An in-depth guide to supervised machine learning classification

An exhaustive understanding of classification algorithms in machine learning.

Written by Badreesh Shetty
Published on Jul. 17, 2019

Machine learning is the science (and art) of programming computers so they can learn from data.

> [Machine learning is the] field of study that gives computers the ability to learn without being explicitly programmed. — Arthur Samuel, 1959

A better definition:

after being given examples of spam emails that are flagged by users, and examples of regular non-spam (also called "ham") emails. The examples the system uses to learn are called the training set. In this case, the task (*T*) is to flag spam for new emails, the experience (*E*) is the training data, and the performance measure (*P*) needs to be defined. For example, you can use the ratio of correctly classified emails as P. This particular performance measure is called accuracy and it is often used in classification tasks as it is a supervised learning approach.

DIVE DEEPER
An Introduction to Machine Learning for Beginners

Supervised Learning

In supervised learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data by associating patterns to the unlabeled new data.

Supervised learning can be divided into two categories: classification and regression.

CLASSIFICATION PREDICTS THE CATEGORY THE DATA BELONGS TO.

Some examples of classification include spam detection, churn prediction, sentiment analysis, dog breed detection and so on.

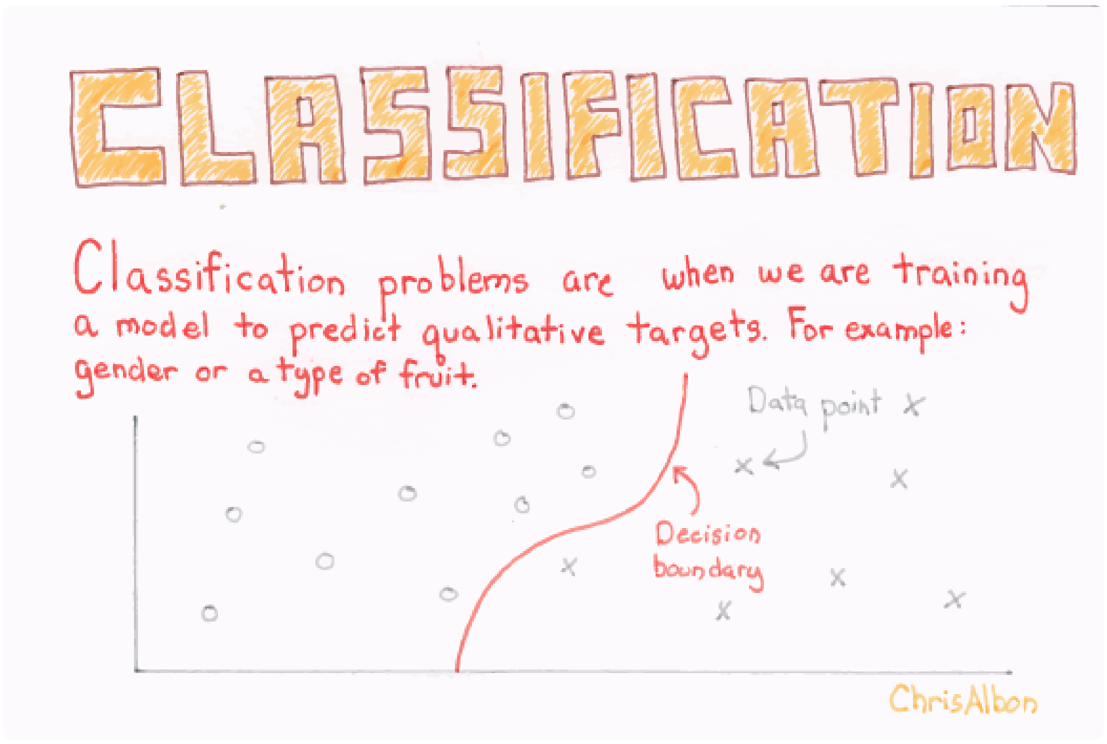
REGRESSION PREDICTS A NUMERICAL VALUE BASED ON PREVIOUSLY OBSERVED DATA.

Some examples of regression include house price prediction, stock price prediction, height-weight prediction and so on.

DIVE DEEPER
A Tour of the Top 10 Algorithms for Machine Learning Newbies

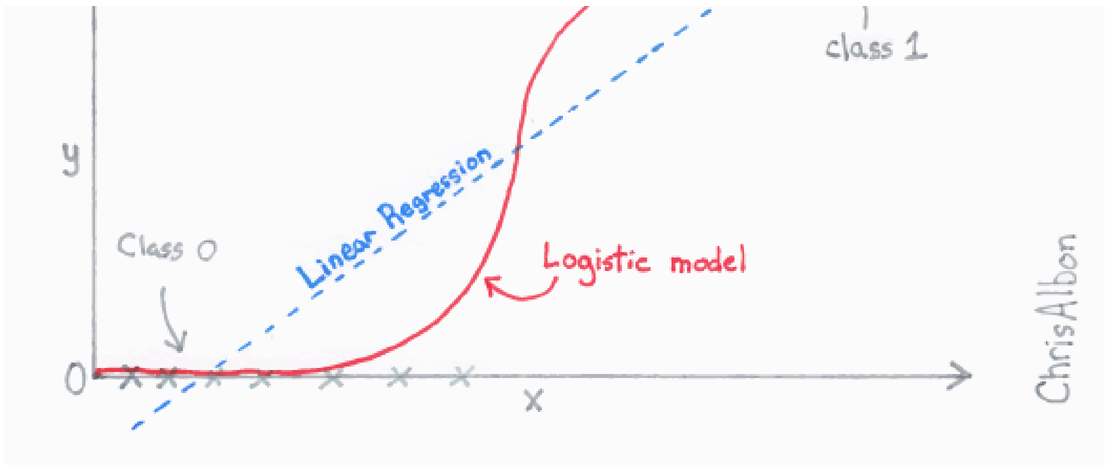
based on one or more independent variables.

> Classification is used for predicting discrete responses.


CLASSIFICATION
Classification problems are when we are training a model to predict qualitative targets. For example: gender or a type of fruit.
Data point
Decision boundary
ChrisAlbon

1. LOGISTIC REGRESSION

Logistic regression is kind of like linear regression, but is used when the dependent variable is not a number but something else (e.g., a "yes/no" response). It's called regression but performs classification based on the regression and it classifies the dependent variable into either of the classes.



Logistic regression is used for prediction of output which is binary, as stated above. For example, if a credit card company builds a model to decide whether or not to issue a credit card to a customer, it will model for whether the customer is going to "default" or "not default" on their card.

$$y = b_0 + b_1 x$$

Linear Regression

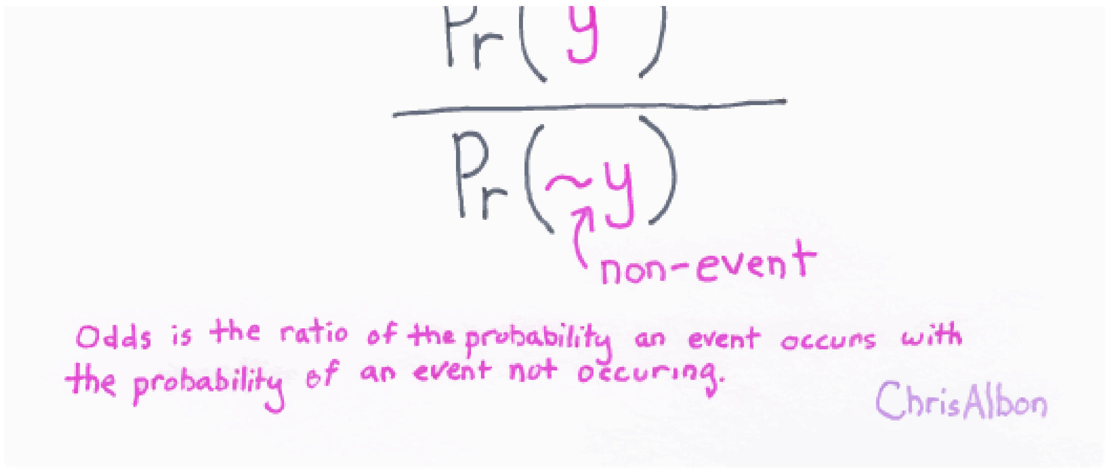
Firstly, linear regression is performed on the relationship between variables to get the model. The threshold for the classification line is assumed to be at 0.5.

$$p = \frac{1}{1 + e^{-y}}$$

Logistic Sigmoid Function

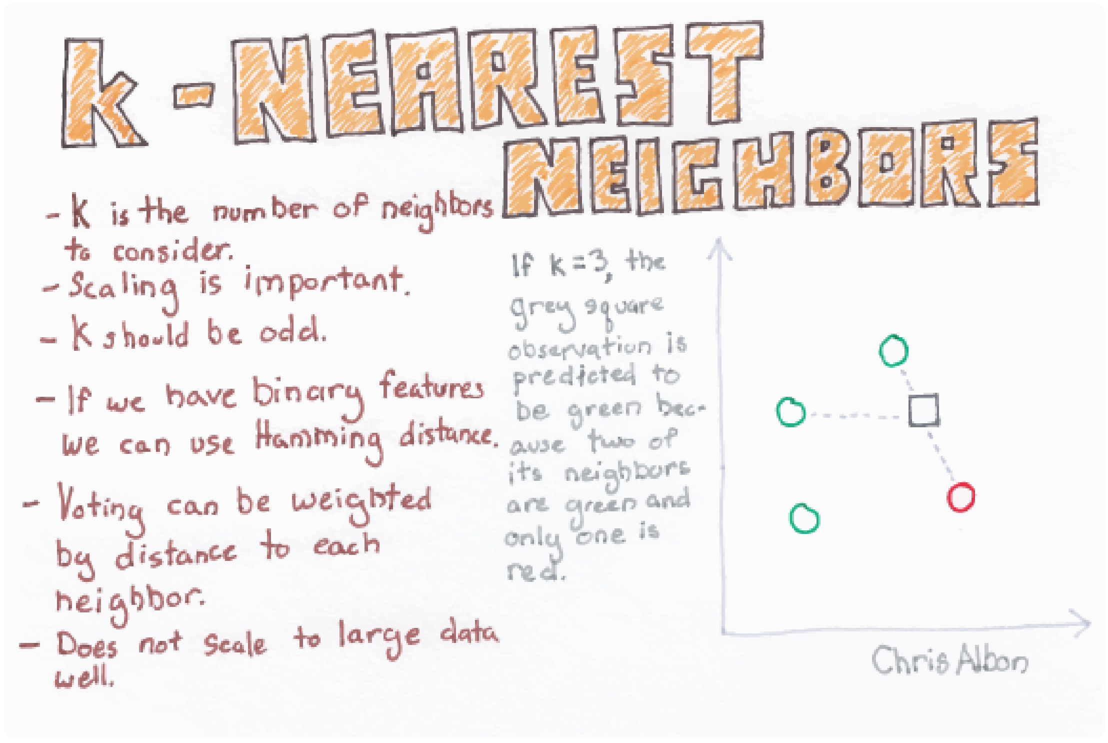
Logistic function is applied to the regression to get the probabilities of it belonging in either class.

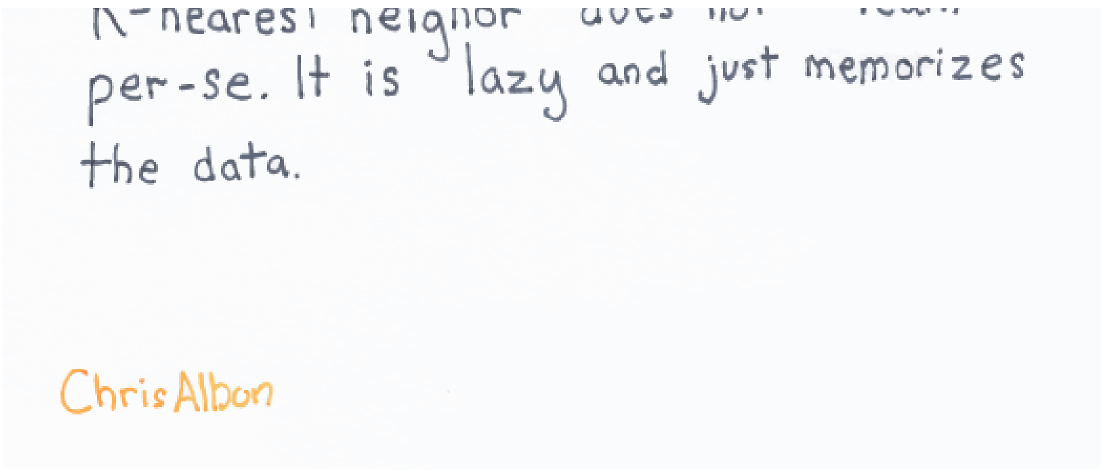
It gives the log of the probability of the event occurring to the log of the probability of it not occurring. In the end, it classifies the variable based on the higher probability of either class.

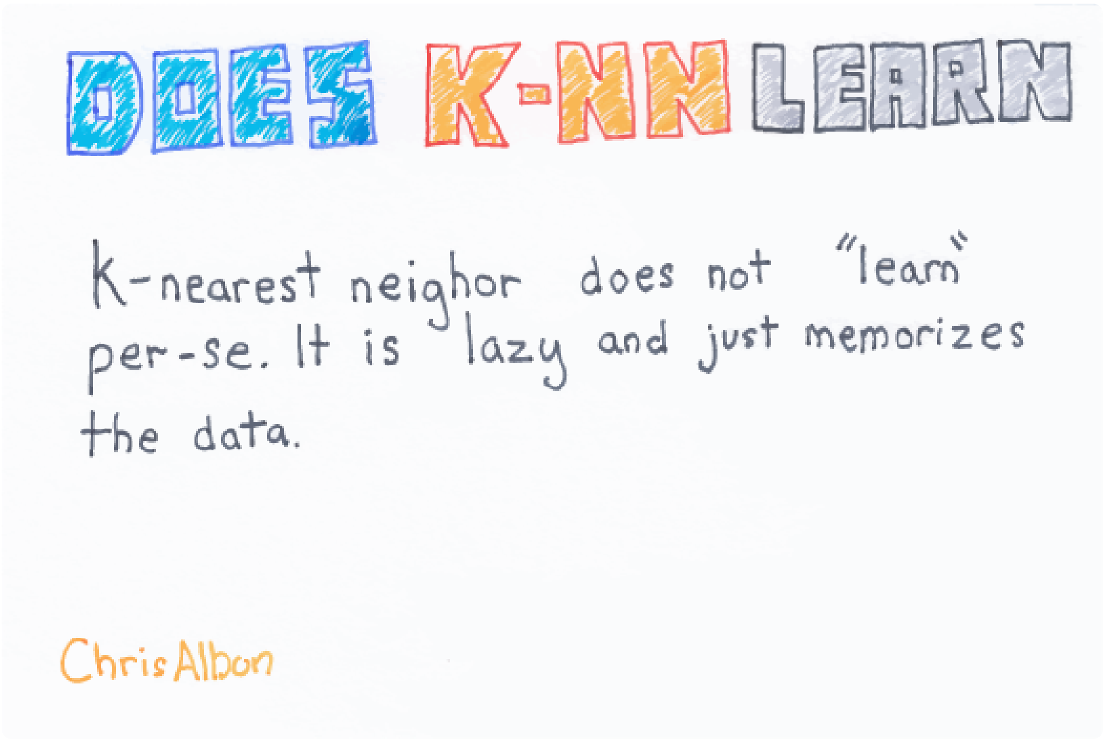

Pr(y)
Pr(~y)
non-event
Odds is the ratio of the probability an event occurs with the probability of an event not occuring.
ChrisAlbon

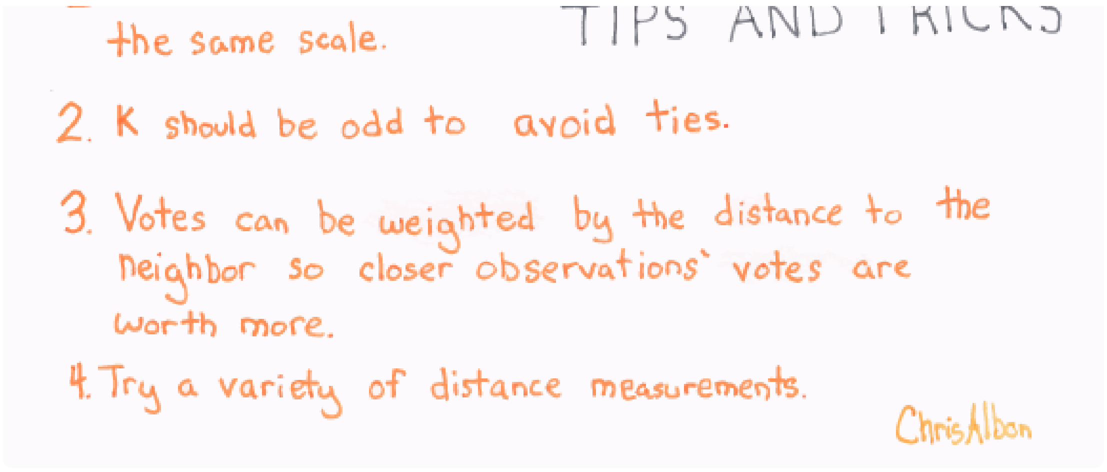
2. K-NEAREST NEIGHBORS (K-NN)

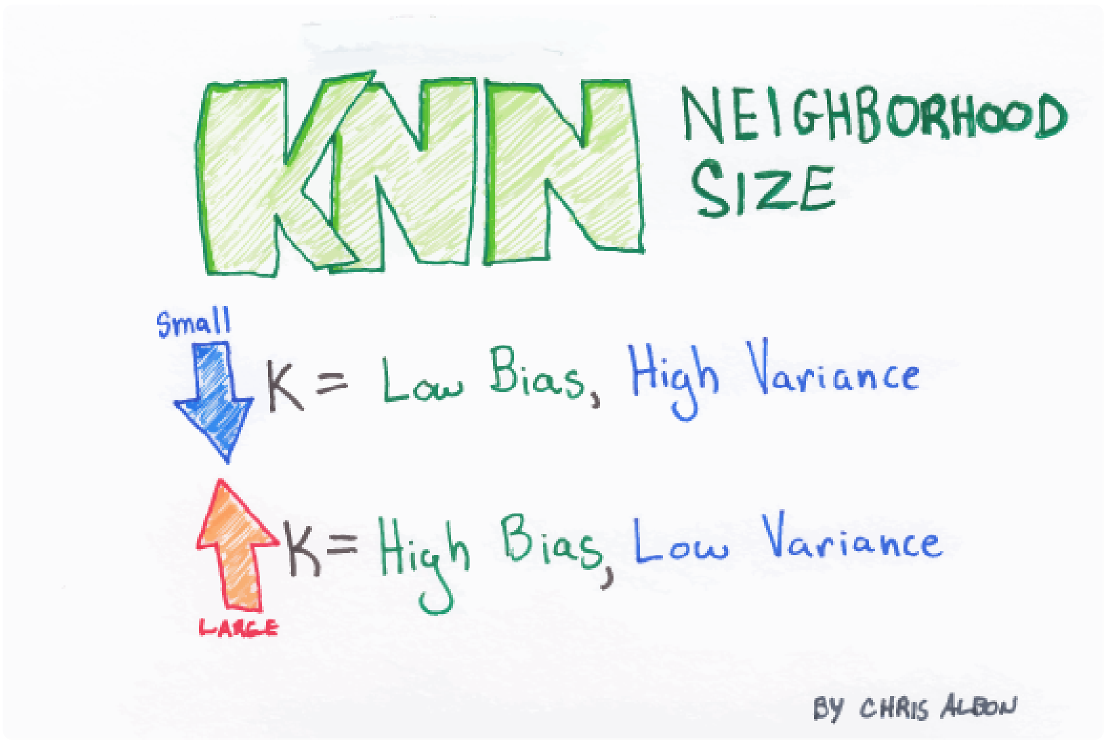
K-NN algorithm is one of the simplest classification algorithms and it is used to identify the data points that are separated into several classes to predict the classification of a new sample point. K-NN is a non-parametric, lazy learning algorithm. It classifies new cases based on a similarity measure (i.e., distance functions).


k-NEAREST NEIGHBORS
- K is the number of neighbors to consider.
- Scaling is important.
- K should be odd.
- If we have binary features we can use Hamming distance.
- Voting can be weighted by distance to each neighbor.
- Does not scale to large data well.
If k=3, the grey square observation is predicted to be green because two of its neighbors are green and only one is red.
Chris Albon


per-se. It is lazy and just memorizes
the data.
Chris Albon


DOES K-NN LEARN
K-nearest neighbor does not "learn" per-se. It is lazy and just memorizes the data.
Chris Albon


the same scale.
TIPS AND TRICKS
2. K should be odd to avoid ties.
3. Votes can be weighted by the distance to the neighbor so closer observations' votes are worth more.
4. Try a variety of distance measurements.
ChrisAlbon



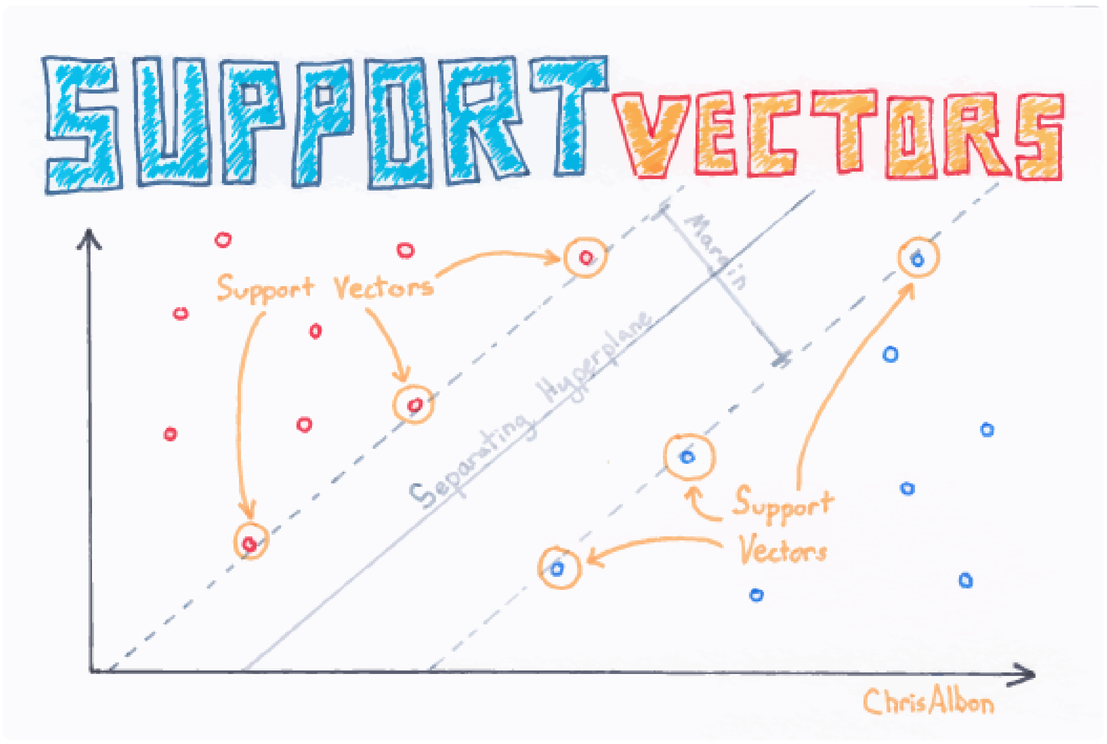
K-NN works well with a small number of input variables (*p*), but struggles when the number of inputs is very large.

Find out who's hiring.

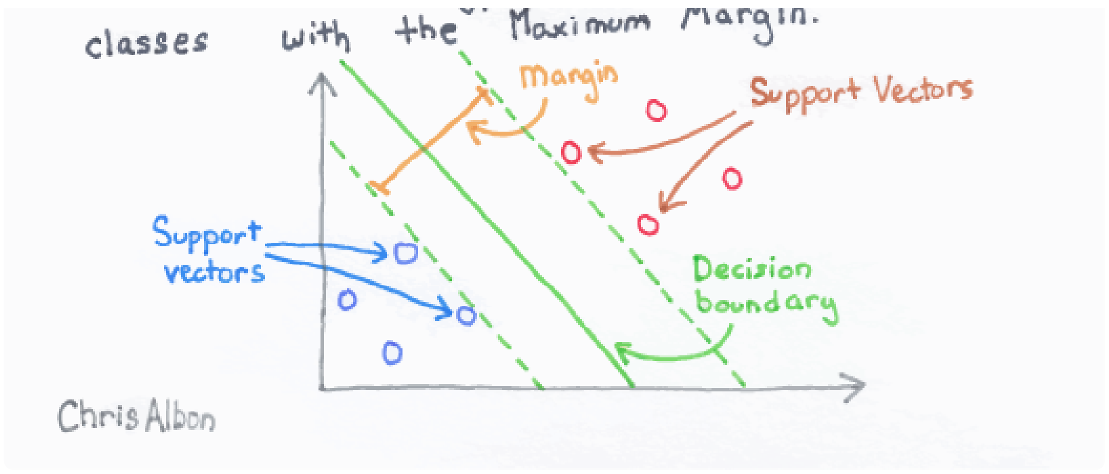
See all **Data + Analytics** jobs at top tech companies & startups

3. SUPPORT VECTOR MACHINE (SVM)

Support vector is used for both regression and classification. It is based on the concept of decision planes that define decision boundaries. A decision plane (hyperplane) is one that separates between a set of objects having different class memberships.



It performs classification by finding the hyperplane that maximizes the margin between the two classes with the help of support vectors.



The learning of the hyperplane in SVM is done by transforming the problem using some linear algebra (i.e., the example above is a linear kernel which has a linear separability between each variable).

For higher dimensional data, other kernels are used as points and cannot be classified easily. They are specified in the next section.

Kernel SVM

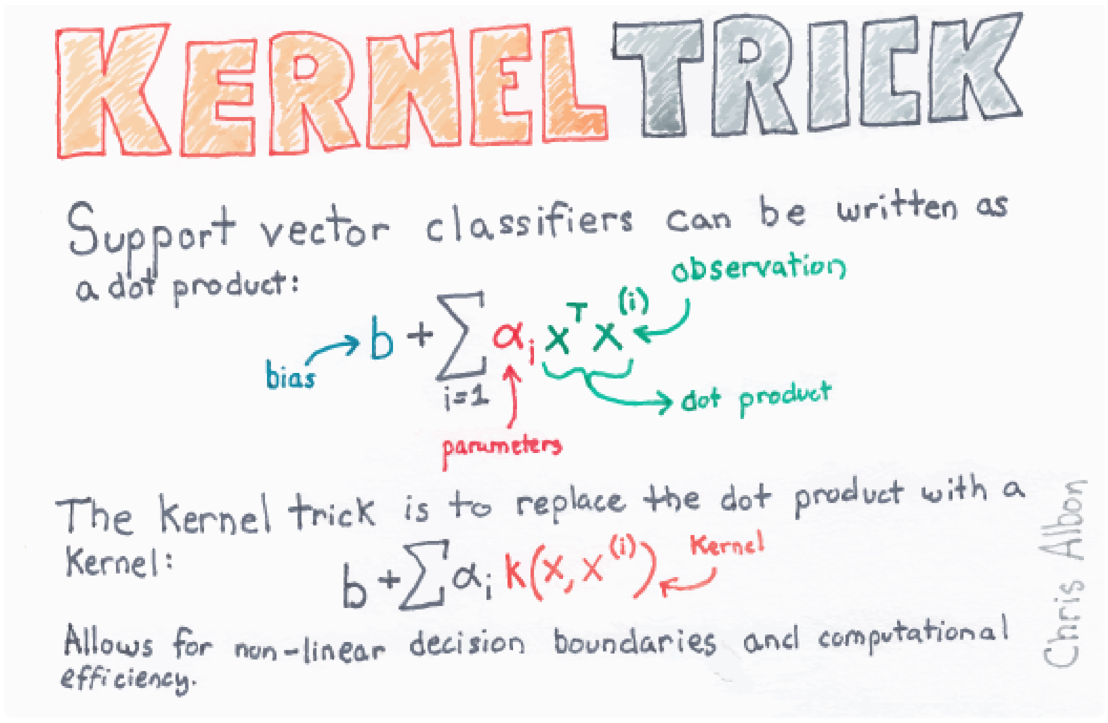
Kernel SVM takes in a kernel function in the SVM algorithm and transforms it into the required form that maps data on a higher dimension which is separable.

Types of kernel function are:

$$K(\mathbf{X}_i, \mathbf{X}_j) = \left\{ \begin{array}{ll} \mathbf{X}_i \cdot \mathbf{X}_j & \text{Linear} \\ (\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C)^d & \text{Polynomial} \\ \exp(-\gamma \,|\, \mathbf{X}_i - \mathbf{X}_j \,|^2) & \text{RBF} \\ \tanh(\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C) & \text{Sigmoid} \end{array} \right\}$$

Type of kernel functions

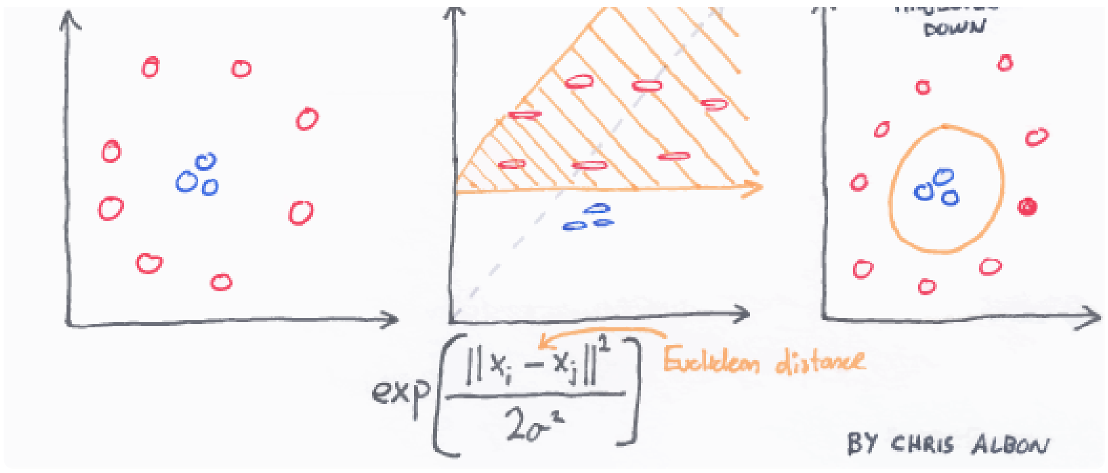
1. Linear SVM is the one we discussed earlier.
2. In polynomial kernel, the degree of the polynomial should be specified. It allows for curved lines in the input space.
3. In the radial basis function (RBF) kernel, it is used for non-linearly separable variables. For distance, metric squared Euclidean distance is used. Using a


KERNEL TRICK
Support vector classifiers can be written as a dot product:
$b + \sum_{i=1} \alpha_i x^T x^{(i)}$
bias
observation
dot product
parameters
The kernel trick is to replace the dot product with a kernel:
$b + \sum \alpha_i k(x, x^{(i)})$
Kernel
Allows for non-linear decision boundaries and computational efficiency.
Chris Albon

Kernel trick uses the kernel function to transform data into a higher dimensional feature space and makes it possible to perform the linear separation for classification.

Radial Basis Function (RBF) Kernel

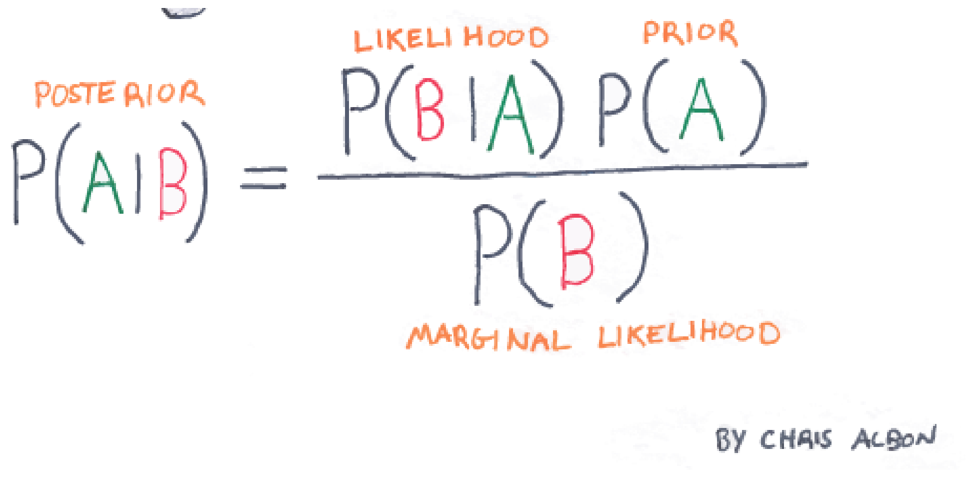
The RBF kernel SVM decision region is actually also a linear decision region. What RBF kernel SVM actually does is create non-linear combinations of features to uplift the samples onto a higher-dimensional feature space where a linear decision boundary can be used to separate classes.



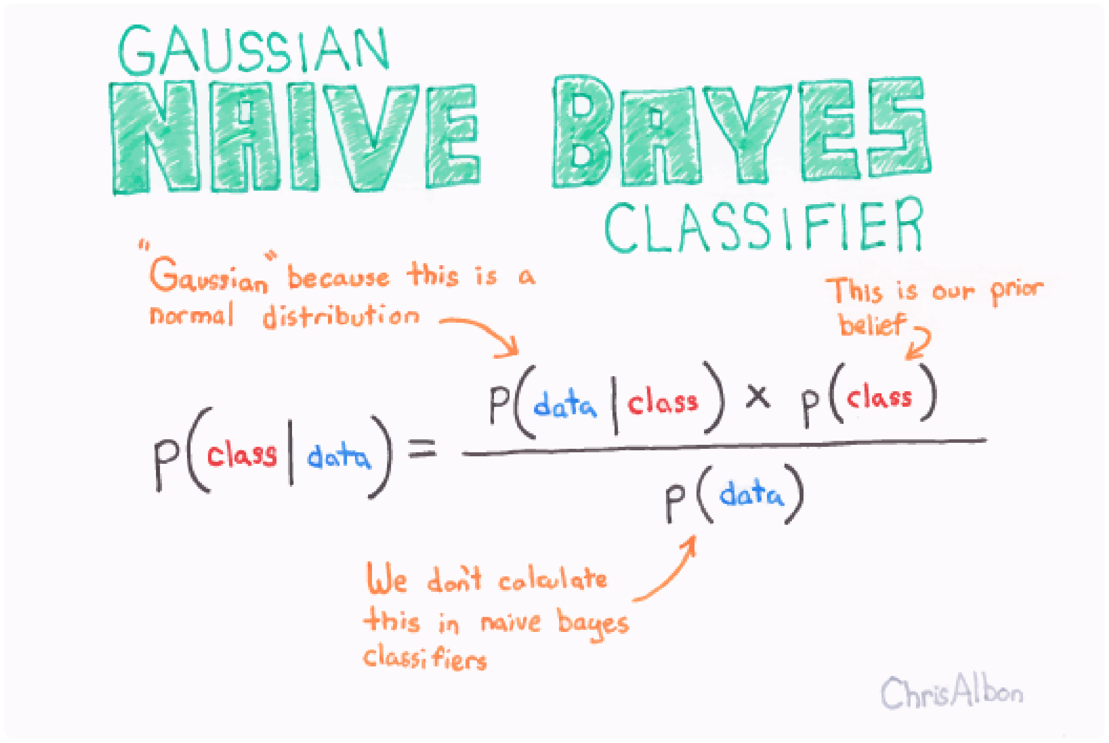
So, the rule of thumb is: use linear SVMs for linear problems, and nonlinear kernels such as the RBF kernel for non-linear problems.

4. NAIVE BAYES

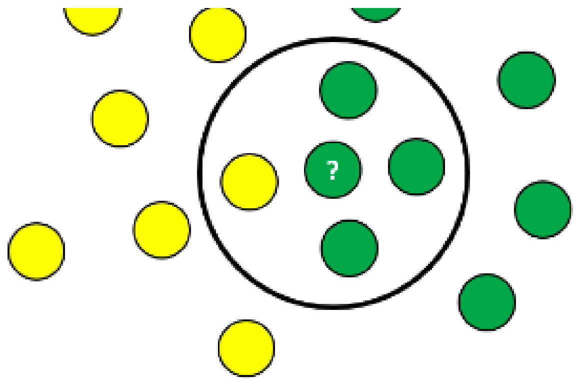
The naive Bayes classifier is based on Bayes' theorem with the independence assumptions between predictors (i.e., it assumes the presence of a feature in a class is unrelated to any other feature). Even if these features depend on each other, or upon the existence of the other features, all of these properties independently. Thus, the name naive Bayes.



Based on naive Bayes, Gaussian naive Bayes is used for classification based on the binomial (normal) distribution of data.


GAUSSIAN
NAIVE BAYES
CLASSIFIER
"Gaussian" because this is a normal distribution
This is our prior belief
p(class | data) = P(data | class) × p(class) / p(data)
We don't calculate this in naive bayes classifiers
ChrisAlbon

- *P(class|data)* is the posterior probability of *class*(*target*) given *predictor*(*attribute*). The probability of a data point having either class, given the data point. This is the value that we are looking to calculate.
- *P(class)* is the prior probability of *class*.
- *P(data|class)* is the likelihood, which is the probability of *predictor* given *class*.
- *P(data)* is the prior probability of *predictor or marginal likelihood*.



NB Classification Example

Steps

1. Calculate Prior Probability

P(*class*) = Number of data points in the class/Total no. of observations

P(*yellow*) = 10/17

P(*green*) = 7/17

2. Calculate Marginal Likelihood

P(*data*) = Number of data points similar to observation/Total no. of observations

P(?) = 4/17

The value is present in checking both the probabilities.

3. Calculate Likelihood

P(*data/class*) = Number of similar observations to the class/Total no. of points in the class.

P(?/*yellow*) = 1/7

P(?/*green*) = 3/10

$$P(yellow/?) = \frac{1/7 * 7/17}{4/17} = 0.25$$

$$P(green/?) = \frac{3/10 * 10/17}{4/17} = 0.75$$

5. Classification

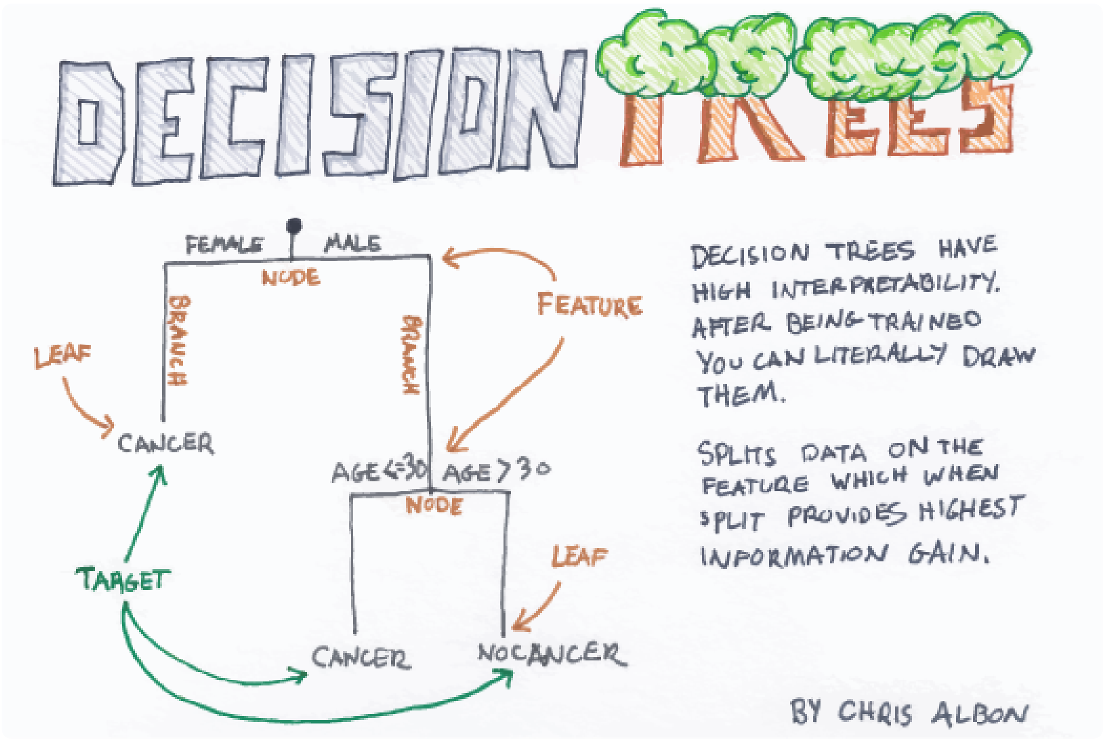
$$P(class1/data) > P(class2/data)$$

$$P(green/?) > P(yellow/?)$$

The higher probability, the class belongs to that category as from above 75% probability the point belongs to class green.

Multinomial, Bernoulli naive Bayes are the other models used in calculating probabilities. Thus, a naive Bayes model is easy to build, with no complicated iterative parameter estimation, which makes it particularly useful for very large datasets.

5. DECISION TREE CLASSIFICATION


DECISION TREES
FEMALE
MALE
NODE
BRANCH
BRANCH
FEATURE
LEAF
CANCER
AGE<=30
AGE>30
NODE
LEAF
TARGET
CANCER
NO CANCER
DECISION TREES HAVE HIGH INTERPRETABILITY. AFTER BEING TRAINED YOU CAN LITERALLY DRAW THEM.
SPLITS DATA ON THE FEATURE WHICH WHEN SPLIT PROVIDES HIGHEST INFORMATION GAIN.
BY CHRIS ALBON

Entropy and information gain are used to construct a decision tree.

Entropy

Entropy is the degree or amount of uncertainty in the randomness of elements. In other words, it is a measure of impurity.

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Entropy

Intuitively, it tells us about the predictability of a certain event. Entropy calculates the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero, and if the sample is equally divided it has an entropy of one.

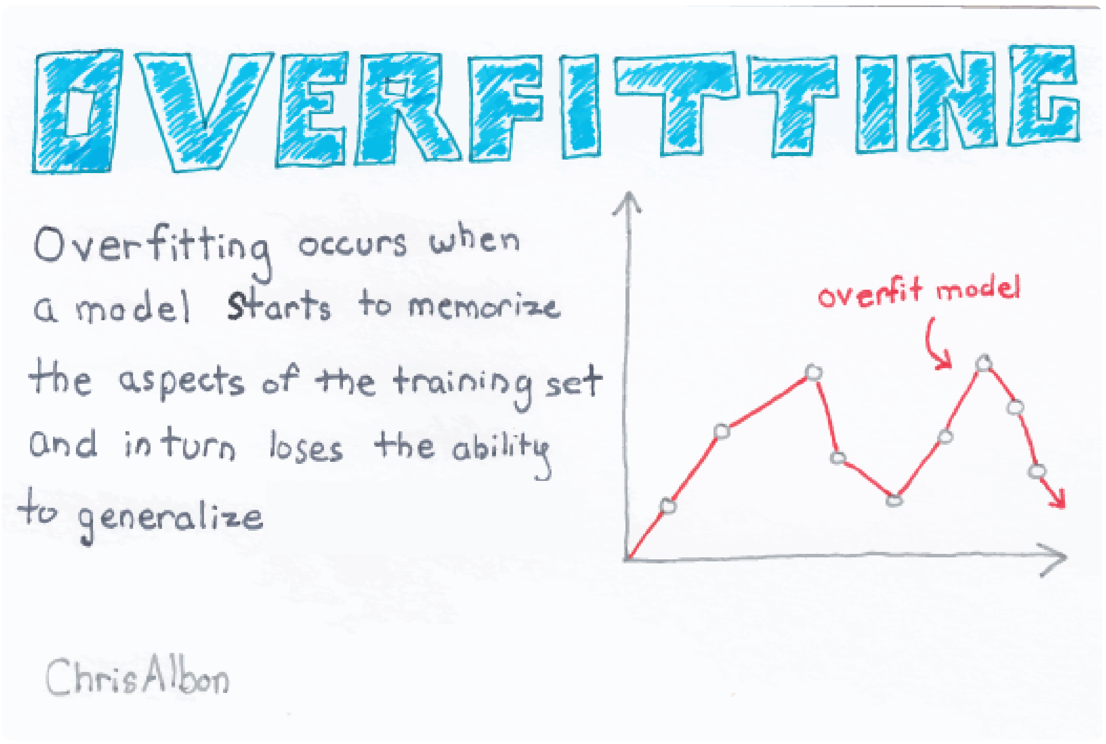
Information Gain

$$Gain(T,X) = Entropy(T) - Entropy(T,X)$$

Where *Gain*(T, X) is the information gain by applying feature X. *Entropy*(T) is the entropy of the entire set, while the second term calculates the entropy after applying the feature X.

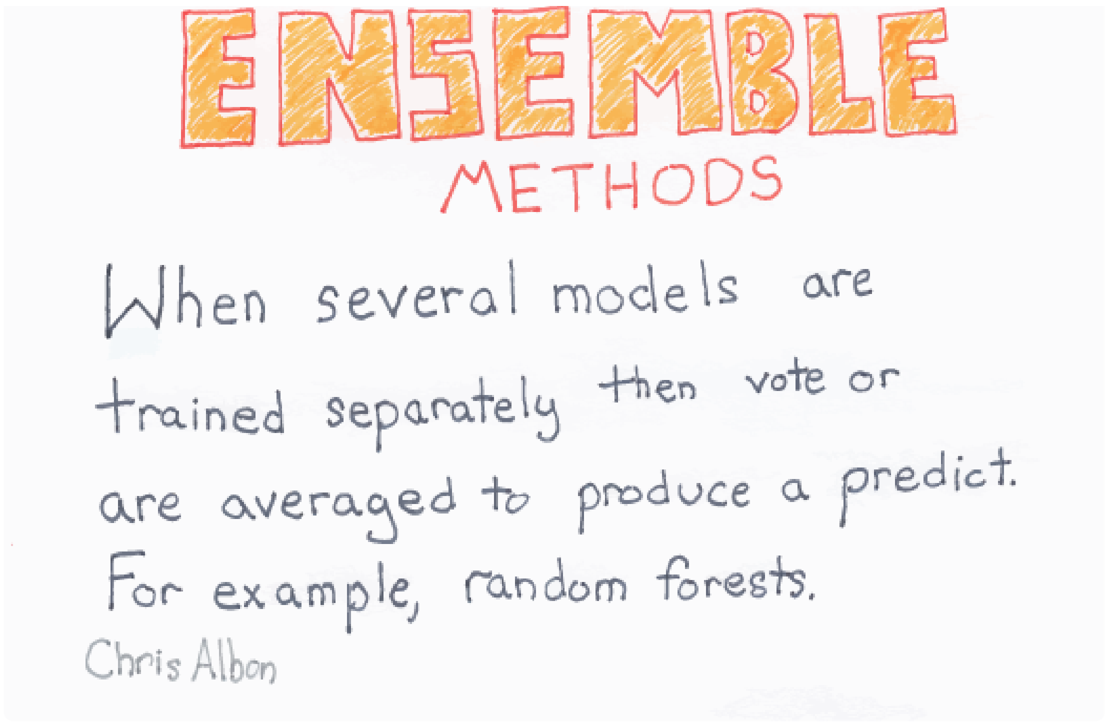
Information gain ranks attributes for filtering at a given node in the tree. The ranking is based on the highest information gain entropy in each split.

The disadvantage of a decision tree model is overfitting, as it tries to fit the model by going deeper in the training set and thereby reducing test accuracy.


OVERFITTING
Overfitting occurs when a model starts to memorize the aspects of the training set and in turn loses the ability to generalize
overfit model
Chris Albon

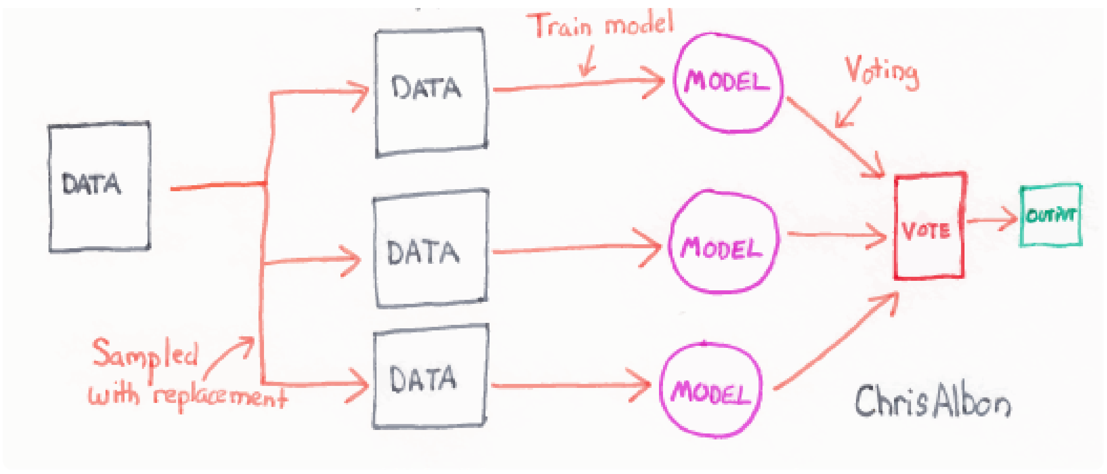
Overfitting in decision trees can be minimized by pruning nodes.

Ensemble Methods for Classification

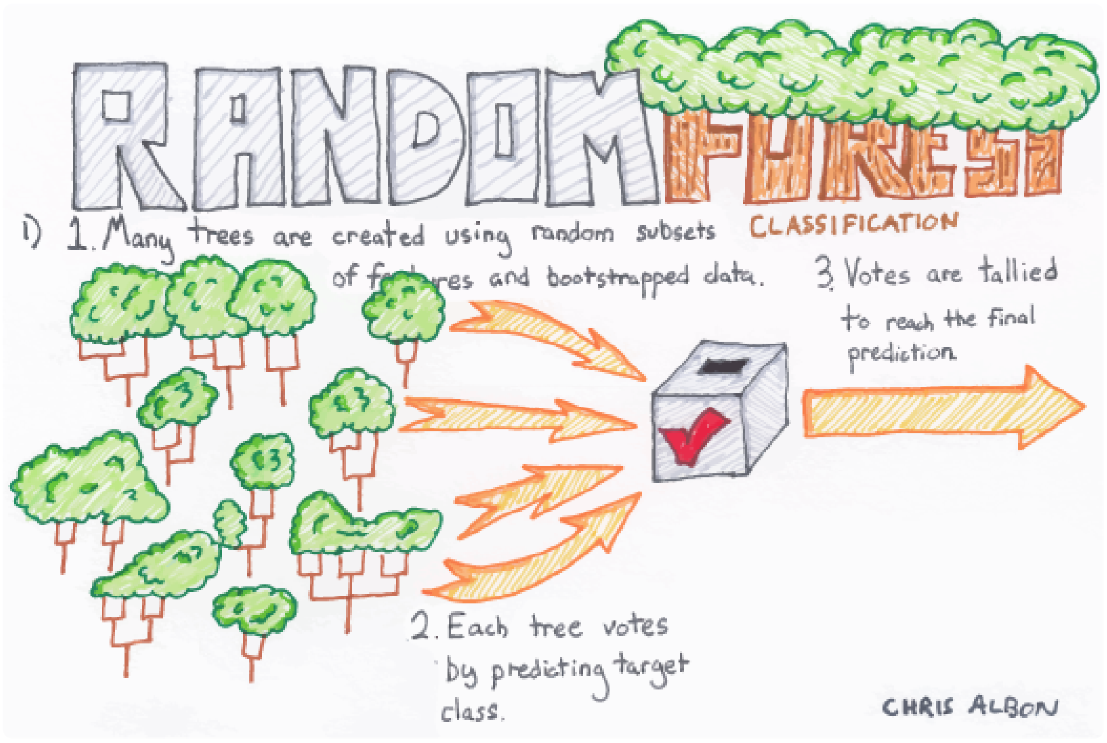

ENSEMBLE
METHODS
When several models are trained separately then vote or are averaged to produce a predict. For example, random forests.
Chris Albon

1. RANDOM FOREST CLASSIFICATION

Random forest classifier is an ensemble algorithm based on bagging i.e bootstrap aggregation. Ensemble methods combines more than one algorithm of the same or different kind for classifying objects (i.e., an ensemble of SVM, naive Bayes or decision trees, for example.)



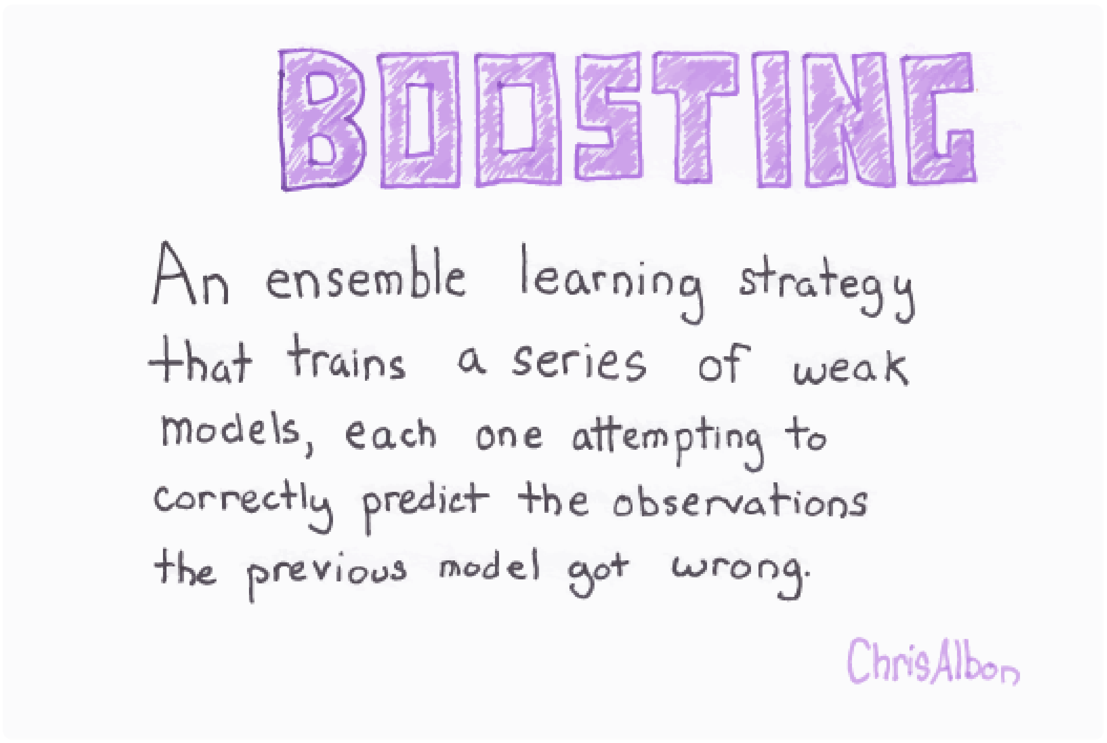
The general idea is that a combination of learning models increases the overall result selected.


RANDOM FOREST
CLASSIFICATION
1. Many trees are created using random subsets of features and bootstrapped data.
2. Each tree votes by predicting target class.
3. Votes are tallied to reach the final prediction.
CHRIS ALBON

Deep decision trees may suffer from overfitting, but random forests prevent overfitting by creating trees on random subsets. The main reason is that it takes the average of all the predictions, which cancels out the biases.

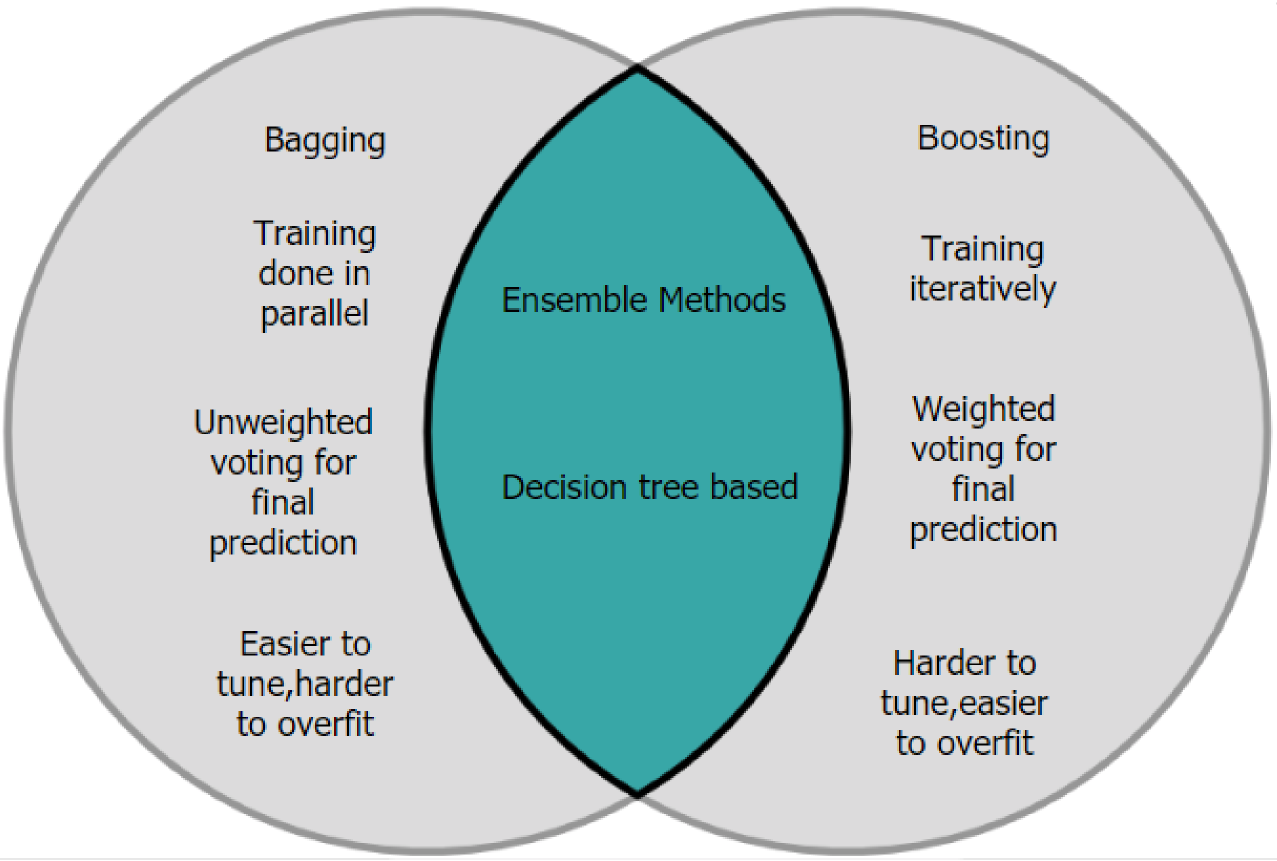
Random forest adds additional randomness to the model while growing the trees. Instead of searching for the most important feature while splitting a node, it searches

Gradient boosting classifier is a boosting ensemble method. Boosting is a way to combine (ensemble) weak learners, primarily to reduce prediction bias. Instead of creating a pool of predictors, as in bagging, boosting produces a cascade of them, where each output is the input for the following learner. Typically, in a bagging algorithm trees are grown in parallel to get the average prediction across all trees, where each tree is built on a sample of original data. Gradient boosting, on the other hand, takes a sequential approach to obtaining predictions instead of parallelizing the tree building process. In gradient boosting, each decision tree predicts the error of the previous decision tree—thereby *boosting* (improving) the error (gradient).


BOOSTING
An ensemble learning strategy that trains a series of weak models, each one attempting to correctly predict the observations the previous model got wrong.
ChrisAlbon

Working of Gradient Boosting

1. Initialize predictions with a simple decision tree.
2. Calculate residual (actual-prediction) value.
3. Build another shallow decision tree that predicts residual based on all the independent values.
4. Update the original prediction with the new prediction multiplied by learning rate.


Bagging
Training done in parallel
Unweighted voting for final prediction
Easier to tune,harder to overfit
Ensemble Methods
Decision tree based
Boosting
Training iteratively
Weighted voting for final prediction
Harder to tune,easier to overfit

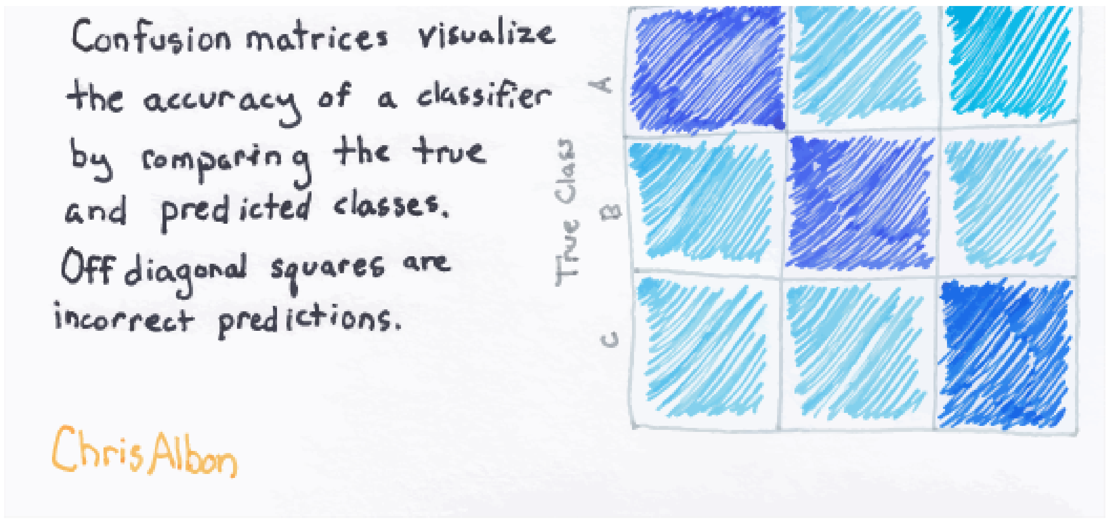
Difference between RF & GB

Checkout this post: Gradient Boosting From Scratch

Classification Model Performances

1. CONFUSION MATRIX

A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. It is a table with four different combinations of predicted and actual values in the case for a binary classifier.

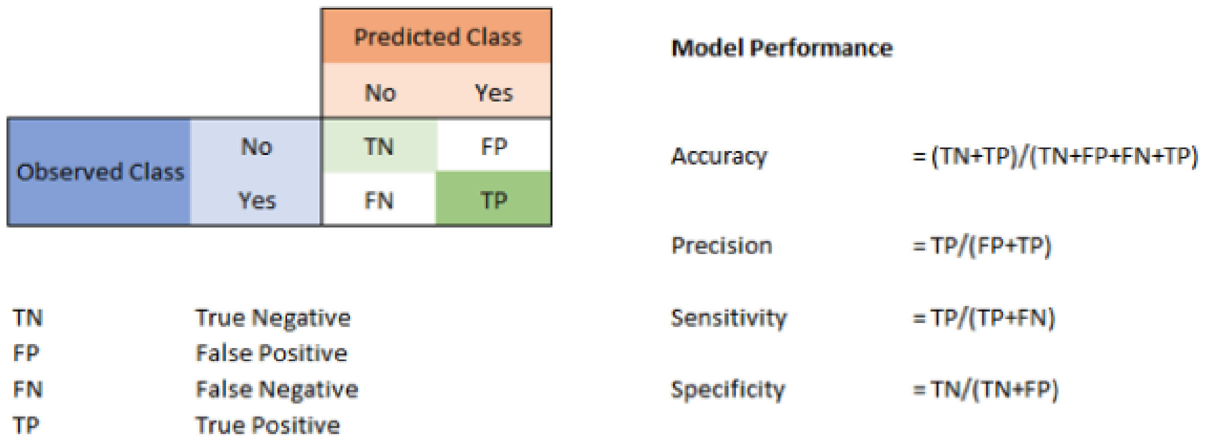

Confusion matrices visualize the accuracy of a classifier by comparing the true and predicted classes. Off diagonal squares are incorrect predictions.

Chris Albon

General Multiclass Confusion Matrix

The confusion matrix for a multi-class classification problem can help you determine mistake patterns.

For a binary classifier:



		Predicted Class	
		No	Yes
Observed Class	No	TN	FP
	Yes	FN	TP

TN	True Negative
FP	False Positive
FN	False Negative
TP	True Positive

Model Performance

Accuracy	= (TN+TP)/(TN+FP+FN+TP)
Precision	= TP/(FP+TP)
Sensitivity	= TP/(TP+FN)
Specificity	= TN/(TN+FP)

Binary Confusion Matrix

A true positive is an outcome where the model correctly predicts the positive class. Similarly, a true negative is an outcome where the model correctly predicts

The terms false positive and false negative are used in determining how well the model is predicting with respect to classification. A false positive is an outcome where the model *incorrectly* predicts the *positive* class. And a false negative is an outcome where the model *incorrectly* predicts the *negative* class. The more values in main diagonal, the better the model, whereas the other diagonal gives the worst result for classification.

FALSE POSITIVE

An example in which the model mistakenly predicted the positive class. For example, the model inferred that a particular email message was spam (the positive class), but that email message was actually not spam. It's like a warning sign that the mistake should be rectified as it's not much of a serious concern compared to false negative.

False positive (*type I error*)—when you reject a true null hypothesis

$$FPR = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$

FALSE NEGATIVE

An example in which the model mistakenly predicted the **negative class**. For example, the model inferred that a particular email message was not spam (the negative class), but that email message actually was spam. It's like a danger sign that the mistake should be rectified early as it's more serious than a false positive.

False negative (*type II error*)—when you accept a false null hypothesis.

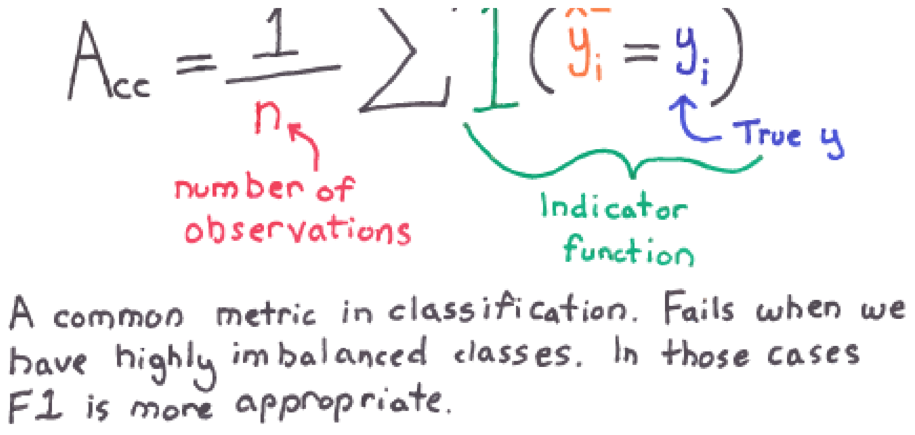
You're pregnant

This picture perfectly easily illustrates the above metrics. The man's test results are a false positive since a man cannot be pregnant. The woman's test results are a false negative because she's clearly pregnant.

From the confusion matrix, we can infer accuracy, precision, recall and F-1 score.

Accuracy

Accuracy is the fraction of predictions our model got right.


$Acc = \frac{1}{n} \sum \mathbb{1}(\hat{y}_i = y_i)$
number of observations
Indicator function
True y
A common metric in classification. Fails when we have highly imbalanced classes. In those cases F1 is more appropriate.

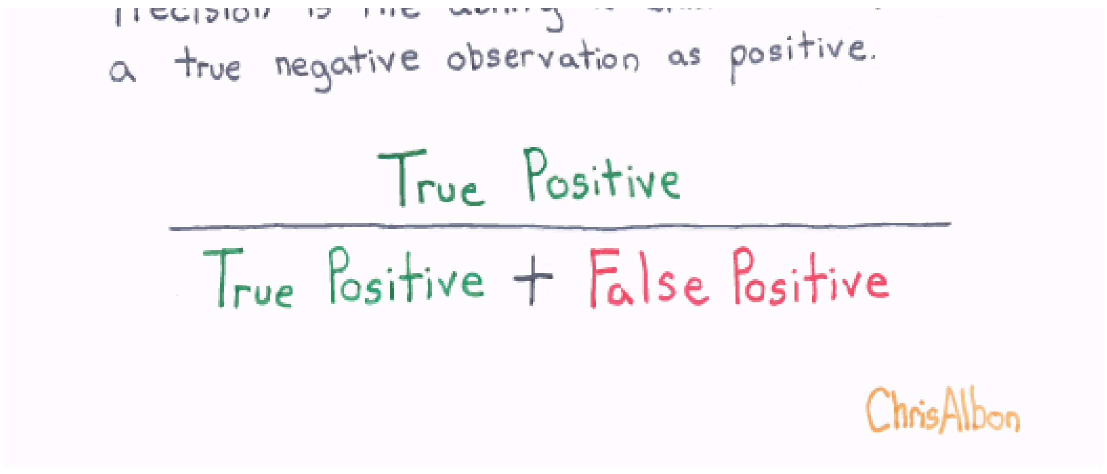
Accuracy can also be written as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy alone doesn't tell the full story when working with a class-imbalanced data set, where there is a significant disparity between the number of positive and negative labels. Precision and recall are better metrics for evaluating class-imbalanced problems.

Precision

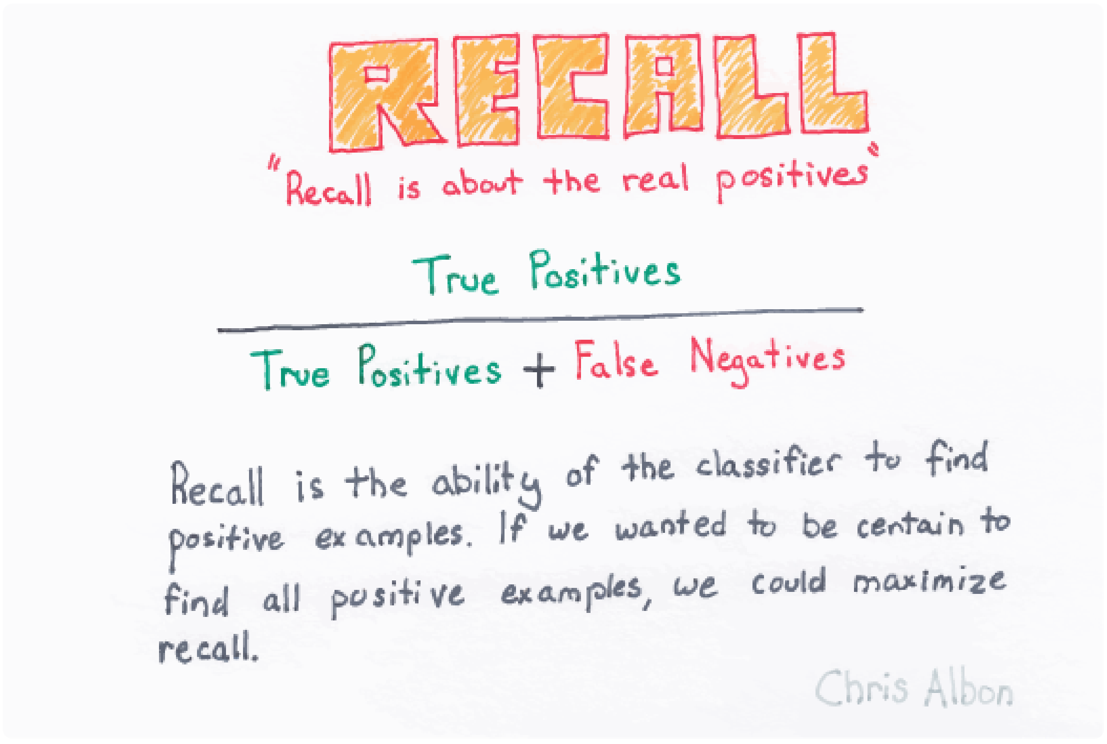
Out of all the classes, precision is how much we predicted correctly.


a true negative observation as positive.
True Positive
True Positive + False Positive
ChrisAlbon

Precision should be as high as possible.

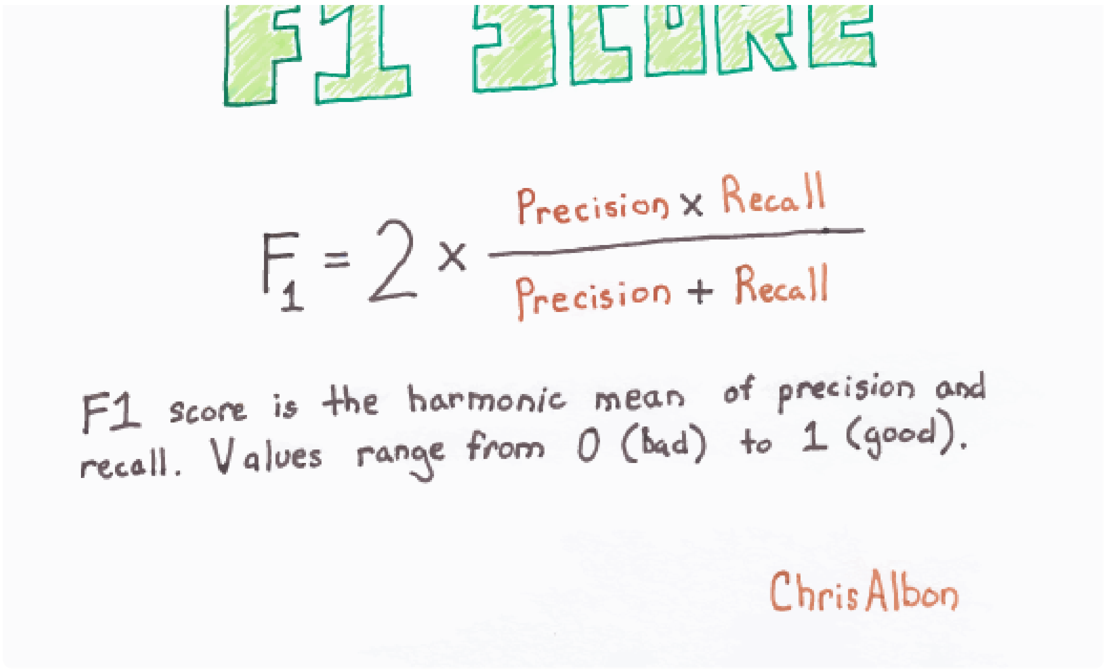
Recall

Out of all the positive classes, recall is how much we predicted correctly. It is also called sensitivity or true positive rate (TPR).


RECALL
"Recall is about the real positives"
True Positives
True Positives + False Negatives
Recall is the ability of the classifier to find positive examples. If we wanted to be certain to find all positive examples, we could maximize recall.
Chris Albon

Recall should be as high as possible.

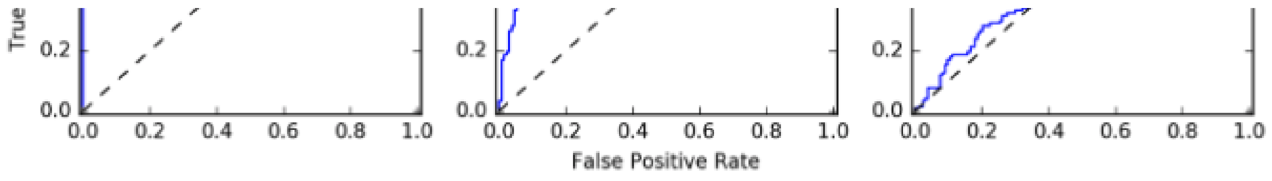
F-1 Score

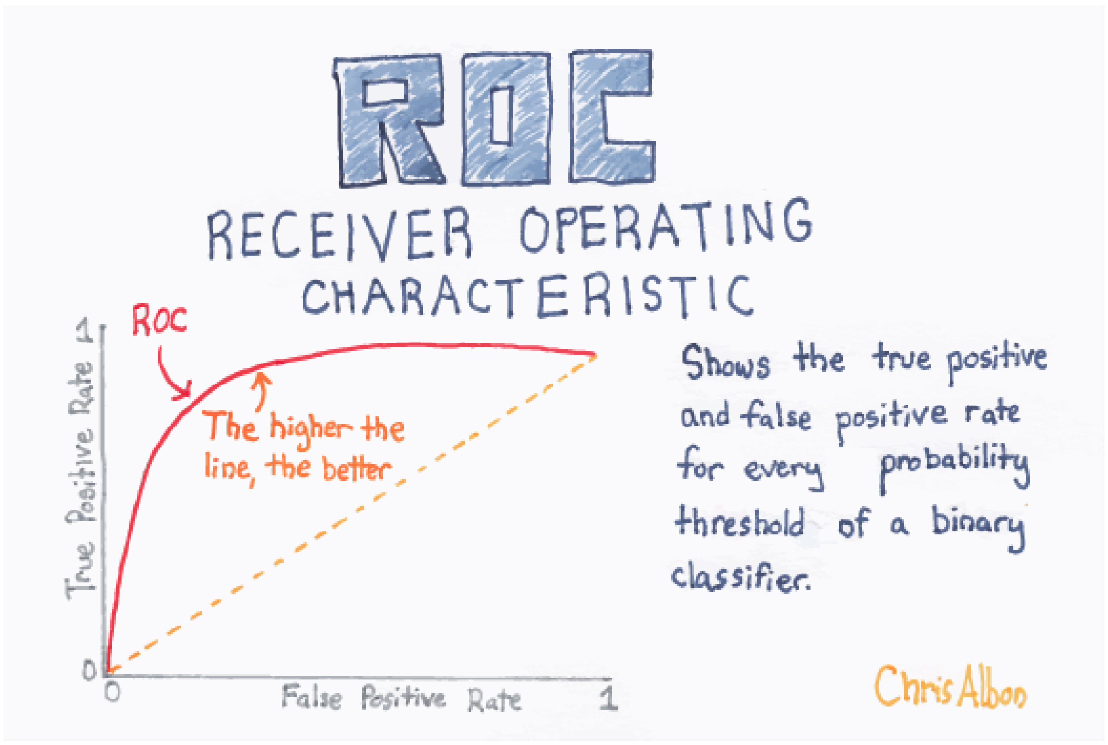

F1 SCORE
$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$
F1 score is the harmonic mean of precision and recall. Values range from 0 (bad) to 1 (good).
Chris Albon

The regular mean treats all values equally, while the harmonic mean gives much more weight to low values thereby punishing the extreme values more. As a result, the classifier will only get a high F-1 score if both recall and precision are high.

3. RECEIVER OPERATOR CURVE (ROC) & AREA UNDER THE CURVE (AUC)

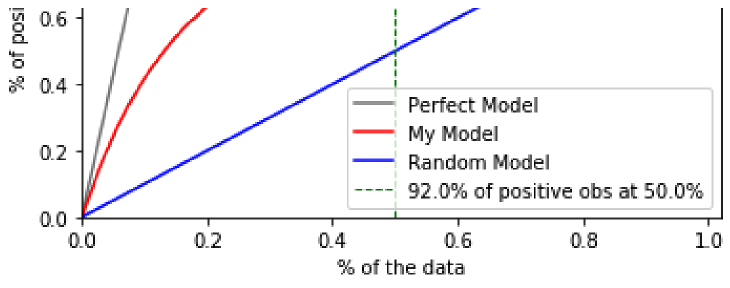
ROC curve is an important classification evaluation metric. It tells us how well the model has accurately predicted. The ROC curve shows the sensitivity of the classifier by plotting the rate of true positives to the rate of false positives. If the classifier is outstanding, the true positive rate will increase, and the area under the curve will be close to one. If the classifier is similar to random guessing, the true positive rate will increase linearly with the false positive rate. The better the AUC measure, the better the model.




ROC
RECEIVER OPERATING CHARACTERISTIC
ROC
The higher the line, the better
True Positive Rate
False Positive Rate
0
1
Shows the true positive and false positive rate for every probability threshold of a binary classifier.
Chris Albon

4. CUMULATIVE ACCURACY PROFILE CURVE

The CAP of a model represents the cumulative number of positive outcomes along the y-axis versus the corresponding cumulative number of a classifying parameters along the x-axis. The CAP is distinct from the receiver operating characteristic (ROC), which plots the true-positive rate against the false-positive rate. CAP curve is rarely used as compared to ROC curve.



CAP Curve

$$CAPCurve(x,y) = (\frac{(TP+FP)}{n}, \frac{TP}{(TP+FN)})$$

$$Where, n = TP+FN+FP+TN$$

Consider a model that predicts whether a customer will purchase a product. If a customer is selected at random, there is a 50% chance they will buy the product. The cumulative number elements for which the customer buys would rise linearly toward a maximum value corresponding to the total number of customers. This distribution is called the “random” CAP. Its the blue line in the above diagram. A perfect prediction, on the other hand, determines exactly which customer will buy the product, such that the maximum customer buying the property will be reached with a minimum number of customer selection among the elements. This produces a steep line on the CAP curve that stays flat once the maximum is reached, which is the “perfect” CAP. It's also called the “ideal” line and is the grey line in the figure above.

In the end, a model should predict where it maximizes the correct predictions and gets closer to a perfect model line.

References: Classifier Evaluation With CAP Curve in Python

Classification Implementation: Github Repo.

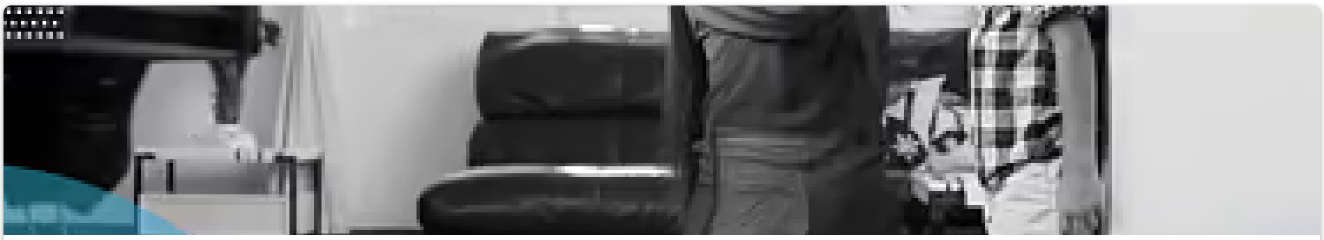
Curse of Dimensionality

RELATED

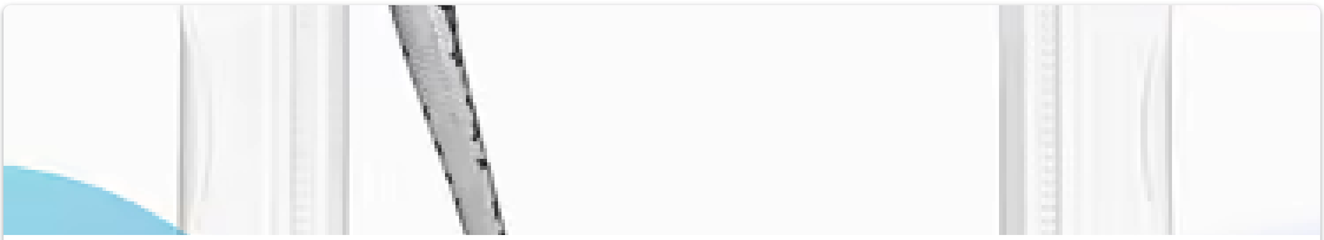
Read More About Data Science

RECENT DATA SCIENCE ARTICLES

How to Ace a Data Scientist Job Interview With American Express



Ordinal Data Versus Nominal Data: What's the Difference?



How to Define Empty Variables and Data Structures in Python

EXPERT CONTRIBUTOR NETWORK

Expert Contributors

Built In's expert contributor network publishes thoughtful, solutions-oriented stories written by innovative tech professionals. It is the tech industry's definitive destination for sharing compelling, first-person accounts of problem-solving on the road to innovation.

LEARN MORE

Great Companies Need Great People. **That's Where We Come In.**

RECRUIT WITH US

Content Descriptions

Company News

Get Involved

Recruit With Built In

Become an Expert Contributor

Send Us a News Tip

Resources

Customer Support

Share Feedback

Report a Bug

Browse Jobs

Tech Hubs

Built In Austin

Built In Boston

Built In Chicago

Built In Colorado

Built In LA

Built In NYC

Built In San Francisco

Built In Seattle