Pros and Cons of KNN

Pros

- As said earlier, it is lazy learning algorithm and therefore requires no training prior to making real time predictions. This makes the KNN algorithm much faster than other algorithms that require training e.g SVM, linear regression, etc.
- Since the algorithm requires no training before making predictions, new data can be added seamlessly.
- There are only two parameters required to implement KNN i.e. the value of K and the distance function (e.g. Euclidean or Manhattan etc.)

Cons

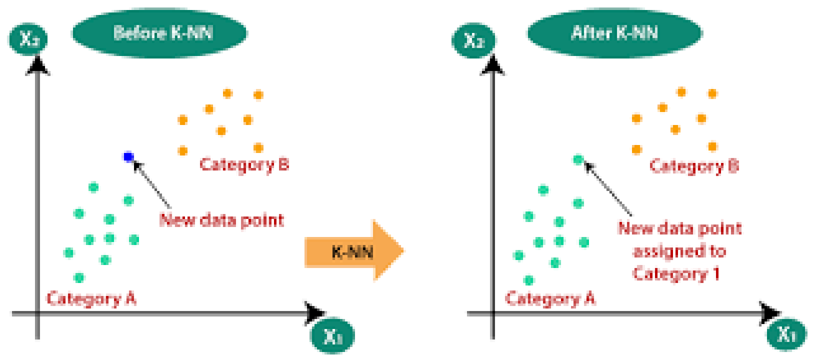
- The KNN algorithm doesn't work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate distance in each dimension.
- The KNN algorithm has a high prediction cost for large datasets. This is because in large datasets the cost of calculating distance between new point and each existing point becomes higher.
- Finally, the KNN algorithm doesn't work well with categorical features since it is difficult to find the distance between dimensions with categorical features.

How to improve KNN?

- For better results, normalizing data on the same scale is highly recommended.
- Generally, the normalization range considered between 0 and 1.
- KNN is not suitable for the large dimensional data. In such cases, dimension needs to reduce to improve the performance. Also, handling missing values will help us in improving results.

What is the KNN Algorithm?

- KNN(K-nearest neighbours) is a supervised learning and non-parametric algorithm that can be used to solve both classification and regression problem statements.
- It uses data in which there is a target column present i.e, labelled data to model a function to produce an output for the unseen data. It uses the euclidean distance formula to compute the distance between the data points for classification or prediction.
- The main objective of this algorithm is that similar data points must be close to each other so it uses the distance to calculate the similar points that are close to each other.



Why is KNN a non-parametric Algorithm?

- The term “non-parametric” refers to not making any assumptions on the underlying data distribution. These methods do not have any fixed numbers of parameters in the model.
- Similarly in KNN, the model parameters grow with the training data by considering each training case as a parameter of the model. So, KNN is a non-parametric algorithm.

What is "K" in the KNN Algorithm?

K represents the number of nearest neighbors you want to select to predict the class of a given item, which is coming as an unseen dataset for the model.

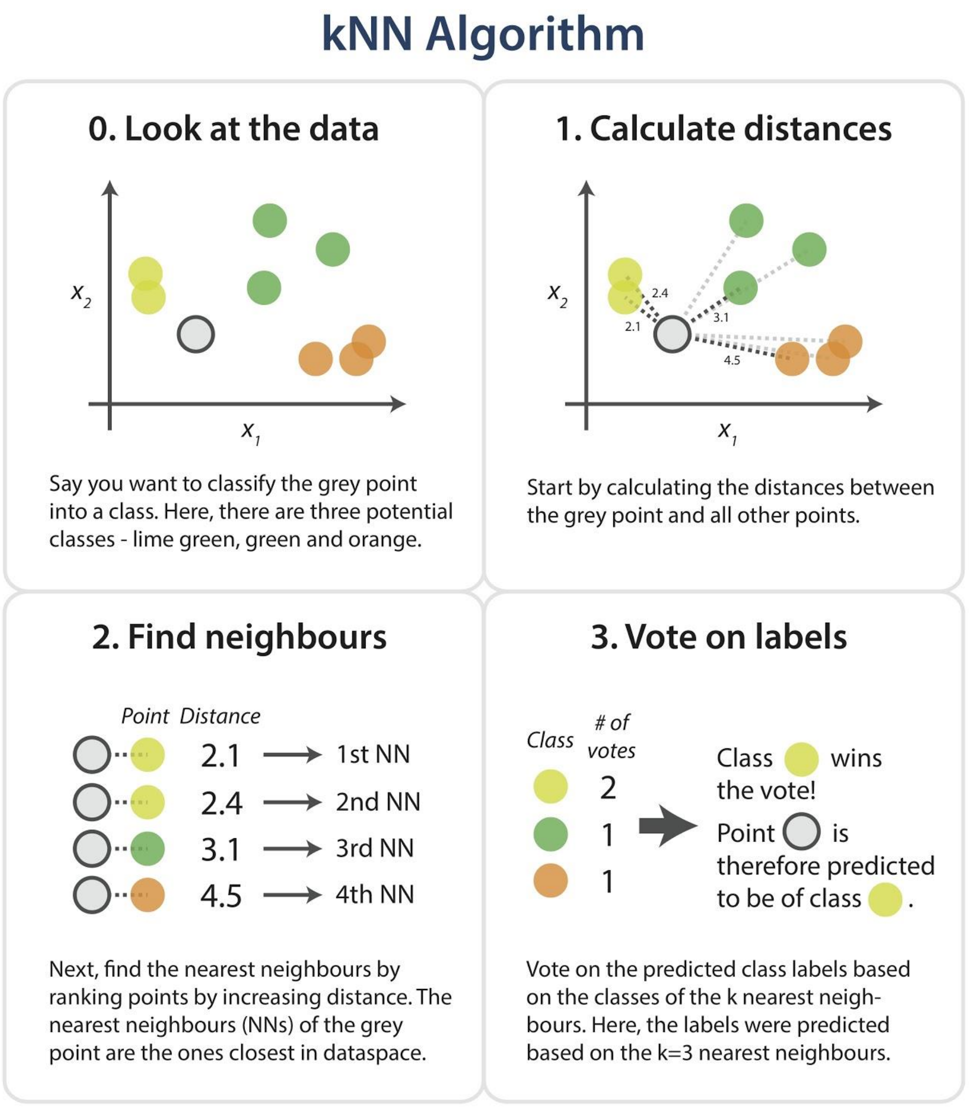
Why is the odd value of "K" preferred over even values in the KNN Algorithm?

The odd value of K should be preferred over even values in order to ensure that there are no ties in the voting. If the square root of a number of data points is even, then add or subtract 1 to it to make it odd.

How does the KNN algorithm make the predictions on the unseen dataset?

The following operations have happened during each iteration of the algorithm. For each of the unseen or test data point, the kNN classifier must:

1. Step-1: Calculate the distances of test point to all points in the training set and store them
2. Step-2: Sort the calculated distances in increasing order
3. Step-3: Store the K nearest points from our training dataset
4. Step-4: Calculate the proportions of each class
5. Step-5: Assign the class with the highest proportion


kNN Algorithm
0. Look at the data
x_2
x_1
Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.
1. Calculate distances
x_2
x_1
2.4
2.1
3.1
4.5
Start by calculating the distances between the grey point and all other points.
2. Find neighbours
Point Distance
2.1 → 1st NN
2.4 → 2nd NN
3.1 → 3rd NN
4.5 → 4th NN
Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.
3. Vote on labels
Class
# of votes
2
1
1
Class wins the vote!
Point is therefore predicted to be of class .
Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

Is Feature Scaling required for the KNN Algorithm? Explain with proper justification.

- Yes, feature scaling is required to get the better performance of the KNN algorithm.
- For Example, Imagine a dataset having n number of instances and N number of features. There is one feature having values ranging between 0 and 1. Meanwhile, there is also a feature that varies from -999 to 999. When these values are substituted in the formula of Euclidean Distance, this will affect the performance by giving higher weightage to variables having a higher magnitude.

What is space and time complexity of the KNN Algorithm?

Time complexity: The distance calculation step requires quadratic time complexity, and the sorting of the calculated distances requires an O(N log N) time. Together, we can say that the process is an O(N3 log N) process, which is a monstrously long process.

Space complexity: Since it stores all the pairwise distances and is sorted in memory on a machine, memory is also the problem. Usually, local machines will crash, if we have very large datasets.

Can the KNN algorithm be used for regression problem statements?

- Yes, KNN can be used for regression problem statements.
- In other words, the KNN algorithm can be applied when the dependent variable is continuous. For regression problem statements, the predicted value is given by the average of the values of its k nearest neighbours.

Why is the KNN Algorithm known as Lazy Learner?

- When the KNN algorithm gets the training data, it does not learn and make a model, it just stores the data. Instead of finding any discriminative function with the help of the training data, it follows instance-based learning and also uses the training data when it actually needs to do some prediction on the unseen datasets.
- As a result, KNN does not immediately learn a model rather delays the learning thereby being referred to as Lazy Learner.

Why is it recommended not to use the KNN Algorithm for large datasets?

The Problem in processing the data: KNN works well with smaller datasets because it is a lazy learner. It needs to store all the data and then make a decision only at run time. It includes the computation of distances for a given point with all other points. So if the dataset is large, there will be a lot of processing which may adversely impact the performance of the algorithm.

Sensitive to noise: Another thing in the context of large datasets is that there is more likely a chance of noise in the dataset which adversely affects the performance of the KNN algorithm since the KNN algorithm is sensitive to the noise present in the dataset.

How to handle categorical variables in the KNN Algorithm?

- To handle the categorical variables we have to create dummy variables out of a categorical variable and include them instead of the original categorical variable. Unlike regression, create k dummies instead of (k-1).
- For example, a categorical variable named “Degree” has 5 unique levels or categories. So we will create 5 dummy variables. Each dummy variable has 1 against its degree and else 0.

How to choose the optimal value of K in the KNN Algorithm?

- There is no straightforward method to find the optimal value of K in the KNN algorithm.
- You have to play around with different values to choose which value of K should be optimal for my problem statement. Choosing the right value of K is done through a process known as Hyperparameter Tuning.

- The optimum value of K for KNN is highly dependent on the data itself. In different scenarios, the optimum K may vary. It is more or less a hit and trial method.
- There is no one proper method of finding the K value in the KNN algorithm. No method is the rule of thumb but you should try the following suggestions:
    - 1. Square Root Method: Take the square root of the number of samples in the training dataset and assign it to the K value.
    - 2. Cross-Validation Method: We should also take the help of cross-validation to find out the optimal value of K in KNN. Start with the minimum value of k i.e, K=1, and run cross-validation, measure the accuracy, and keep repeating till the results become consistent. As the value of K increases, the error usually goes down after each one-step increase in K, then stabilizes, and then raises again. Finally, pick the optimum K at the beginning of the stable zone. This technique is also known as the Elbow Method.
    - 3. Domain Knowledge: Sometimes with the help of domain knowledge for a particular use case we are able to find the optimum value of K (K should be an odd number).

How can you relate KNN Algorithm to the Bias-Variance tradeoff?

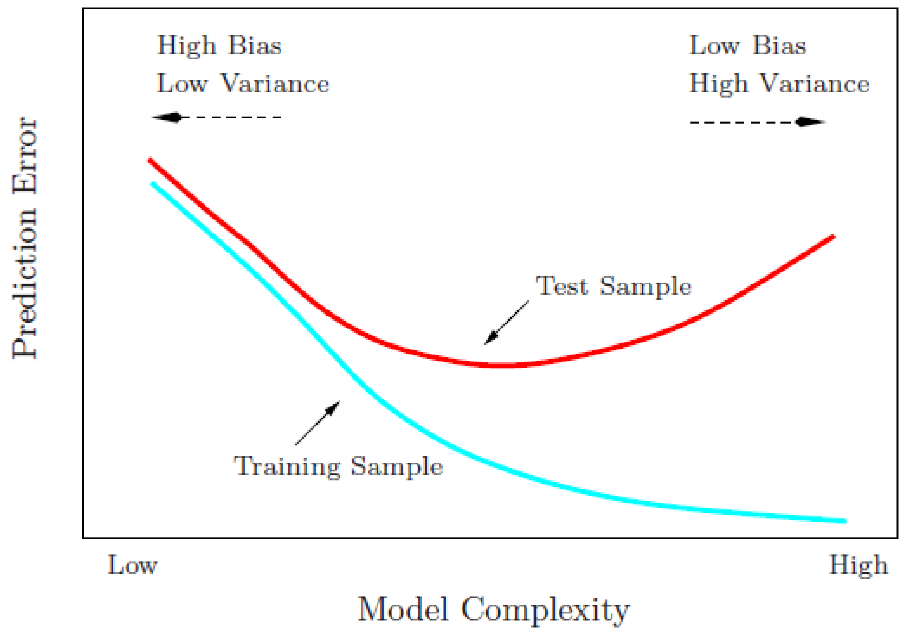
Problem with having too small K: The major concern associated with small values of K lies behind the fact that the smaller value causes noise to have a higher influence on the result which will also lead to a large variance in the predictions.

Problem with having too large K: The larger the value of K, the higher is the accuracy. If K is too large, then our model is under-fitted. As a result, the error will go up again. So, to prevent your model from under-fitting it should retain the generalization capabilities otherwise there are fair chances that your model may perform well in the training data but drastically fail in the real data. The computational expense of the algorithm also increases if we choose the k very large.

So, choosing k to a large value may lead to a model with a large bias(error).

The effects of k values on the bias and variance is explained below:

- As the value of k increases, the bias will be increases
- As the value of k decreases, the variance will increase
- With the increasing value of K, the boundary becomes smoother
- So, there is a tradeoff between overfitting and underfitting and you have to maintain a balance while choosing the value of K in KNN. Therefore, K should not be too small or too large.



Which algorithm can be used for value imputation in both categorical and continuous categories of data?

- KNN is the only algorithm that can be used for the imputation of both categorical and continuous variables. It can be used as one of many techniques when it comes to handling missing values.
- To impute a new sample, we determine the samples in the training set "nearest" to the new sample and averages the nearby points to impute. A Scikit learn library of Python provides a quick and convenient way to use this technique.
- Note: NaNs are omitted while distances are calculated. Hence we replace the missing values with the average value of the neighbours. The missing values will then be replaced by the average value of their "neighbours".

Explain the statement- "The KNN algorithm does more computation on test time rather than train time".

The above-given statement is absolutely true.

The basic idea behind the kNN algorithm is to determine a k-long list of samples that are close to a sample that we want to classify. Therefore, the training phase is basically storing a training set, whereas during the prediction stage the algorithm looks for k-neighbours using that stored data. Moreover, KNN does not learn anything from the training dataset as well.

What are the things which should be kept in our mind while choosing the value of k in the KNN Algorithm?

- If K is small, then results might not be reliable because the noise will have a higher influence on the result. If K is large, then there will be a lot of processing to be done which may adversely impact the performance of the algorithm.
- So, the following things must be considered while choosing the value of K:
- K should be the square root of n (number of data points in the training dataset).
- K should be chosen as the odd so that there are no ties. If the square root is even, then add or subtract 1 to it.

What are the advantages of the KNN Algorithm?

Some of the advantages of the KNN algorithm are as follows:

- No Training Period: It does not learn anything during the training period since it does not find any discriminative function with the help of the training data. In simple words, actually, there is no training period for the KNN algorithm. It stores the training dataset and learns from it only when we use the algorithm for making the real-time predictions on the test dataset. As a result, the KNN algorithm is much faster than other algorithms which require training. For Example, SupportVector Machines(SVMs), Linear Regression, etc.
- Moreover, since the KNN algorithm does not require any training before making predictions as a result new data can be added seamlessly without impacting the accuracy of the algorithm.
- Easy to implement and understand: To implement the KNN algorithm, we need only two parameters i.e. the value of K and the distance metric(e.g. Euclidean or Manhattan, etc.). Since both the parameters are easily interpretable therefore they are easy to understand.

What are the disadvantages of the KNN Algorithm?

- Does not work well with large datasets: In large datasets, the cost of calculating the distance between the new point and each existing point is huge which decreases the performance of the algorithm.
- Does not work well with high dimensions: KNN algorithms generally do not work well with high dimensional data since, with the increasing number of dimensions, it becomes difficult to calculate the distance for each dimension.
- Need feature scaling: We need to do feature scaling (standardization and normalization) on the dataset before feeding it to the KNN algorithm otherwise it may generate wrong predictions.
- Sensitive to Noise and Outliers: KNN is highly sensitive to the noise present in the dataset and requires manual imputation of the missing values along with outliers removal.

k-NN algorithm does more computation on test time rather than train time.

A) TRUE B) FALSE

Solution: A)The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the testing phase, a test point is classified by assigning the label which are most frequent among the k training samples nearest to that query point – hence higher computation.

Which of the following distance metric can not be used in k-NN?

A) Manhattan
B) Minkowski
C) Tanimoto
D) Jaccard
E) Mahalanobis
F) All can be used

Solution: F)All of these distance metric can be used as a distance metric for k-NN.

Which of the following statement is true about k-NN algorithm?

1. k-NN performs much better if all of the data have the same scale
2. k-NN works well with a small number of input variables (p), but struggles when the number of inputs is very large
3. k-NN makes no assumptions about the functional form of the problem being solved

A) 1 and 2

B) 1 and 3

C) Only 1

D) All of the above

Solution: D) The above mentioned statements are assumptions of kNN algorithm

Which of the following machine learning algorithm can be used for imputing missing values of both categorical and continuous variables?

A) K-NN

B) Linear Regression

C) Logistic Regression

Solution: A), k-NN algorithm can be used for imputing missing value of both categorical and continuous variables.

Which of the following is true about Manhattan distance?

A) It can be used for continuous variables

B) It can be used for categorical variables

C) It can be used for categorical as well as continuous

D) None of these

Solution: A) Manhattan Distance is designed for calculating the distance between real valued features.

Which of the following distance measure do we use in case of categorical variables in k-NN?

Hamming Distance

Euclidean Distance

Manhattan Distance

A) 1

B) 2

C) 3

D) 1 and 2

E) 2 and 3

F) 1,2 and 3

Solution: A) Both Euclidean and Manhattan distances are used in case of continuous variables, whereas hamming distance is used in case of categorical variable.

Which of the following will be Euclidean Distance between the two data point A(1,3) and B(2,3)?

A) 1

B) 2

C) 4

D) 8

Solution: A), sqrt((1-2)^2 + (3-3)^2) = sqrt(1^2 + 0^2) = 1

Which of the following will be Manhattan Distance between the two data point A(1,3) and B(2,3)?

A) 1

B) 2

C) 4

D) 8

Solution: A) sqrt(mod((1-2)) + mod((3-3))) = sqrt(1 + 0) = 1

A company has build a kNN classifier that gets 100% accuracy on training data. When they deployed this model on client side it has been found that the model is not at all accurate. Which of the following thing might gone wrong? Note: Model has successfully deployed and no technical issues are found at client side except the model performance

A) It is probably a overfitted model

B) It is probably a underfitted model

C) Can't say

D) None of these

Solution: A), In an overfitted module, it seems to be performing well on training data, but it is not generalized enough to give the same results on a new data.

You have given the following 2 statements, find which of these option is/are true in case of k-NN?

1. In case of very large value of k, we may include points from other classes into the neighborhood.
2. In case of too small value of k the algorithm is very sensitive to noise

A) 1

B) 2

C) 1 and 2

D) None of these

Solution: C), Both the options are true and are self explanatory.

Which of the following statements is true for k-NN classifiers?

A) The classification accuracy is better with larger values of k

B) The decision boundary is smoother with smaller values of k

C) The decision boundary is linear

D) k-NN does not require an explicit training step

Solution: D)

True-False: It is possible to construct a 2-NN classifier by using the 1-NN classifier?

A) TRUE

B) FALSE

Solution: A) You can implement a 2-NN classifier by ensembling 1-NN classifiers

In k-NN what will happen when you increase/decrease the value of k?

A) The boundary becomes smoother with increasing value of K

B) The boundary becomes smoother with decreasing value of K

C) Smoothness of boundary doesn't dependent on value of K

D) None of these

Solution: A) The decision boundary would become smoother by increasing the value of K

Following are the two statements given for k-NN algorthm, which of the statement(s) is/are true?

1. We can choose optimal value of k with the help of cross validation
2. Euclidean distance treats each feature as equally important

A) 1

B) 2

C) 1 and 2

D) None of these

Solution: C) Both the statements are true

Suppose, you have trained a k-NN model and now you want to get the prediction on test data. Before getting the prediction suppose you want to calculate the time taken by k-NN for predicting the class for test data.

Note: Calculating the distance between 2 observation will take D time.

What would be the time taken by 1-NN if there are N(Very large) observations in test data?

A) N*D

B) N*D*2

C) (N*D)/2

D) None of these

Solution: A) The value of N is very large, so option A is correct

What would be the relation between the time taken by 1-NN,2-NN,3-NN.

A) 1-NN >2-NN >3-NN

B) 1-NN < 2-NN < 3-NN

C) 1-NN ~ 2-NN ~ 3-NN

D) None of these

Solution: C) The training time for any value of k in kNN algorithm is the same.

Why should we not use the KNN algorithm for large datasets?

Here is an overview of the data flow that occurs in the KNN algorithm:

- Calculate the distances to all vectors in a training set and store them
- Sort the calculated distances
- Store the K nearest vectors
- Calculate the most frequent class displayed by K nearest vectors

Imagine you have a very large dataset. Therefore, it is not only a bad decision to store a large amount of data but it is also computationally costly to keep calculating and sorting all the values.

The k-NN algorithm does more computation on test time rather than train time.

That is absolutely true. The idea of the kNN algorithm is to find a k-long list of samples that are close to a sample we want to classify. Therefore, the training phase is basically storing a training set, whereas while the prediction stage the algorithm looks for k-neighbours using that stored data.

Why do you need to scale your data for the k-NN algorithm?

Imagine a dataset having m number of “examples” and n number of “features”. There is one feature dimension having values exactly between 0 and 1. Meanwhile, there is also a feature dimension that varies from -99999 to 99999. Considering the formula of Euclidean Distance, this will affect the performance by giving higher weightage to variables having a higher magnitude.

The k-NN algorithm can be used for imputing the missing value of both categorical and continuous variables.

That is true. k-NN can be used as one of many techniques when it comes to handling missing values. A new sample is imputed by determining the samples in the training set “nearest” to it and averages these nearby points to impute. A scikit learn library provides a quick and convenient way to use this technique.

Note: NaNs are omitted while distances are calculated.

Example:

```
from sklearn.impute import KNNImputer
# define imputer
imputer = KNNImputer() #default k is 5=> n_neighbors=5
# fit on the dataset
imputer.fit(X)
# transform the dataset
Xtrans = imputer.transform(X)
```

Thus, missing values will be replaced by the mean value of its “neighbours”.

Is Euclidean Distance always the case?

Although Euclidean Distance is the most common method used and taught, it is not always the optimal decision. In fact, it is hard to come up with the right metric just by looking at data, so I would suggest trying a set of them. However, there are some special cases. For instance, hamming distance is used in case of a categorical variable.

What is a Naïve Bayes Classifier?

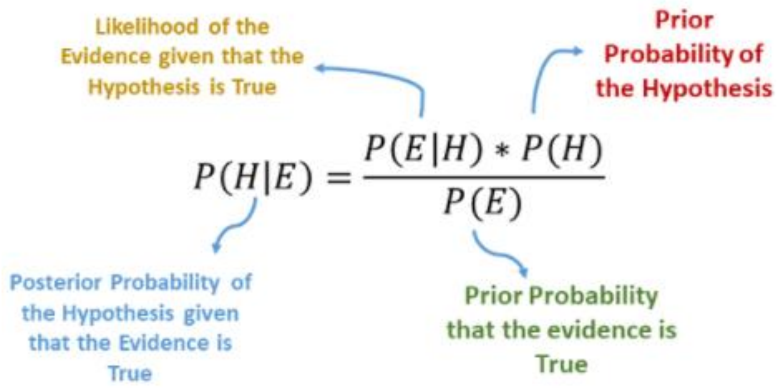
Naive Bayes Classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong independence assumptions between the features.

Bayes' theorem is given by the following equation:

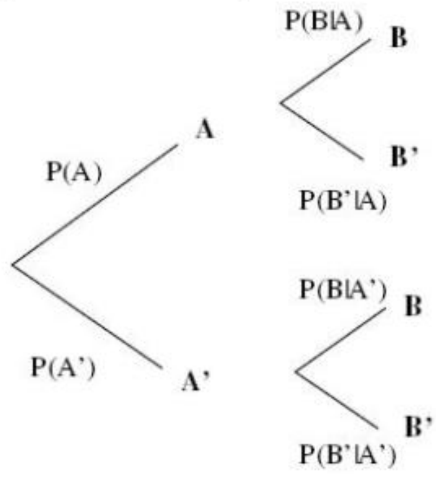
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using Bayes' theorem, the probability of A happening given that B has occurred can be found.

An example of the way a Naive Bayes Classifier can be used is, given that it has rained, the probability of temperature being low is P(Temperature|Rain).


Likelihood of the Evidence given that the Hypothesis is True
Prior Probability of the Hypothesis
$P(H|E) = \frac{P(E|H) * P(H)}{P(E)}$
Posterior Probability of the Hypothesis given that the Evidence is True
Prior Probability that the evidence is True

Bayes' Theorem



Using the tree diagram:

- $P(A \text{ and } B) = P(A) \times P(B \mid A)$

This is the multiplication law of probability

Rearranging gives:

- $P(A \mid B) = \dfrac{P(A \text{ and } B)}{P(B)}$
- $P(B \mid A) = \dfrac{P(A \text{ and } B)}{P(A)}$

Why Naive Bayes is called Naive?

We call it naive because its assumptions (it assumes that all of the features in the dataset are equally important and independent) are really optimistic and rarely true in most real-world applications:

1. we consider that these predictors are independent
2. we consider that all the predictors have an equal effect on the outcome (like the day being windy does not have more importance in deciding to play golf or not)

How would you use Naive Bayes classifier for categorical features? What if some features are numerical?

We can use any kind of predictor in a Naive Bayes classifier. All we need is the conditional probability of a feature given the class, i.e., P(F | Class).

- For the categorical features, we can estimate P(F | Class) using a distribution such as multinomial or Bernoulli.
- For the numerical features, we can estimate P(F | Class) using a distribution such as Normal or Gaussian.
- For a numerical feature that follows a different specific distribution, for example, an exponential, then that specific distribution may be used instead.
- For a numerical or categorical without a well-defined distribution, then a kernel density estimator can be used to estimate the probability distribution instead.