Machine Learning – Assignment 1

Matteo Bottacini

April 26, 2020

To assess the three tasks regarding the regression problem has been provided a linear model (`T1`), a non-linear model (`T2`) and explanation to answer why I think this neural network is better than the baseline model (`T3`).

The report is structured in this way:

- In the first chapter is discussed the linear model: how it works, and the performance obtained.
- In the second chapter the non-linear model: how it works, and the performance obtained.
- In the third one which model is statistically better through t-test and p-value.
- In the fourth one an explanation is given as to why the neural network model obtained a better performance than the random forest one.

For both models the initial dataset has been split in two different sets:

1. Training set (`80%`);
2. Test set (`20%`).

The training set is used to fit the parameters and then test set to predict the output and analyze the performance of the model.
If the model is unable to acquire the relationship between features and labels it is subjected to training data underfitting. Otherwise, if the model works well with the training set but is unable to generalize data it hasn't seen it is subjected to overfitting.

The whole project is run in Python 3.7 and modules used are: Numpy, Pandas, Sklearn, Tensorflow, Keras, Scipy, Joblib.

Inside the `data` folder there are data that are used to train and evaluate both models.
Inside `deliverable` there are two `.pickle` files containing the linear and the non-linear model respectively created from `linear_regression.py` and `nonlinear_regression.py` which are in the `src` folder.
These two python scripts create the models here described.
In `utils.py` are written some important function that enable the user to save and load models created with Keras and Sklearn.

In the end, inside `deliverable` there's `run_model.py`.
Once the model are stored in `deliverable` as `.pickle` files running this code models are evaluated and MSEs is printed out as well as which model is statistically better and the reason.

Models used for the analysis descripted in this report are already stored. Running `run_model.py` enable the user to achieve the same results.

1. Linear model

1.1. OLS method

To assess the task has been used the Ordinary least squares (OLS) method for estimating the unknown parameters in a linear regression model.
The principle of least squares is minimizing the sum of squares of the differences between the observed training set and those predicted by the linear model.

Given a function in vector form:

$$y = X\beta$$

Where β is a `4x1` vector of unknown parameters and `X` is `nx3` matrix.
The function to minimize is:

$$\widehat{\beta} = \arg\min_{\beta} S(\beta)$$

$$S(\beta) = \sum_{i=1}^{n} |y_i - \sum_{j=1}^{p} X_{ij}\,\beta_j|^2 = ||y - X\beta||^2$$

The solution is given by:

$$(X^T X)\hat{\beta} = X^T y$$

$\hat{\beta}$ is the coefficient vector of the least-squared and is called the OLS estimator:

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

Since OLS is invariant there is no need for standardization the training set.
The coefficients estimates rely on the independence of the features. When features are correlated and columns of the matrix `X` are linearly dependent, the design matrix becomes close to singular and as a result the estimator becomes highly sensitive to random errors in the observed target, producing a large variance.
Since the features of our dataset are not linearly dependent there is no need for regularization, like Ridge or LASSO.

Given the dataset the coefficients estimates obtained using the linear regression from sklearn are:

theta_0	1.5290488509016684
theta_1	-0.18889522
theta_2	-0.37143427
theta_3	-01185147

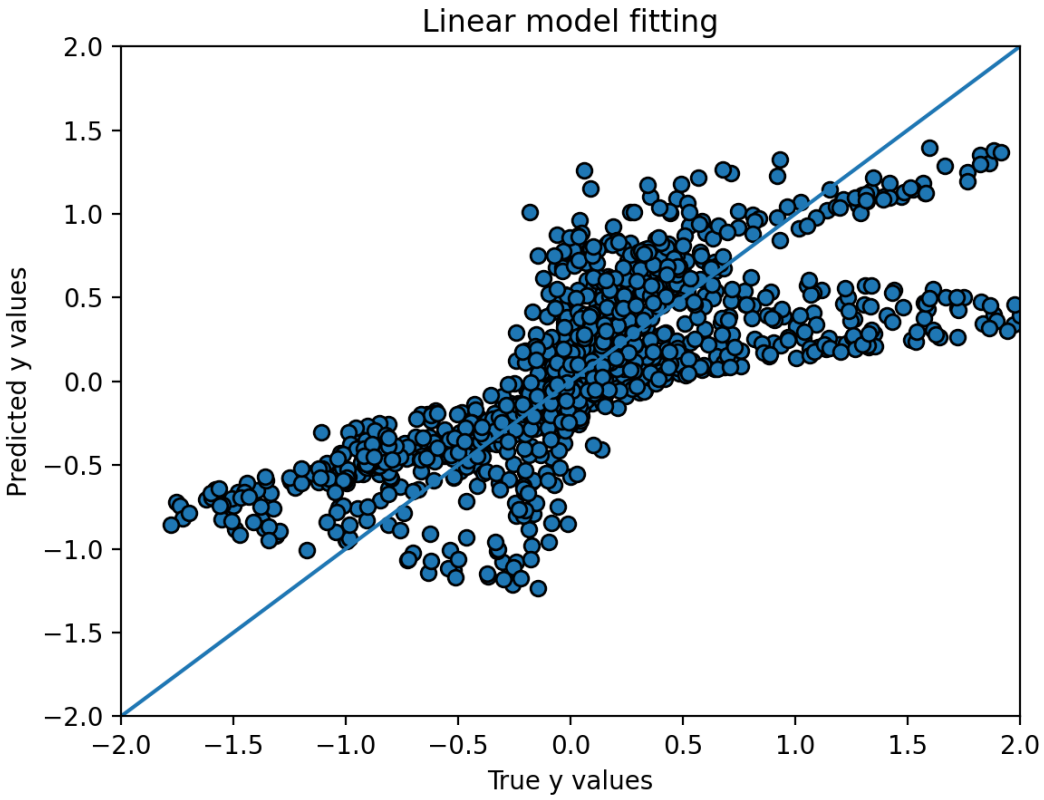
1.2. Performance and error distribution

The mean squared error (MSE) has been used as measure of performance. It indicates the mean square discrepancy between the observed data values and the estimated data values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2$$

The MSE obtained is: `0.20669043719895588`

To visualize the result let's consider a scatterplot where on the `x-axis` lie the `true y values`, and on the `y-axis` lie the `predicted y values`.
The more the points lie on the diagonal the more the result is correct.
If all the points lie on the line with slope 1 the model generalize really well.

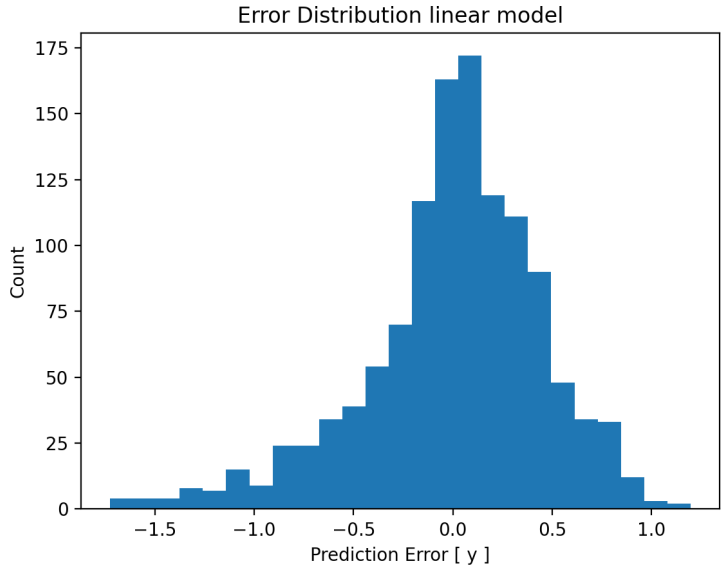


There is a lot of dispersion and this means that this model doesn't explain the relationship between features and labels really well.

Inspect the distribution of the error which is evaluated as:

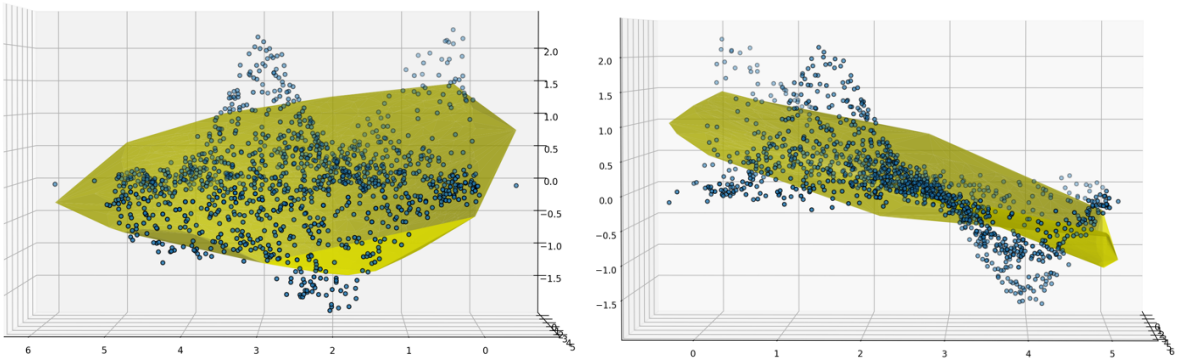
$$error = y - \hat{y}$$

It's not quite a gaussian, and the skewness, as measure of the asymmetry, is negative.
The left tail is longer, the mass of the distribution is concentrated on the right of the figure. The distribution is said to be left-skewed, despite the fact that the curve itself appears to be skewed or leaning to the right.



mean	0.002759
std	0.454852
min	-1.727896
25%	-0.199077
50%	0.043812
75%	0.295115
max	1.188542

Then, since there are two features and one label it is possible to visualize the regression problem in a three-dimensional space.
Here we have another demonstration that the linear model doesn't fit well the data and that this linear model is not a good solution for this task.



2. Non-linear model

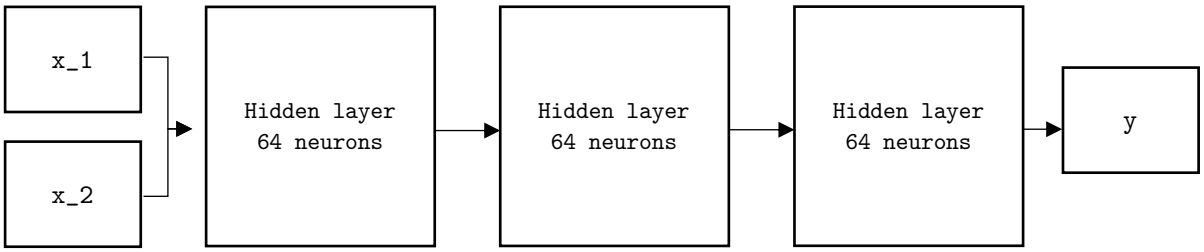
2.1. Neural Network structure and architecture

To assess the task, among many different models has been used a sequential model in Keras as a sequence of layers, the so-called neural network (NN).

The network architecture is composed by three densely connected hidden layers and an output layers that returns a single continuous value.

The input layer features for training are those in the training set. Fully connected layers are defined using the Dense class with 64 neurons. The activation function for the three hidden layers is the rectified linear unit (ReLU).

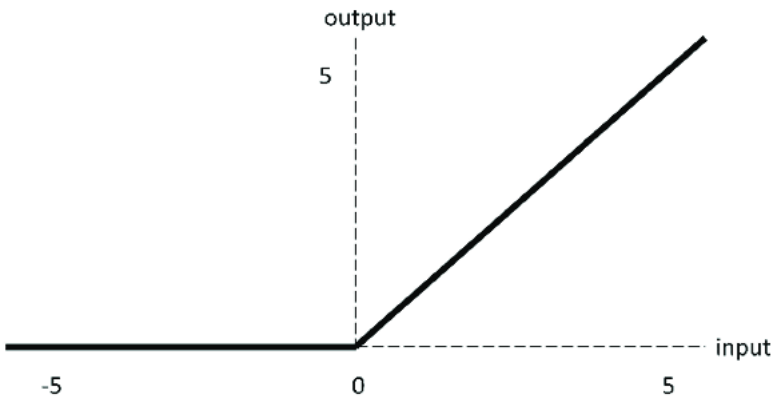
Here is the NN architecture described.



ReLU is defined as:

$$y = \max(0, x)$$

$$\frac{\partial ReLU(x)}{\partial x} = \begin{cases} 0, & x < 0 \\ 1, & x > 0 \end{cases}$$



Source: https://www.researchgate.net

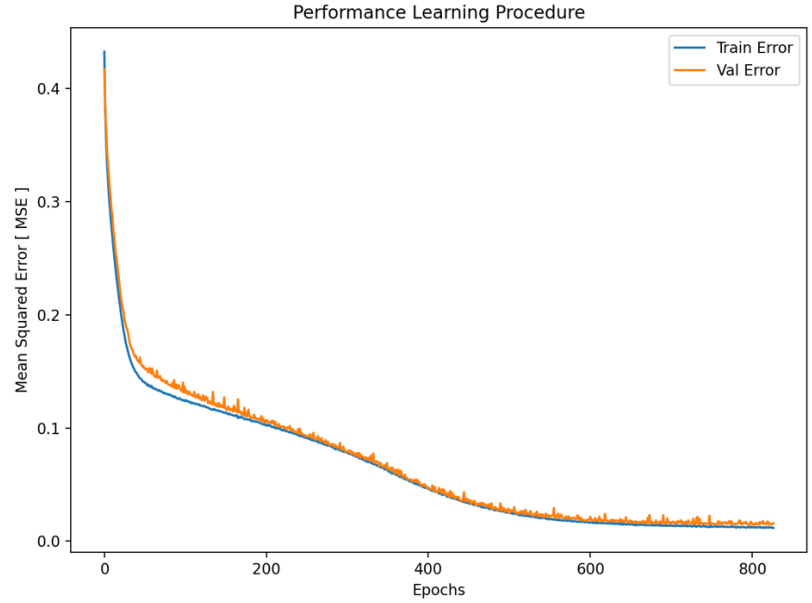
As shown in the figure and described mathematically, ReLU is linear for all positive values and zero for all negative values. This is an advantage in computational terms since it converges faster. It is linear so it doesn't have the vanishing gradient problem like sigmoid or tanh. Since it is zero for all negative values once a neuron gets negative it's unlikely for it to become positive, and the risk is that a large

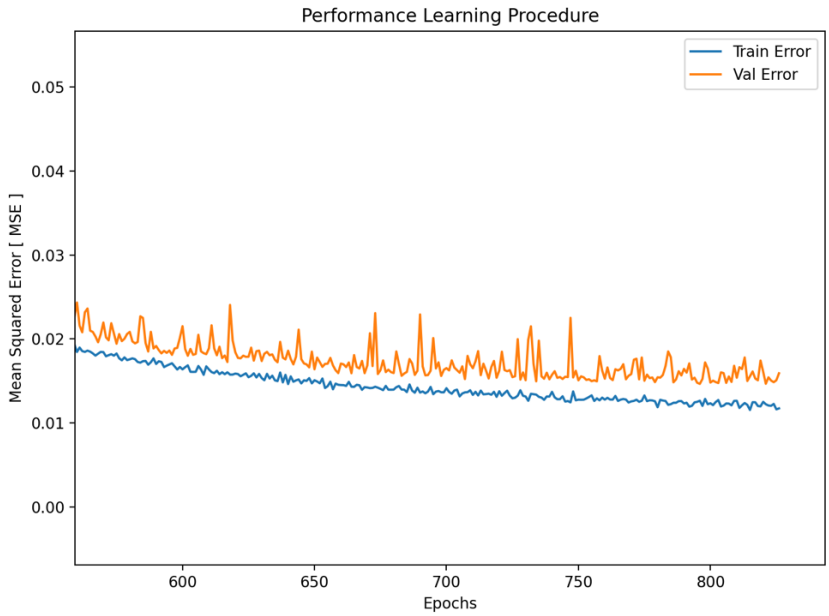
part of the NN is useless. This problem could be avoided with a small learning rate (in this case: `0.0001`). As optimizer has been used RMSprop optimizer which divide the gradient by a running average of its recent magnitude. The loss function to minimize which is the objective function is the MSE and is monitored during the training procedure.

Then, to train the model, data are normalized. Those statistics used to normalize sets are intentionally generated from only the training set and will be used to normalize the test set too. It's done to project the test set into the same distribution that the model has been trained on.

2.2. Performance and error distribution

The model is trained for at most `1000` epochs and to avoid the overfitting problem the early stopping is implemented. In case of overfitting the error on the validation set increase and the learning procedure stops. Here are the performances of the learning procedure of the model, and as shown, the training interrupt just over `800` epochs.

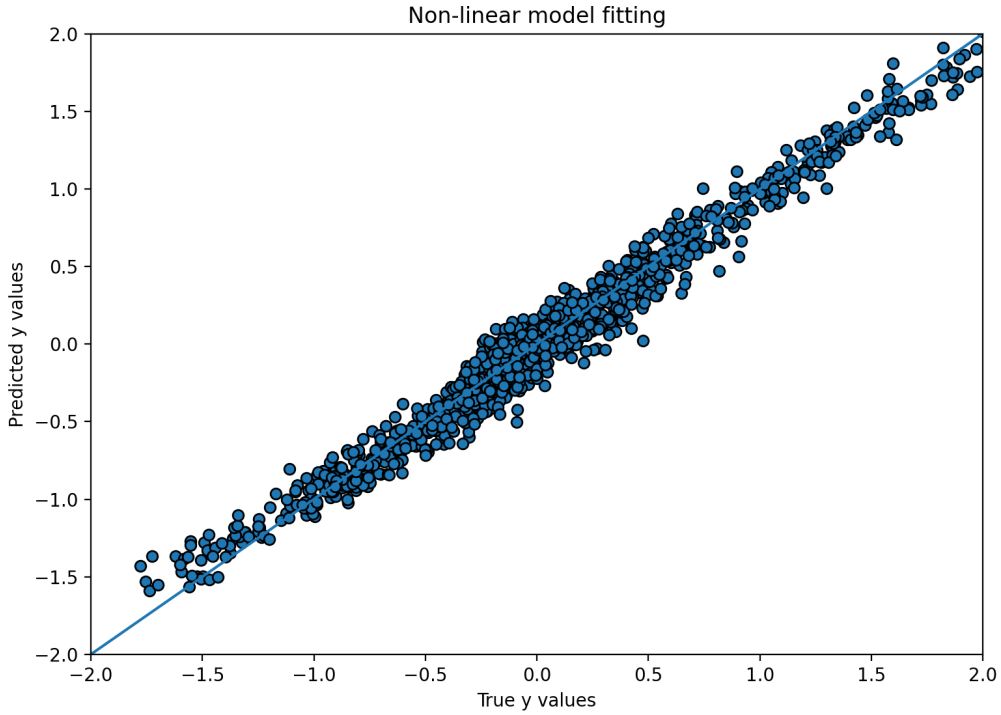




The MSE has been used as measure of performance, as previous in the case of the linear model.

The MSE obtained is: `0.01291963024587587`

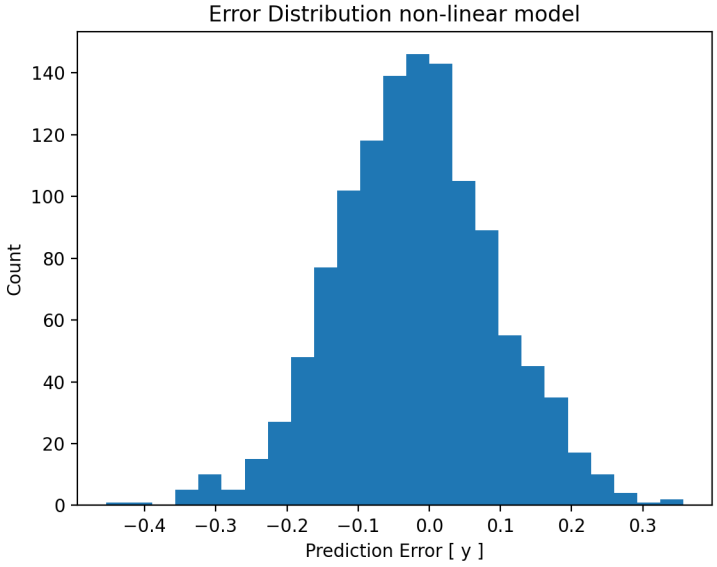
Visualize the results putting `true y values` on the `x-axis` and `predicted y values` on the `y-axis`.



Points of the scatterplot do not lie all on the diagonal, but they are very close to it. There is still a little dispersion.

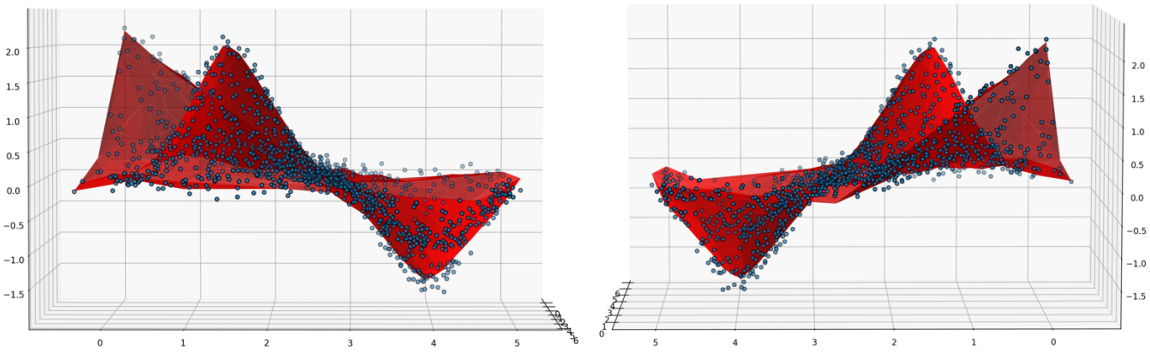
Inspect the error distribution.

It's quite a gaussian, as shown from the descriptive statistics. This result is might expected and a perfect gaussian distribution is not reached because of the small set.



mean	-0.020079
std	0.111924
min	-0.453893
25%	-0.094617
50%	-0.019093
75%	0.051316
max	0.356792

Then, visualize the regression problem in a three-dimensional space.
Here we have another demonstration that the non-linear model fit well the data and that this neural network is one of the ways to assess the task.



3. The model statistically better

In order to assess which model is statically better are performed the t-test and the p-value.

The first assumption is that both models have no bias:

$$E[\varepsilon_{linear}] = E[\varepsilon_{non-linear}] = 0$$

The hypothesis testing is:

$$H_0: Var[\varepsilon_{linear}] = Var[\varepsilon_{non-linear}]$$

$$H_1: Var[\varepsilon_{linear}] \neq Var[\varepsilon_{non-linear}]$$

Under the Central Limit Theorem (CLT):

$$T = \frac{\bar{e}_{linear} - \bar{e}_{non-linear}}{\sqrt{\frac{s^2_{linear}}{l} + \frac{s^2_{non-linear}}{l}}} \sim N(0,1)$$

$$e_{model,i} = (y_i - \hat{y}_i)^2, \qquad i = 1,2,\dots,l$$

$$\bar{e}_{model} = \frac{1}{l}\sum_{i=1}^{l} e_{model,i}$$

$$s^2_{model} = \frac{1}{l-1}\sum_{i=1}^{l}(e_{model,i} - \bar{e}_{model})^2$$

H_0 is rejected when T is outside the 95% confidence interval (-1.96, 1.96) and it is selected the model with the smaller variance.

Under those assumption t-test reveals

$$T=18.35$$

With a large evidence the NN model is statistically better, according to the t-test.

Moreover, it has been evaluated the p-value which provide the smallest level of significance at which the H_0 hypothesis would be rejected. As small is the p-value as strong is the evidence in rejected H_0.

The p-value between the two models is:

$$p\text{-value}=1.6953816236538262e-66$$

Once again, it is proof with a huge evidence that the non-linear model is statistically better.

4. Neural Network vs Random Forest

Empirically the MSE obtained by the NN model is lower than the one obtained with the Random Forest one. These two algorithms work in a radical different way.

Random Forest is the set of random trees: each random tree processes the sample and predict the output label. But, each decision tree in the set is independent from another, so each one can predict the output label.

The neural network is made up of connected neurons in which one neuron cannot operate indipendently. In the model there are three densely connected hidden layers and an output layer.
Each layer process the data and then pass forward the next layer, and only the last layer of neurons is making decisions giving one single continuous value.

For this reason, I think that a NN, which is more complex than a Random Forest, if it's set in the right way it is able to get better performance. Notice that changing each parameter in the NN, like the activation function, the optimizer, the number of neurons, etc. make results differ a lot since the network is the combined result of all the parameters.

References:

1. SciPy documentation: https://scipy.org
2. Keras documentation: https://keras.io
3. Scikit-learn documentation: https://scikit-learn.org
4. Course repository: https://github.com/andreacini/ml-19-20
5. Neural Netwroks for Machine Learning: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf