Supervised Learning
Linear Regression And Logistic Regression

Supervised Learning is a well defined problem in machine learning where we need to find the best model using data from a training set. However finding the best model means that we need to generalize to unseen examples. Today we will focus on building the best model with the focus on getting the best score on an unseen set, the so-called test set.

Exercise 1: Regression on Boston Dataset

The Boston dataset is composed of 506 instances and 14 columns (13 features and 1 target variable). The target variable corresponds to the median value of owner-occupied homes in $1000s. We are going to find a model able to explain and thus predict this target variables using one and/or more variables.

1. Load the Boston dataset. We consider that the first 300 examples are our training set, the next 100 are our validation set and the remaining examples are the test set. What is the range of the target variable? Describe the target variable?
2. Right now we will be using only the first features called `CRIM` for modelling the target variable. Plot `CRIM` vs `target`. What do you observe?
3. Use `LinearRegression()` for modeling `target` with `CRIM` on the training set and compute the predicted values for the validation set. Plot the predicted data points against the real ones.
4. Implement a function which is computing the Mean-Square-Error (MSE). What is the MSE on the training set and on the validation set?
5. Build the model: `log(target) = w . CRIM + b`. Compute the MSE on the train and val set? Do you get betters scores?
6. Create a function `train_and_metrics()` and build the model `log(target) = w . log(CRIM) + b`. Do you improve your performance on the validation set?
7. Create a function which is doing min-max normalization? Does applying min-max normalization on `CRIM` decreases your MSE on the validation set?
8. Finally create a model with 2 variables `CRIM` and `LSTAT` such as `log(target) = w1 . log(CRIM) + w2 . log(LSTAT) + b`. Does it decrease the MSE on the validation set? You can know compute the MSE on the test set!

Exercise 2: Binary Classification on Breast Cancer

The Breast Cancer Wisconsin dataset has 569 examples and 30 variables plus the target variable which is a binary variable (cancer yes or no).

1. Load the dataset and split it into a training, validation and test set as usual.
2. Predict the target variable using one of the variable of your choice.
3. Write a function able to create a confusion matrix given predictions and ground-truth.
4. Write a function which takes a confusion matrix and return the accuracy, prevision, recall and F-measure.
5. How big is the discrepancy in term of F-measure between the train and val set?
6. Employ normalization trick to make sure that you are fitting every variable into the range 0-1
7. Does normalization improve performance of you model using the same variable as before? Note: do not forget to "denormalize" the prediction if they are normalized.

8. Implement a algorithm able to find the best model given the set of possible variables? Note: Assume that there is a fixed-validation set.
9. Modify the previous algorithm by doing 5-cross fold validation
10. What is your final performance on the test set?

Exercise 3: Classification on the Titanic dataset

For this dataset we do not have direct access to the test set so you cannot cheat! On the Titanic dataset we are going to predict wether a person will survive the Titanic disaster.

1. Go on the kaggle website and download the csv file (train, val, test) and load them into python (using pandas library).
2. Do-it-yourself: Given what you have done on the previous exercises find the best model using the train and val set.
3. Once you find the best model, you can predict for the test set. Then you can create a csv file and upload it to kaggle. What is your score on the leaderbord?

Supplementary exercise: Simple Logistic Regression with Gradient Descent

Without using `scikit_learn` implement a simple logistic regression algorithm with Gradient Descent using the Sigmoid function. Reminder: we are using the following modelling - $P(y_i = 1|\mathbf{x}_i; \mathbf{w}, b) = p_{\mathbf{w},b}(\mathbf{x}_i) = \frac{1}{1+e^{-(\mathbf{w}\mathbf{x}_i+b)}}$
The first thing to do is to compute by hand the gradients for w and b according to the loss function. In our case the loss function is equal to: $J(\mathbf{w}, b) = -\frac{1}{N}\sum_{i=1}^{N} y_i log(p_{\mathbf{w},b}(\mathbf{x}_i)) + (1 - y_i)log(1 - p_{\mathbf{w},b}(\mathbf{x}_i))$
You can create synthetic dataset where a point is a 2D coordinates $\mathbf{x} = (x^1, x^2)$ where x^i is between 0 and 1. You can associate a label to data point $\mathbf{x}$ such that $y = 1$ if $x^1 + x^2 > 1$ else $y = 0$.