Precision and recall

Imagine that, your girlfriend gave you a birthday surprise every year in last **10** years.However, one day, your girlfriend asks you:

'Sweetie, do you remember all birthday surprises from me?'

This simple question makes your life in danger.

*So, recall is the ratio of a number of **events you can correctly recall** to a number of **all correct events**.*

If you can recall all **10** events correctly, then, your recall ratio is **1.0** (**100%**). If you can recall **7** events correctly, your recall ratio is **0.7** (**70%**).

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$

True positive rate (TPR), Recall, Sensitivity, probability of detection $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$

Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$

$F_1 \text{ score} = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$ harmonic mean

However, you might be wrong in some answers.

*So, precision is the ratio of a number of **events you can correctly recall** to a number **all events you recall** (mix of correct and wrong recalls). In other words, it is how precise of your recall.*

From the previous example (10 real events, 15 answers: 10 correct answers, 5 wrong answers), you get **100%** recall but your precision is only **66.67%** (10 / 15).

Yes, you can guess what I'm going to say next. If a machine learning algorithm is good at recall, it doesn't mean that algorithm is good at precision. That's why we also need F1 score which is the *(harmonic)* mean of recall and precision to evaluate an algorithm.

Bias-Variance Trade-Off

Error in prediction can be of three types:

- Bias Error: Algo. has limited flexibility to learn features from dataset
- Variance Error: Algo. is specific to certain sets in dataset(failed to generalize)
- Irreducible Error

If any one is high then either underfitting, overfitting or model complexity increases

Low Bias(use more complex algo): **Accuracy,** Decision tree, KNN, Non-linear algo
Low Variance(use less complex algo): **Consistency,** Regression, NB, Linear algo

TradeOff can be handled using: Pruning/Regularisation, Systematic cross validation

In k-fold cross-validation, you split the input data into k subsets of data . You train an ML model on all but one (k-1) of the subsets, and then evaluate the model on the subset that was not used for training. This process is repeated k times, with a different subset.

Regularization

It constrains or shrinks the **coefficient estimates towards zero** to prevent Overfitting

Lasso(L1)

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|$$

Ridge(L2)

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2$$

The **key difference** between these techniques is that Lasso shrinks the less important feature's coefficient to zero thus, removing some feature altogether. So, this works well for **feature selection** in case we have a huge number of features.

Why is Nearest Neighbor a Lazy Algorithm?

K-NN is a lazy learner because it doesn't learn a discriminative function from the training data but "memorizes" the training dataset instead.

For example, the logistic regression algorithm learns its model weights (parameters) during training time.

The "prediction" step in K-NN is relatively expensive! Each time we want to make a prediction, K-NN is searching for the nearest neighbor(s) in the entire training set! (Note that there are certain tricks such as BallTrees and KDtrees to speed this up a bit.)

Naive Bayes classifiers

Simple "probabilistic classifiers" based on applying Bayes' theorem with assumptions that the features are independent of each other.

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

Gaussian naive Bayes

When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution.

Multinomial naive Bayes

With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial

Bernoulli naive Bayes

In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs.

Unsupervised Machine Learning

Unsupervised learning problems can be further grouped into clustering and association problems.

- **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.
- Word2vec

Some popular examples of unsupervised learning algorithms are:

- k-means for clustering problems.
- **Apriori** algorithm for association rule learning problems.(It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database.)

XGBoost

Open-source software library which provides the gradient boosting framework.

In every branch, there are two possible choices (left or right of the split); where the **optimal default directions** are learned from the training data. Also automatic decides "**default** " value incase of missing data.

Classification and Regression Trees or CART

https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

1) Classification Model Using Lazy learner – KNN
2) Classification model Probabilistic Learning using-Naïve Bayes
3) Classification using Decision Trees-C5.0,Ripper Algorithms
4) Predicting Numeric Data : Regression Methods –Linear Model , Logistic Model, Cart Algorithms
5) Black Box Methods: Neural Networks and Support Vector Machines: ANN
6) Finding Patterns : Market Basket Analysis Using Associate Rules: **Apriori** Algorithm

7) Cluster Analysis : K-means Algorithm, Hierarchical Clustering Methods
8) Evaluating Model Performance :Confusion Matrices, ROC, K fold Cross validation, Bootstrap Sampling
9) Improving Model Performance : Bagging ,Boosting and Random Forests
10) Text Mining: Word Clouds, Topic Modeling, LDA, Clustering, N Gram Tokenization.

What is Support Vector Machine?

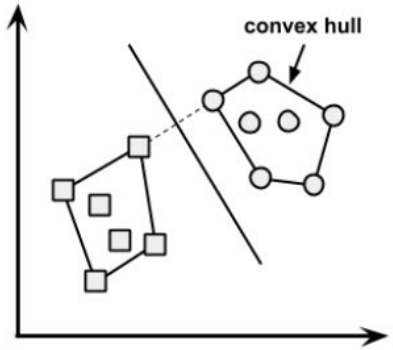
It is a **supervised** machine learning algorithm which can be used for both **classification** or **regression** challenges.

In this algorithm, we **plot each data item as a point in n-dimensional space** (where n is number of features you have). Then, we perform classification by finding the **hyperplane that differentiate** the classes very well .

Params to tune:

- **kernel**:"linear", "**rbf**","poly"(type of plane)
- **gamma**: 'rbf', 'poly' and 'sigmoid'. Higher the value of gamma, will try to exact fit the as per training data set i.e. generalization error and cause over-fitting problem.
- **C:** Penalty parameter C of the error term. It also controls the trade off between smooth decision boundary and classifying the training points correctly.

convex hull:



Answer: In case of linearly separable data, convex hull represents the outer boundaries of the two group of data points. Once convex hull is created, we get maximum margin hyperplane (MMH) as a perpendicular bisector between two convex hulls. MMH is the line which attempts to create greatest separation between two groups.

Pros:

- Well with **clear margin of separation**
- It is effective in **high dimensional spaces**.
- Effective when **number of dimensions is greater than the number of samples**.
- It uses a **subset of training points** in the decision function (called support vectors), so it is also memory efficient.

Cons:

- With large data set , **training time** is higher
- Data set has **more noise** i.e. target classes are **overlapping**
- SVM **doesn't** directly **provide probability** estimates, these are calculated using an expensive five-fold cross-validation. It is related SVC method of Python scikit-learn library.

Hierarchical clustering

HCA is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies :

- **Agglomerative**: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- **Divisive**: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy. Eg DIANA (DIvisive ANAlysis Clustering) algorithm

Names	Formula
Euclidean distance	$\|a-b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$
Squared Euclidean distance	$\|a-b\|_2^2 = \sum_i (a_i - b_i)^2$
Manhattan distance	$\|a-b\|_1 = \sum_i \|a_i - b_i\|$
Maximum distance	$\|a-b\|_\infty = \max_i \|a_i - b_i\|$
Mahalanobis distance	$\sqrt{(a-b)^\top S^{-1}(a-b)}$ where S is the Covariance matrix

Metric
For text or other non-numeric data, metrics such as the Hamming distance or Levenshtein distance are often used.

Time Series Forecasting

Making predictions about the future is called **extrapolation** in the classical statistical handling of time series data.
Forecasting involves taking models fit on historical data and using them to predict future observations.

Descriptive models can borrow for the future (i.e. to smooth or remove noise), they only seek to best describe the data.

Generative/discriminative approach

In statistical classification, two main approaches are called the **generative** approach and the **discriminative** approach.

The task is to determine the language that someone is speaking

- Generative approach: – is to learn each language and determine as to which language the speech belongs to
    - Gaussians, Naïve Bayes, Mixtures of multinomials
    - Mixtures of Gaussians, Mixtures of experts, Hidden Markov Models (HMM)
    - Sigmoidal belief networks, Bayesian networks, Markov random fields
- Discriminative approach: – is determine the linguistic differences without learning any language
    - Logistic regression, SVMs
    - Traditional neural networks, Nearest neighbor
    - Conditional Random Fields (CRF)

Given an observable variable *X* and a target variable *Y*, a **generative model** is a statistical model of the joint probability distribution on $X \times Y$, $P(X,Y)$

A **discriminative model** is a model of the conditional probability of the target *Y*, given an observation *x*, symbolically, $P(Y|X=x)$. Classifiers computed without using a probability model are also referred to loosely as "discriminative".

Latent Dirichlet allocation(Topic modelling)

Based on the words present in a document, LDA assigns topic to each of them. It tells that each document is a mixture of a small number of topics and that each word's creation is attributable to one of the document's topics.

Principal Component Analysis Tutorial

The main idea of principal component analysis (PCA) is to **reduce the dimensionality** of a data set consisting of many **variables correlated** with each other, either heavily or lightly, **while retaining the variation present in the dataset, up to the maximum extent**.

The same is done by transforming the variables to a **new set of variables**, which are known as the **principal components** (or simply, the PCs) and are **orthogonal**, ordered such that the retention of variation present in the original variables decreases as we move down in the order.

Importantly, the dataset on which PCA technique is to be used must be scaled. As a layman, it is a method of summarizing data.

Q18. While working on a data set, how do you select important variables? Explain your methods.

Answer: Following are the methods of variable selection you can use:

- **Remove the correlated variables prior** to selecting important variables
- Use linear regression and **select variables based on p values**
- Use Forward Selection, Backward Selection, Stepwise Selection
- Use Random Forest, Xgboost and plot variable importance chart
- Use Lasso Regression
- Measure information gain for the available set of features and select top n features accordingly.

Q21. Both being tree based algorithm, how is random forest different from Gradient boosting algorithm (GBM)?

Answer: The fundamental difference is, random forest uses bagging technique to make predictions. GBM uses boosting techniques to make predictions.

In bagging technique, a data set is divided into n samples using randomized sampling. Then, using a single learning algorithm a model is build on all samples. Later, the resultant predictions are combined using voting or averaging. Bagging is done is parallel. In boosting, after the first round of predictions, the algorithm weighs misclassified predictions higher, such that they can be corrected in the succeeding round. This sequential process of giving higher weights to misclassified predictions continue until a stopping criterion is reached.

Random forest improves model accuracy by reducing variance (mainly). The trees grown are uncorrelated to maximize the decrease in variance. On the other hand, GBM improves accuracy my reducing both bias and variance in a model.

Q27. What cross validation technique would you use on time series data set? Is it k-fold or LOOCV?

Answer: Neither.

In time series problem, k fold can be troublesome because there might be some pattern in year 4 or 5 which is not in year 3. Resampling the data set will separate these trends, and we might end up validation on past years, which is incorrect. Instead, we can use forward chaining strategy with 5 fold as shown below:

- fold 1 : training [1], test [2]
- fold 2 : training [1 2], test [3]
- fold 3 : training [1 2 3], test [4]

where 1,2,3,4,5,6 represents “year”.

LOOCV (Leave One Out Cross Validation) : This is the same as a **K-fold cross-validation** with K being **equal to the number of observations** in the original sample.

An **advantage** of using this method is that we make **use of all data points** and hence it is **low bias**.

The major drawback of this method is that it leads to **higher variation in the testing model** as we are testing against one data point. If the data point is an outlier it can lead to higher variation. Another drawback is it takes a **lot of execution time** as it iterates over ‘the number of data points’ times.

29. ‘People who bought this, also bought…’ recommendations seen on amazon is a result of which algorithm?

Answer: The basic idea for this kind of recommendation engine comes from collaborative filtering.

Collaborative Filtering algorithm considers “User Behavior” for recommending items.

They exploit behavior of other users and items in terms of transaction history, ratings, selection and purchase information.

In this case, features of the items are not known.

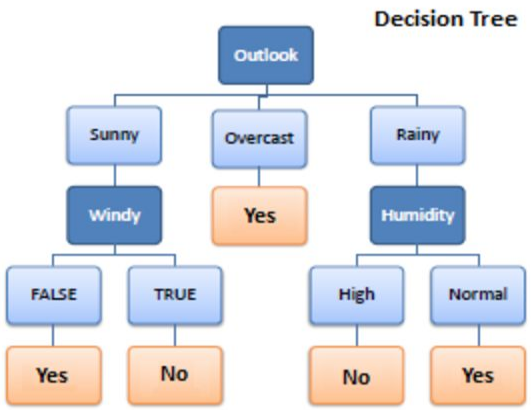
Content Based Recommendations

The feature of items are mapped with feature of users in order to obtain user – item similarity.The top matched pairs are given as recommendations

Q. What do you understand by Type I vs Type II error ?Answer: Type I error is committed when the null hypothesis is true and we reject it, also known as a ‘False Positive’. Type II error is committed when the null hypothesis is false and we accept it, also known as ‘False Negative’. In the context of confusion matrix, we can say Type I error occurs when we classify a value as positive (1) when it is actually negative (0). Type II error occurs when we classify a value as negative (0) when it is actually positive(1).

Decision Trees and Random Forests

Decision trees are a type of model used for both **classification** and **regression**.Tree depth represents how many questions are asked before we reach our predicted classification.



Advantages to using decision trees:

- 1. Easy to **interpret** and make for **straightforward visualizations**.
- 3. Can handle both **numerical** and **categorical** data.
- 4. Perform well on large datasets
- 5. Are extremely **fast**

Disadvantages of decision trees:

- **Hunt's algorithm**. This is a greedy model, meaning it makes the most optimal decision at each step, but does not take into account the global optimum. What does this mean? At each step the algorithm chooses the best result. However, choosing the best result at a given step does not ensure you will be headed down the route that will lead to the optimal decision when you make it to the final node of the tree, called the leaf node.
- Decision trees are **prone to overfitting**, especially when a tree is particularly **deep**. Way to combat this issue is by setting a max depth. This is a simpler model with less variance sample to sample but ultimately will not be a strong predictive model.

Random forests

Ideally, we would like to minimize both error due to bias and error due to variance. **Random forests** mitigate this problem well. A random forest is simply a **collection of decision trees** whose **results are aggregated into one final result.** Their ability to **limit overfitting** without substantially increasing error due to bias is why they are such powerful models.

- One way Random Forests reduce variance is by training on different samples of the data.
- A second way is by using a **random subset of features**. This means if we have 30 features, **random forests will only use a certain number of those features in each model, say five**. If we use **many trees in our forest**, eventually many or all of our features will have been included. This inclusion of many features will help limit our error due to bias and error due to variance. **They aggregate many decision trees to limit overfitting as well as error due to bias and therefore yield useful results.**

Pruning is what happens in decision trees when **branches that have weak predictive power are removed** in order to **reduce the complexity** of the model and **increase** the **predictive accuracy** of a decision tree model. Pruning can happen **bottom-up** and **top-down**, with **approaches** such as **reduced error pruning** and **cost complexity pruning**. **Reduced error pruning** is perhaps the simplest version: replace each node. If it doesn't decrease predictive accuracy, keep it pruned.

Q. OLS is to linear regression. Maximum likelihood is to logistic regression. Explain the statement.

Answer: OLS and Maximum likelihood are the methods used by the respective regression methods to approximate the unknown parameter (coefficient) value. In simple words,

Ordinary least square(OLS) is a method used in linear regression which approximates the parameters resulting in minimum distance between actual and predicted values. Maximum Likelihood helps in choosing the the values of parameters which maximizes the likelihood that the parameters are most likely to produce observed data.

Q. What is the "Curse of Dimensionality?"

The difficulty of searching through a solution space becomes much harder as you have more features (dimensions).The more dimensions you have, the higher volume of data you'll need.

Q. What are the advantages and disadvantages of neural networks?

Advantages: Neural networks (specifically deep NNs) have led to performance breakthroughs for unstructured datasets such as images, audio, and video. Their incredible **flexibility allows them to learn patterns** that no other ML algorithm can learn.

Disadvantages: However, they **require a large amount of training data** to converge. It's also **difficult to pick the right architecture**, and the internal "hidden" layers are incomprehensible.

For example, below are some **standard loss functions** for different predictive model types:

- Regression: Mean Squared Error or mean squared error, mse for short.
- Binary Classification (2 class): Logarithmic Loss, also called cross entropy or binary_crossentropy.
- Multiclass Classification (> 2 class): Multiclass Logarithmic Loss or categorical_crossentropy

LSTM

1. Sequence Prediction (Weather Forecasting,Stock Market Prediction.,Product Recommendation)
2. Sequence Classification(Anomaly Detection,Sentiment Analysis)
3. Sequence Generation(Text Generation,Music Generation,Image Caption Generation)
4. Sequence-to-Sequence Prediction(Text Summarization,Multi-Step Time Series Forecasting)

There are three gates:
Forget Gate: Decides what information to discard from the cell.
Input Gate: Decides which values from the input to update the memory state.
Output Gate: Decides what to output based on input and the

Backpropagation refers to the training algorithm for updating network weights to minimize error.

https://machinelearningmastery.com/truncated-backpropagation-through-time-in-keras/

```
from pandas import Series
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
scaler = scaler.fit(values)
normalized = scaler.transform(values)
```

Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler = scaler.fit(values)
standardized = scaler.transform(values)
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
# integer encode
label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values)

# binary encode
onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
onehot_encoded = onehot_encoder.fit_transform(integer_encoded)
```

```
from keras.preprocessing.sequence import pad_sequences
```

```
padded = pad_sequences(sequences, padding= post )
```

```
from pandas import DataFrame
# define the sequence
df = DataFrame()
df[ t ] = [x for x in range(10)]
# shift backward
df[ t+1 ] = df[ t ].shift(-1)
```

```
history = model.fit(X, y, batch_size=10, epochs=100, verbose=0)
```

By default, the internal state of all LSTM memory units in the network is reset after each batch

```
model.add(LSTM(2, stateful=True, batch_input_shape=(10, 5, 1)))
for i in range(1000):
        model.fit(X, y, epochs=1, batch_input_shape=(10, 5, 1))
        model.reset_states()
```

One-to-One Model

```
model = Sequential()
model.add(LSTM(..., input_shape=(1, ...)))
model.add(Dense(1))
```

One-to-Many Model

```
model = Sequential()
...
model.add(LSTM(...))
```

```
model.add(TimeDistributed(Dense(1)))
```

```
Many-to-One Model
model = Sequential()
model.add(LSTM(..., input_shape=(steps, ...)))
model.add(Dense(1))

Many-to-Many Model
model = Sequential()
model.add(LSTM(..., input_shape=(in_steps, ...)))
model.add(RepeatVector(out_steps))
model.add(LSTM(..., return_sequences=True))
model.add(TimeDistributed(Dense(1)))
```

```
model = Sequential()
model.add(TimeDistributed(Conv2D(...))
model.add(TimeDistributed(MaxPooling2D(...)))
model.add(TimeDistributed(Flatten()))
model.add(LSTM(...))
model.add(Dense(...))
```

Arrays

Remove, Search, Range Search, insert position, Rotate, Search in rotated, Max subarray, Set Matrix, Pascal's Triangle, 2 Sum, 3 Sum, Single number, Intersection, Majority, Duplicates, Missing, Consecutive sequence, valid sudoku, plus one

Sorting

Bubble, Insertion, Counting, Quick, Merge

Heap

Kth largest in array, Kth smallest in matrix, Median, Ugly Number, Super Ugly Number

Strings

Reverse, First unique char, anagram, pallindrome, last word, common prefix, substring without repeating chars,

Numbers

Pallindrome, Power, Sqrt, Reverse, Happy number, Guess number, Next permutation, single

Linked Lists

Delete, Reverse, Remove, Cycle, Reverse range, Rotate, Partition, Merge, Swap Node, Add, Add one, Remove duplicates, Pallindrome, Odd-even

Binary Trees

Max depth, Min depth, Invert, Same, LCA, Level order, Inorder, Preorder, Postorder, Balanced, Symmetric, Validate, Paths, Path sum, Max path sum, Right side view, Flatten to linked list, Kth smallest, Next Right

Graphs

Islands

Dynamic Programming

Climb Stairs, House Robber, Combination Sum, Palindromic Substring, Max product subarray, Frog Jump, Coin Change, Unique Paths, LIS, minimum path sum

Design

Chess, Twitter, LRU cache, Swimming Pool, Payment Gateway, ATM

Customer centricity (listen invent personalize to/for customer)
Thinking long term(invent-experiment-fail)
Passion for invention
Story of kindle (don't make money from harware)
Story of AWS

$$\left\{ {n \atop k} \right\} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n.$$ Stirling numbers of the second kind

In how many ways can the number 1;000;000 (one million) be written as the product of three positive integers a,b,c,

Decision Tree or Naive Bayes

Consider the following (very typical?) example: you have N data points which each have D dimensions, all values that are either 0 or 1. Without noise, assuming each dimension is conditionally independent, and an accurate prior (i.e. the probability of your labels, without knowing any evidence), **you literally cannot do better than Naive Bayes** (using square loss, 0–1 loss, or log loss) — at least, this is true if you're a fan of the Bayesian statistics (I don't think Frequentist statistics can make such bold statements).

(note that this is because Naive Bayes is simply a special case "ideal" Bayesian reasoning (given the Naive Bayes assumptions, namely independence)).

It turns out adding noise (in the sense of "adding some epsilon probability of flipping from 1s to 0s) doesn't matter, because this has an equivalent formulation as Naive Bayes without noise (see bottom of post). In this sense, Naive Bayes is *very* (one might say "perfectly") capable of handling noise.

You might say:

Gee golly, what an impressive claim. It's not often you hear such an absolute claim in machine learning. I'm just going to use Naive Bayes to solve all my problems!

Or maybe you're more cynical and say

There is absolutely no way that can be true. If Naive Bayes is so great, why are all the hard problems being solved by deep neural networks?

The answer is that… your assumptions (*especially* the conditional independence one) are generally wrong… often wildly. Consider classifying an image as either a dog-image or cat-image with Naive Bayes on the pixel values. This is clearly going to perform terribly… but that's because the pixel values aren't independent of each other. Similarly, Naive Bayes for spam-email classification isn't the best thing since sliced bread (it *is* good though), because a word isn't independent from the words around them. In this 2nd case, however, independence is "truer" than in the first, with predictable results.

The absolute *biggest* question in your mind when deciding to use Naive Bayes is this: "to what extent are my variables conditionally independent". If your assumptions (especially the conditional independence) assumption is (close to) true, Naive Bayes will likely amaze you or, if there is a lot of noise, at least outperform anything else you try.

But as you edge away from independence, Naive Bayes starts to get overconfident with its predictions (at the very least) and you lose all guarantees. If you're not confident in your independence assumption, **consider using alternative models**. If you *are* confident in it, **use Naive Bayes**.

Decision trees are high-variance models — that is, relatively modest changes in your data can yield large changes in the resulting tree. As a result (or equivalently?) they can be very sensitive to noise.

In fact, this is why decision trees/stumps (more than any other model) benefit so much from being used in ensembles (i.e. random forests), since the strength of an ensemble depends heavily on the variance of its members.

Unfortunately, what makes decision trees so good in ensembles makes them unreliable individually (*especially* without pruning). Unfortunately, I'm not aware of any formal explanation for the high variability, but at least part of it comes from the idea that two completely dissimilar splits might be similar by your split-metric (e.g. information gain), so relatively small changes in your data can send your tree-constructing algorithm down radically different paths.

Fortunately, there is rarely a good reason to train a tree rather than a random forest, apart from interpretability.

BOTTOM OF POST

As an example, suppose we have a label "y" and two dimensions (1 and 2).

Suppose $P(y|x_1,x_2)=t_1t_2$P(y|x1,x2)=t1t2 (and $P(y|x_1,\neg x_2=t_1(1-t_2)$P(y|x1,¬x2=t1(1−t2), etc.). Then adding epsilon noise to x_1x1 gives us:

$P(y|x_1,x_2)=((1-\epsilon)t_1+\epsilon(1-t_1))t_2$P(y|x1,x2)=((1−ϵ)t1+ϵ(1−t1))t2

$\therefore P(y|x_1,x_2)=(t_1+\epsilon(1-2t_1))t_2 \therefore$ P(y|x1,x2)=(t1+ϵ(1−2t1))t2

This is another Naive Bayes problem, but instead of $P(y|x_1)=t_1$ P(y|x1)=t1, we have $P(y|x_1)=t_1+\epsilon(1-t_1)$ P(y|x1)=t1+ϵ(1−t1). The epsilon noise will certainly hurt our model's certainty, but nobody can do better (because our model is the Bayes optimal model, and nobody can do better than the Bayes Optimal model).

Alternatively, we can add the noise to the label itself:

$P(y|x_1,x_2)=P(z|x_1,x_2)(1-\epsilon)+P(\neg z|x_1,x_2)\epsilon$ P(y|x1,x2)=P(z|x1,x2)(1−ϵ)+P(¬z|x1,x2)ϵ

And the same idea applies:

$P(y|x_1,x_2)=P(z|x_1,x_2)(1-\epsilon)+(1-P(z|x_1,x_2))\epsilon$ P(y|x1,x2)=P(z|x1,x2)(1−ϵ)+(1−P(z|x1,x2))ϵ

$\therefore P(y|x_1,x_2)=P(z|x_1,x_2)+\epsilon(1-2P(z|x_1,x_2)) \therefore$ P(y|x1,x2)=P(z|x1,x2)+ϵ(1−2P(z|x1,x2))

Becomes just another Naive Bayes problem (again, you will necessarily be less confident because of the noise, and your probabilities will be closer to 50–50 (assuming labels are uniformly distributed)).

Master theorem (analysis of algorithms)