Note Machine Learning from A-Z

Advantages of train-test split

Avoid over-fitting and under-fittign from modelling

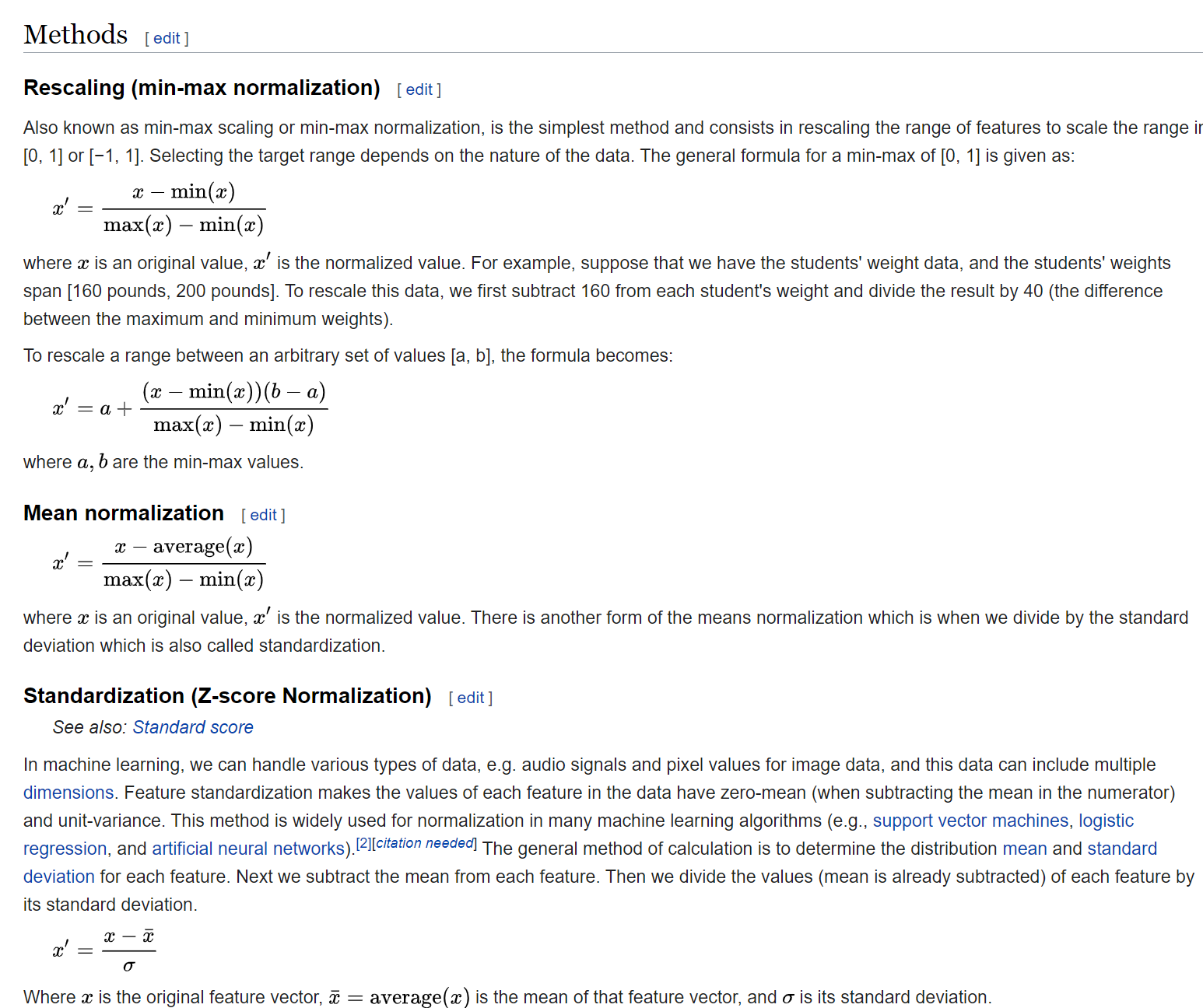
Increase the computatuinal efficiency

Predict the modelling performance

Condition:

Large-size of data sets -enough to split two set

Original set can represent the demographic of its’ train set



UCL machine learning respiratory

* Benign or malignant
* kNN- Kernel nearest neighbours

SVM for classification

* Hyperplanes
* Maximum　Margin

Most of the times work better than other machine learning classification.

Modeling- classification questions

In this Part 3 you learned about 7 classification models. Like for Part 2 - Regression, that's quite a lot so you might be asking yourself the same questions as before:

1. What are the pros and cons of each model ?
2. How do I know which model to choose for my problem ?
3. How can I improve each of these models ?

Again, let's answer each of these questions one by one:

1. What are the pros and cons of each model ?

Please find attached at the bottom of this article a cheat-sheet that gives you all the pros and the cons of each classification model.

2. How do I know which model to choose for my problem ?

Same as for regression models, you first need to figure out whether your problem is linear or non linear. You will learn how to do that in Part 10 - Model Selection. Then:

If your problem is linear, you should go for Logistic Regression or SVM.

If your problem is non linear, you should go for K-NN, Naive Bayes, Decision Tree or Random Forest.

Then which one should you choose in each case ? You will learn that in Part 10 - Model Selection with k-Fold Cross Validation.

Then from a business point of view, you would rather use:

- Logistic Regression or Naive Bayes when you want to rank your predictions by their probability. For example if you want to rank your customers from the highest probability that they buy a certain product, to the lowest probability. Eventually that allows you to target your marketing campaigns. And of course for this type of business problem, you should use Logistic Regression if your problem is linear, and Naive Bayes if your problem is non linear.

- SVM when you want to predict to which segment your customers belong to. Segments can be any kind of segments, for example some market segments you identified earlier with clustering.

- Decision Tree when you want to have clear interpretation of your model results,

- Random Forest when you are just looking for high performance with less need for interpretation.

3. How can I improve each of these models ?

Same answer as in Part 2:

In Part 10 - Model Selection, you will find the second section dedicated to Parameter Tuning, that will allow you to improve the performance of your models, by tuning them. You probably already noticed that each model is composed of two types of parameters:

* the parameters that are learnt, for example the coefficients in Linear Regression,
* the hyperparameters.

The hyperparameters are the parameters that are not learnt and that are fixed values inside the model equations. For example, the regularization parameter lambda or the penalty parameter C are hyperparameters. So far we used the default value of these hyperparameters, and we haven't searched for their optimal value so that your model reaches even higher performance. Finding their optimal value is exactly what Parameter Tuning is about. So for those of you already interested in improving your model performance and doing some parameter tuning, feel free to jump directly to Part 10 - Model Selection.

For class objection

Beaware of intent level and unintent level

Intent to right

Example

def inspect(results):

    lhs         = [tuple(result[2][0][0])[0] for result in results]

    rhs         = [tuple(result[2][0][1])[0] for result in results]

    supports    = [result[1] for result in results]

    confidences = [result[2][0][2] for result in results]

    lifts       = [result[2][0][3] for result in results]

    return list(zip(lhs, rhs, supports, confidences, lifts))

resultsinDataFrame = pd.DataFrame(inspect(results), columns = ['Left Hand Side', 'Right Hand Side', 'Support', 'Confidence', 'Lift'])

Hello students,

Congratulations for having completed Part 7 - Natural Language Processing.

If you are up for some practical activities, here is a little challenge:

1. Run the other classification models we made in Part 3 - Classification, other than the one we used in the last tutorial.

2. Evaluate the performance of each of these models. Try to beat the Accuracy obtained in the tutorial. But remember, Accuracy is not enough, so you should also look at other performance metrics like Precision (measuring exactness), Recall (measuring completeness) and the F1 Score (compromise between Precision and Recall). Please find below these metrics formulas (TP = # True Positives, TN = # True Negatives, FP = # False Positives, FN = # False Negatives):

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

F1 Score = 2 \* Precision \* Recall / (Precision + Recall)

3. Try even other classification models that we haven't covered in Part 3 - Classification. Good ones for NLP include:

* CART
* C5.0
* Maximum Entropy

Submit your results in the Q&A for this Lecture or by pm and justify in few words why you think it's the most appropriate model.

Enjoy Machine Learning!

 F1 Score is the 2\*((precision\*recall)/(precision+recall))

https://archive.ics.uci.edu/ml/datasets.php?format=&task=reg&att=num&area=&numAtt=&numIns=&type=&sort=nameUp&view=table