[L-1]}}(\mathrm{W}^{[L]}, \mathrm{b}^{[L]}, a^{[L-1]}) = \mathrm{W}^{[L]}$

The recursion is as follows:

$$\delta^{[3]} = a^{[3]} - y = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -0.5 \\ 0.5 \end{pmatrix}$$

$$\delta^{[2]} = \left(\mathrm{W}^{[3]}\right)^{\mathrm{T}} \cdot \delta^{[3]} \circ \left(1 - \tanh^2(\mathrm{net}^{[2]})\right) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} -0.5 \\ 0.5 \end{pmatrix} \circ \left(\begin{pmatrix} 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 3.76156957926 \\ 3.76156957926 \end{pmatrix}^2\right) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\delta^{[1]} = \left(\mathrm{W}^{[2]}\right)^{\mathrm{T}} \cdot \delta^{[2]} \circ \left(1 - \tanh^2(\mathrm{net}^{[1]})\right) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \end{pmatrix} \circ \left(\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 6 \\ 1 \\ 6 \end{pmatrix}^2\right) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Finally, we can go to the last phase and perform the updates. Starting off with the first layer (same as 1a)):

$$\frac{\partial E}{\partial W^{[1]}} = \delta^{[1]} \cdot \frac{\partial net^{[1]}}{\partial W^{[1]}}^{T} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \frac{\partial E}{\partial b^{[1]}} = \delta^{[1]} \cdot \frac{\partial net^{[1]}}{\partial b^{[1]}}^{T} = \delta^{[1]} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$W^{[1]}_{new} = W^{[1]}_{old} - \eta \frac{\partial E}{\partial W^{[1]}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad b^{[1]}_{new} = b^{[1]}_{old} - \eta \frac{\partial E}{\partial b^{[1]}} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Off to the second layer (also same as 1a)):

$$\frac{\partial E}{\partial W^{[2]}} = \delta^{[2]} \cdot \frac{\partial net^{[2]}}{\partial W^{[2]}}^{T} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \frac{\partial E}{\partial b^{[2]}} = \delta^{[2]} \cdot \frac{\partial net^{[2]}}{\partial b^{[2]}}^{T} = \delta^{[2]} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$W^{[2]}_{new} = W^{[2]}_{old} - \eta \frac{\partial E}{\partial W^{[2]}} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad b^{[2]}_{new} = b^{[2]}_{old} - \eta \frac{\partial E}{\partial b^{[2]}} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

And lastly for the output layer:

$$\frac{\partial E}{\partial W^{[3]}} = \delta^{[3]} \cdot \frac{\partial net^{[3]}}{\partial W^{[3]}}^{T} = \delta^{[3]} \cdot (a^{[2]})^{T} = \begin{pmatrix} -0.5 \\ 0.5 \end{pmatrix} \cdot \begin{pmatrix} 0.99891971703 & 0.99891971703 \end{pmatrix}$$

$$= \begin{pmatrix} -0.49945985851 & -0.49945985851 \\ 0.49945985851 & 0.49945985851 \end{pmatrix}$$

$$\frac{\partial E}{\partial b^{[3]}} = \delta^{[3]} \cdot \frac{\partial net^{[3]}}{\partial b^{[3]}}^{T} = \delta^{[3]} = \begin{pmatrix} -0.5 \\ 0.5 \end{pmatrix}$$

Which leads to us to:

$$W^{[3]}_{new} = W^{[3]}_{old} - \eta \frac{\partial E}{\partial W^{[3]}} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} -0.49945985851 & -0.49945985851 \\ 0.49945985851 & 0.49945985851 \end{pmatrix}$$

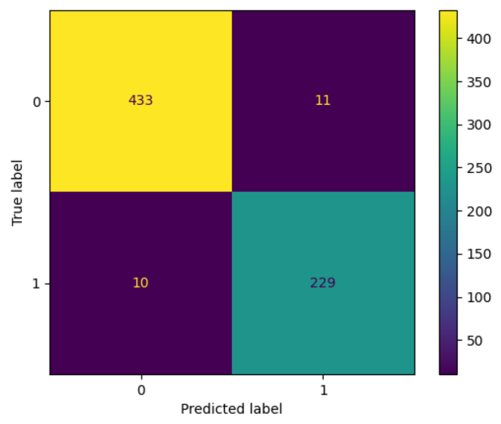
$= \begin{pmatrix} 0.049945985851 & 0.049945985851 \\ -0.49945985851 & -0.049945985851 \end{pmatrix}$, and:

$$b^{[3]}_{new} = b^{[3]}_{old} - \eta \frac{\partial E}{\partial b^{[3]}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} -0.5 \\ 0.5 \end{pmatrix} = \begin{pmatrix} 0.05 \\ -0.05 \end{pmatrix}$$

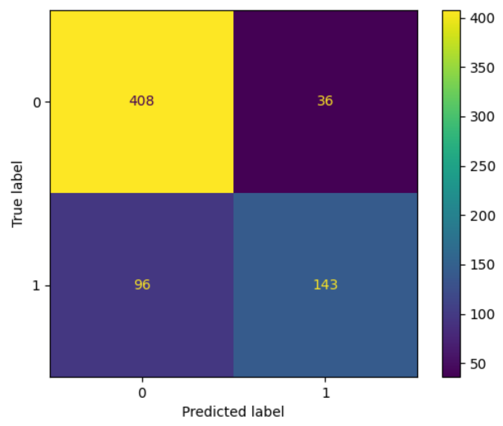
II. Programming and critical analysis

2)

Without early stopping:



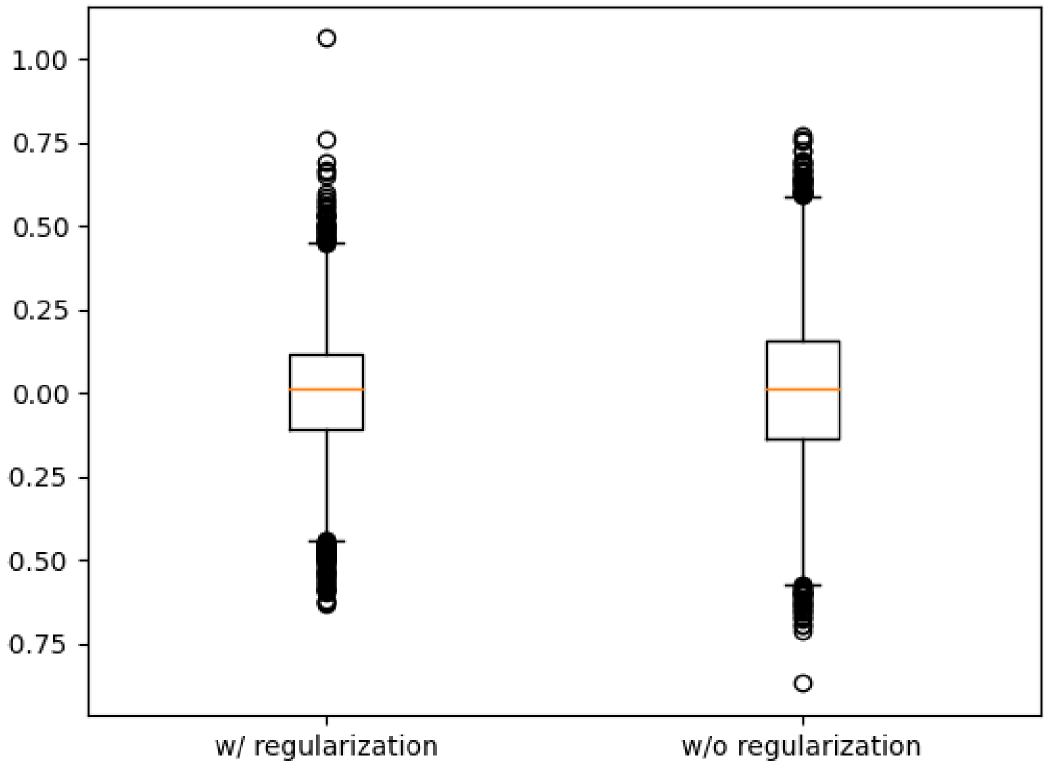
With early stopping:



Observing both matrices there is a better performance (higher accuracy) for the model without early stopping. Usually, early stopping prevents overfitting achieving better results. However, the small size of dataset used is too small to properly obtain the global minimum or a close enough local one.

Another possible reason may reside on the mixing of cross-validation with early stopping as they are at odds because cross-validation assumes we don't know the generalization error and early stopping is trying to fit us the best model based on the generalization error. We could instead use the number of training epochs as a hyperparameter over the use of early stopping to repeatedly evaluate the performance of the model.

3) The boxplot is as shown:



According to the box plot, when applying regularization to the model we obtain a smaller interquartile range, even though there are no significant changes to the number of outliers. We could try to improve the model's performance (reduce its error) by a variety of ways. One would be to change the initial weights and learning rates as to not land on an irrelevant local minimum. Since the dataset is big enough, we could also use early stopping to avoid overfitting to our training set. Also, we could normalize the data to reduce the effect of noise and/or remove outliers.

III. APPENDIX

```
# 2
data = arff.loadarff('breast.w.arff')
df = pd.DataFrame(data[0])
X = df.iloc[:,:-1].values.tolist()
y = df.iloc[:,-1].values.tolist()

cv_fold = KFold(n_splits=5, random_state=0, shuffle=True)
classifier_nes = MLPClassifier(hidden_layer_sizes=(3, 2), max_iter=1500,
early_stopping=False, random_state=53)
classifier_es = MLPClassifier(hidden_layer_sizes=(3, 2), max_iter=1500,
early_stopping=True, random_state=53)
prediction_nes = cross_val_predict(classifier_nes, X, y, cv = cv_fold)
prediction_es = cross_val_predict(classifier_es, X, y, cv = cv_fold)

cm_nes = metrics.confusion_matrix(y, prediction_nes)
cm_es = metrics.confusion_matrix(y, prediction_es)

disp_nes = ConfusionMatrixDisplay(confusion_matrix=cm_nes).plot()
plt.savefig('cm_nes.png')

plt.clf()

disp_nes = ConfusionMatrixDisplay(confusion_matrix=cm_es).plot()
plt.savefig('cm_es.png')
```

```
# 3
data = arff.loadarff('kin8nm.arff')
df = pd.DataFrame(data[0])
X = df.iloc[:,:-1].values.tolist()
y = df.iloc[:,-1].values.tolist()

cv_fold = KFold(n_splits=5, random_state=0, shuffle=True)
regressor_reg = MLPRegressor(hidden_layer_sizes=(3, 2), max_iter=1500, alpha=0.1)
regressor_noreg = MLPRegressor(hidden_layer_sizes=(3, 2), max_iter=1500, alpha=0)
prediction_reg = cross_val_predict(regressor_reg, X, y, cv=cv_fold)
prediction_noreg = cross_val_predict(regressor_noreg, X, y, cv=cv_fold)

residuals_reg = [a_i - b_i for a_i, b_i in zip(y, prediction_reg)]
residuals_noreg = [a_i - b_i for a_i, b_i in zip(y, prediction_noreg)]

plt.boxplot([residuals_reg, residuals_noreg])
plt.xticks([1,2], ['w/ regularization', 'w/o regularization'])
plt.savefig('boxplot.png')
```

END