Basic Practices

# 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.

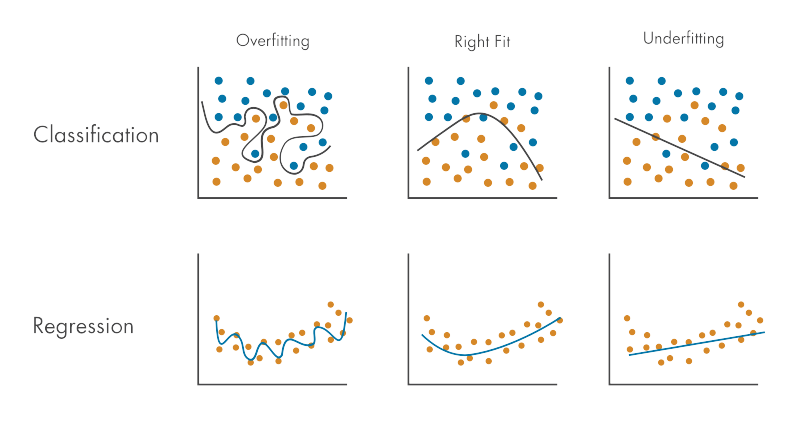
# 2. Three Datasets

1) Training set

2) Validation set

3) Test set

# 3. 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.**

# 4. Regularisation

Regularization is an umbrella term that encompasses methods that force the learning algorithm to build a less complex model. 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=1w2**

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

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

# 5. Model’s Assessment

The test set contains the examples that the learning algorithm has never seen before, so if our model performs well on predicting the labels of the examples from the test set, we say that our model generalizes well or, simply, that it’s good.

#### Regression

First, we compute the mean squared error (or any such error measure) for the training, and, separately, for the test data. If the MSE of the model on the test data is substantially higher than the MSE obtained on the training data, this is a sign of overfitting. If both underperforms, it is underfitting.

#### Classification

**Confusion Matrix**

The confusion matrix is a table that summarizes how successful the classification model is at predicting examples belonging to various classes. One axis of the confusion matrix is the label that the model predicted, and the other axis is the actual label. Confusion matrix is used to calculate two other performance metrics: precision and recall.

It also helps us to have an idea about where the prediction is going wrong. This can help us address that problem with more data focusing on that problem.



**Precision/Recall – Best for skewed class prediction**

Precision is the ratio of **correct positive prediction**s to the **overall number of positive predictions**.

Precision = TP/ (TP+FP)

Recall is the ratio of **correct positive predictions** to the overall number of **correct predictions** in the dataset.

Recall = TP/ (TP+FN)

**High precision => if prediction == Positive: result = Positive # says + only if sure**

**High recall => if result = Positive: prediction == Positive # says + if there is a chance**

Almost always, in practice, we have to choose between a high precision or a high recall. It’s usually impossible to have both.

**Accuracy**

Accuracy is given by the number of correctly classified examples divided by the total number of classified examples.

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

**Cost-sensitive Accuracy**

For dealing with the situation in which different classes have different importance, a useful metric is cost-sensitive accuracy. To compute a cost-sensitive accuracy, you first assign a cost (a positive number) to both types of mistakes: FP and FN. You then compute the counts TP, TN, FP, FN as usual and multiply the counts for FP and FN by the corresponding cost before calculating the accuracy

**Area under the ROC Curve**

didn’t read

# 7. Hyperparameter Tuning

Hyperparameters are variables that are not optimized by the learning algorithm itself. The data analyst has to “tune” hyperparameters by experimentally finding the best combination of values, one per hyperparameter.

One method is **grid search**, but trying all combinations of hyperparameters, especially if there are more than a couple of them, could be time-consuming.

There are more efficient techniques, such as random search and Bayesian hyperparameter optimization. **Random search** differs from grid search in that you no longer provide a discrete set of values to explore for each hyperparameter; instead, you provide a statistical distribution for each hyperparameter from which values are randomly sampled and set the total number of combinations you want to try. **Bayesian techniques** differ from random or grid search in that they use past evaluation results to choose the next values to evaluate.

There are also **gradient-based techniques, evolutionary optimization techniques**, and other algorithmic hyperparameter tuning techniques.