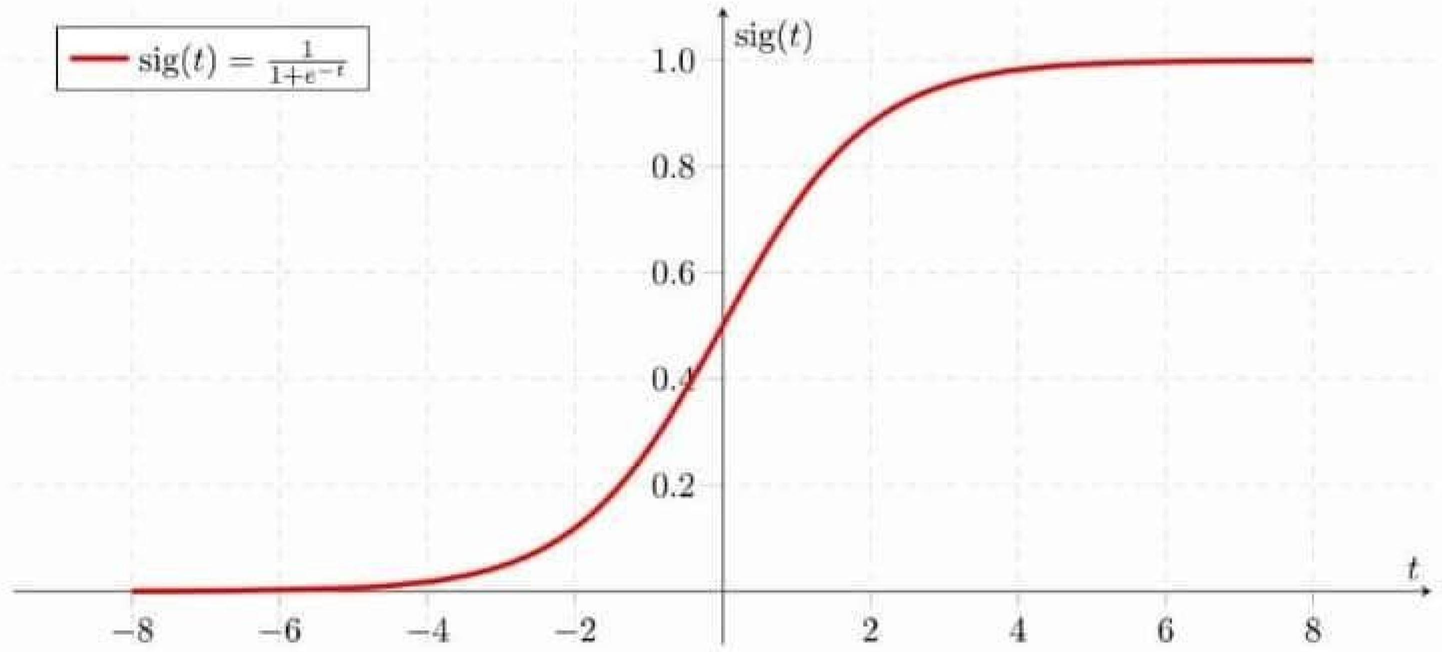
Logistic Regression

- By name, everyone thinks this is a regression algorithm but not.
- Logistic Regression is used when the dependent variable(target) is categorical.
- logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.
- sigmoid function = 1/(1+e(-x))
- It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

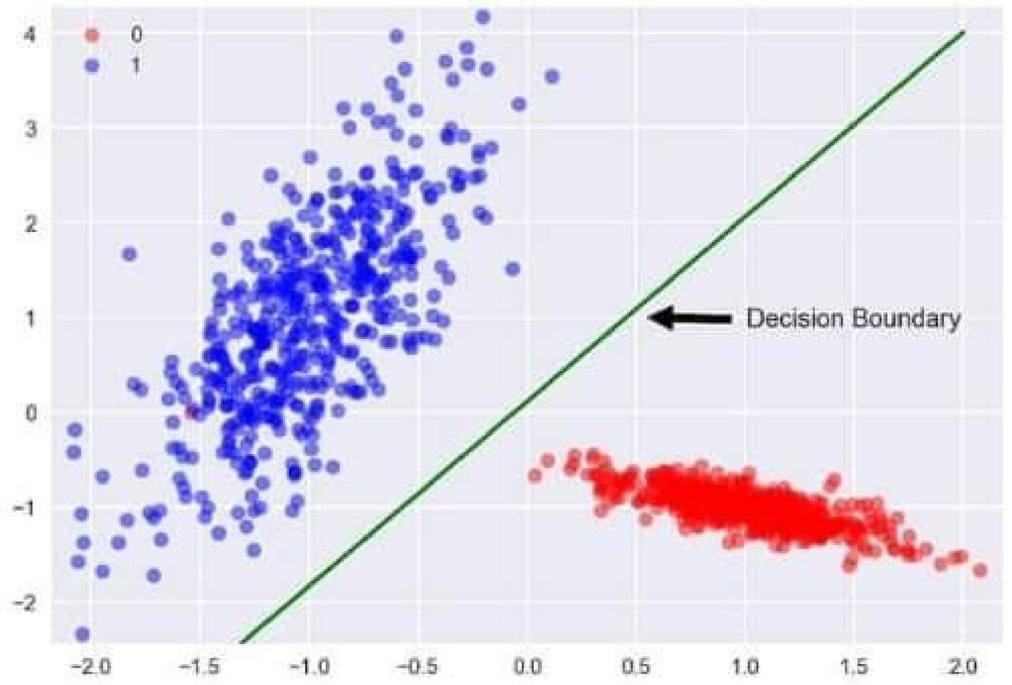


Logistic Regression

- You can keep your own threshold value as a separator for predicted classes.
- So here the sigmoid returns a number between 0 and 1.
- For example, if our threshold was .5 and our prediction function returned .7, we would classify this observation as positive. If our prediction was .2 we would classify the observation as negative.
- For logistic regression with multiple classes, we could select the class with the highest predicted probability.
- If we have multiple classes more than 2 then we build multiple logistic regression models with 2 classes and pics the highest avg probability as a class.
- This is just an outline of how an algorithm works. from the next post, I will teach in detail with Math.

Logistic Regression

- Let's see why we are using a sigmoid function instead of just how we are doing linear regression.



- The algorithm assumes that data should be almost separable.
- So we will draw a plan to separate the 2 classes like blue points and red points as shown in the above figure. And the green line is the decision boundary.
- The equation can be written as W*T x X + b. W is the weight vector or coefficient vector, X is the input values and b is the intersect.

Logistic Regression

- So how do we find the line which separates the data?
- We find the distance between the point and the plane and if we get W^T x X > 0 then it is correctly classified or else incorrectly classified.
- For example, we have seen data in the last figure where blue points are positive and red points are negative and of course, we will have some misclassified points in the training dataset.
- So here we check for the accuracy by sum(yi x W^T x Xi > 0) for all points. if the point is correctly classified by the line we get yi x W^T x Xi > 0 as positive or else we get as yi x W^T x Xi < 0. yi is the original label.
- let's check a few cases if the point is positive and it is above the plain. so the distance between the point and plan will be positive and the label is positive so we get the result as > 0. if the point is negative and the point lies below the plane and we get distance as a negative and original label is negative and we get result as > 0

Logistic Regression

- If the point is positive and lies below the line so we get the result as < 0 which is misclassified and if the point is negative and lies above the line we get the result as < 0 which is also misclassified.
  So here we want to maximize the output. if every point is correctly classified we get highest score sum(yi x W^T x Xi).
  We find the line using gradient descent as how we find the line in linear regression.
  So the problem with this method is when we have extreme outliers. which will change the shape of the line because we can get high value as output because of one extreme point and we think this the perfect line but intuitively we misclassified most of the points.
  To solve this problem we introduce a concept called sigmoid and a loss function to calculate the loss of a model.

Methods to avoid Over-fitting

Lasso Regression

- Lasso, or Least Absolute Shrinkage and Selection Operator, is quite similar conceptually to ridge regression. It also adds a penalty for non-zero coefficients, but unlike ridge regression which penalizes sum of squared coefficients (the so-called L2 penalty), lasso penalizes the sum of their absolute values (L1 penalty).
- As a result, for high values of λ, many coefficients are exactly zeroed under lasso, which is never the case in ridge regression.
- The only difference in ridge and lasso loss functions is in the penalty terms. Under lasso, the loss is defined as
- Loss = OLS + λ|sum(Coefficients)|.
- group of predictors are highly correlated, lasso picks only one of them and shrinks the others to zero

Methods to avoid Over-fitting

Elastic Net

- ElasticNet is a hybrid of Lasso and Ridge Regression techniques. It is trained with L1 and L2 prior as regularizer. Elastic-net is useful when there are multiple features which are correlated. Lasso is likely to pick one of these at random, while elastic-net is likely to pick both.
- Loss = OLS + λ1|sum(Coefficients)| + λ2||Coefficients)||
- A practical advantage of trading-off between Lasso and Ridge is that it allows Elastic-Net to inherit some of Ridge's stability under rotation.
- It encourages group effect in case of highly correlated variables
- There are no limitations on the number of selected variables
- It can suffer from double shrinkage

Methods to avoid Over-fitting

Regularization

- This is a form of regression, that constrains / regularizes or shrinks the coefficient estimates towards zero. In other words, this technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting.
- Lets take a simple linear regression equation of multivariate.
- Y = a1X1 + a2X2 + a3X3 + ...
- So here we fit a model to get the best fit line which has the low error value (R square or mean square).
- Here a1, a2, a3 ..... are the coefficients and X1, X2, X3,..... are data points where we we estimate a1, a2 , a3 …based on an objective function(loss function).
- Now, this will adjust the coefficients based on your training data. If there is noise in the training data, then the estimated coefficients won't generalize well to the future data.
- Now, this optimization might simply overfit the equation if x1 , x2 , x3 (independent variables) are too many in numbers.
- This is where regularization comes in and shrinks or regularizes these learned estimates towards zero.

Methods to avoid Over-fitting

Stratified k-fold cross validation

- Stratification is the process of rearranging the data so as to ensure that each fold is a good representative of the whole. For example, in a binary classification problem where each class comprises of 50% of the data, it is best to arrange the data such that in every fold, each class comprises of about half the instances.
- It is generally a better approach when dealing with both bias and variance. A randomly selected fold might not adequately represent the minor class, particularly in cases where there is a huge class imbalance.
- The mean response value is approximately equal in all the folds is another way of saying the proportion of each class in all the folds are approximately equal.

Methods to avoid Over-fitting

Cross-Validation

- Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.
- In this method we will stop one set of samples as validation and other samples as testing. Ans we do that K times (k is the only parameter in k-fold cross validation)
- It is used to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.
- few common methods used for Cross Validation

  1. validation set approach
  2. Leave one out cross validation (LOOCV)
  3. k-fold cross validation
  4. Stratified k-fold cross validation
  5. Adversarial Validation
  6. Cross Validation for time series

Methods to avoid Over-fitting

validation set approach

- In this approach, we reserve 50% of the dataset for validation and the remaining 50% for model training. However, a major disadvantage of this approach is that since we are training a model on only 50% of the dataset, there is a huge possibility that we might miss out on some interesting information about the data which will lead to a higher bias (underfitting).

Leave one out cross validation (LOOCV)

- Here we use only one data point in complete data set as a validation and remaining data as training.
- We do this iteratively for every data point.
- We use all data points so we can reduce the bias, but it takes lot of time to execute because we iterate this N(size of dataset) time.
- if we leave out few point as P instead of one point we call it as LPOCV

Methods to avoid Over-fitting

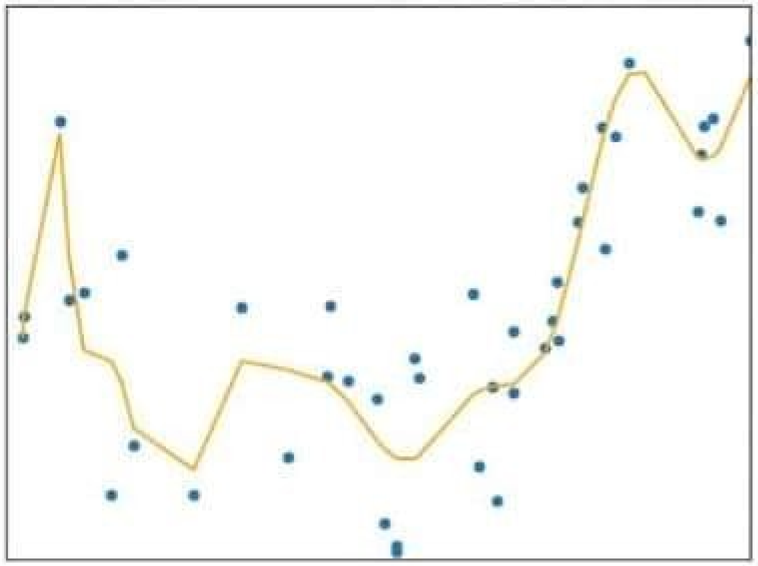
k-fold cross validation

- In the last two approaches we used to have high bias and high variance issues.
- So we can remove them here in this approach.
- The procedure for doing this method is as follows.

1. Shuffle the dataset randomly
2. Randomly split your entire dataset into k”folds”
3. For each k-fold in your dataset, build your model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for kth fold
4. Record the error you see on each of the predictions
5. Repeat this until each of the k-folds has served as the test set.
6. The average of your k recorded errors is called the cross-validation error and will serve as your performance metric for the model.

To choose the right value of K it depends on many parameters like dataset size, test and validation dataset distribution, maintaining the balance between bias and variance as small K leads to high bias and large k leads to high variance.

Polynomial regression



- What if the distribution of the data was more complex like the above picture
- We can't use linear models to get good accuracy. which also leads the model to underfit.
- We need to generate the curve to fit the data which w we can't do with linear models.
- Most of the time the equations look like

  $Y=\theta_0 + \theta_1 X + \theta_2 X^2 + \dots + \theta_m X^m$ + residual error

Polynomial regression

- We use the same methods to find the weight vector of how we do in linear regression.
- While there might be a temptation to fit a higher degree polynomial to get a lower error, this can result in over-fitting.
- We also use L1 and L2 regularizers to maintain the balance between bias(underfit) and variance(overfit)

Polynomial regression

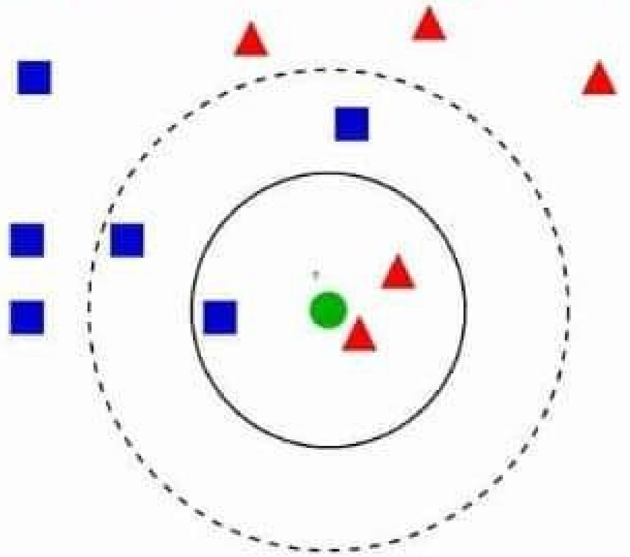
Advantages of using Polynomial Regression:

- Polynomial basically fits a wide range of curvature.
- Polynomial provides the best approximation of the relationship between the dependent and independent variable.
- A Broad range of function can be fit under it.

DisAdvantages of using Polynomial Regression:

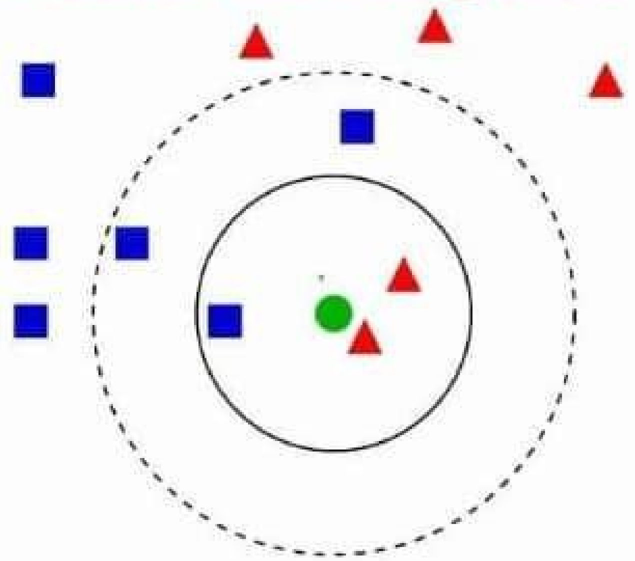
- These are too sensitive to the outliers.
- The presence of one or two outliers in the data can seriously affect the results of nonlinear analysis.
- In addition, there are unfortunately fewer model validation tools for the detection of outliers in nonlinear regression than there are for linear regression.

K Nearest Neighbors



- The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement and can be used to solve both classification and regression problems.
- Let's take an example and learn this algorithm.
- You can see we have blue squares as one class and a red triangle as another class. And now we need to find the green circle belongs to which class?
- Here comes the K in KNN it is used to find the K nearest neighbors for that green circle and take the majority class label and assign it to the green circle.

K Nearest Neighbors



- Let's take K = 3 and you can see that we have 2 red triangles and 1 blue square as the nearest neighbors to green circle and we take the majority and assign green circle as a red triangle.
- When we take K = 5 then you can see it belongs to the blue square.
- So how to find the nearest neighbors?
- KNN works based on a similarity measure (e.g., distance functions)
- There are multiple distance functions like Euclidean, Minkowski, Manhattan etc..

Imbalanced vs balanced dataset

- Classification is one of the most common machine learning problems. One of the common issues found in datasets that are used for classification is imbalanced classes issue.
- Imbalanced dataset means when we have unequal distribution of classes.
- For example, let's consider a dataset called anomaly detection on AIOPS data(server data). Mostly we have 2 classes like anomaly and not anomaly.
- whenever you see this kind of data we used to have a lot of non-anomolous data and very few anomaly data.
- This leaves us with something like 50:1 ratio between the anomaly and non-anomaly classes.

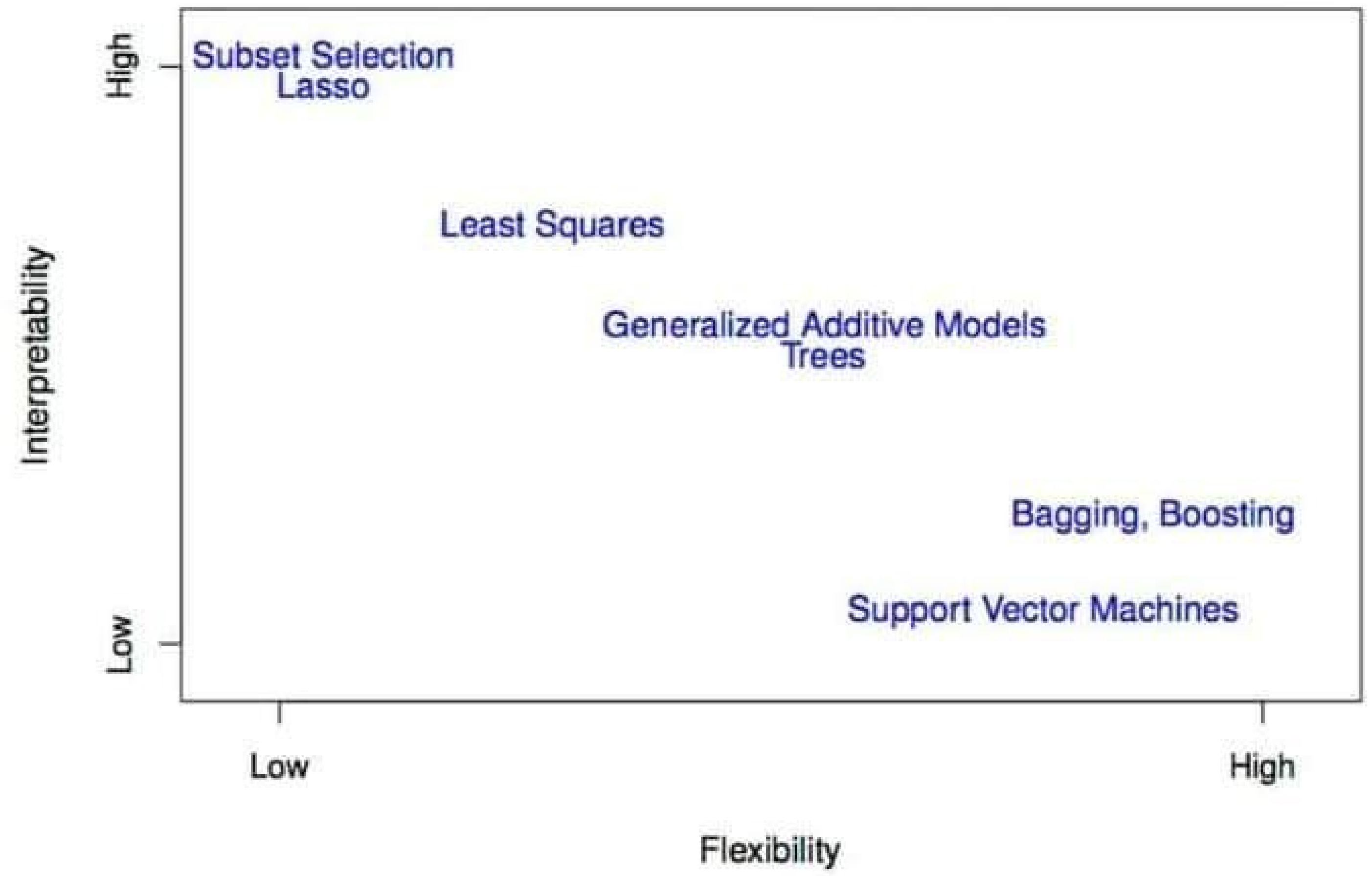
Imbalanced vs balanced dataset

- If we train a binary classification model without fixing this problem, the model will be completely biased and mostly it always predicts the majority class.
- The features of the minority class are treated as noise and are often ignored. Thus, there is a high probability of misclassification of the minority class as compared to the majority class.
- Let's see how to fix this in the next series of posts.

Statistical learning

- Non-parametric Methods: It do not make explicit assumptions about the functional form of f.
- Instead they seek an estimate of f that gets as close to the data points as possible without being too rough or wiggly.
- Such approaches can have a major advantage over parametric approaches: by avoiding the assumption of a particular functional form for f, they have the potential to accurately fit a wider range of possible shapes for f.

Trade-Off Between Prediction Accuracy and Model Interpretability



- When interpretability is the goal, there are clear advantages to using simple and relatively inflexible statistical learning methods and vice versa.

Decision trees

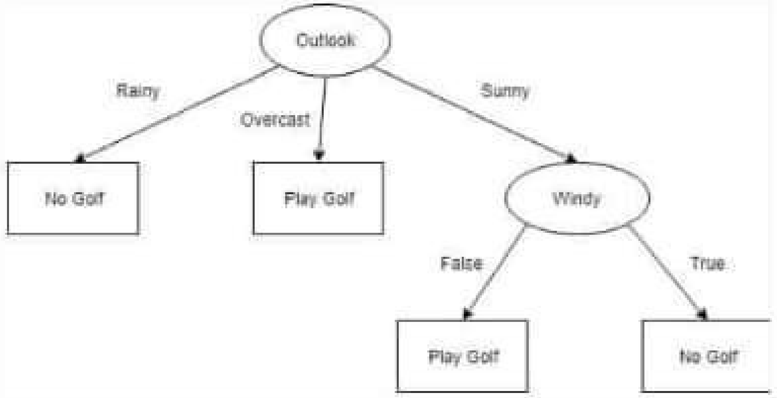
- Advantages
- **Easy to Understand:** Decision tree output is very easy to understand even for people from the non-analytical background. It does not require any statistical knowledge to read and interpret them. Its graphical representation is very intuitive and users can easily relate their hypothesis.
- **Useful in Data exploration:** Decision tree is one of the fastest ways to identify the most significant variables and the relation between two or more variables. With the help of decision trees, we can create new variables/features that have a better power to predict the target variable. You can refer article (Trick to enhance the power of regression model) for one such trick. It can also be used in the data exploration stage. For example, we are working on a problem where we have information available in hundreds of variables, their decision tree will help to identify the most significant variable.
- **Less data cleaning required:** It requires less data cleaning compared to some other modelling techniques. It is not influenced by outliers and missing values to a fair degree.

Decision trees

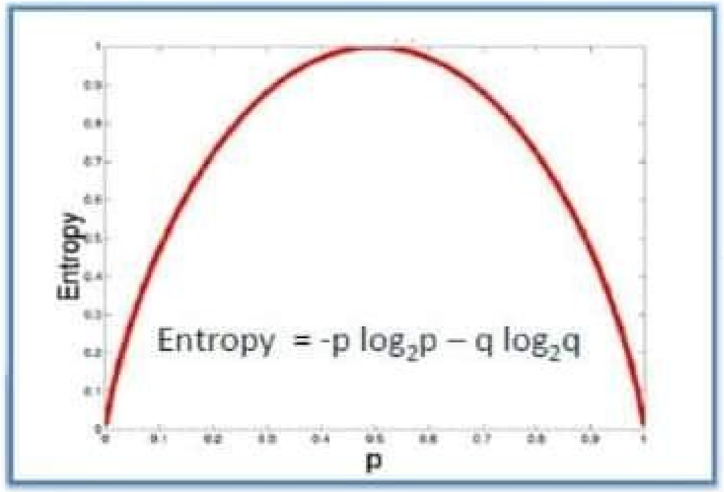
- Advantages
- data type is not a constraint: It can handle both numerical and categorical variables.
- Non-Parametric Method: Decision tree is considered to be a non-parametric method. This means that decision trees have no assumptions about space distribution and the classifier structure.
- DisAdvantages
- Overfitting: Overfitting is one of the most practical difficulties for decision tree models. This problem gets solved by setting constraints on model parameters and pruning.
- Not fit for continuous variables: While working with continuous numerical variables, decision tree loses information when it categorizes variables in different categories.

Decision trees

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes



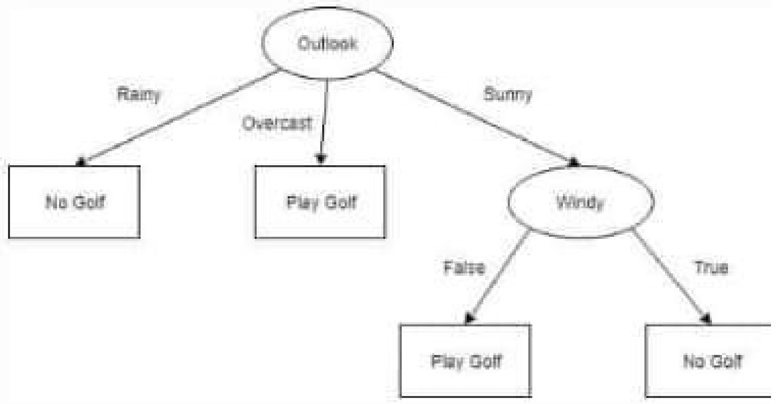
- So the first step is to calculate the entropy of dataset.
- Let's say we have binary classification problem with 6 samples as "YES" and 4 samples as "NO".
- the entropy of dataset will be -(6/10)*log2(6/10) - (4/10)*log2(6/10) = 0.73
- The entropy of dataset is 0.73
- if we have equal samples let say we have 5 as YES and 5 as NO. The entropy will be 1.
- Entropy controls how a Decision Tree decides to split the data. It actually effects how a Decision Tree draws its boundaries.



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

Decision trees

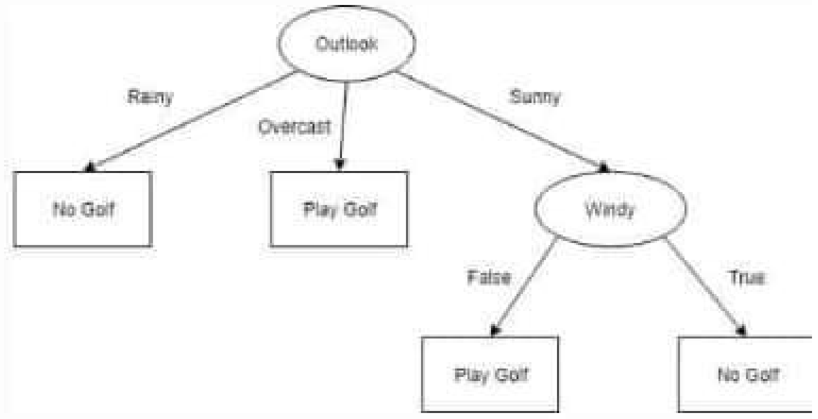
Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes



- The next step is to calculate the entropy for each feature. Then it is added proportionally, to get total entropy for the split (weighted entropy).
- The resulting entropy is subtracted from the entropy before the split(parent). The result is the Information Gain and we try to maximize this at every point.
- First, we calculate the IG when outlook feature is used to split the dataset. If the feature is categorical we split using the categories. If it is continuous there are many methods to split, we discuss this later.
- Entropy for the category(Rainy) -(3/4)*log2(3/4)-(1/4)*log2(1/4) is 0.81 as we have 3 NO samples for rainy and 1 YES sample for rainy

Decision trees

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes



- Entropy for the category(Overcast) -(2/2)*log2(2/2)-(0/2)*log2(0/2) is 0 as we have 2 YES samples and 0 NO samples.
- Entropy for the category(sunny) -(3/4)*log2(3/4)-(1/4)*log2(1/4) is 0.81 as we have 3 YES samples and 1 NO sample.
- final entropy of outlook will be (4/10)*0.81 + 2(10)*0+(4/10)*0.81 = 0.648.
- let's calculate the entropy of other features.
- The entropy of Temperature (3/10)*0.91+ (3/10)*0.91+ (4/10)*0.81 = 0.87
- The entropy of Humidity (5/10)*0.97 + (5/10)*0.72 = 0.845
- The entropy of windy (7/10)*0.86 + (3/10)* = 0.91

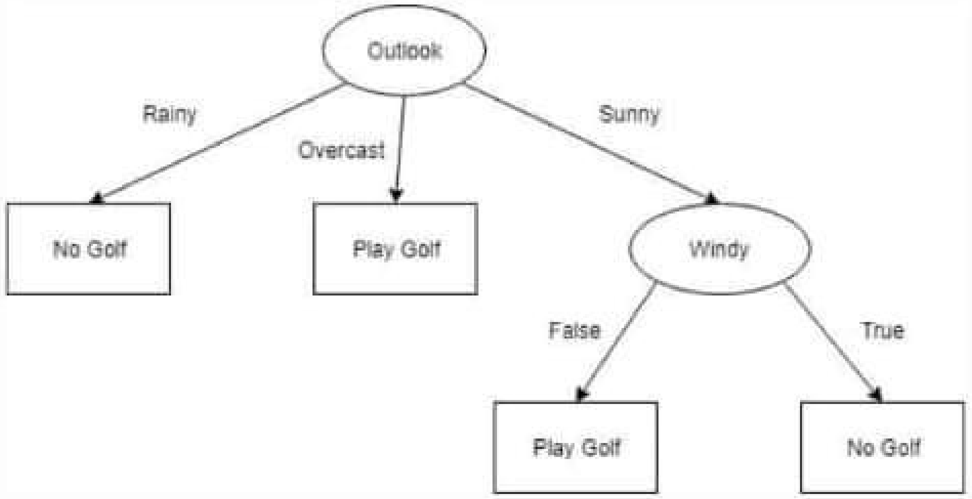
Decision trees

- So how does a Decision tree works?
- The important things you need to know before learning how decision tree works are Entropy and Information gain.
- Let's take a common example to lean how DT works.

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes

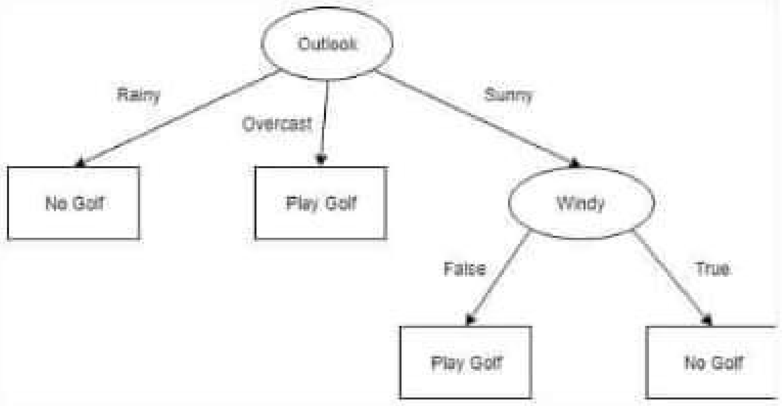
Say we have the above dataset, and would like to predict if we would play golf or not given the weather.

The decision tree for this problem might look like the one below.



Decision trees

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes



- Information gain of the features will be root entropy - weighted entropy.
- !G for outlook is 0.73 - 0.648 = 0.082
- IG for temperature is 0.73 - 0.87 = -0.14
- IG for humidity is 0.73 - 0.845 = -0.115
- IG for windy is 0.73 - 0.91 = -0.18
- So the highest IG is for Outlook feature and we use this feature to split the dataset as you can see in the above decision tree because we are gaining more information through this split.
- A branch with the entropy of 0 is a leaf node(overcast).
- And in the next level we stopped splitting the rainy and overcast this is because of pruning.
- And we do the same for the next iterations. As depth increases, it leads to overfitting as we are creating too complex rules.