Generalisation, Training and Test Set Representation

1 Generalisation

- Goal - to predict well on new data drawn from (hidden) true distribution
- Issue - we don't see the truth, but we only get to sample from it
- If it fits current sample well, how can we trust it will predict well on other new samples?

How do we know if our model is good?

- Theoretically
  - Generalisation theory - based on ideas of measuring model simplicity/complexity
- Intuition: formalisation of Ockham's razor principle
  - The less complex a model is, the more likely a good empirical result is
- Empirically
  - Asking: will our model do well on a new sample of data
  - Evaluate: get a new sample of data - call it the set set
  - Good performance on the test set is a useful indicator of good performance

Three basic assumptions in all of the above

1. We draw examples independently and identically at random from the distribution
2. The distribution is stationary - it doesn't change over time
3. We always pull from the same distribution, including training, validation and test sets

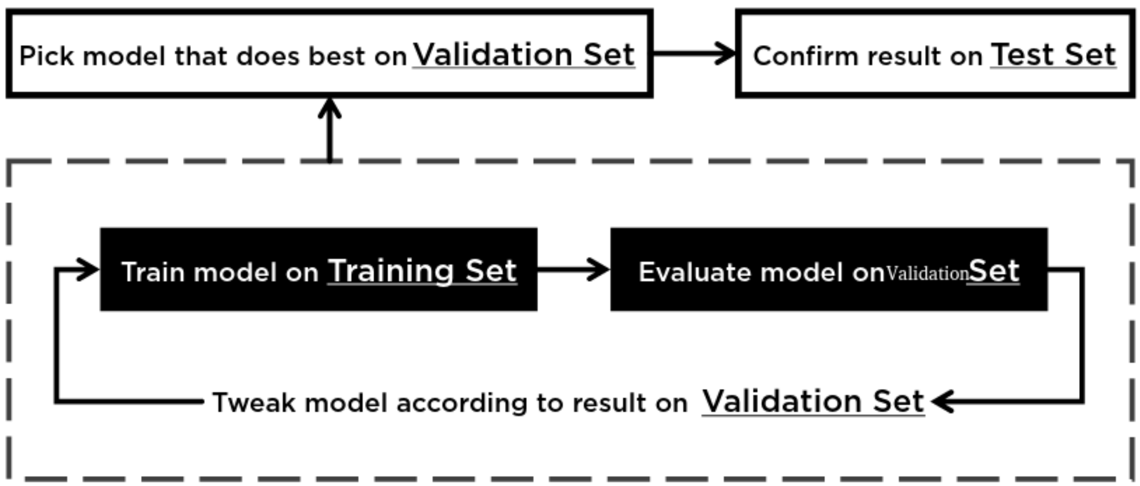
2 Training and Test set

Larger Training Set - The better model we will be able to learn
Larger Test Set - The better we will be able to have confidence in evaluation metrics and tighter confidence intervals

Ensure the test set meets the following 2 conditions:

- Is large enough to yield statistically meaningful results
- Is representative of the data set as a whole

2.1 Validation Set



1. Keeping the test data way off to the side (completely unused)
2. Pick the model that does best on the validation set
3. Double check that model against the test set

This is a better workflow because it creates fewer exposures to the test set

3 Representation

We must create a representation of the data to provide the model with a useful vantage point into the data's key qualities. That is, in order to train a model, we must choose the set of features that best represent the data

3.1 Numeric

This works for some models, but in some cases the gradient will change throughout, so would not work

3.2 Bucketing

One categorical feature is created for each bucket (sections). Then a fitting can be created for each bucket

3.3 Categorical

One hot encoding - Only one category selected at a time (e.g. a person can only have one blood type)

If there are a small number of categories, then use the raw value, for larger numbers, hashing may be needed.

3.4 Feature Crossing

Two different features (e.g. age and blood type), then connect together as one feature (e.g. young people with blood type A)

3.5 Hashing

- Save memory and time
- Adds some noise, but limits the maximum number of possibilities

3.6 Embedding

- Powerful ways to represent large vocabularies
- Tell the model that objects with different names mean the same thing (group together)