Machine Learning in Python

Idea behind it:

The aim of this thesis is to summarize the most common Machine Learning implementations. In this sense example tasks are presented, in which certain algorithms for the applications occur. In addition, the most important contents behind the algorithms used are briefly explained in this document. The source of this summary is the Applied ML with Python course from the University of Michigan (Coursera).

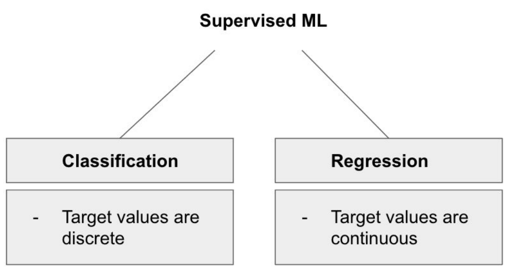
Intro to Applied Machine Learning:

What is Machine Learning (ML):

- In short, algorithms that can be trained with labelled data. Always with the aim that the algorithms are able to generalize in a later stage. Or with other words, they can make accurate predictions for new objects that were not seeing during training.
- ML cover the fields of statistics, computer science, psychology and even economics.

Key concepts of ML:

→ **Supervised Machine Learning:** learn to predict target values from labelled training data



→ **Unsupervised Machine Learning:** The data is not labelled. So the algorithms is able to find groups and similar instances (clustering) or find unusual patterns (outlier detection)

Hyper-parameters: Choices which will influence the performance such as the number of batches or epochs used.

- Usually, the training set is split-up into two parts. A smaller training set which is for the update of the weights in the training and a validation set which is mainly for hyper-parameter tuning.

Overfitting: When a learner outputs a classification which is 100 percent accurate on the training data and poorly accurate on the test data. With other words, the model did not learn to generalize, but to memorize.

Bias vs. Variance:

- Bias is a learner's tendency to consistently learn something wrong.
- Variance is the tendency to learn random things irrespective of the real signal.

Machine Learning Workflow

- The workflow can be interpreted as 3 different blocks. The first block is the *Representation*. There the focus is the feature representation and the question which type of classifier to apply. The second block can be seen as *Evaluation.* At this stage, the criteria for the distinction between good and bad results is central. Finally, the last block is *Optimization.* There, the settings and parameters are selected which give the best results, or with other words the best classifier set-up.

Further Note:

- Before applying machine learning algorithms to a dataset, it is always a good idea to examine the data first. As a result, additional cleaning or preprocessing may be necessary. This could be the clear noisy data, change datatypes or even understand that the use case is solvable without ML.

K-nearest neighbour classification:

- KNN can be used for classification but also for regression tasks
- It is a memory, or instance based supervised learning algorithm

K: number of the nearest neighbours the classifier will take into account in order to make its prediction.

→ This works in three different steps:

1. Find the most similar instance to x_test that are in the training set X_train
2. Get the labels y_NN for the instance X_NN
3. Predict the label for x_test by combining the labels of y_NN. This can be for example with majority vote.

The zone where the transition from one class to another is detectable, is called the *Decision boundary.*

The nearest neighbour algorithm needs four things:

1) A clear distance metrics,
2) The fact how many nearest neighbours will be included,
3) Optional weighting function on the neighbour points,
4) A method for aggregating the classes of the neighbour points.

L1 - Distance and Euclidean Distance

- The distance between two data points can be defined as

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

Or with the euclidean norm

$$d_1(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

Important note: If k = 1, the prediction is very sensitive to noise, mislabelled data and of course individual data points. A bigger *k* means that the separation between the classes becomes smoother and more robust but also increases the chance of wrong detection. This conflict is called the *Biased Variance Tradeoff.* Furthermore, a low value for *k* is more likely to overfit and so lead to worse accuracy on the predictions on the test data.

The R^2 Regression Score

- Measures how well a prediction model for regression fits the given data. The score can be between 0 and 1. Whereas, 1 means perfect prediction.

Linear Models

- A linear model is a sum of weighted variables that predict a target output value given an input data instance (feature vector)

Linear Regression

The linear regression model can be denoted as

$$\widehat{\mathbf{y}} = (\widehat{w_0}x_0 + \widehat{w_1}x_1 + \ldots + \widehat{w_n}x_n)$$

,

Where *y* denotes the predicted output, *w* the feature weights, *b* a constant term and *x* the features.

- Talking about Linear Regression, one usually means *Least Squares Linear Regression*

This comes from the fact that there is an ordinary least squares problem behind it. The aim is to find *w* and *b* that are able to minimize the squared error of the model, like:

$$\lim_{\widehat{x} \to x} |\, x - \widehat{x}\, |^2 = 0$$

- But there are no additional parameter to control model complexity → the predicted model always be a straight line

→ The question is now how does linear regression parameters *w* and *b*?

- The learning algorithm finds the parameters that optimize an *Objective Function*, typically minimize some kind of loss function of the predicted target values vs. actual target values

$$R(\mathbf{w}, b) = \sum_{i=1}^{n} (\mathbf{y}_i - (\mathbf{w}\mathbf{x}_i + b)^2$$

Another model which is similar to the Least Square Linear Regression is *Ridge Regression.*

Ridge Regression

The model is defined as

$$R(\mathbf{w}, b) = \sum_{i=1}^{n} (\mathbf{y}_i - (\mathbf{w}\mathbf{x}_i + b)^2 + \alpha \sum_{j=1}^{n} w_j^2$$

- The additional penalty parameter is for regularization. This means that it shall prevent over-fitting of the model by restricting the model. This type of regularization is also called *L2 Regularization.* The idea is to minimize the squares of *w* entries. However, the influence of the regularization term is based on ***α***. Regularization works genuinely good at small amounts of data with a lot of features. Though, it becomes “less” important with an increasing amount of data.

→ the higher ***α*** the stronger is the regularizing effect or with other words the simpler gets the model.

The Need for Feature Normalization

- It is important that all features should be on the same scale so that there is a weighted influence of all weights. One way to do normalization is with *MinMax Scaling.*

MinMax Scaling

- For each feature *x* compute the minimum and maximum value across all the instances of the training set. Then each feature will be transformed into a scaled version of itself. This can be done with

$$x_i' = \frac{x_i - x_i^{min}}{x_i^{max} - x_i^{min}}$$

→ As a result all the features will be at the same scale!

Lasso Regression

- Another version of regression is *Lasso Regression.* It uses an *L1 Regularization* penalty for the training. The *L1 Regularization* penalty minimizes the sum of the absolute values of the coefficients. This can be denotes as

$$R(\mathbf{w}, b) = \sum_{i=1}^{n} (\mathbf{y}_i - (\mathbf{w}\mathbf{x}_i + b)^2 + \alpha \sum_{j=1}^{n} |w_j|$$

- The effect of this representation is that it sets the parameters weights to 0 for the least influential weights. As a result, there is a sparse solution which is based on feature selection.

Lasso vs. Ridge

- The question is when is it appropriate to apply *Ridge* and when to apply *Lasso?*
- If there are many variables with small to medium sized effects it is usually better to implement with *Ridge Regression.* However, if there are fewer variables with medium to large effects it can be better to use *Lasso Regression.*

Polynomial features with Linear Regression

- One step further is to update the linear model with additional features of all polynomial combinations. There, the degree specifies how many variables participate at a time in each new feature. Important to note here is that it is till a linear model

This can be defined as

$$\mathbf{x} = (x_0, x_1) \rightarrow \mathbf{x}' = (x_0, x_1, x_0 x_1, x_0^2, x_1^2)$$

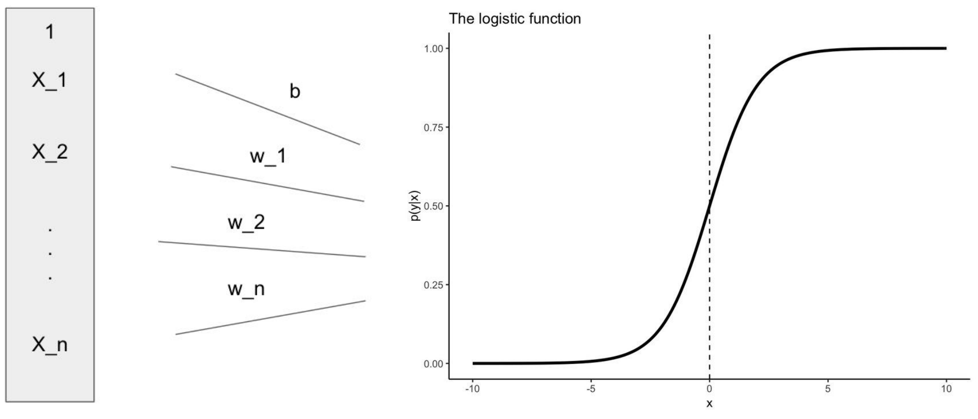
$$\widehat{\mathbf{y}} = \widehat{w_0} x_0 + \widehat{w_1} x_1 + \widehat{w_{00}} x_0^2 + \widehat{w_{01}} x_0 x_1 + \widehat{w_{11}} x_1^2 + b$$

But why?

- The idea is to capture interactions between the original features. Or generally speaking, we can apply other non-linear transformations to create new features.

- But polynomial feature expansion with high degree can lead quickly to very complex models that can have the notion to overfit. Therefore it is important to apply a regularized learning method, such as *Ridge Regression*

Logistic Regression



→ y = (0,1)

- The idea behind it to compress a linear function to a range between 0 and 1.
- In that sense, it can be interpreted as the probability that the input object belongs to a certain class.

Regularization in Logistic Regression

- *L2* is usually integrated by default
- The parameter *C* controls the amount of regularization
  - In addition, there is the need of normalizing the features so that they are all at the same scale

C Parameter

- As the parameter defines the impact of regularization in the model, there is the option to define it very big, but also very small
  - Larger values of C have the impact there is less regularization. This means that the model tries to fit the training data as well as possible. Each individual data point is taken into account for the classification
  - Smaller values of C mean more regularization. This means that the model becomes more tolerant of errors for individual data points.

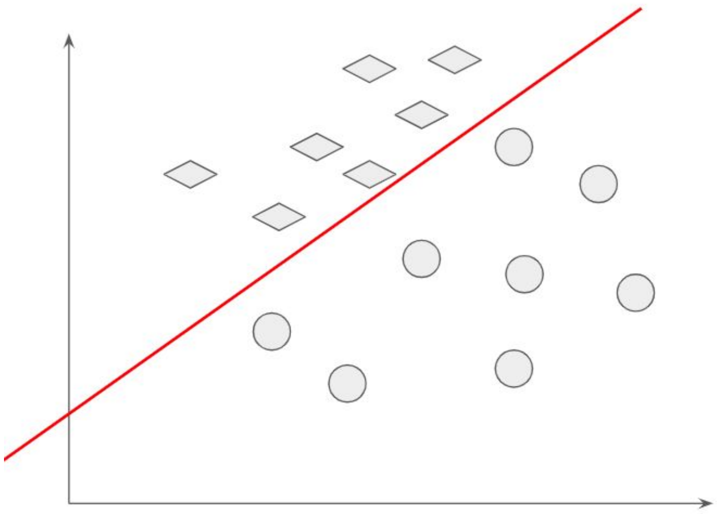
Support Vector Machine

- Looking at the model representation for a linear model.

$$\begin{tikzcd} x \arrow[r] & \boxed{f} \arrow[r] & y_{\_pred} \end{tikzcd}$$

And mathematically, it can be denoted:

$$f(x, w, b) = sign(w * x + b)$$



Classifier margin:

- It defines the maximum width of the decision boundary that can be increased before hitting the data point.
- The linear classifier with maximum margin is a linear Support Vector Machine (LSVM).
- How tolerant the SVM is of misclassifying training points, as its objective of minimizing the margin between classes is controlled by the parameter C.

Multi-class Classification

- In Scikit, multi-class problems are treated like multiple binary classification problems.
- Each data point will be run through each individual classifier. The one with the highest score indicates the class. Or with other words, there is a matching.

Kernelized Support Vector Machine

- This kind of SVM can transform the data in a way that it will be interpreted in a higher dimension.
- There are different kernels available which correspond to different types of transformation. The most commonly used one is the *Radial Basis Function* kernel. Or short, the RBF kernel.

RBF Kernel

$$K(\mathbf{x}, \mathbf{x}') = exp(-\gamma \cdot \left\|\mathbf{x} - \mathbf{x}'\right\|^2)$$

- The kernel can be interpreted as similarity measure between data points.
- The similarity between two data points and the transformed feature space is an exponentially decaying function of the distance between the vectors and the original original input space.

γ Parameter

- The γ value controls how far a single trending example reaches which in turn affects how tightly the decision boundaries end up surrounding the input space.

γ Parameter vs. C Parameter

- If γ value is large then the C parameter has no impact and if γ value is small, C would affect model like in a LSVM.

Cross Validation

- *Cross Validation* is a method that goes beyond evaluation a single model using a single Train/Test set split. It uses multiple splits and each of those are individually trained and tested. As a result, there is a stable and reliable understanding of the average performance of the classifier.
- It also gives insight of how sensitive the model is given the nature of the data.
- However, the implementation needs more time and computational power.

K-fold validation

- Original data is split up into folds. For example, for model 1:
    - Fold 1 is the test set and folds between 2 and 5 are for training purposes.

Stratified cross-validation

- K - Fold can be problematic when the given input data is already in any kind of order. As a result, there would be a bias.
- When splitting the data, the proportion of the data in the original dataset will be considered in each fold.

Leave-one-out cross-validation

- It is especially good for small data sets, but needs a lot of computation.

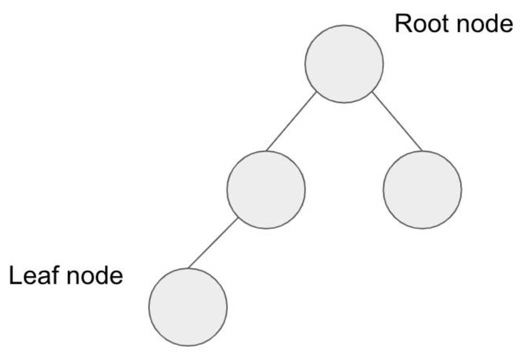
Validation Curve

- It shows how sensitive the model is to changes in important hyper-parameters.

- It also shows the mean cross-validation accuracy for training and testing as a function of the given parameters.
- Lastly, it shows a variation of means because of the different folds.

Decision Tree

- Decision Trees (DT) can be used for either classification or regression purposes.
- They consider if-else clauses for features. The more information a feature holds the earlier it should be compared in the tree.



- The aim in Decision Trees is to find a sequence of questions that have the best accuracy at classifying the data in the fewest steps.
- DT decisions are easy to interpret.
- Furthermore, it is important to know of how informative a split moment is.
- In the Decision Tree implementation, it can be noted how many samples of each class ends up at this leaf node during training.

→ Therefore, it is important to identify the most informative splits. This can lead to two different nodes. A *pure node* is a node which holds all samples in one class. A *mixed node* has a mixture of different classes. In order to improve splits, a feature importance calculation can be applied.

Feature Importance Calculation

- A number between 0 and 1 is assigned to each feature. The value of 0 means that the feature was not used in the prediction and 1 means that the feature predicted the target perfectly.
- Furthermore, all feature importances are normalized to the total sum of 1.

- A common problem of DTs is that they tend to overfit. There are two strategies to counteract. One is to pre-prune which means to set the tree a limit of levels which it can build. Another is post-processing. The tree can fully evolve and afterwords will be reduced.

Random Forest

Ensemble: The idea is to take multiple learning models and combine them in order to create an aggregated model that is more powerful than a single one.

→ This also reduces the risk of overfitting.

- A *Random Forest* is an ensemble of trees. It is very widely used and because there are more DT, the generalization is more stable and better. Importantly, an ensemble of DT should be diverse, so a random variation of tree buildings.
  - This means that the data to build the trees is randomly selected.
  - Also, the features chosen in the split are randomly chosen.

Gradient Boosted Decision Trees

- It is an ensemble of multiple trees.
- The idea is that with a series of trees, each one gets individually trained but tries to correct the mistakes from the previous ones.
- There is a learning rate which can control how hard each new tree tries to correct the remaining errors.
- It is a very good method, but also sometimes hard to interpret.

Evaluation methods

- Different applications have very different understanding of goals.
    - E.g. user satisfaction, amount of revenue or increase in patient survival rate

Confusion Matrix:

Confusion Matrix	
TN	FP
FN	TP

- *Label 1 is the positive class. This means that it is the class of interest. Label 0 is then everything else.*

Classification Error:

$$= \frac{FP + FN}{TN + FN + TP + FP}$$

True Positive Rate:

$$= \frac{TP}{FN + TP}$$

- It is also called Recall, sensitivity or probability of detection

Precision:

$$= \frac{TP}{FP + TP}$$

- It is also called the precision. The precision can either be increased by improving the amount truly positives decisions or by reducing the false positive detections.

- For the question of what is the faction of all negative instances that the classifier identified as positives, it makes sense to look at the *False Positive Rate* (FPR).

False Positive Rate:

$$= \frac{FP}{FP + TN}$$

- It is also called the specificity

→ There is always a trade-off in the evaluation of the model. If one tries to improve the accuracy by e.g. adapting the decision boundaries, there will be some loss on the recall.

Some tasks could be:

- Search engine ranking
- Document classification

However, there are some tasks which are more recall-oriented because of a desired FN = 0.

- Search and information extraction in legal discovery
- Tumor detection

Furthermore, there is the *F1 and F Score.*

F1 Score :

$$= 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

F Score :

$$= (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$$

- The **β** allows adjustments of the metric to control the emphasis on recall vs. precision.

→ For Precision-oriented users: **β** = 0.5

- In that way, the false positives hurt the performance more than false negatives.

→ For Recall-oriented users: **β** = 2

- In that way, the false negatives hurt the performance more than false positives.

Classifier Decision Functions

- Each classifier score value per test point indicates how confidently the classifier predicts the positive class or the negative class.
- Choosing a fixed decision threshold gives a classification rule

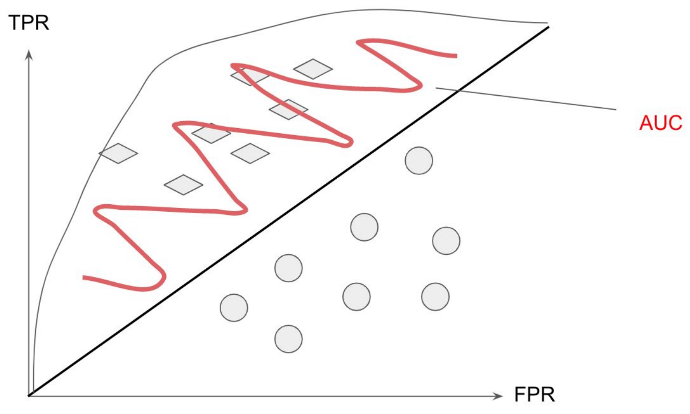
→ Typically, the threshold is set to > 0.5, so that the most likely class is chosen.

- Adjusting a threshold affects the predictions of the classifier.

  - A higher threshold means that the classifier becomes more conservative in its decisions.
  - Varying the decision threshold leads to the *Precision / Recall Curve.*

Precision / Recall Curve (PR Curve) and ROC Curve

- The ideal point would be where Precision is 1.0 and Recall is 1.0
- The steepness of the PR Curve is also very important.



- The ideal point is where FPR = 0 and TPR = 1.
- The steepness indicates how strongly TPR gets maximized and FPR minimized.
- The Area Under Curve (AUC) measures the performance of the classifier. The bigger the better.