Basic Practices

1. Feature Engineering
   * One hot encoding
   * Binning
   * Normalisation
   * Standardisation
   * Dealing with missing features
2. Underfitting and Overfitting
3. Regularisation
4. Vectorisation and Broadcasting

# 1. Feature Engineering

A feature vector is an ordered list of numerical properties of an observed phenomenon. It represents input features to a machine learning model that makes a prediction. The problem of transforming raw data into a dataset, which can be used by a ML model is called feature engineering.

#### One-Hot Encoding

Some learning algorithms only work with numerical feature vectors. So we reperesent each class in a feature as a separate feature and assign that feature to have 0 or 1.

For eg: Instead of

colors = [“red”, “blue”, “green”]

we design three features as

red = [1, 0, 0]

blue = [0, 1, 0]

green = [0, 0, 1]

You should not transform red into 1, yellow into 2, and green into 3 to avoid increasing the dimensionality because that would imply that there’s an order among the values in this category and this specific order is important for the decision making. If the order of a feature’s values is not important, using ordered numbers as values is likely to confuse the learning algorithm, because the algorithm will try to find a regularity where there’s no one, which may potentially lead to overfitting.

#### Binning

An opposite situation, occurring less frequently in practice, is when you have a numerical feature but you want to convert it into a categorical one. Binning (also called bucketing) is the process of converting a continuous feature into multiple binary features called bins or buckets, typically based on value range.

For eg, A dataset with individuals' income datacan be categorized by their income levels into three bins: "Low Income," "Medium Income," and "High Income."

In some cases, a carefully designed binning can help the learning algorithm to learn using fewer examples. It happens because we give a “hint” to the learning algorithm that if the value of a feature falls within a specific range, the exact value of the feature doesn’t matter.

#### Normalization

Normalization is the process of converting an actual range of values which a numerical feature can take, into a standard range of values, typically in the interval [−1, 1] or [0, 1]. Normalizing the data is not a strict requirement. However, in practice, it can lead to an increased speed of learning. Additionally, it’s useful to ensure that our inputs are roughly in the same relatively small range to avoid problems which computers have when working with very small or very big numbers (known as numerical overflow).

#### Standardization

Standardization (or z-score normalization) is the procedure during which the feature values are rescaled so that they have the properties of a standard normal distribution with µ = 0 and σ = 1, where µ is the mean and σ is the standard deviation from the mean.

#### Dealing with Missing Features

The typical approaches of dealing with missing values for a feature include:

• removing the examples with missing features from the dataset

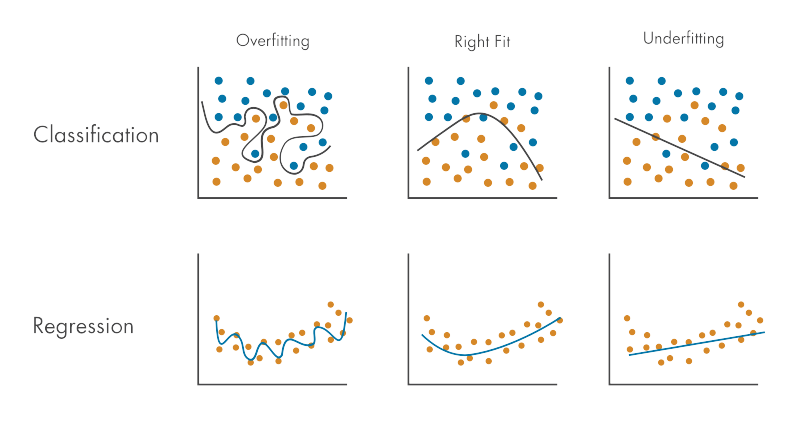
• using a learning algorithm that can deal with missing feature values

• using a data imputation technique.

#### **Data Imputation Techniques**

* One data imputation technique consists in replacing the missing value of a feature by an average value of this feature in the dataset.
* Another technique is to replace the missing value with a value outside the normal range of values.For example, if the normal range is [0, 1], then you can set the missing value to 2 or −1.
* Alternatively, you can replace the missing value by a value in the middle of the range. For example, if the range for a feature is [−1, 1], you can set the missing value to be equal to 0.Here, the idea is that the value in the middle of the range will not significantly affect the prediction.
* A more advanced technique is to use the missing value as the target variable for a regression problem. You can use all remaining features to form a feature vector, where j is the feature with a missing value. Then you build a regression model to predict ŷ from x̂. Of course, to build training examples (x̂, ŷ), you only use those examples from the original dataset, in which the value of feature j is present.
* Finally, if you have a significantly large dataset and just a few features with missing values, you can increase the dimensionality of your feature vectors by adding a binary indicator feature for each feature with missing values.

# 2. Underfitting and Overfitting



#### Underfitting

If the model makes many mistakes on the training data, we say that the model has a high bias or that the model underfits.Underfitting is the inability of the model to predict well the labels of the data it was trained on.

Reasons:

* Model is too simple
* The features you engineered are not informative enough

#### Overfitting

The model that overfits predicts very well the training data but poorly the data from at least one of the two holdout sets. It is also called the problem of high variance.

Reasons:

* The model is too complex for the data
* You have too many features but a small number of training examples

**Regularization is the most widely used approach to prevent overfitting.**

# 3. Regularisation

Regularization is an umbrella term that encompasses methods that force the learning algorithm to **build a less complex model, to reduce overfitting**. In practice, that often leads to slightly higher bias but significantly reduces the variance.

**To create a regularized model, we modify the objective function by adding a penalizing term whose value is higher when the model is more complex.**

A regularised cost function is:

**J = (1/m) Σmi=1( [f(x) – y ]2 ) + (**λ**/2m)Σnj=1 ||w||2 (L2 regularisation)**

where λ is the regularisation parameter, and is manully set.

when λ == 0: no regularisation (potential overfitting)

when λ == very big(1010): high regularisation(potential underfitting)

# Another regularisation technique is **dropout regularisation**, which is used on NN. Here, neurons from layers are randomly ‘switched off’. This leads to simpler networks – reducing overfitting.

# 4. Vectorisation and Broadcasting

Vectorization and broadcasting are techniques used in Python, particularly with libraries like NumPy, to boost the speed and efficiency of numerical operations on arrays or matrices.

#### Vectorization

Vectorization is the process of rewriting a loop-based, scalar computation to work on entire arrays or vectors at once. Instead of performing operations element-wise, you apply operations directly to the entire array. It significantly speeds up numerical computations because it takes advantage of low-level optimizations provided by libraries like NumPy.

Eg: Using np.dot(m1,m2) instead of multiplying over a loop

#### Broadcasting

Broadcasting is a technique used in NumPy (and similar libraries) to perform operations on arrays with different shapes, making them compatible without creating unnecessary copies of data. **It allows you to perform element-wise operations on arrays of different shapes, which can simplify code and reduce memory usage.** It's particularly useful when working with arrays of differing dimensions.

Eg: arr + 3, adds 3 to every element in the array by broadcasting.