Tecnológico de Monterrey

Modelling and Evaluation II

CS5056 Data Analytics

Francisco J. Cantú, Héctor Ceballos
Tecnológico de Monterrey

March 10, 2021

Februrary-June, 2021

How do we Select and Evaluate the best Model that Fits a Dataset

Tecnológico de Monterrey

CS5056 Data Analytics

March 10, 2021

Class 5/16

Agenda

ORDINARY LEAST SQUARES

OVERFITTING & MODEL VALIDATION

UNBALANCED CLASSES

DECISION TREES, BAGGING, RANDOM FOREST, BOOSTING

LOGISTIC REGRESSION

CS5056 Data Analytics

March 10, 2021

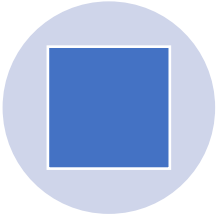
Class 5/16

Agenda

SUPPORT VECTOR MACHINES

K-NEAREST NEIGHBORS

MODEL SELECTION



HANDS-ON EXERCISES IN PYTHON, R, WEKA

CS5056 Data Analytics

March 17, 24 2021

Class 6,7/16

Agenda

PANEL DATA

PROBABILISTIC MODELLING

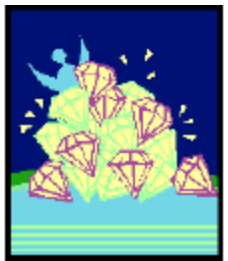
BAYES CLASSIFIER

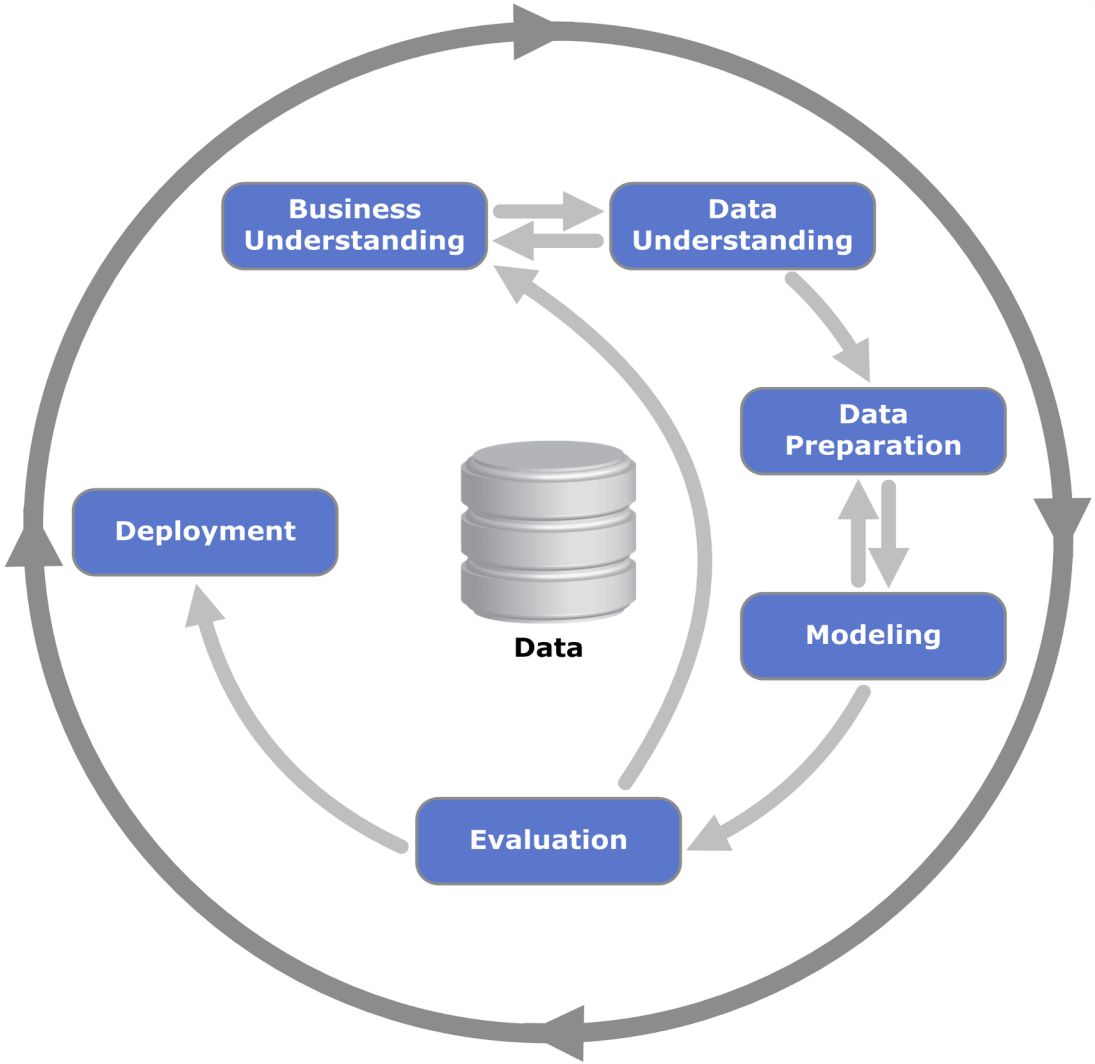
BAYESIAN NETWORKS

PARAMETER OPTIMIZATION: LASSO, Ridge, Bayesian

HANDS-ON EXERCISES USING PYTHON, R, WEKA

Data Mining Cycle: CRISP-DM





Ordinary Least Squares (OLS)

- OLS is a type of linear least squares method for estimating the unknown parameters in a linear regression model.

- OLS chooses the parameters of a linear function of a set of explanatory variables by the principle of least squares: minimizing the sum of the squares of the differences between the observed dependent variable (values of the variable being observed) in the given dataset and those predicted by the linear function of the independent variable.

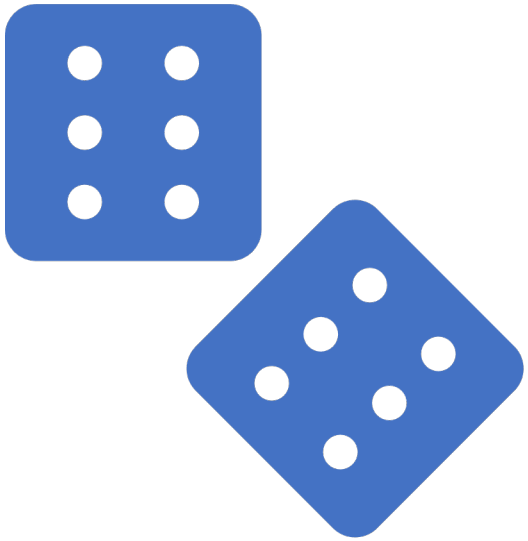
https://en.wikipedia.org/wiki/Ordinary_least_squares

Ordinary Least Squares (OLS)

- The OLS estimator is consistent when the regressors are **exogenous**, and—by the Gauss–Markov theorem—optimal in the class of linear unbiased estimators when the errors are **homoscedastic** and **serially uncorrelated.**
- A process is homoscedastic if all its random variables have the same finite variance. If not, it is called *heteroscedastic*.
- Under these conditions, the method of OLS provides minimum-variance and mean-unbiased estimation when the errors have finite variances.
- Under the additional assumption that the errors are normally distributed, OLS is the *maximum likelihood estimator*.

Ordinary Least Squares (OLS)

- There can be any desired relationship between the regressors so long as it is not a linear relationship, called **colinearity**.
- For instance, we might suspect the response depends linearly both on a value and its square; in which case we would include one regressor whose value is just the square of another regressor.
- In that case, the model would be *quadratic* in the second regressor, but none-the-less is still considered a *linear* model because the model *is* still linear in the parameters.



Ordinary Least Squares (OLS)

Suppose the data consists of n observations $\{ y_i, x_i \}_{i=1}^{n}$. Each observation i includes a scalar response y_i and a column vector x_i of values of p parameters (regressors) x_{ij} for $j = 1, ..., p$. In a linear regression model, the response variable, y_i, is a linear function of the regressors:

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i,$$

or in vector form,

$$y_i = \mathbf{x}_i^{\mathsf{T}} \boldsymbol{\beta} + \varepsilon_i,$$

where $\boldsymbol{x}_i$ is a column vector of the ith observations of all the explanatory variables; $\boldsymbol{\beta}$ is a $p \times 1$ vector of unknown parameters; and the scalars ε_i represent unobserved random variables (errors) which account for influences upon the responses y_i from sources other than the explanators $\boldsymbol{x}_i$.

Ordinary Least Squares (OLS)

Matrix/vector formulation [edit]

Consider an overdetermined system

$$\sum_{j=1}^{p} X_{ij}\beta_j = y_i, \ (i = 1, 2, \ldots, n),$$

of n linear equations in p unknown coefficients, $\beta_1, \beta_2, ..., \beta_p$, with $n > p$. (Note: for a linear model as above, not all of X contains information on the data points. The first column is populated with ones, $X_{i1} = 1$, only the other columns contain actual data, so here p = number of regressors + 1.) This can be written in matrix form as

$$\mathrm{X}\boldsymbol{\beta} = \mathbf{y},$$

where

$$\mathrm{X} = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1p} \\ X_{21} & X_{22} & \cdots & X_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{np} \end{bmatrix}, \qquad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

Such a system usually has no exact solution, so the goal is instead to find the coefficients $\boldsymbol{\beta}$ which fit the equations "best", in the sense of solving the quadratic minimization problem

$$\hat{\boldsymbol{\beta}} = \operatorname{arg\,min}_{\boldsymbol{\beta}} S(\boldsymbol{\beta}),$$

where the objective function S is given by

$$S(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left| y_i - \sum_{j=1}^{p} X_{ij}\beta_j \right|^2 = \|\mathbf{y} - \mathrm{X}\boldsymbol{\beta}\|^2.$$

Ordinary Least Squares (OLS)

If the data matrix X contains only two variables, a constant and a scalar regressor x_i, then this is called the "simple regression model".[12] This case is often considered in the beginner statistics classes, as it provides much simpler formulas even suitable for manual calculation. The parameters are commonly denoted as (α, β):

$$y_i = \alpha + \beta x_i + \varepsilon_i .$$

The least squares estimates in this case are given by simple formulas

$$\hat{\beta} = \frac{\sum x_i y_i - \frac{1}{n}\sum x_i \sum y_i}{\sum x_i^2 - \frac{1}{n}(\sum x_i)^2} = \frac{\operatorname{Cov}[x, y]}{\operatorname{Var}[x]}$$

$$\hat{\alpha} = \overline{y} - \hat{\beta}\,\overline{x}\ ,$$

where Var(.) and Cov(.) are sample parameters.

Hands-on Exercise with Ordinary Linear Squares Model (OLS) using Python

- Statsmodel
- Download script 1-ols.py

OLS Regression Results

Dep. Variable:	y	R-squared:	0.862
Model:	OLS	Adj. R-squared:	0.806
Method:	Least Squares	F-statistic:	15.56
Date:	Fri, 05 Mar 2021	Prob (F-statistic):	0.00713
Time:	18:27:04	Log-Likelihood:	-24.316
No. Observations:	8	AIC:	54.63
Df Residuals:	5	BIC:	54.87
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.5226	4.431	1.246	0.268	-5.867	16.912
x1	0.4471	0.285	1.567	0.178	-0.286	1.180
x2	0.2550	0.453	0.563	0.598	-0.910	1.420

Omnibus:	0.561	Durbin-Watson:	3.268
Prob(Omnibus):	0.755	Jarque-Bera (JB):	0.534
Skew:	0.380	Prob(JB):	0.766
Kurtosis:	1.987	Cond. No.	80.1

Interpretation

- **R-squared:** It signifies the “percentage variation in dependent that is explained by independent variables”. Here, 73.2% variation in y is explained by X1, X2, X3, X4 and X5. This statistic has a drawback, it increases with the number of predictors(dependent variables) increase. Therefore, it becomes inconclusive in case when it is to be decided whether additional variable is adding to the predictability power of the regression.

- **Adj. R-squared:** This is the modified version of R-squared which is adjusted for the number of variables in the regression. It increases only when an additional variable adds to the explanatory power to the regression.

- **Prob(F-Statistic):** This tells the overall significance of the regression. This is to assess the significance level of all the variables together unlike the t-statistic that measures it for individual variables. The null hypothesis under this is “all the regression coefficients are equal to zero”. Prob(F-statistics) depicts the probability of null hypothesis being true. As per the above results, probability is close to zero. This implies that overall, the regressions is meaningful.

https://medium.com/@jyotiyadav99111/statistics-how-should-i-interpret-results-of-ols-3bde1ebeec01

Interpretation

- **AIC/BIC:** It stands for Akaike's Information Criteria and is used for model selection. It penalizes the errors mode in case a new variable is added to the regression equation. It is calculated as number of parameters minus the likelihood of the overall model. A lower AIC implies a better model. BIC stands for Bayesian information criteria and is a variant of AIC where penalties are made more severe.

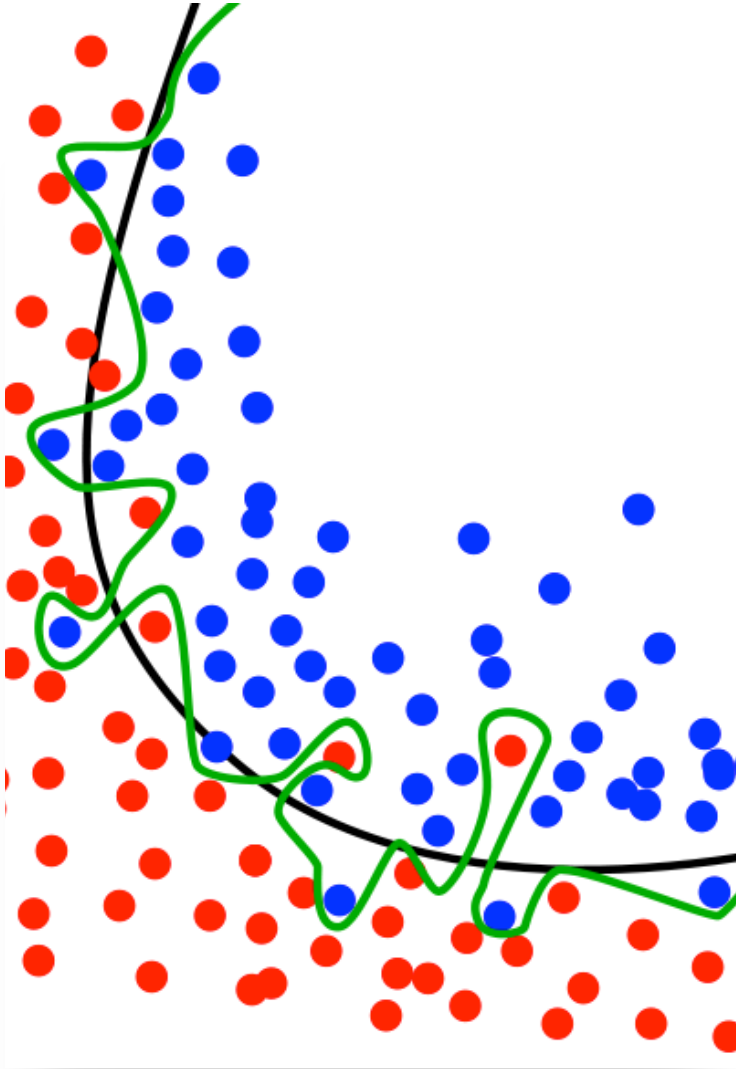
- **Prob(Omnibus):** One of the assumptions of OLS is that the errors are normally distributed. Omnibus test is performed in order to check this. Here, the null hypothesis is that the errors are normally distributed. Prob(Omnibus) is supposed to be close to the 1 in order for it to satisfy the OLS assumption. In this case Prob(Omnibus) is 0.062, which implies that the OLS assumption is not satisfied. Due to this, the coefficients estimated out of it are not Best Linear Unbiased Estimators(BLUE).

- **Durbin-watson:** Another assumption of OLS is of homoscedasticity. This implies that the variance of errors is constant. A value between 1 to 2 is preferred. Here, it is ~1.8 implying that the regression results are reliable from the interpretation side of this metric.

- **Prob(Jarque-Bera):** It is in line with the Omnibus test. It is also performed for the distribution analysis of the regression errors. It is supposed to agree with the results of Omnibus test. A large value of JB test indicates that the errors are not normally distributed.

https://medium.com/@jyotiyadav99111/statistics-how-should-i-interpret-results-of-ols-3bde1ebeec01

Overfitting



https://en.wikipedia.org/wiki/Overfitting

Underfitting

Overfitting

Ok-fitting

How do we deal with Overfitting?

Model Validation

Cross Validation

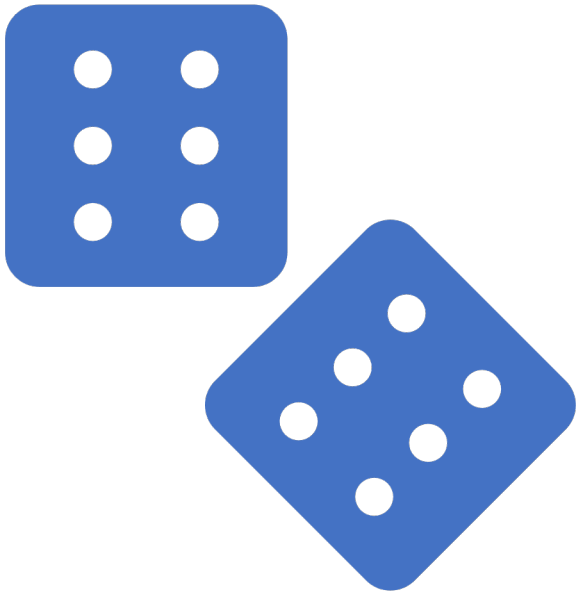
- Assesses how the results of a data model will generalize to an independent data set.
- It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice
- In a prediction problem, a model is usually given a dataset of known data on which training is run (training dataset), and a dataset of unknown data against which the model is tested (called the testing set)
- The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset

https://en.wikipedia.org/wiki/Cross-validation_(statistics)

Hold-out Model Validation

- https://medium.com/the-code-monster/split-a-dataset-into-train-and-test-datasets-using-sk-learn-acc7fd1802e0

- Randomly assign data points to two sets usually called the training set and the test set
- The size of each of the sets is arbitrary although typically the test set is smaller than the training set. For instance, 70-30% or 80-20% for training and test data respectively
- We then fit a model (build the model) on the train set and test (evaluate its performance) on the test dataset



Hands-on Exercise with Hold-out Validation using Python

Using Sklearn and Statsmodels

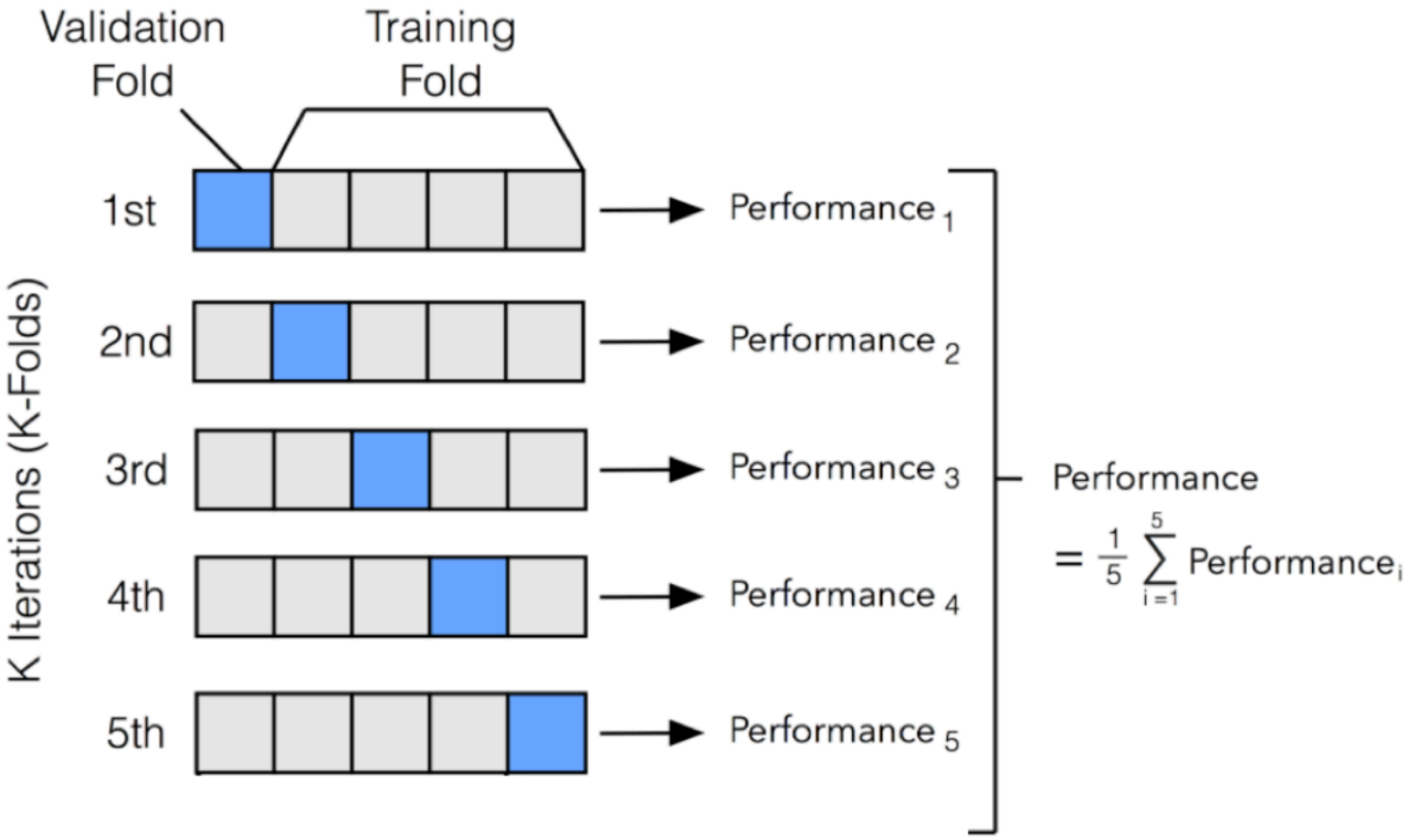
- Download script 2-ols-holdout.py

K-Fold Cross Validation

- In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples
- Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k − 1 subsamples are used as training data
- The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as the validation data
- The k results can then be averaged to produce a single estimation
- All observations are used for both training and validation, and each observation is used for validation exactly once.
- 10-fold cross-validation is commonly used, but in general k remains an unfixed parameter

https://en.wikipedia.org/wiki/Cross-validation_(statistics)

K-Fold Cross Validation



K-Fold Cross Validation

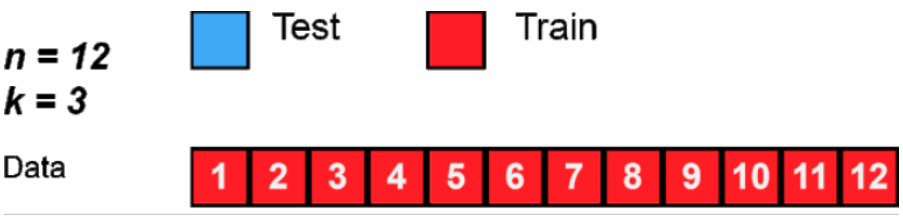
It generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a hold-out split.

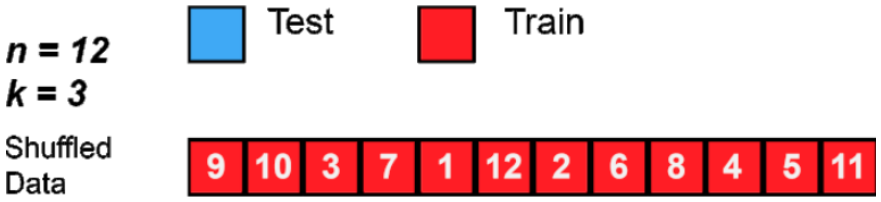
The general procedure is as follows:

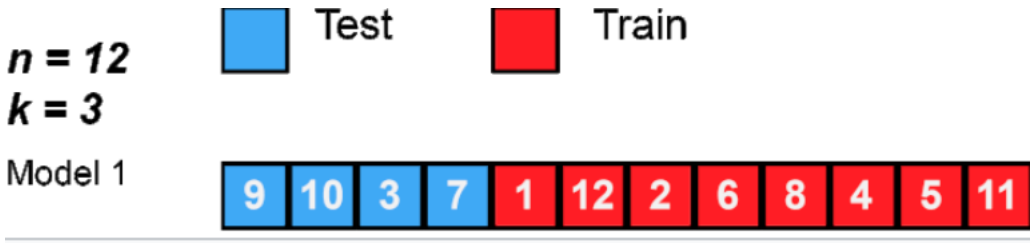
- Shuffle the dataset randomly.
- Split the dataset into k groups
- For each unique group:
- Take the group as a hold out or test data set
- Take the remaining groups as a training data set
- Fit a model on the training set and evaluate it on the test set
- Retain the evaluation score and discard the model
- Summarize the skill of the model using the sample of model evaluation scores

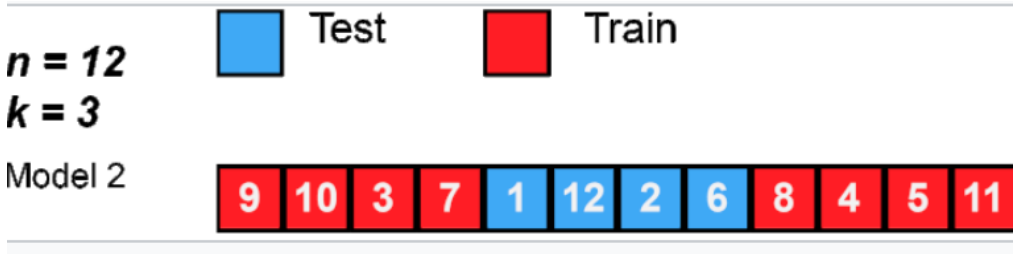
- Importantly, each observation in the data sample is assigned to an individual group and stays in that group for the duration of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times.

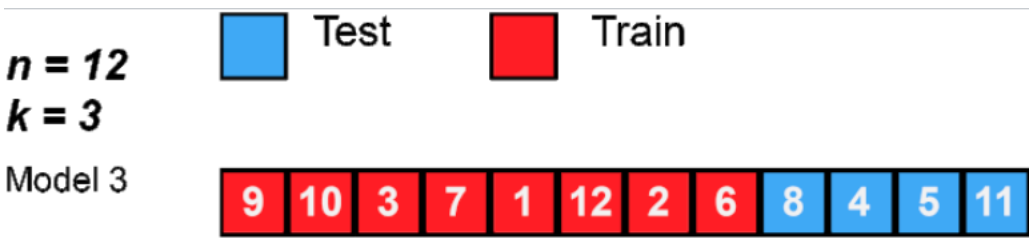
https://machinelearningmastery.com/k-fold-cross-validation/


n = 12
k = 3
Test
Train
Data
1 2 3 4 5 6 7 8 9 10 11 12


n = 12
k = 3
Test
Train
Shuffled Data
9 10 3 7 1 12 2 6 8 4 5 11


n = 12
k = 3
Test
Train
Model 1
9 10 3 7 1 12 2 6 8 4 5 11


n = 12
k = 3
Test
Train
Model 2
9 10 3 7 1 12 2 6 8 4 5 11


n = 12
k = 3
Test
Train
Model 3
9 10 3 7 1 12 2 6 8 4 5 11

Variations of K-Fold Cross Validation

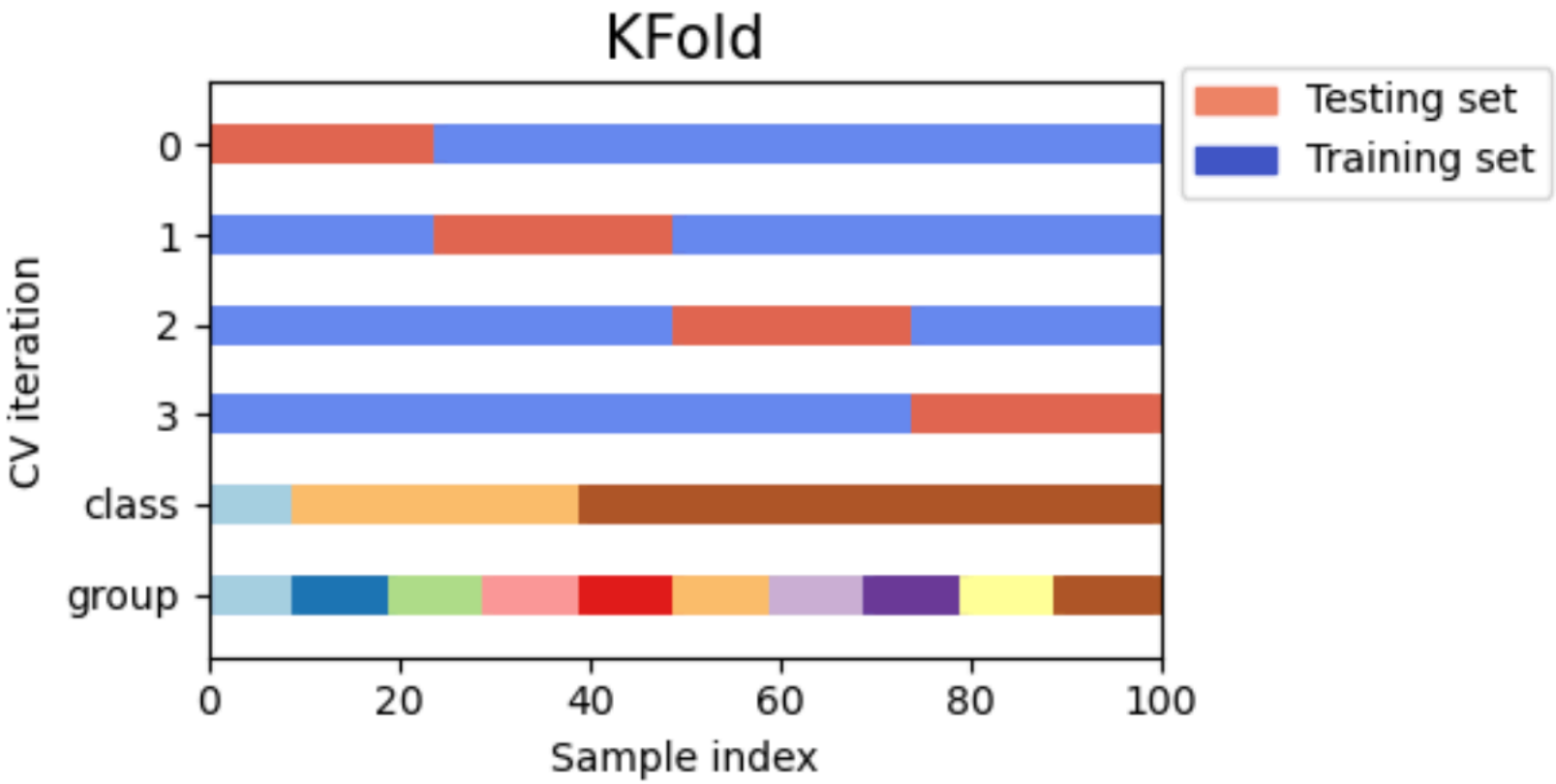
- **Hold-out Split:** Taken to one extreme, k may be set to 2 (not 1) such that a single train/test split is created to evaluate the model.
- **LOOCV:** Taken to another extreme, k may be set to the total number of observations in the dataset such that each observation is given a chance to be the held out of the dataset. This is called leave-one-out cross-validation, or LOOCV for short.
- **Stratified:** The splitting of data into folds may be governed by criteria such as ensuring that each fold has the same proportion of observations with a given categorical value, such as the class outcome value. This is called stratified cross-validation.
- **Repeated:** This is where the k-fold cross-validation procedure is repeated n times, where importantly, the data sample is shuffled prior to each repetition, which results in a different split of the sample.
- **Nested:** This is where k-fold cross-validation is performed within each fold of cross-validation, often to perform hyperparameter tuning during model evaluation. This is called nested cross-validation or double cross-validation.

https://machinelearningmastery.com/k-fold-cross-validation/

Hands-on Exercise with K-Fold Cross Validation using Python

- Using Sklearn and Statsmodels
- Download script 3-k-fold-xval.py

https://scikit-learn.org/stable/auto_examples/model_selection/plot_cv_indices.html


KFold
Testing set
Training set
CV iteration
0
1
2
3
class
group
0
20
40
60
80
100
Sample index

Metrics for Evaluation Classification Models

Hypothesis Testing

Null Hypothesis H_o is true

Alternative Hypothesis H_1 is true

Decision Errors

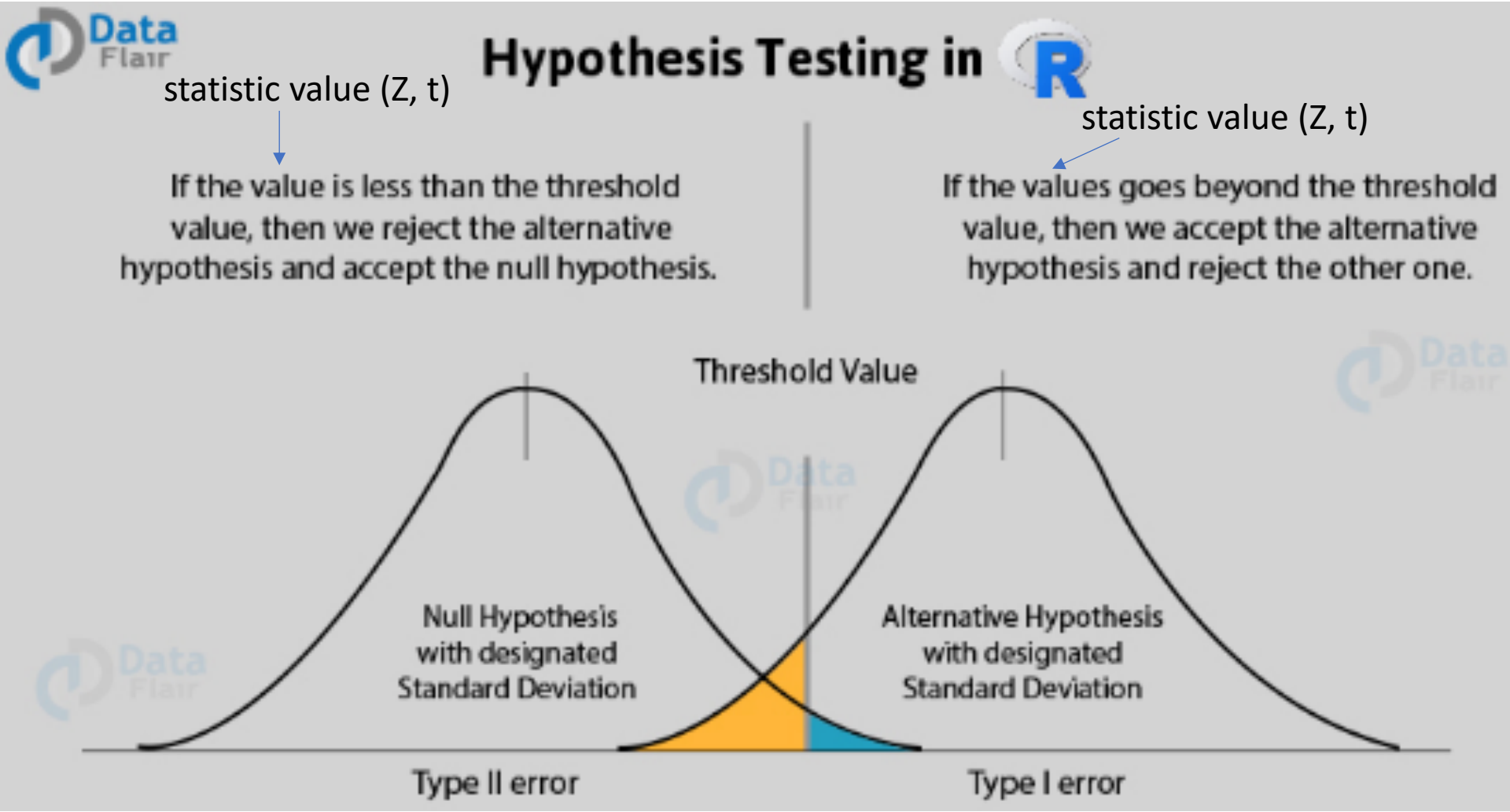
The two types of error that can occur from hypothesis testing:

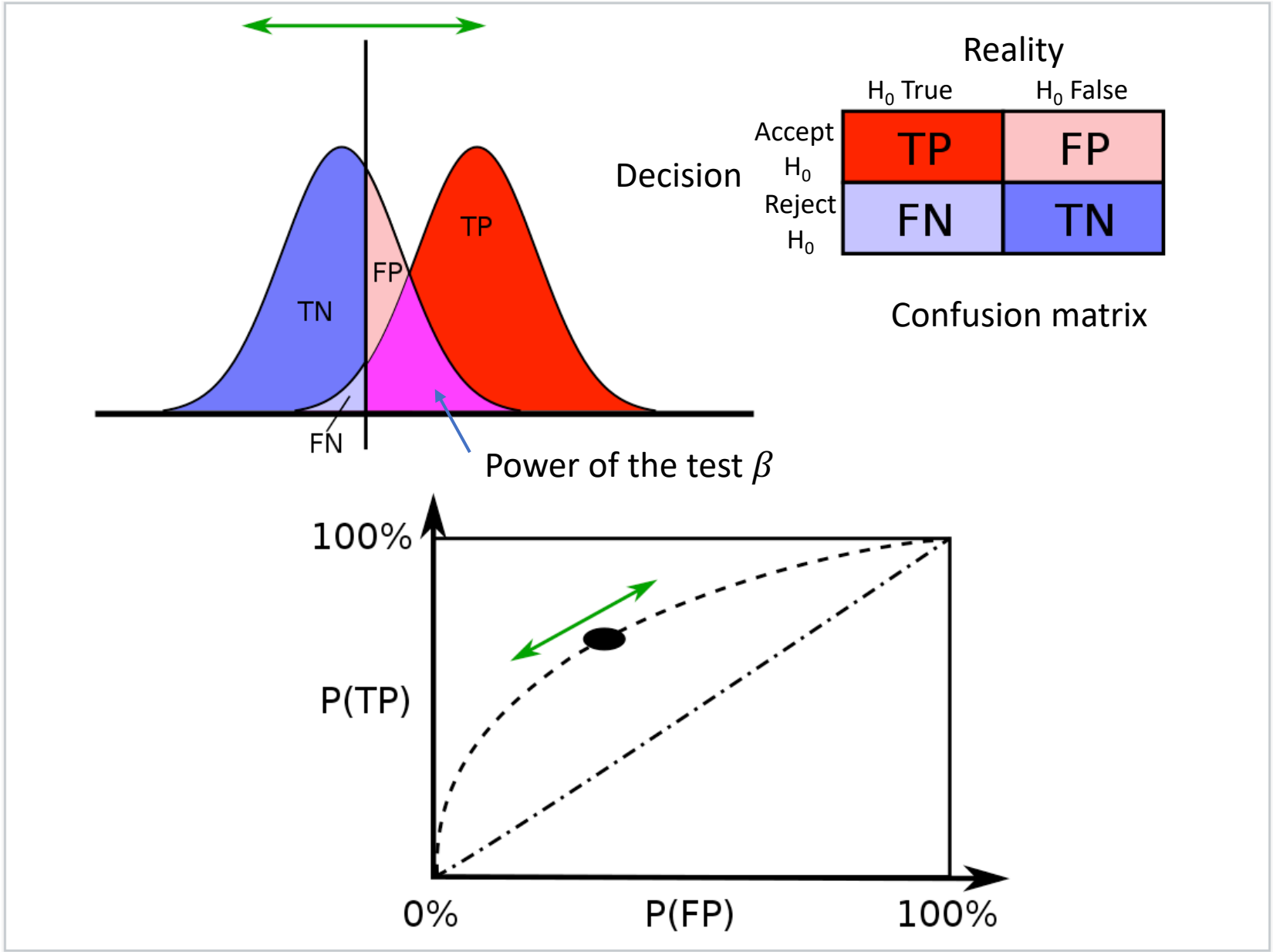
- **Type I Error**
  - Occurs when we reject a true null hypothesis
  - It is called a False Positive (FP)
  - **Significance lev**el is the probability of a Type I error
  - The significance level is represented by the symbol α (Blue area)

- **Type II Error**
  - Occurs when we accept a false null hypothesis
  - It is called a False Negative
  - Accepting a false null hypothesis H0 is referred to as the **Type II** error

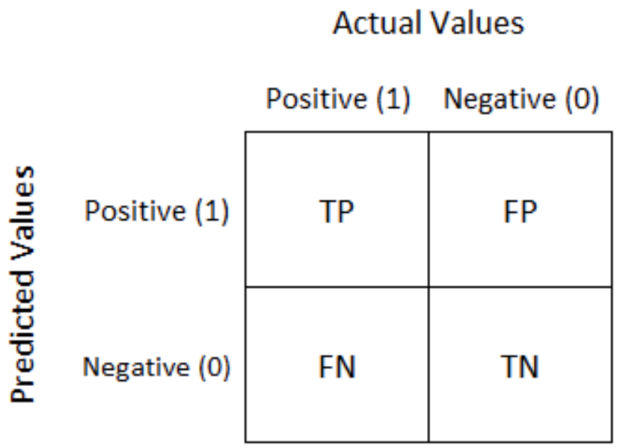
- **Power of the test**
  - The power of the test is the probability of Type II error
  - The power of the test is represented by the symbol β (Yellow area).

Hypothesis Testing in R





The results obtained from positive sample (left curve) overlap with the results obtained from negative samples (right curve). By moving the result cutoff value (vertical bar), the rate of false positives (FP) can be decreased, at the cost of raising the number of false negatives (FN), or vice-versa.



True Positive:

- Interpretation: You predicted positive and it's true.
- You predicted that the person is a worth client, and they actually are.

True Negative:

- Interpretation: You predicted negative and it's true.
- You predicted that the person is not a worth client, and they're actually are not.

False Positive: (Type 1 Error)

- Interpretation: You predicted positive and it's false.
- You predicted that the person is a worth client, but they actually are not.

False Negative: (Type 2 Error)

- Interpretation: You predicted negative and it's false.
- You predicted that the person is not a worth client, but they actually are

https://towardsdatascience.com/metrics-for-evaluating-machine-learning-classification-models-python-example-59b905e079a5

Metrics for Evaluation Classification Models

$$precision = \frac{TP}{TP + FP}$$

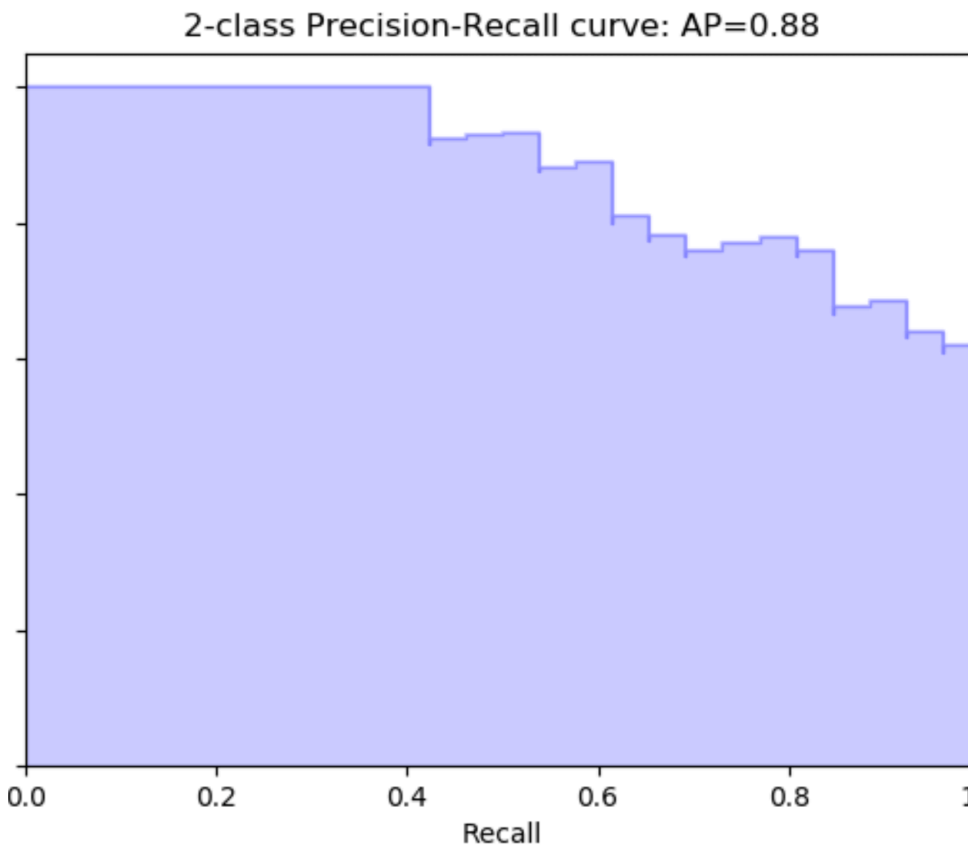
$$recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/

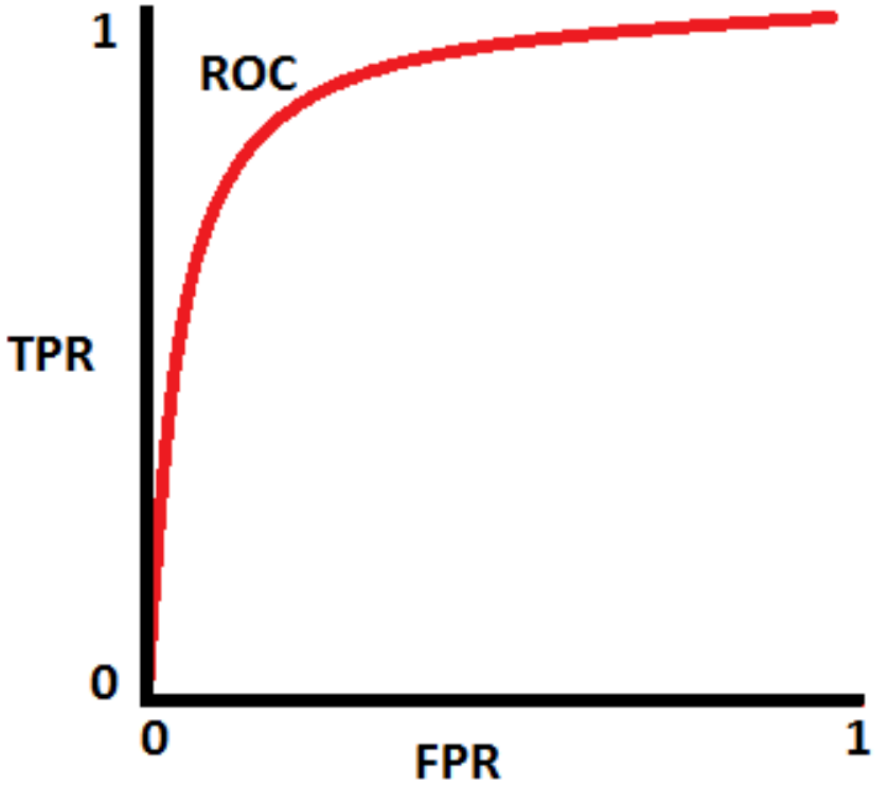
Metrics for Evaluation Classification Models: Precision-Recall curve



- Trade-off between having a high precision or a high recall
- In certain circumstances, it's better to have a high precision. For example, a diagnosis might be better off with a few false positives rather than let anyone with the actual disease go away and avoid getting treated.
- Other times, it's better to have a higher recall as is the case in spam filters. It's more acceptable to have a few spam emails in the user's inbox than it is to classify important emails as junk.
- We can represent the tradeoff between precision and recall graphically in order to form a better judgement.

https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/

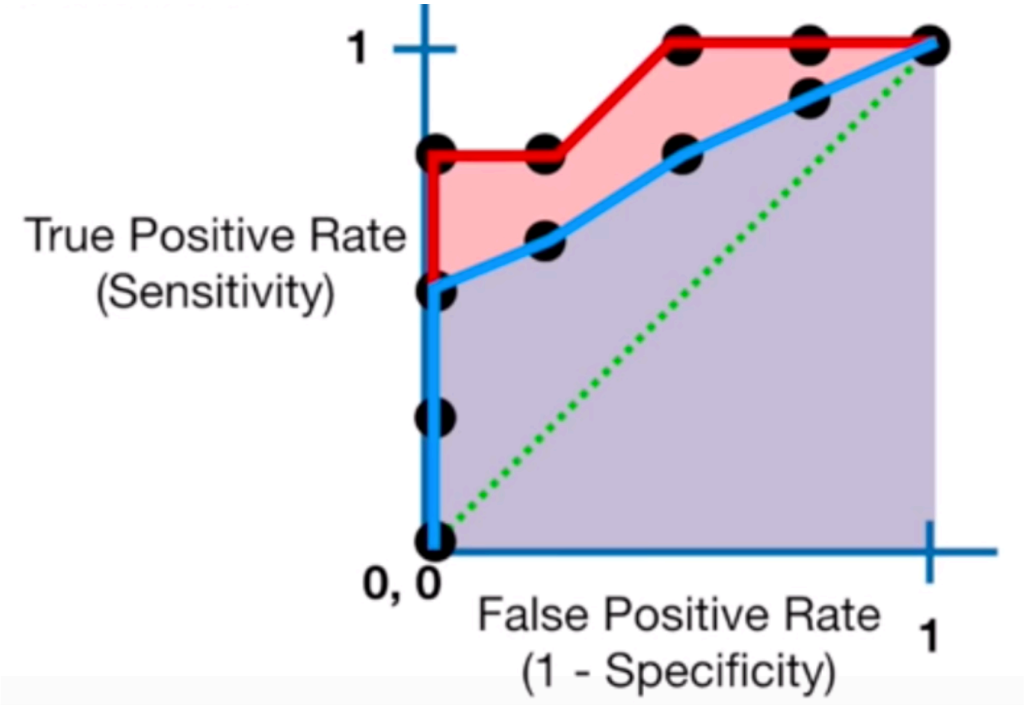
Metrics for Evaluation Classification Models: Receiver Operator Characteristic Curve (ROC)



- Receiver Operator Characteristic (ROC) graph provides an elegant way of presenting multiple confusion matrices produced at different thresholds. A ROC plots the relationship between the true positive rate and the false positive rate.
- True positive rate = Recall = Sensitivity = true positive / (true positive + false negative)
- False positive rate = 1 – specificity = false positive / (false positive + true negative)

https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/

Metrics for Evaluation Classification Models: Area Under the Curve Curve (AUC)



- Area Under the Curve (AUC) makes it easy to compare one ROC curve to another.
- For example, the AUC for the red ROC curve is greater than the AUC for the blue ROC curve.
- Therefore, the model associated with the red curve achieves a higher sensitivity for the same amount of specificity.)

https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/

Hands-on Exercise with Classification Metrics using Python

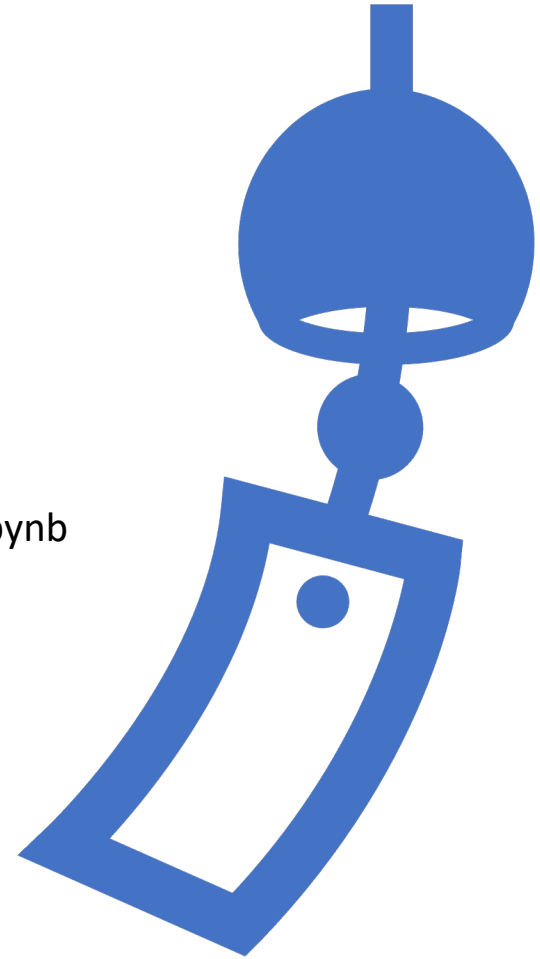
Download script 4-class-metrics.py

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

Unbalanced Datasets

- Records Of Datasets Classes Are Unbalanced
- Unbalanced Datasets Affects Model Prediction
- Resampling
- Undersampling
- Oversampling
- Hands-on Exercise In Python

Hands-on Exercise with Unbalanced Classes using JNP



Download notebook:

5-resampling-strategies-for-imbalanced-datase-48c2b9.ipynb

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

Model Evaluation

Statistical Methods: A Review

Hypothesis Testing

ANOVA

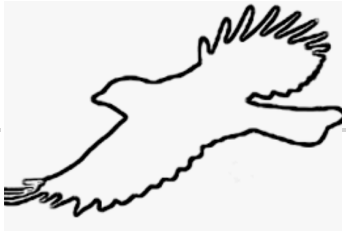
Correlation

Clustering

Principal
Component Analysis

Regression Analysis

Machine Learning Methods: A Bird-eye Overview



DECISION TREES,

BAGGING, RANDOM FOREST, BOOSTING

LOGISTIC REGRESSION

SUPPORT VECTOR MACHINES

K-NEAREST NEIGHBORS

BAYESIAN METHODS

NEURAL NETS

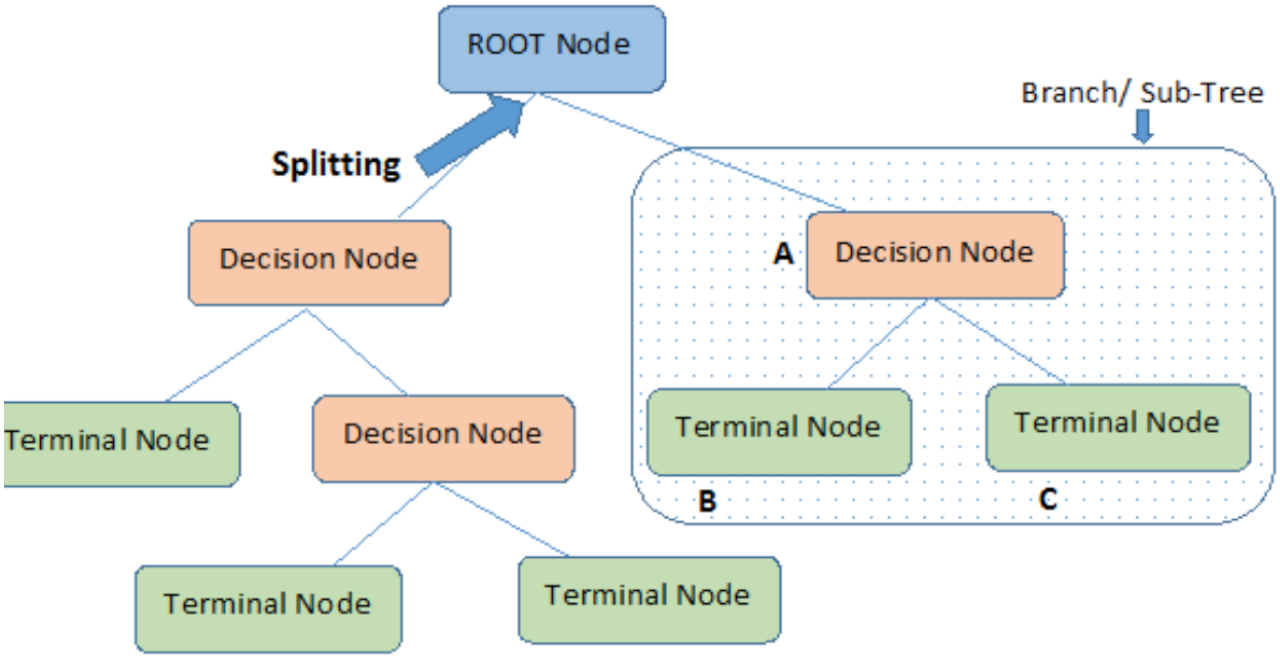
Decision Trees

Decision tree is a type of supervised learning algorithm that can be used in both regression and classification problems (CART)

It works for both categorical and continuous input and output variables

Growing the Tree:

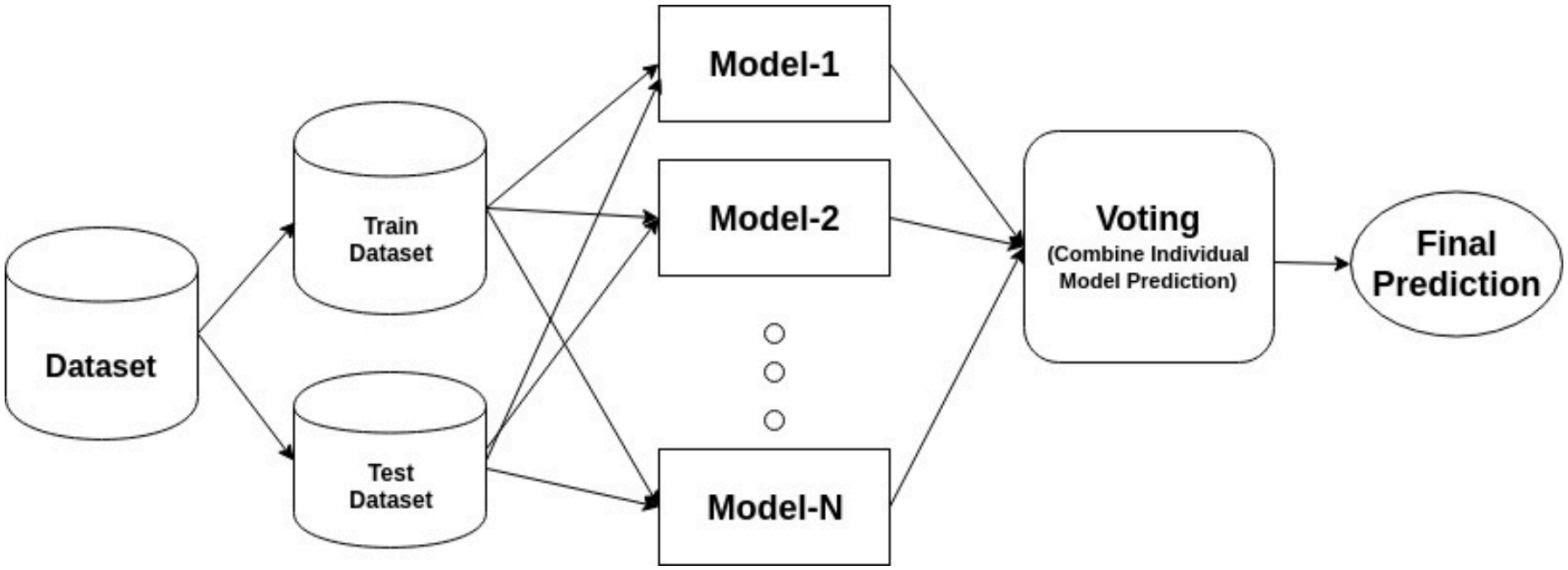
- Classification Error Rate
- Gini Index
- Cross-Entropy

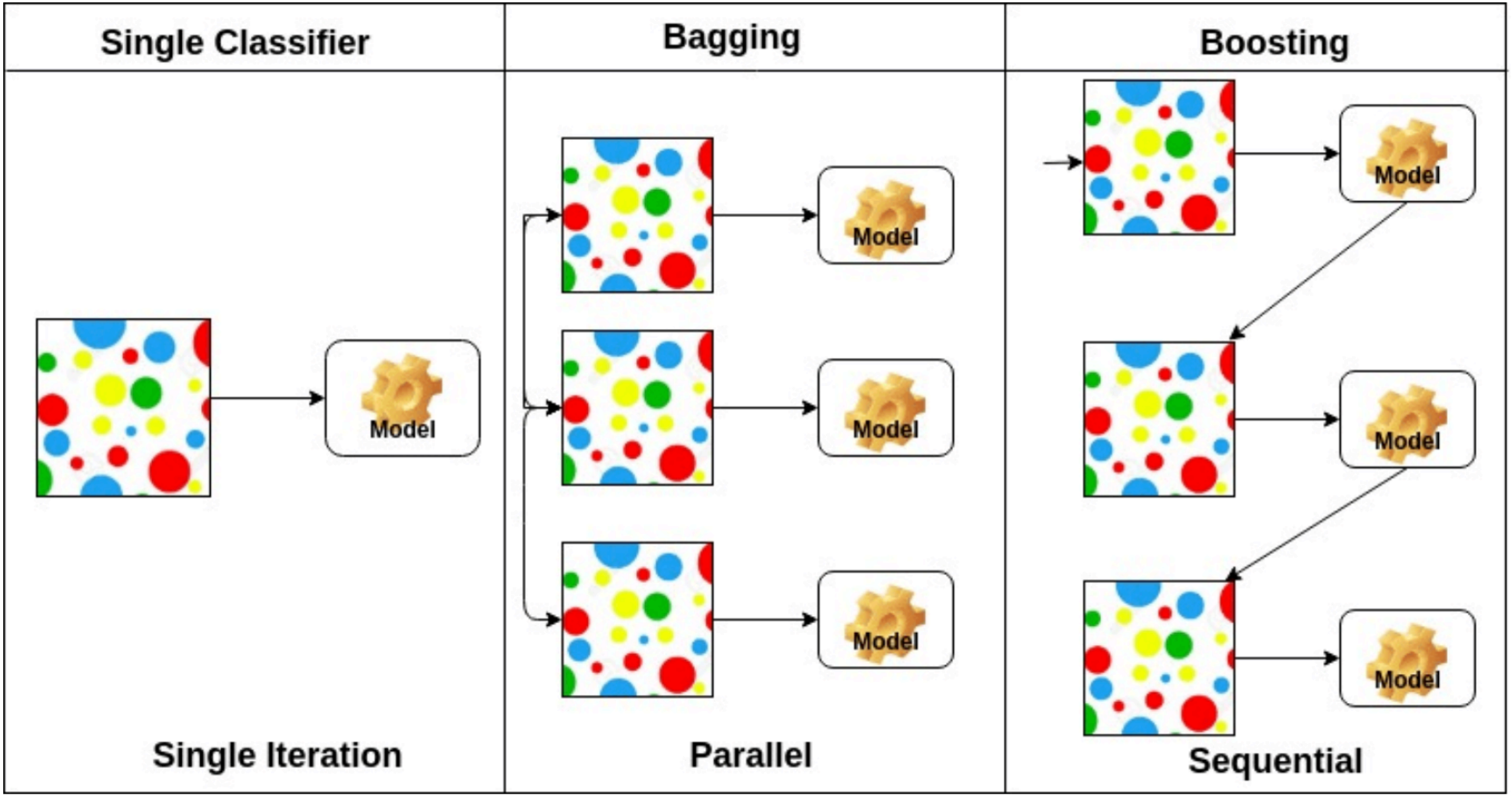


https://www.datacamp.com/community/tutorials/decision-trees-R

Ensemble Method

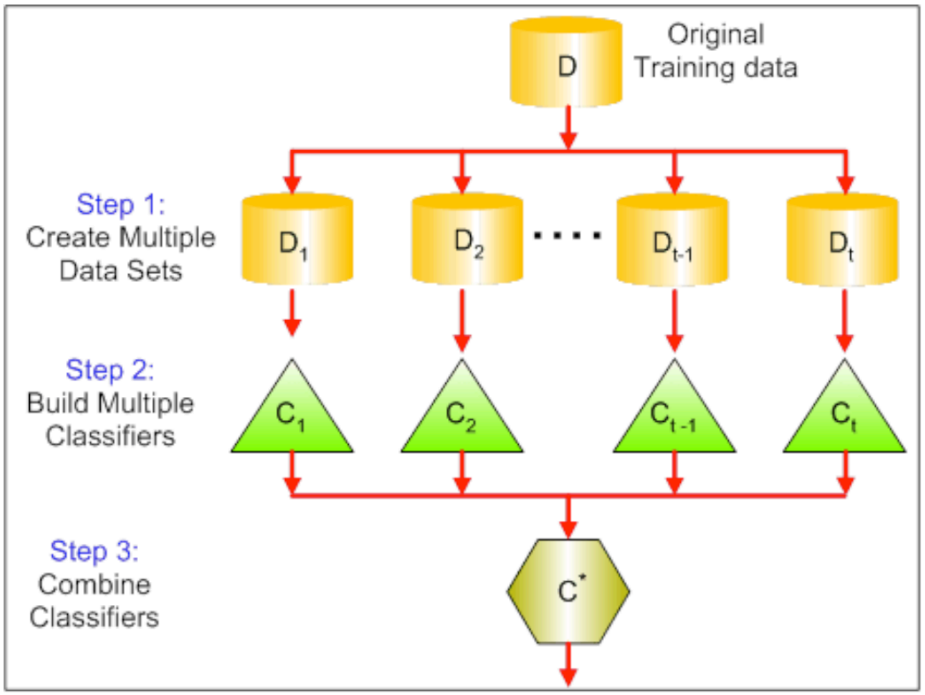
- An ensemble is a composite model that combines a series of low performing classifiers with the aim of creating an improved classifier
- Individual classifier vote and final prediction label returned that performs majority voting.
- Ensembles offer more accuracy than individual or base classifier.
- Ensemble methods can parallelize by allocating each base learner to different-different machines.
- Ensemble methods are meta-algorithms that combine several machine learning methods into a single predictive model to increase performance.
- Ensemble methods can decrease variance using bagging approach, bias using a boosting approach, or improve predictions using stacking approach.




Single Classifier
Bagging
Boosting
Model
Model
Model
Model
Model
Model
Model
Single Iteration
Parallel
Sequential

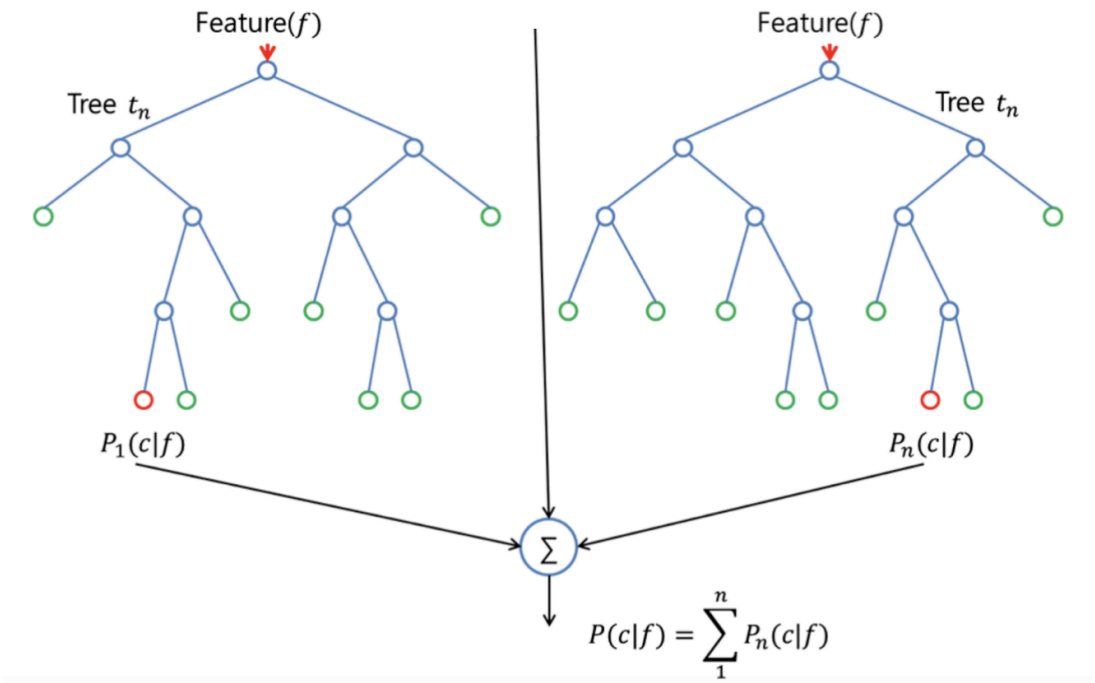
Bagging

- Bagging, or bootstrap aggregation, is a technique used to reduce the variance of your predictions by combining the result of multiple classifiers modeled on different sub-samples of the same dataset



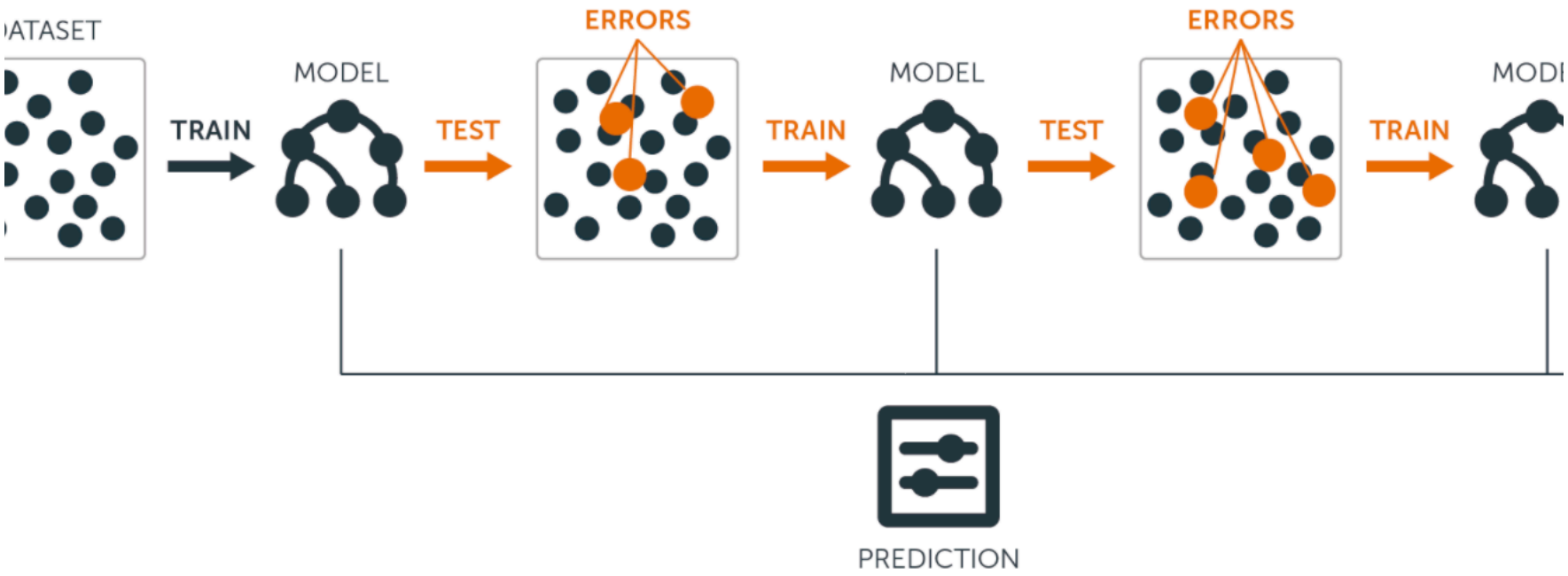
Random Forest

- Random Forests performs both regression and classification tasks
- Provides an improvement over bagged trees by a small tweak that decorrelates the trees.
- A random sample of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors



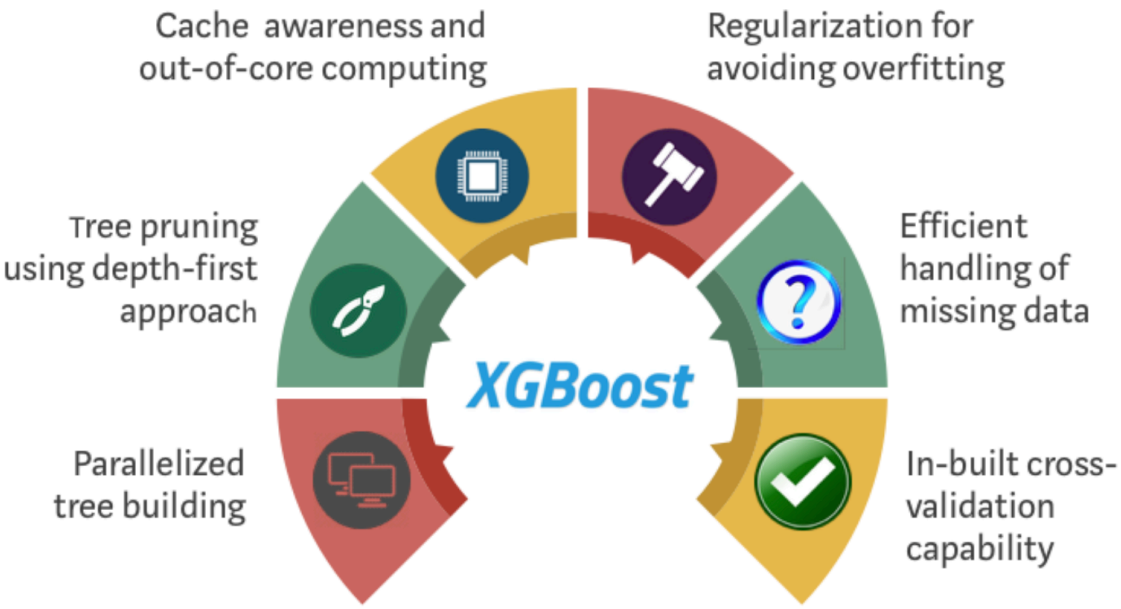
Boosting

- Another approach to improve the predictions resulting from a decision tree for regression or classification
- Boosting works in a similar way to Bagging and Random Forest, except that the trees are grown sequentially: each tree is grown using information from previously grown trees. Boosting does not involve bootstrap sampling; instead, each tree is fitted on a modified version of the original dataset.

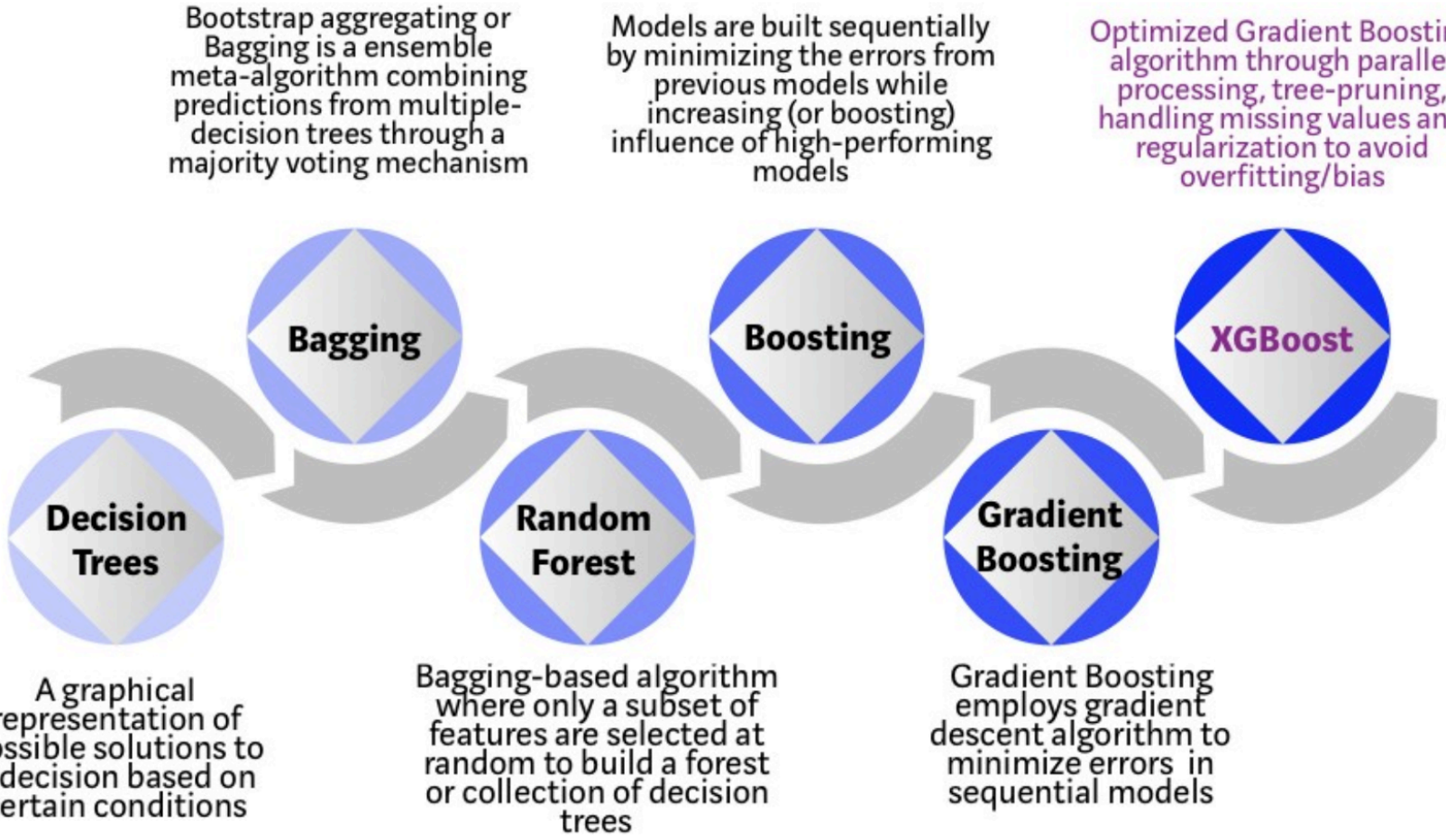


Why does XGBoost perform so well?

XGBoost and Gradient Boosting Machines (GBMs) are both ensemble tree methods that apply the principle of boosting weak learners (CARTs generally) using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.



How XGBoost optimizes standard GBM algorithm


Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism
Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models
Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias
Bagging
Boosting
XGBoost
Decision Trees
Random Forest
Gradient Boosting
A graphical representation of possible solutions to a decision based on certain conditions
Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees
Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

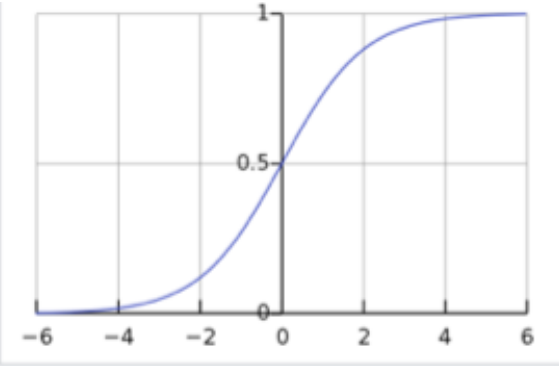
Logistic Regression

$$f(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

- The Logistic Model (or logit model) is used to model the probability of a binary class or event such as pass/fail, win/lose, alive/dead or healthy/sick

- Can model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1 and the sum adding to one

- It uses a **logistic function** to model a binary dependent variable, although many more complex extensions exist



Standard logistic sigmoid function i.e. $L = 1, k = 1, x_0 = 0$

A **logistic function** or **logistic curve** is a common "S" shape (sigmoid curve), with equation:

$$f(x) = \frac{L}{1 + e^{-k(x - x_0)}}$$

where

- e = the natural logarithm base (also known as Euler's number),
- x_0 = the x-value of the sigmoid's midpoint,
- L = the curve's maximum value, and
- k = the logistic growth rate or steepness of the curve.[1]

For values of x in the domain of real numbers from $-\infty$ to $+\infty$, the S-curve shown on the right is obtained, with the graph of f approaching L as x approaches $+\infty$ and approaching zero as x approaches $-\infty$.

https://en.wikipedia.org/wiki/Logistic_regression

Logistic Regression

- Consider a model with two predictors, X_1 and X_2 and one binary (Bernoulli) response variable Y, which we denote p = P(Y = 1)
- We assume a linear relationship between the predictor variables and the log-odds of the event that Y=1
- This linear relationship can be written in the following mathematical form where ℓ is the log-odds, b is the base of the logarithm, and betas the coefficient parameters of the model:

$$\ell = \log_b \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

We can recover the odds by exponentiating the log-odds:

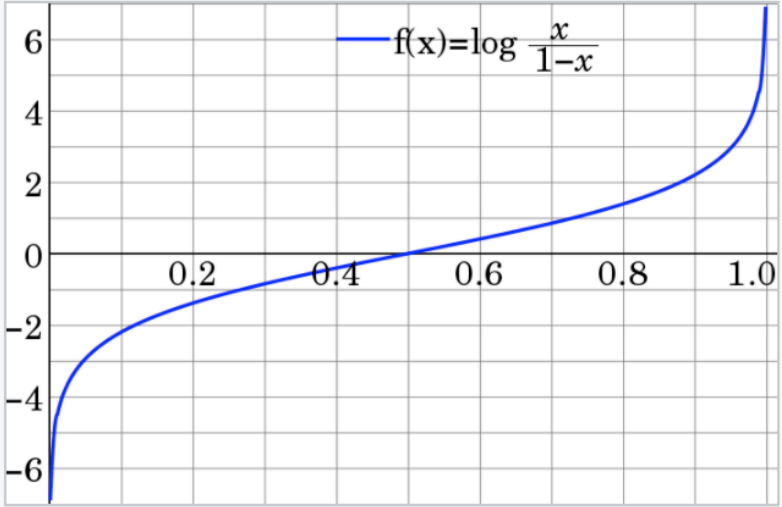
$$\frac{p}{1-p} = b^{\beta_0+\beta_1 x_1+\beta_2 x_2}.$$

By simple algebraic manipulation, the probability that $Y = 1$ is

$$p = \frac{b^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{b^{\beta_0+\beta_1 x_1+\beta_2 x_2}+1} = \frac{1}{1+b^{-(\beta_0+\beta_1 x_1+\beta_2 x_2)}}.$$

Logit Function

- The Logit function or the log-odds is the logarithm of the odds p/(1-p) where p is a probability value
- It is a type of function that creates a map of probability values from (0,1) to (-inf, +inf)
- It is the inverse of the sigmoidal "logistic" function or logistic transform



Plot of logit(*p*) in the domain of 0 to 1, where the base of logarithm is *e*

$$\operatorname{logit}(p) = \log\left(\frac{p}{1-p}\right) = \log(p) - \log(1-p) = -\log\left(\frac{1}{p} - 1\right)$$

Example

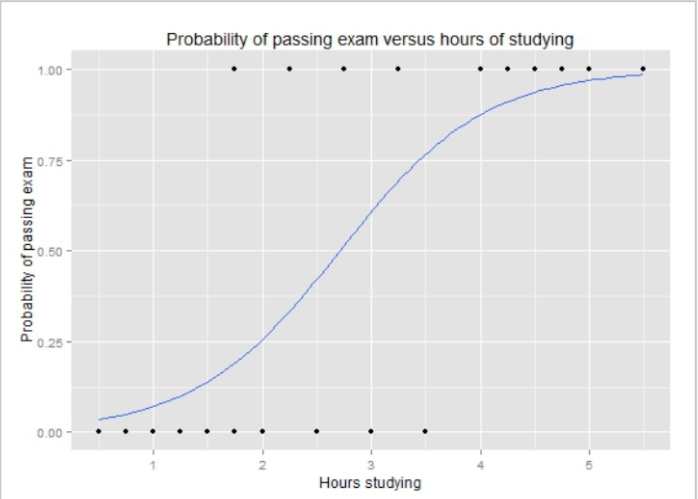
A group of 20 students spends between 0 and 6 hours studying for an exam. How does the number of hours spent studying affect the probability of the student passing the exam?

The reason for using logistic regression for this problem is that the values of the dependent variable, pass and fail, while represented by "1" and "0", are not cardinal numbers. If the problem was changed so that pass/fail was replaced with the grade 0–100 (cardinal numbers), then simple regression analysis could be used.

The table shows the number of hours each student spent studying, and whether they passed (1) or failed (0).

Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Pass	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1

The graph shows the probability of passing the exam versus the number of hours studying, with the logistic regression curve fitted to the data.



Graph of a logistic regression curve showing probability of passing an exam versus hours studying

Example

	Coefficient	Std.Error	z-value	P-value (Wald)
Intercept	–4.0777	1.7610	–2.316	0.0206
Hours	1.5046	0.6287	2.393	0.0167

The output indicates that hours studying is significantly associated with the probability of passing the exam ($p = 0.0167$, Wald test). The output also provides the coefficients for $\text{Intercept} = -4.0777$ and $\text{Hours} = 1.5046$. These coefficients are entered in the logistic regression equation to estimate the odds (probability) of passing the exam:

Hands-on Exercise with Logistic Regression using Python

Download the script
6-logit.py

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

Support Vector Machine

- Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

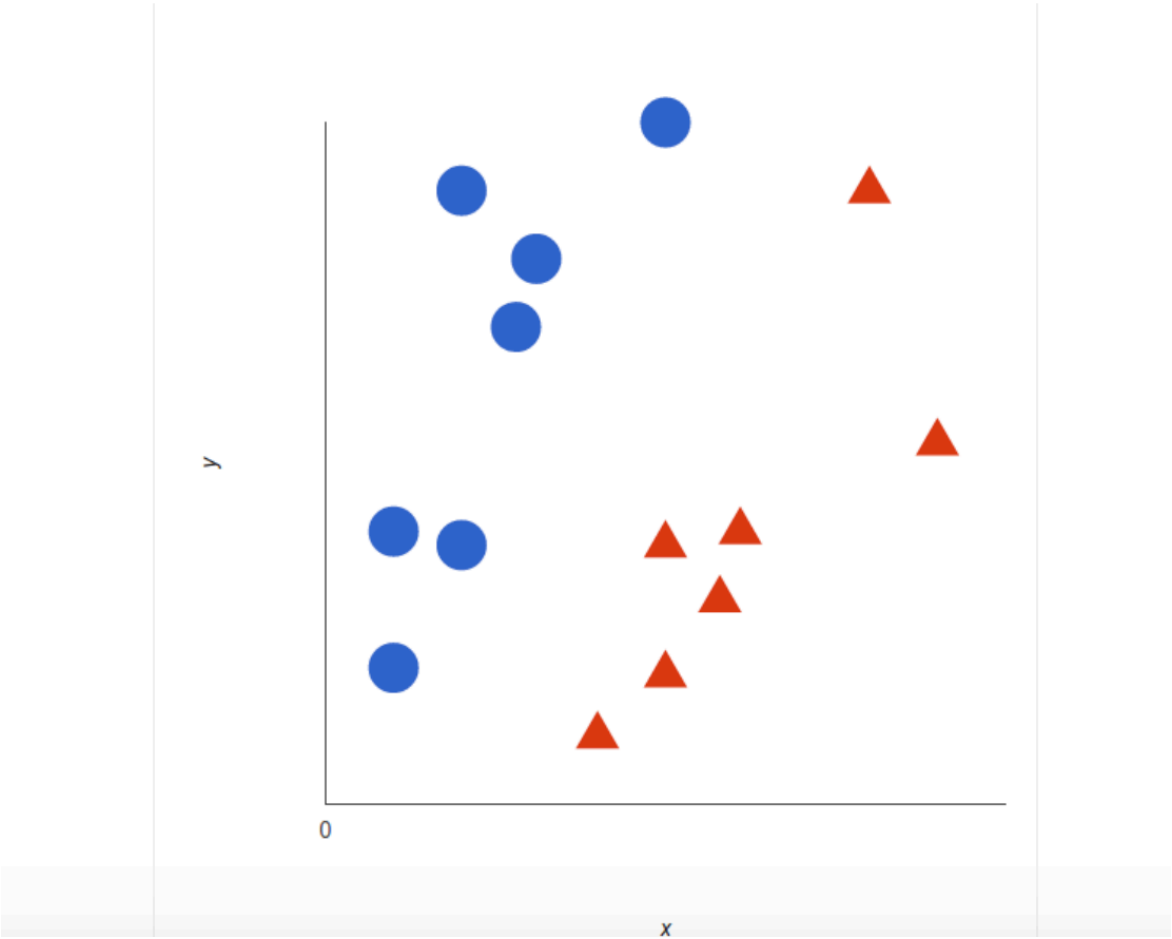
- The advantages of support vector machines are:
  - Effective in high dimensional spaces.
  - Still effective in cases where number of dimensions is greater than the number of samples.
  - Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
  - Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

- If the number of features is much greater than the number of samples, avoiding over-fitting in choosing Kernel functions and regularization term is crucial.

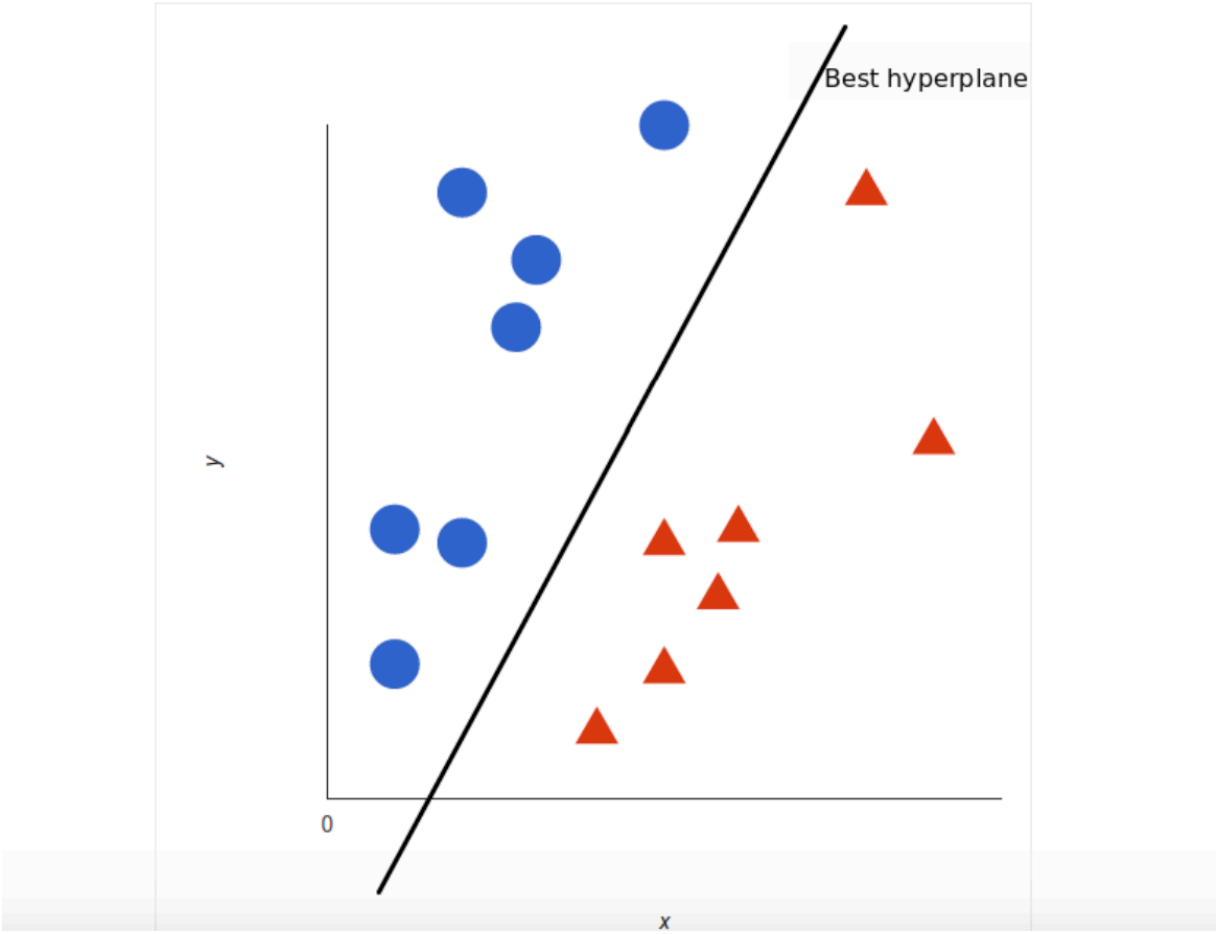
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/

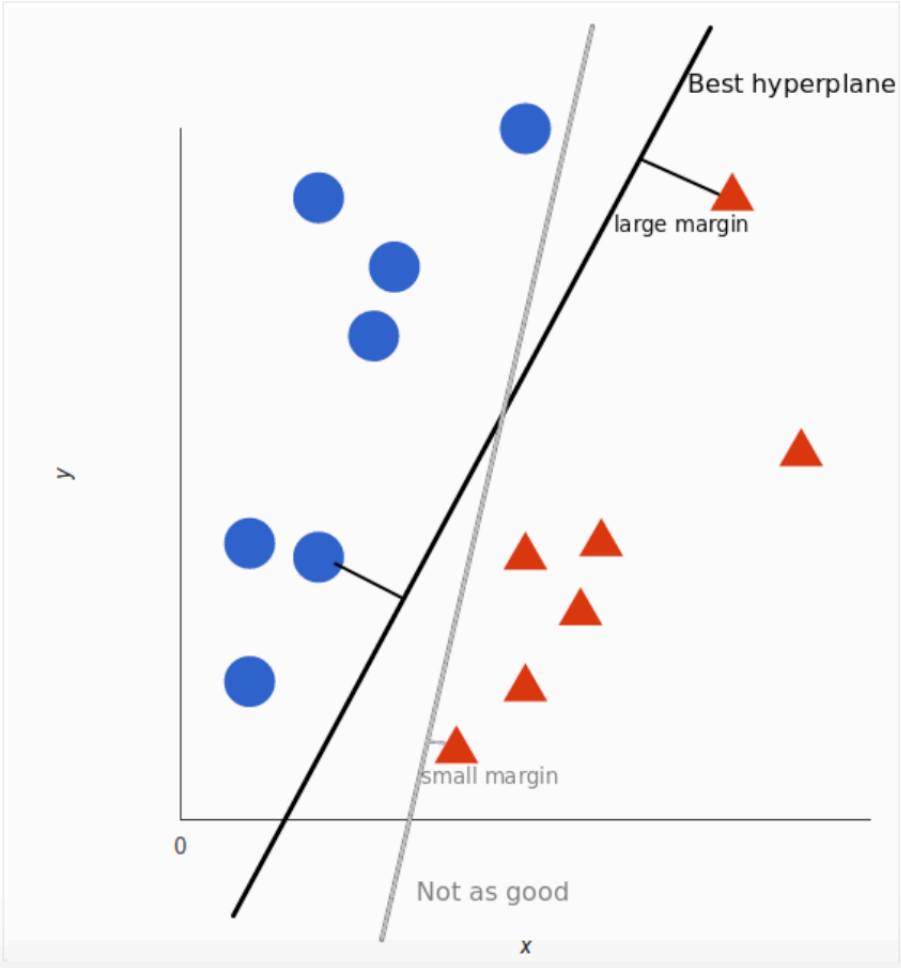
The basics of Support Vector Machines and how it works are best understood with a simple example. Let's imagine we have two tags: *red* and *blue*, and our data has two features: *x* and *y*. We want a classifier that, given a pair of *(x,y)* coordinates, outputs if it's either *red* or *blue*. We plot our already labeled training data on a plane:



A support vector machine takes these data points and outputs the hyperplane (which in two dimensions it's simply a line) that best separates the tags. This line is the **decision boundary**: anything that falls to one side of it we will classify as *blue*, and anything that falls to the other as *red*.

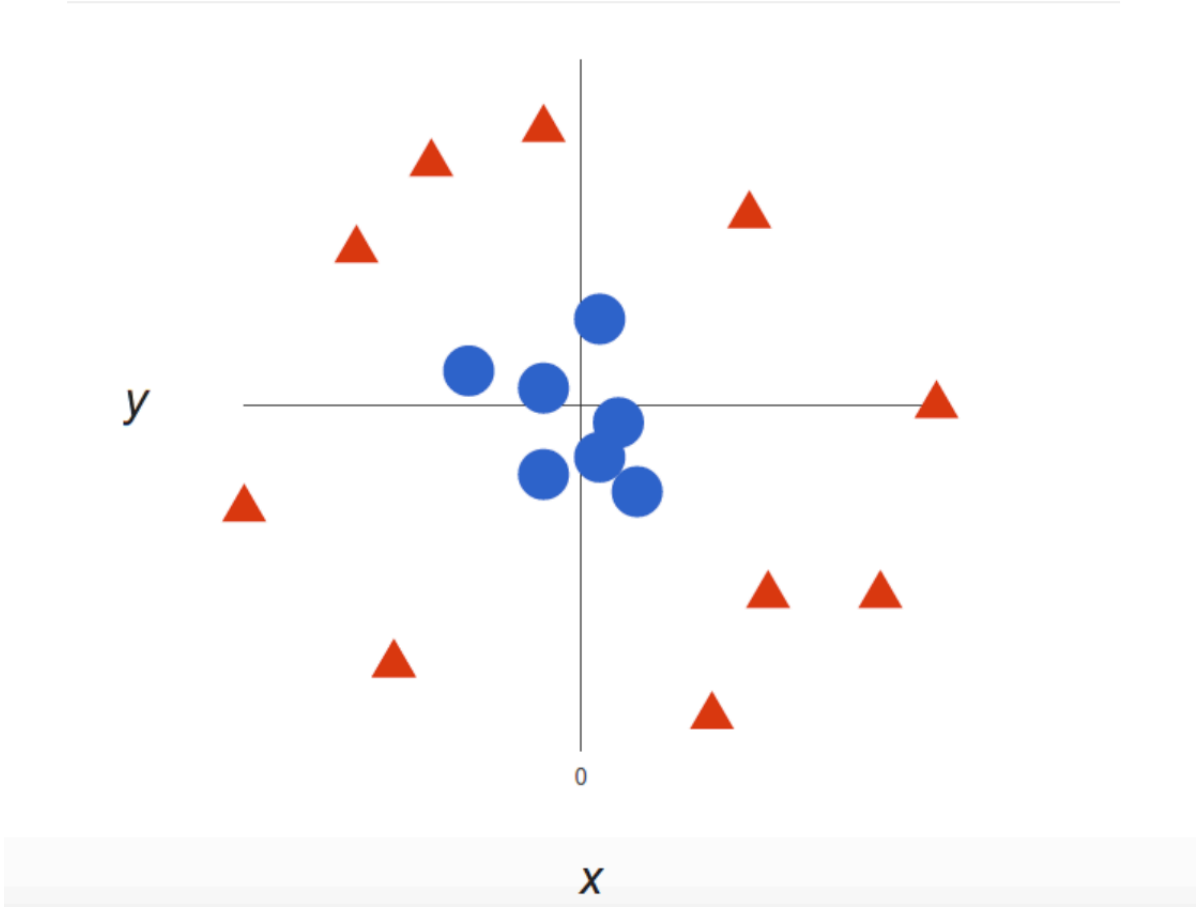


But, what exactly is *the best* hyperplane? For SVM, it's the one that maximizes the margins from both tags. In other words: the hyperplane (remember it's a line in this case) whose distance to the nearest element of each tag is the largest.



Nonlinear data

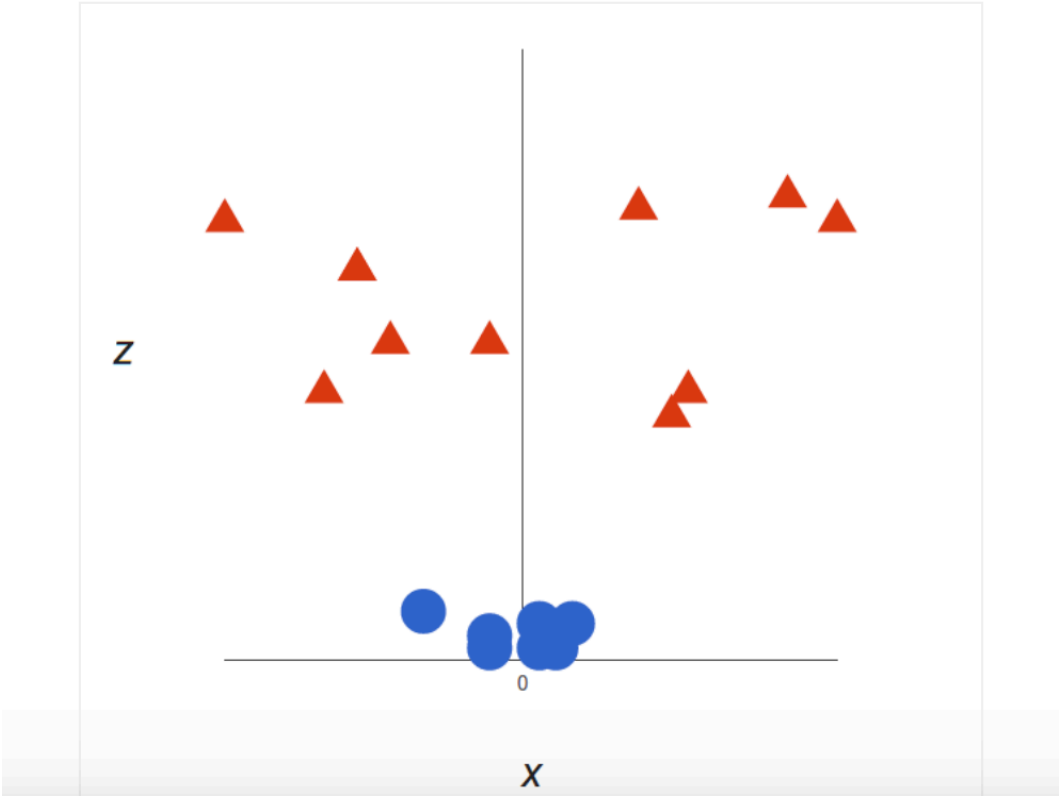
Now this example was easy, since clearly the data was linearly separable — we could draw a straight line to separate *red* and *blue*. Sadly, usually things aren't that simple. Take a look at this case:



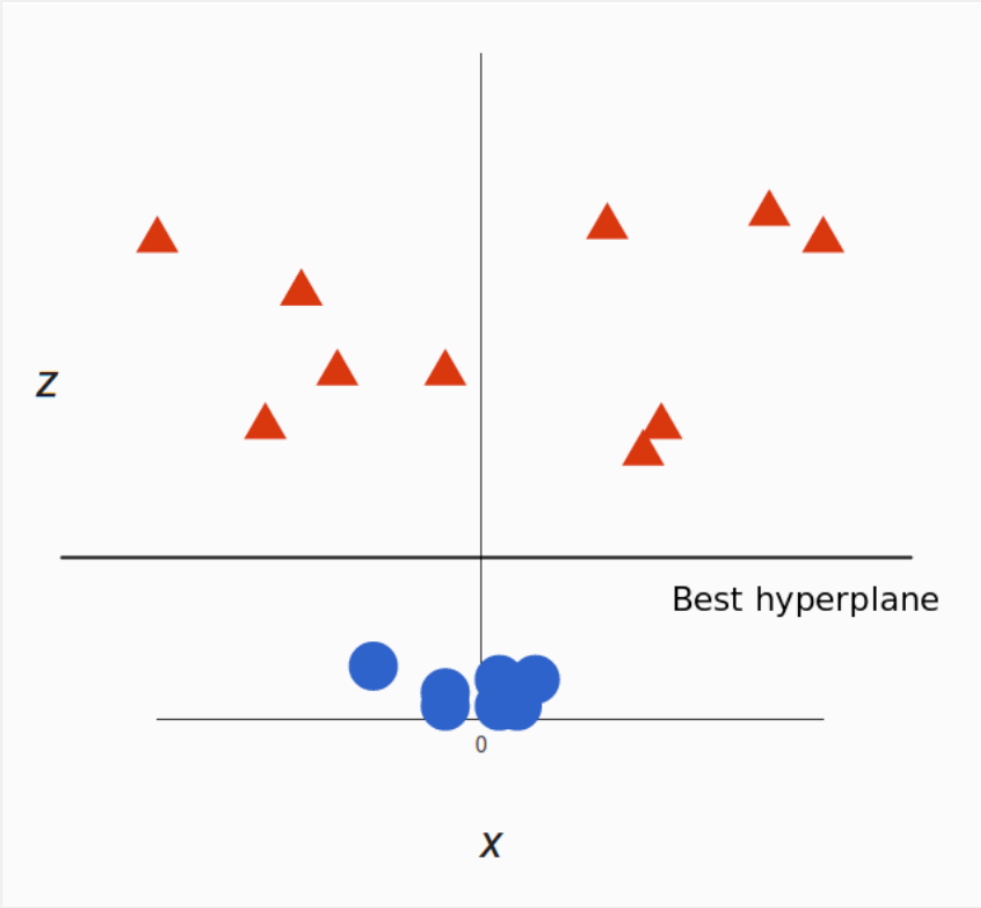
It's pretty clear that there's not a linear decision boundary (a single straight line that separates both tags). However, the vectors are very clearly segregated and it looks as though it should be easy to separate them.

So here's what we'll do: we will add a third dimension. Up until now we had two dimensions: x and y. We create a new z dimension, and we rule that it be calculated a certain way that is convenient for us: $z = x^2 + y^2$ (you'll notice that's the equation for a circle).

This will give us a three-dimensional space. Taking a slice of that space, it looks like this:

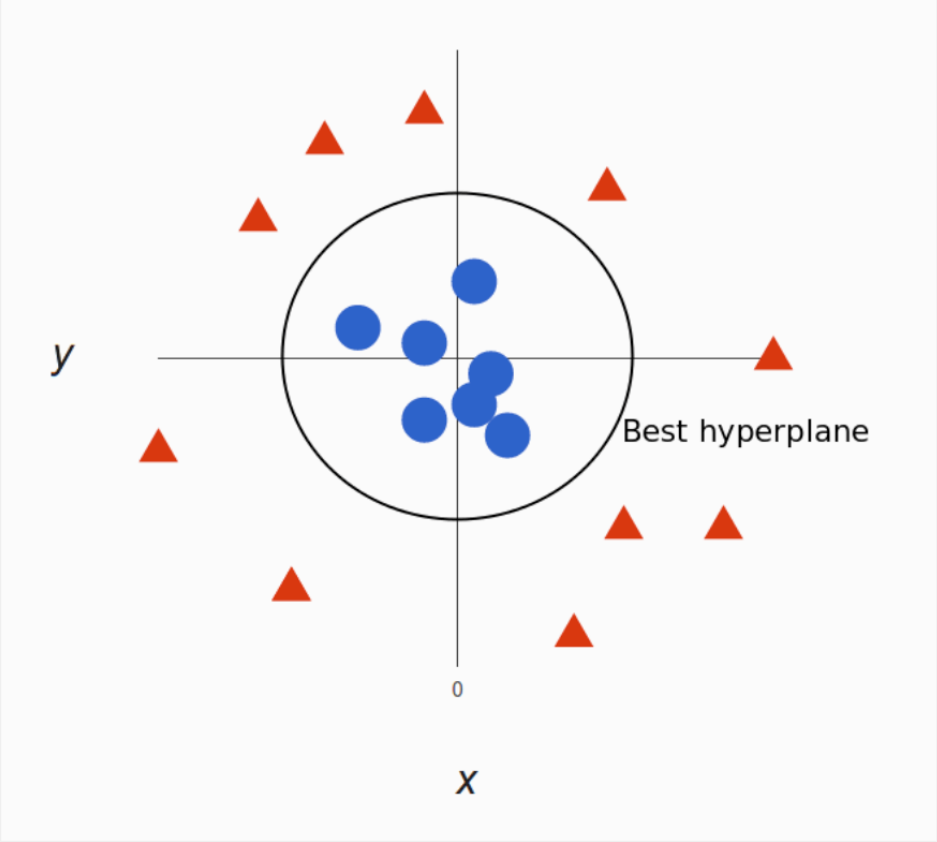


What can SVM do with this? Let's see:



That's great! Note that since we are in three dimensions now, the hyperplane is a plane parallel to the *x* axis at a certain *z* (let's say $z = 1$).

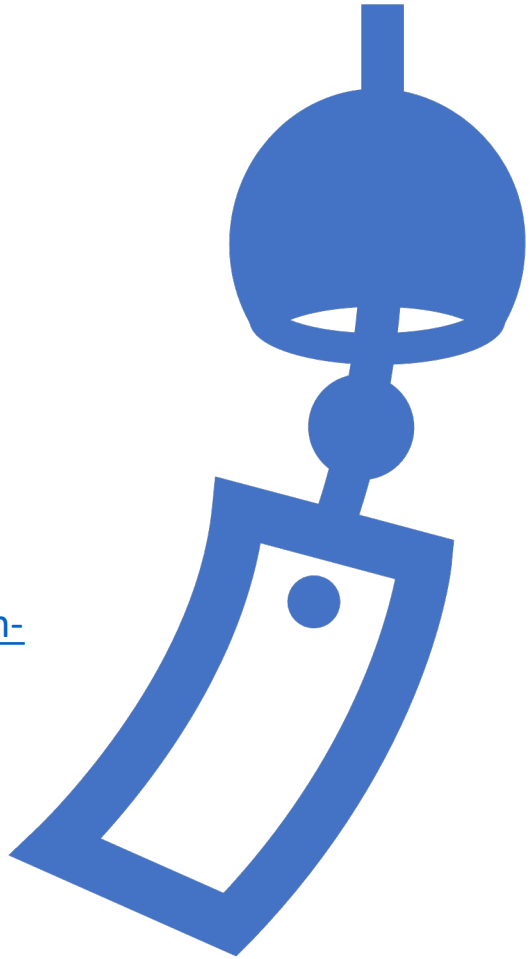
What's left is mapping it back to two dimensions:



Back to our original view, everything is now neatly separated

And there we go! Our decision boundary is a circumference of radius 1, which separates both tags using SVM. Check out this 3d visualization to see another example of the same effect:

Hands-on Exercise with Support Vector Machines using Python



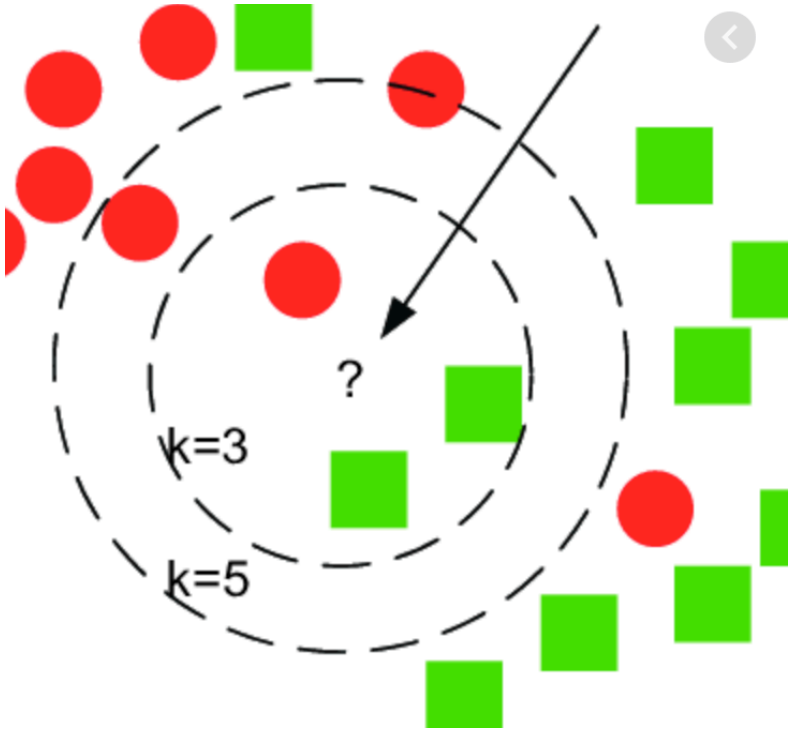
See examples:

1. https://scikit-learn.org/stable/modules/svm.html

2. https://www.cienciadedatos.net/documentos/py24-svm-python.html

K-Nearest Neighbor

- k-nearest neighbor algorithm (k-NN) is a non-parametric method used for classification and regression In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:
  - In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.
  - In k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors
- k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until function evaluation.



https://www.saedsayad.com/k_nearest_neighbors.htm

K-Nearest Neighbor

Algorithm

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

K-Nearest Neighbor

It should also be noted that all three distance measures are only valid for continuous variables. In the instance of categorical variables the Hamming distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.

Hamming Distance

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

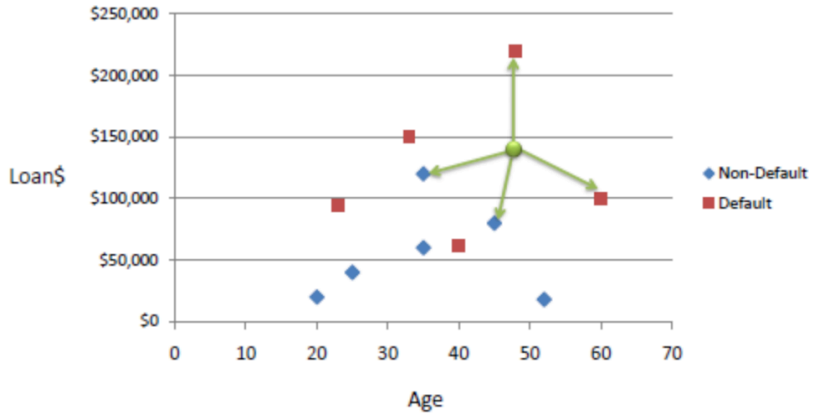
$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

K-Nearest Neighbor

Example:

Consider the following data concerning credit default. Age and Loan are two numerical variables (predictors) and Default is the target.



We can now use the training set to classify an unknown case (Age=48 and Loan=$142,000) using Euclidean distance. If K=1 then the nearest neighbor is the last case in the training set with Default=Y.

D = Sqrt[(48-33)^2 + (142000-150000)^2] = 8000.01 >> Default=Y

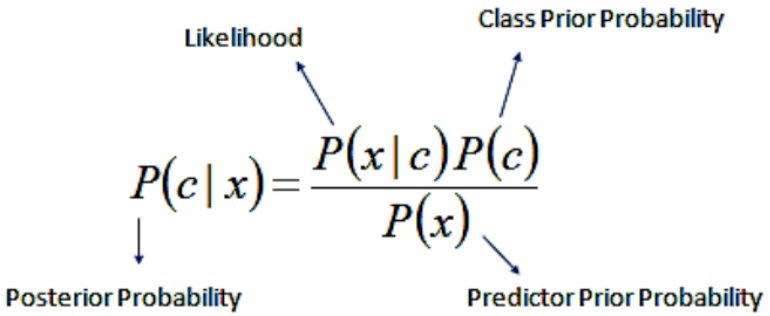
Age	Loan	Default	Distance	
25	$40,000	N	102000	
35	$60,000	N	82000	
45	$80,000	N	62000	
20	$20,000	N	122000	
35	$120,000	N	22000	2
52	$18,000	N	124000	
23	$95,000	Y	47000	
40	$62,000	Y	80000	
60	$100,000	Y	42000	3
48	$220,000	Y	78000	
33	$150,000	Y	8000	1
48	**$142,000**	**?**		

Euclidean Distance $D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$

With K=3, there are two Default=Y and one Default=N out of three closest neighbors. The prediction for the unknown case is again Default=Y.

Naïve Bayes Classifier

The ***Naïve Bayes classifier*** is a simple probabilistic classifier which is based on Bayes theorem but with strong assumptions regarding independence. Historically, this technique became popular with applications in email filtering, spam detection, and document categorization. Although it is often outperformed by other techniques, and despite the naïve design and oversimplified assumptions, this classifier can perform well in many complex real-world problems. And since it is a resource efficient algorithm that is fast and scales well, it is definitely a machine learning algorithm to have in your toolkit.



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Bayes Theorem

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

where A and B are events and $P(B) \neq 0$.

- $P(A \mid B)$ is a conditional probability: the likelihood of event A occurring given that B is true.
- $P(B \mid A)$ is also a conditional probability: the likelihood of event B occurring given that A is true.
- $P(A)$ and $P(B)$ are the probabilities of observing A and B respectively; they are known as the marginal probability.

https://en.wikipedia.org/wiki/Bayes%27_theorem

Hands-on Exercise 1 with Model Validation using Python

Download …

7-churn-modeling-with-crisp-dm.py

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

Hands-on Exercise 2 with Model Validation using Python

Download …

8-churning1.ipynb

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

Hands-on Exercise 3 with Model Validation using Python

Download …

9-churning2.ipynb

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

Hands-on Exercise 4 with Model Validation using Python

Download …

10-churning3.ipynb

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

Francisco J. Cantú-Ortiz, PhD

Professor of Computer Science and Artificial Intelligence
Tecnológico de Monterrey
Enago-Academy Advisor for Strategic Alliances

E-mail: fcantu@tec.mx, fjcantor@gmail.com
Cel: +52 81 1050 8294, SNI-2 CVU: 9804
Personal Page: http://semtech.mty.itesm.mx/fcantu/
Facebook: fcantu; Twitter: @fjcantor; Skype: fjcantor
Orcid: 0000-0002-2015-0562
Scopus ID:6701563520
Researcher ID: B-8457-2009
https://www.researchgate.net/profile/Francisco_Cantu-Ortiz
https://scholar.google.com.mx/citations?hl=es&user=45-uuK4AAAAJ
https://itesm.academia.edu/FranciscoJavierCantuOrtiz
Ave. Eugenio Garza Sada No. 2501, Monterrey N.L., C.P. 64849, México