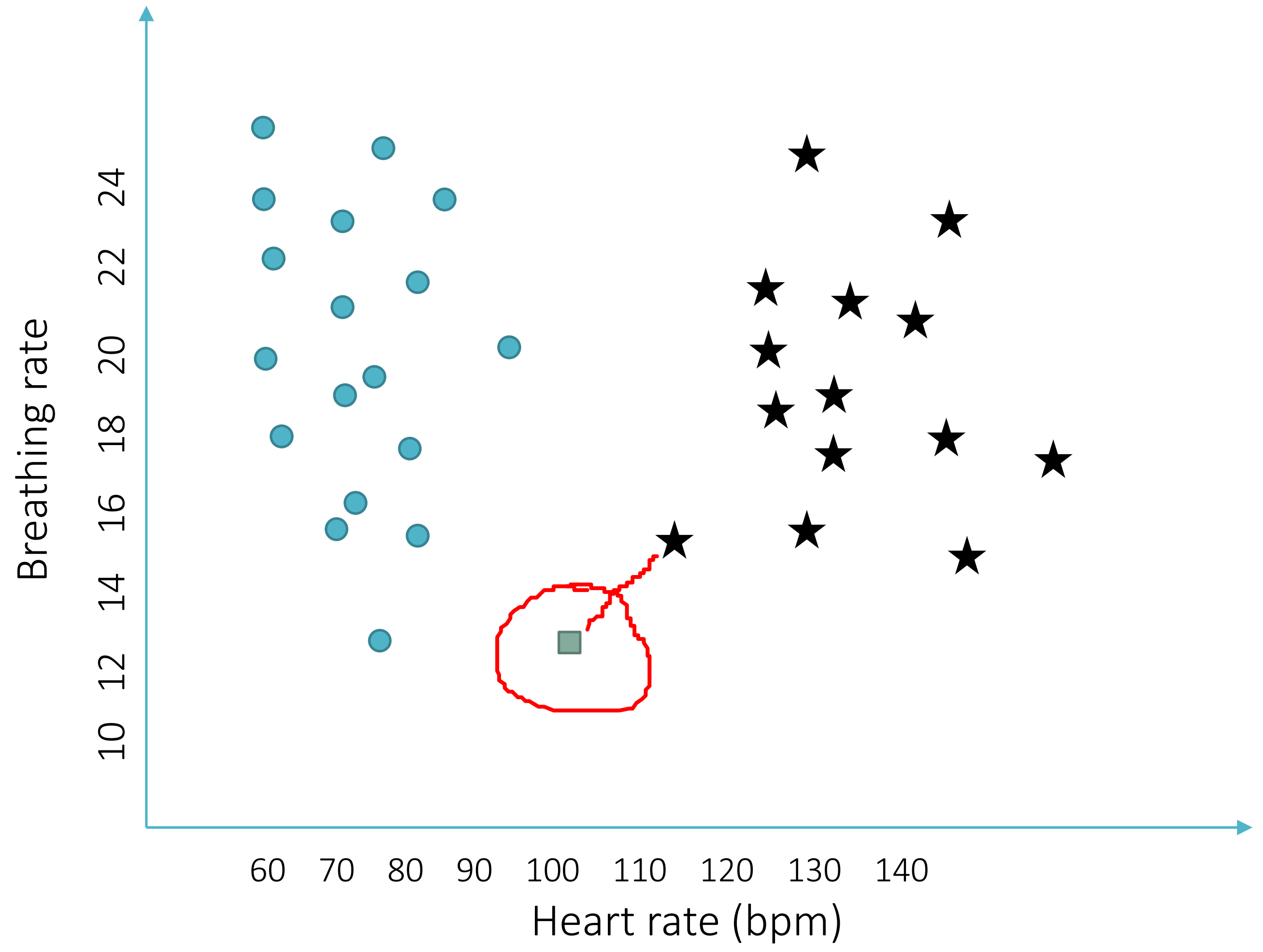
Practical ML – Part 1

Common mistakes to avoid

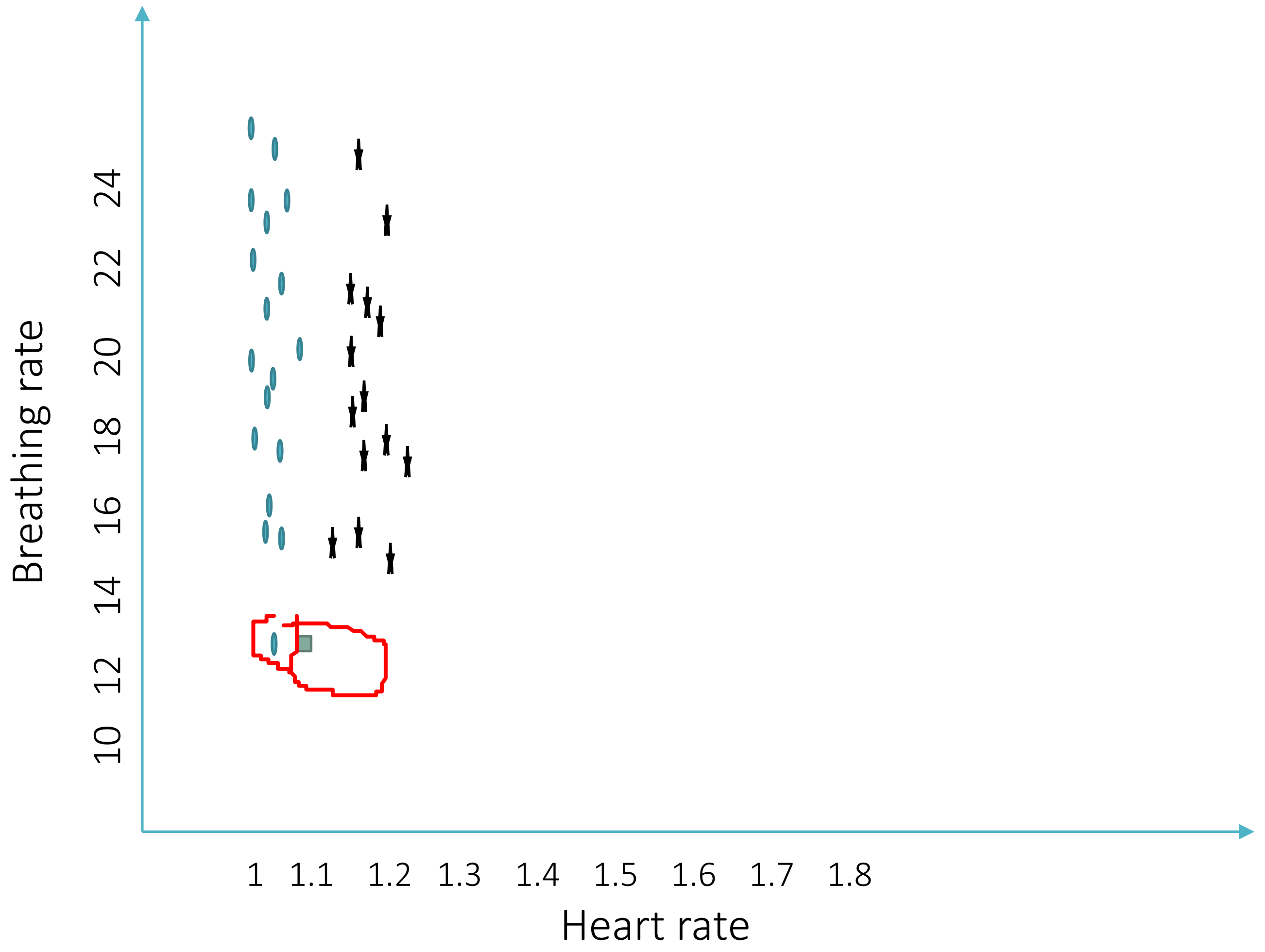
Data Normalization – why?



Two classes – normal (circle), abnormal (stars)

Apply a nearest neighbours classifier

Data Normalization – why?



Heart rate rescaled into different units (beats per second, instead of beats per minute)

New test data point is now categorised into the other class

Breathing rate has greater influence on the model

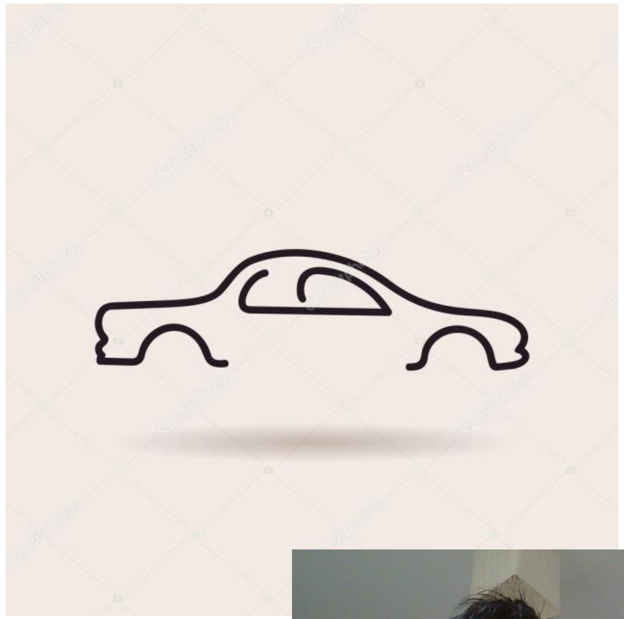
Data Normalization – when does it matter?

Any classification method that uses distances (nearest neighbour

SVMs (affects robustness for linear SVMs, and affects the decision boundary for non-linear SCMs

Usually best to normalize data anyway

Caveat: might **not** scale, if the relative size of features is important.



Data Normalization – how?

Zero mean, unit variance transform

For a feature, X:

$$x_{norm} = \frac{x - \bar{x}}{\sigma}$$

Restrict range:

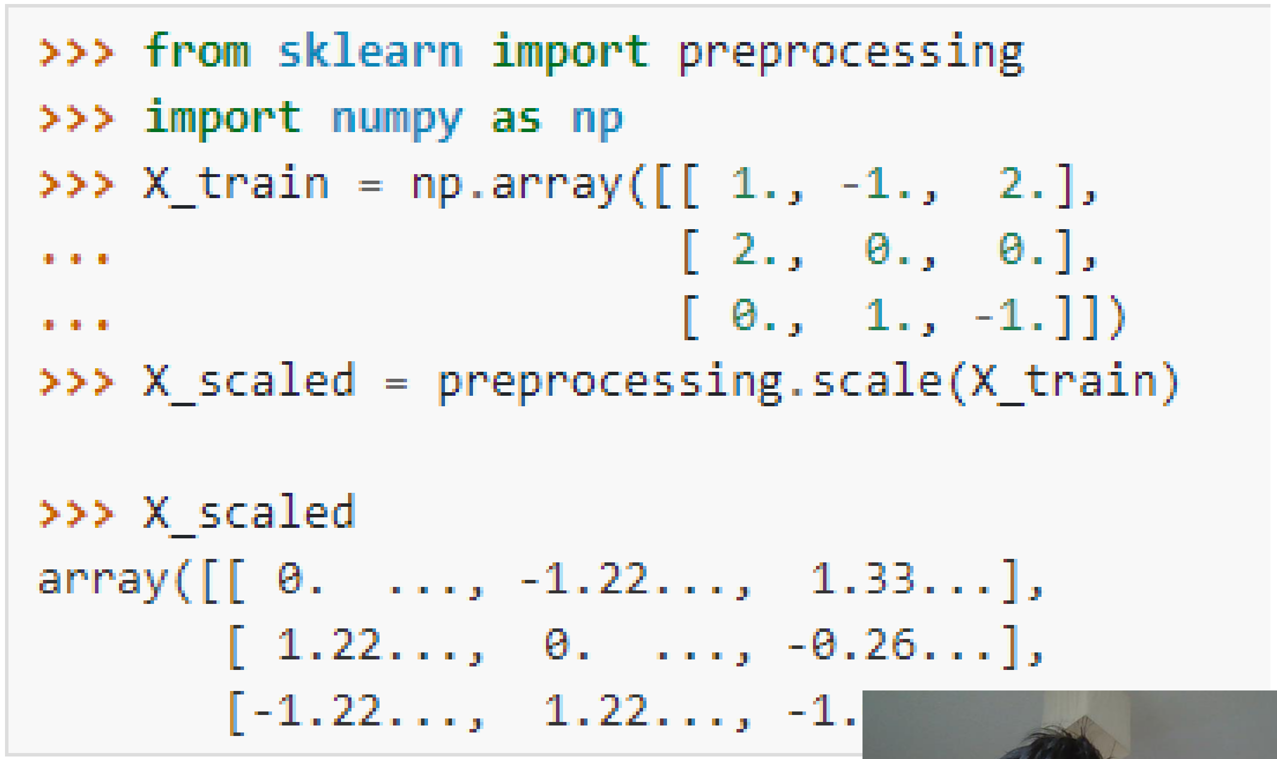
For a feature, X:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

generate scaling factors from the training set ONLY (i.e. calculate mean and variance of training data, NOT the whole set)

Data Normalization in Python scikit

Zero mean, unit variance:



```
>>> from sklearn import preprocessing
>>> import numpy as np
>>> X_train = np.array([[ 1., -1.,  2.],
...                     [ 2.,  0.,  0.],
...                     [ 0.,  1., -1.]])
>>> X_scaled = preprocessing.scale(X_train)

>>> X_scaled
array([[ 0.  ..., -1.22...,  1.33...],
       [ 1.22...,  0.  ..., -0.26...],
       [-1.22...,  1.22..., -1.
```

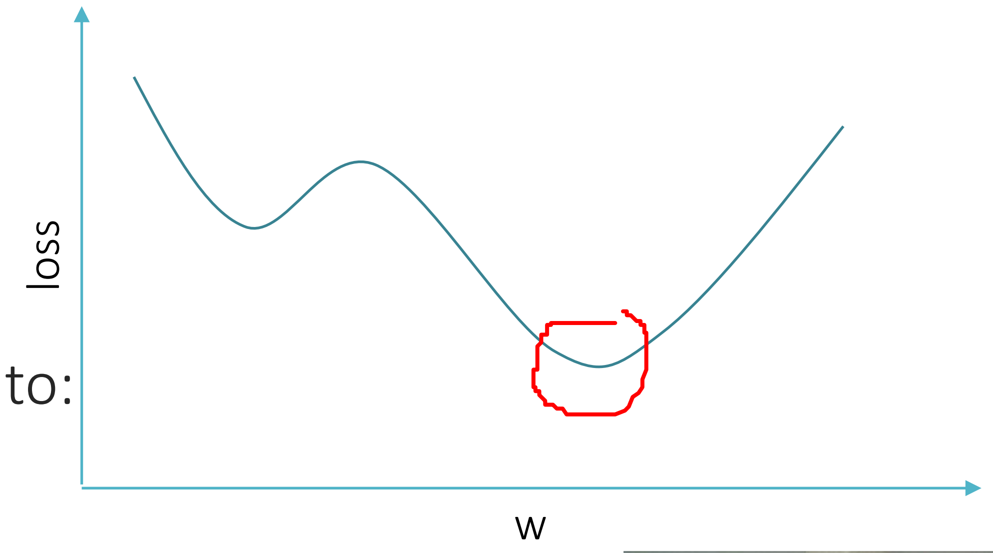
Restrict range

Sklearn.preprocessing.normalize

Algorithm Convergence

For most ML algorithms, we perform an iterative process to train a model (exception : K-NN)

When should we stop training?



Update the weights until any small changes lead to:

> minimal decrease in error/loss

> increase in error/loss

Convergence in scikit

sklearn.linear_model.**LogisticRegression**(*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*)

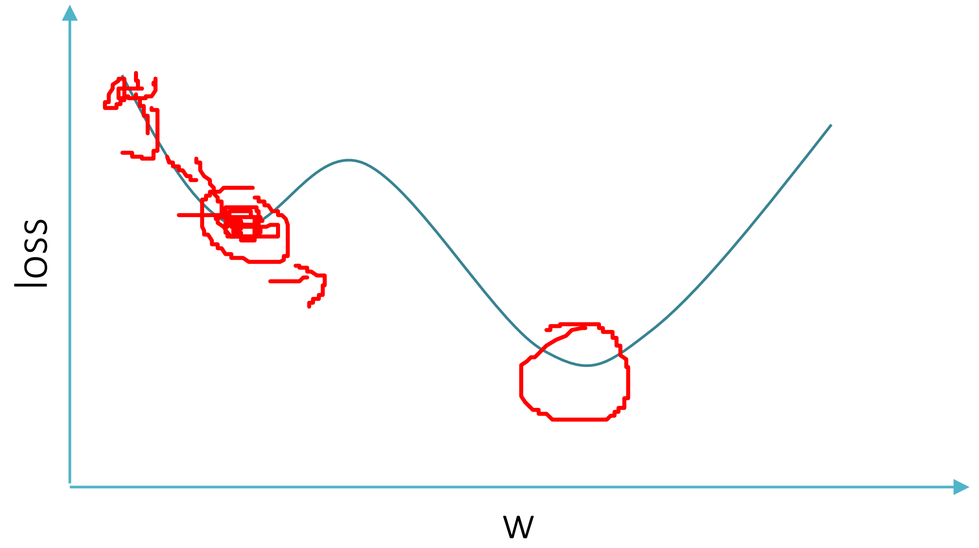
Tol : (tolerance), is the convergence criterion. If the loss function does not change by more than *tol*, the machine learning algorithm is assumed to have finished

Max_iter: (maximum iterations) is another stopping criterion. It sets a hard limit on the number of times the algorithm runs through the training data

```
UserWarning: lbfgs failed to converge. Increase the number of iterations.
```

Convergence – global minimum

Even after algorithm convergence, we may not have the optimum solution



methods such as simulated annealing and momentum reduce the impact of local minima

In scikit, include additional parameter (e.g. nesterovs_momentum)

Summary

Discussed two common pitfalls during practical training on machine learning models:

Data normalisation

Un-normalized data over emphasizes the importance of features with a wide dynamic range

Rule of thumb: normalize your data (even if we do not strictly need to)

Convergence

Make sure that your ML model has converged properly!