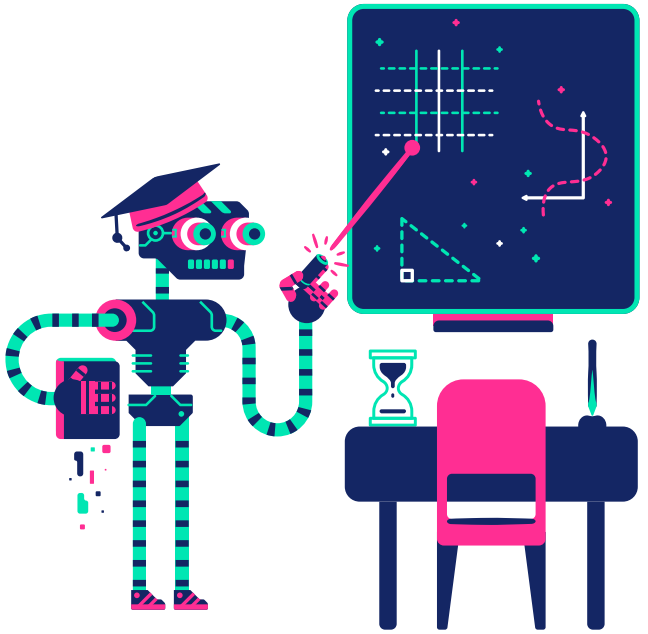
Performance Improvement and Metrics

Penyusun Modul: Chairul Aulia

Editor: Citra Chairunnisa

Machine Learning Performance

Enhancing a model performance can be challenging at times. I'm sure, a lot of you would agree with me if you've found yourself stuck in a similar situation. You try all the strategies and algorithms that you've learned. Yet, you fail at improving the accuracy of your model.
Previously we have learned how to improve through ensemble learning. Today we will see what we can do to improve our model performance even before the model is built!



Adding more useful data

Researchers have demonstrated that **massive data** can lead to **lower estimation variance** and hence **better predictive** performance. More data increases the probability that it contains useful information, which is advantageous.

We might don't get an option to add more data. For example: we do not get a choice to increase the size of training data in data science competitions. But while working on a company project, you are suggested to ask for more data, if possible.

But please take a note that there comes a stage where even adding infinite amounts of data cannot improve any more accuracy.
"It is not just big data, but good (quality) data which helps us build better performing ML models. If we have a huge data repository with features which are too noisy or not having enough variation to capture critical patterns in the data, any ML models will effectively be useless regardless of the data volume."

Data Cleaning

Data Cleaning means the process of identifying the incorrect, incomplete, inaccurate, irrelevant or missing part of the data and then modifying, replacing or deleting them according to the necessity. Data cleaning is considered a foundational element of the basic data science.

(We already covered this topic in detail)

Handling imbalanced data

Imbalanced data refers to those types of datasets where the target class has an **uneven distribution** of observations, i.e one class label has a very high number of observations and the other has a very low number of observations.
Example: Disease diagnosis, Customer churn prediction, Fraud detection, Natural disaster

There are some techniques to deal with the imbalanced problems:

- Using the right evaluation metrics
- Resampling (under-sampling, over-sampling)
- Using Ensemble methods

Under-sampling/Downn-sampling

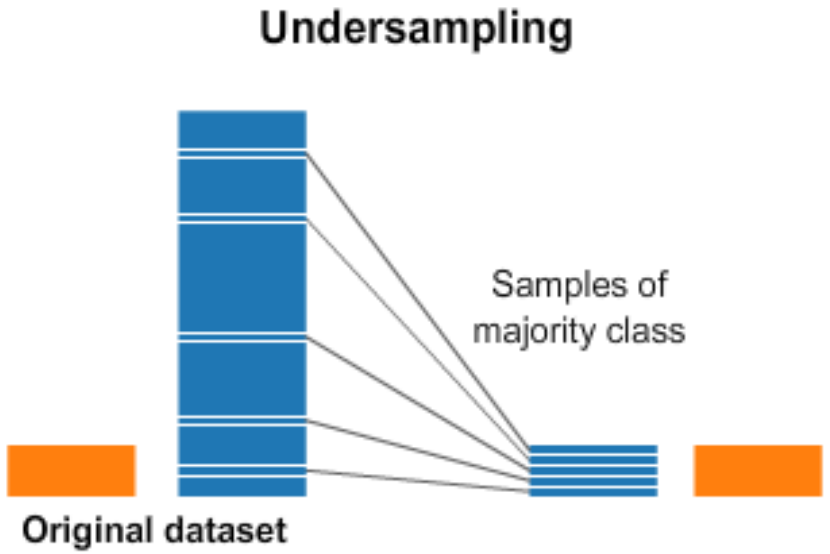
It aims to balance class distribution by randomly **eliminating majority** class examples. This is done until the majority and minority class instances are balanced out.

Advantages

- Run-time can be improved by decreasing the amount of training dataset.
- Helps in solving the memory problems

Disadvantages

- Losing some critical information
- Sample chosen by random under sampling may be a biased sample



Over-sampling/Up-sampling

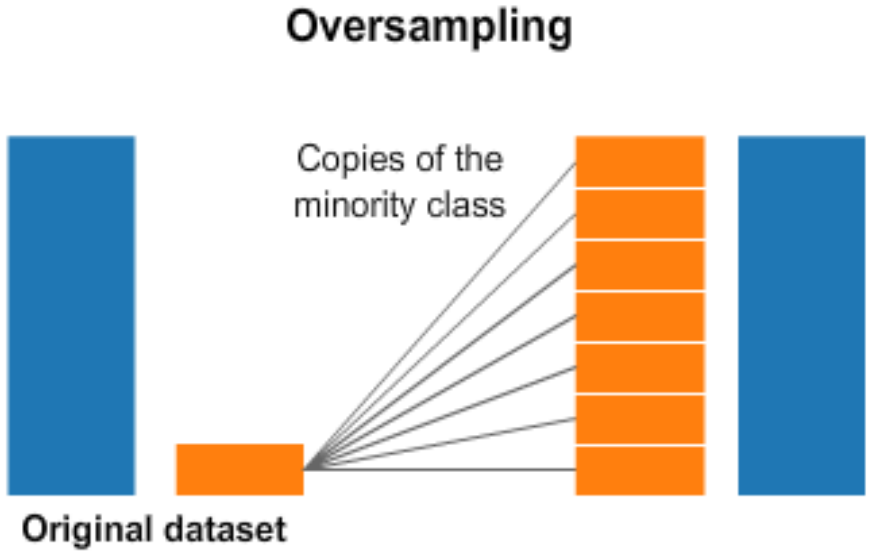
Over-Sampling **increases** the number of instances in the **minority** class by randomly replicating them in order to present a higher representation of the minority class in the sample.

Advantages

- No information loss.
- Outperforms under sampling

Disadvantages

- It increases the likelihood of overfitting since it replicates the minority class events.

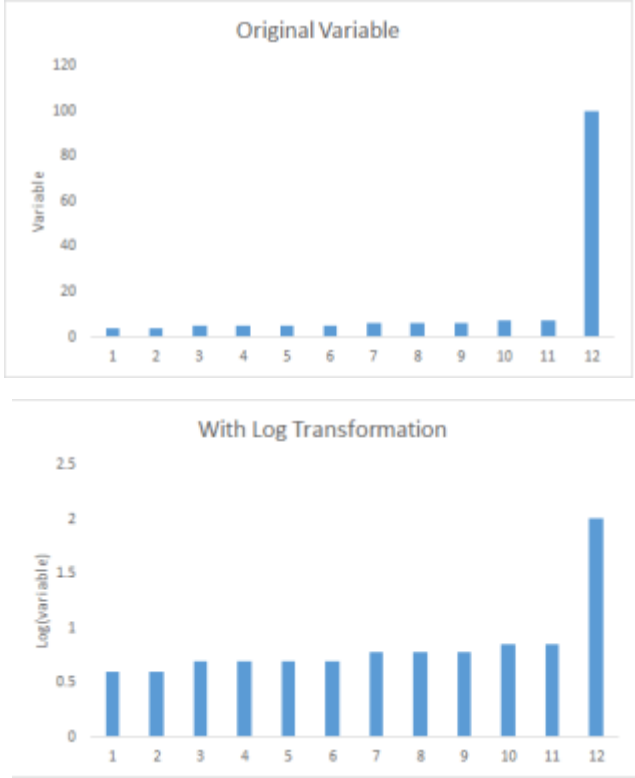


Feature Engineering

This step helps to extract more information from existing data. New information is extracted in terms of new features. These features may have a higher ability to explain the variance in the training data

This process can be divided into 2 steps:

1. Feature transformation. To remove skewness, normalize the variables or change the scale
2. Feature Creation. Deriving new variable(s) from existing variables is known as feature creation. It helps to unleash the **hidden relationship** of a data set.
   Let's say, we want to predict the number of transactions in a store based on transaction dates. Here transaction dates may not have direct correlation with number of transaction, but if we look at the day of a week, it may have a higher correlation. In this case, the information about day of a week is hidden



Remove skewness with log transformation

Feature Selection

Feature Selection is a process of finding out **the best subset of attributes** which better explains the relationship of independent variables with target variable.

You can select features based on:

- Domain knowledge
- Visualization
- Statistical parameters.

Feature Extraction

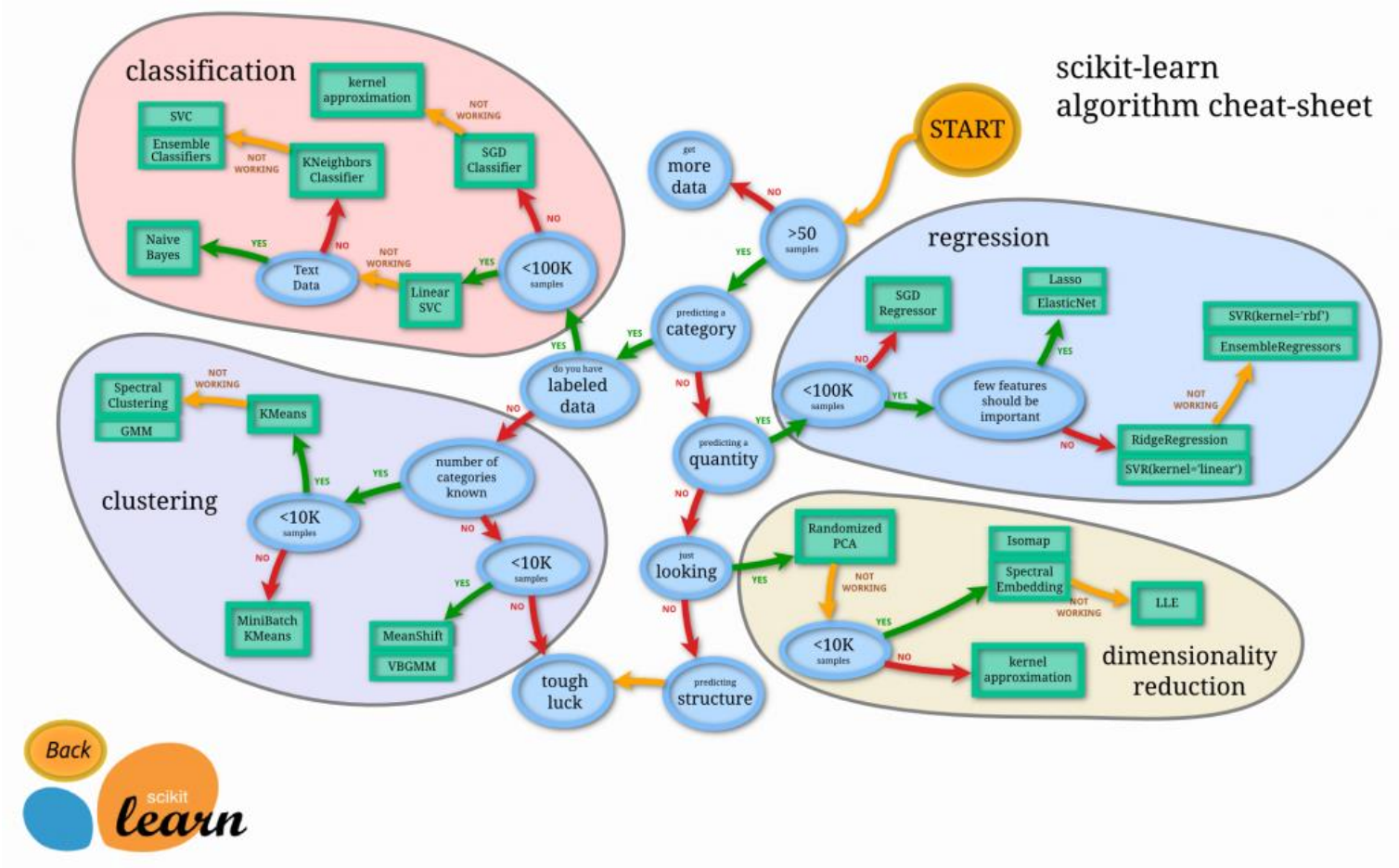
Feature extraction is for creating a new, smaller set of features that stills captures most of the useful information. Again, feature selection keeps a subset of the original features while feature extraction creates new sets of components.

Some methods:
Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) both **reduce the dimensionality** of data by looking for linear combinations of the features which best explain the data.
Main difference:
LDA focuses on finding a feature subspace that **maximizes the separability** between the groups.
PCA focuses on capturing the direction of **maximum variation** in the data set.

Compare Multiple Algorithms

Hitting at the right machine learning algorithm is the ideal approach to achieve higher accuracy. But, it is easier said than done.

This intuition comes with experience and incessant practice. Some algorithms are better suited to a particular type of data sets than others. Hence, we should apply all relevant models and check the performance.

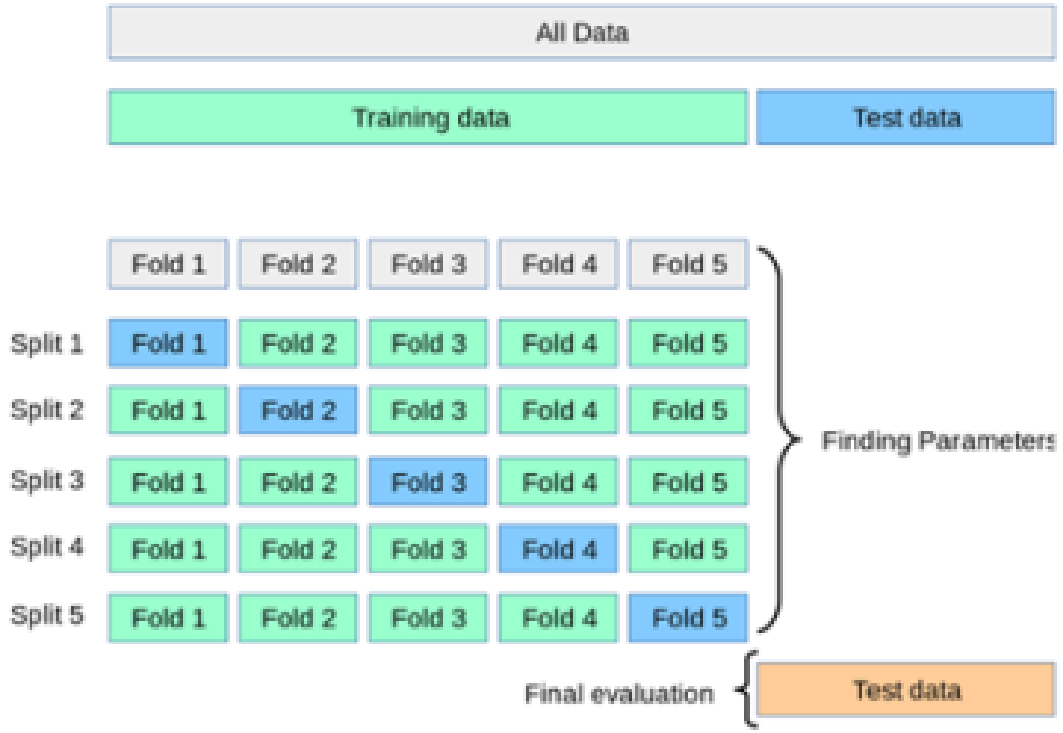


Algorithm Tuning

We know that machine learning algorithms are driven by parameters. These parameters majorly influence the outcome of learning process.
The objective of parameter tuning is to find the optimum value for each parameter to improve the accuracy of the model. To tune these parameters, you must have a good understanding of these meaning and their individual impact on model. You can repeat this process with a number of well performing models.
For example: In random forest, we have various parameters like `max_features`, `number_trees`, `random_state`, `oob_score` and others. Intuitive optimization of these parameter values will result in better and more accurate models.

K-Fold Cross Validation

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.
The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=5 becoming 5-fold cross-validation.



Exercise

Let's implement k-fold cross validation using `model_selection` module in sklearn and choose whatever machine learning model you want!

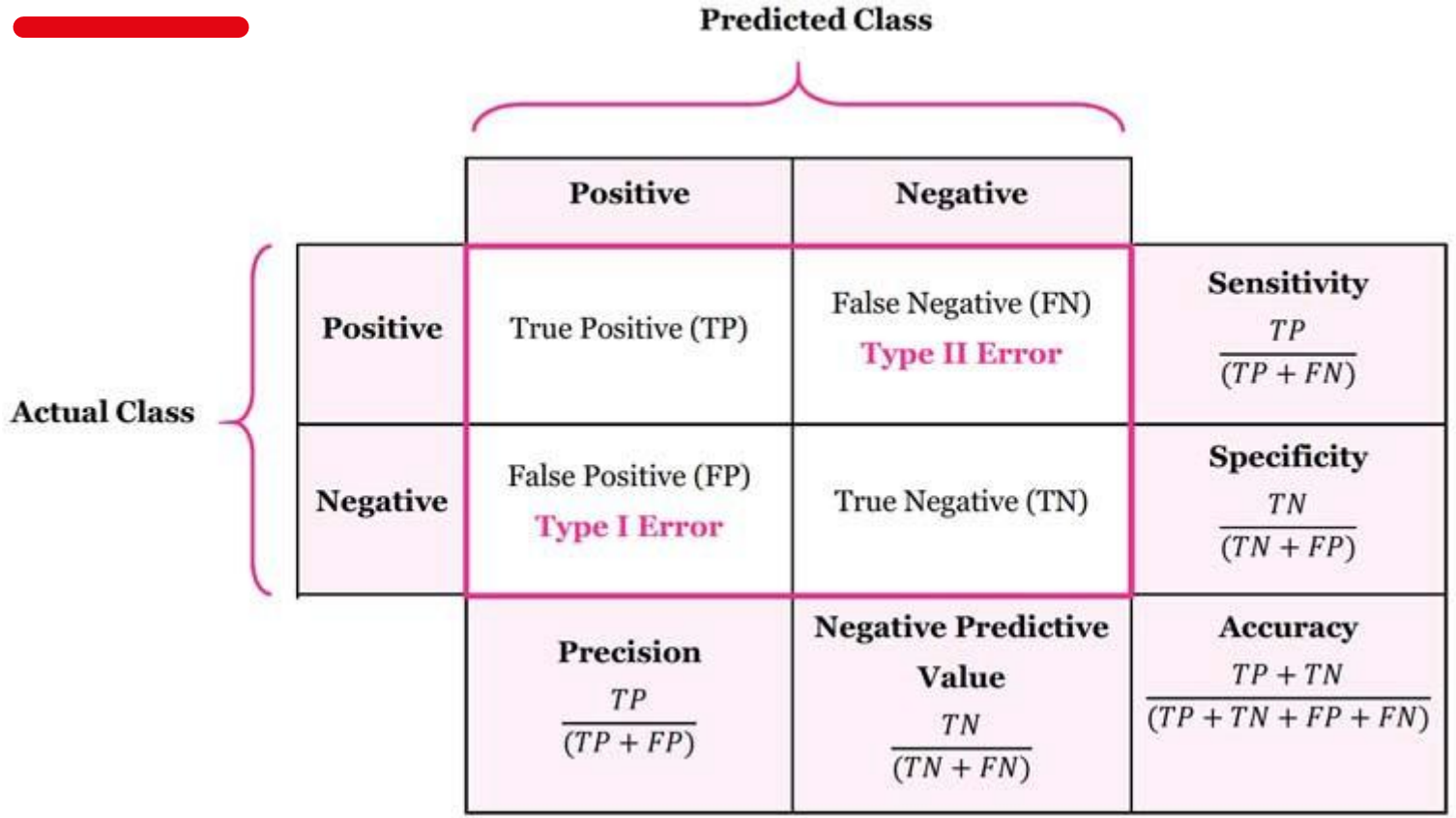
Use sample dataset with `load_breast_cancer()`

Performance Metrics in ML

The model performance gives a measure of how well a Machine Learning model is performing. But, is it all about the accuracy of the model's predictions? Let's see what metrics are used in machine learning other than accuracy.

Please note that **metrics are different with loss functions**. Loss functions also show a measure of model performance, but they are used to **optimize our model** while metrics to **judge the model performance** and has nothing to do with optimization. Loss functions might differ for several algorithms, but metrics don't need to differ. Metrics only differ based on the application (classification/regression)

Confusion Matrix



Confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

Confusion Matrix

Consider the following values for the confusion matrix-
True negatives (TN) = 300
True positives (TP) = 500
False negatives (FN) = 150
False positives (FP) = 50

Accuracy = (500+300)/(500+50+150+300) = 800/1000 = 80%
Recall/Sensitivity = 500/(500+150) = 500/650 = 76.92%
Precision = 500/(500+50) = 500/550 = 90.90%
Specificity = 300/(300+50) = 300/350 = 85.71%

Precision tells us how many of the correctly predicted cases actually turned out to be positive (useful when FP is a higher concern than FN/type I error is not tolerated)

Recall tells us how many of the actual positive cases we were able to predict correctly (useful when FN is a higher concern/type II error is not tolerated)

Measurement	Formula
Accuracy	$\frac{TP+TN}{P+N}$
Misclassification rate	$\frac{FP+FN}{P+N}$
Sensitivity	$\frac{TP}{P}$
Specificity	$\frac{TN}{N}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F, F_1, F-score	$\frac{2 \times precision \times sensitivity}{precision + sensitivity}$
F_β, where β is real non negative	$\frac{(1+\beta^2) \times precision \times sensitivity}{\beta^2 \times precision + sensitivity}$

Regression Metrics

Mean squared error	$\mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\mathrm{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

Exercise

Let’s calculate regression metrics of linear regression model and classification metrics of decision tree model!

Use **Salary_Data** and **White_Wine_Clean** datasets