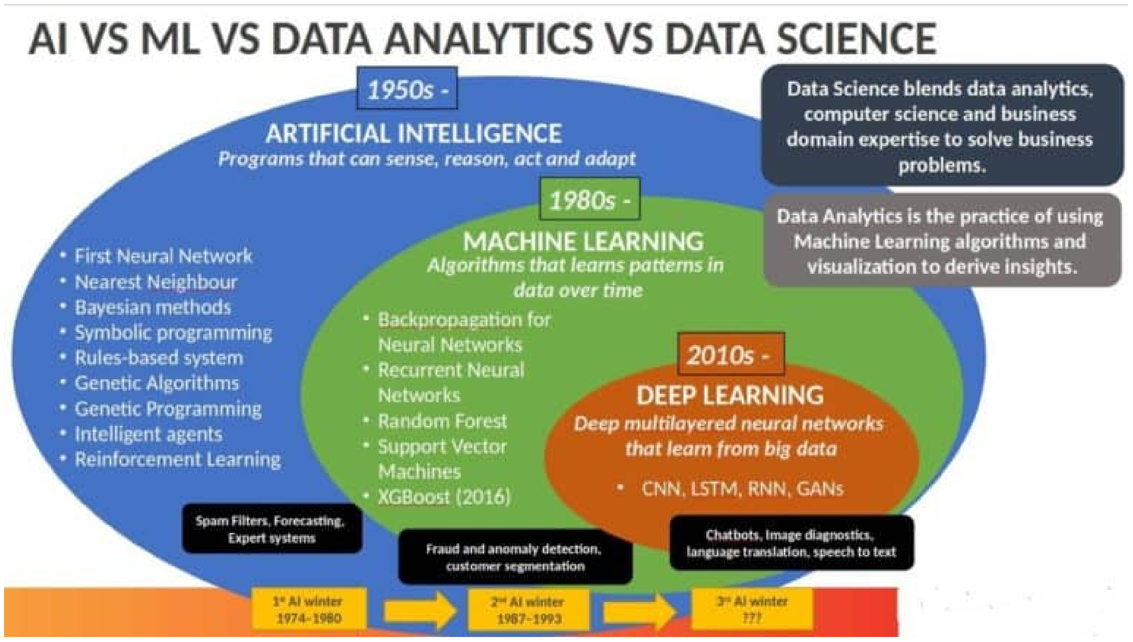
Contents

Day 1

July 31, 2020

1 What is the difference between AI, Data Science, ML, and DL?

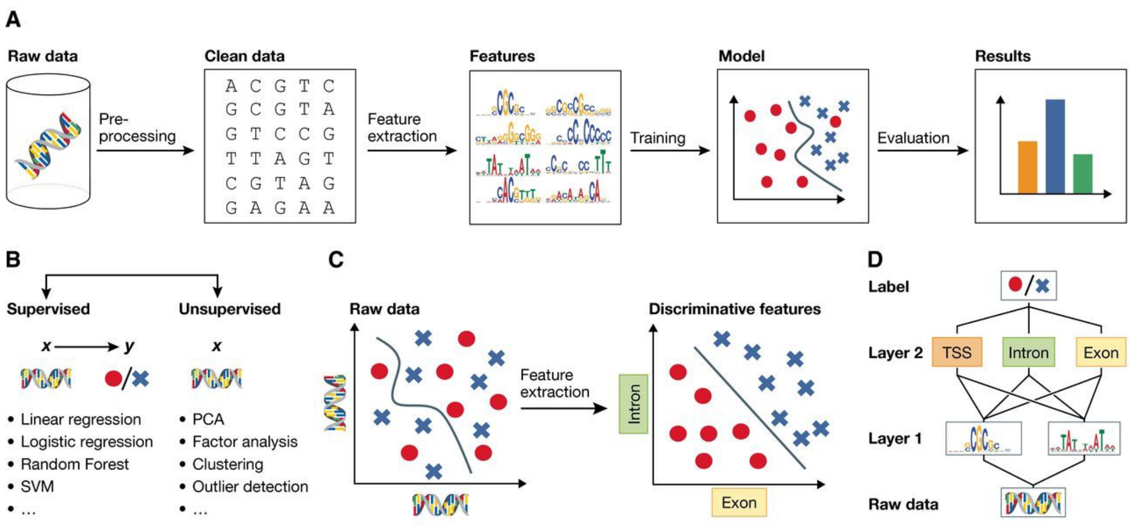

AI VS ML VS DATA ANALYTICS VS DATA SCIENCE
1950s -
ARTIFICIAL INTELLIGENCE
Programs that can sense, reason, act and adapt
• First Neural Network
• Nearest Neighbour
• Bayesian methods
• Symbolic programming
• Rules-based system
• Genetic Algorithms
• Genetic Programming
• Intelligent agents
• Reinforcement Learning
1980s -
MACHINE LEARNING
Algorithms that learns patterns in data over time
• Backpropagation for Neural Networks
• Recurrent Neural Networks
• Random Forest
• Support Vector Machines
• XGBoost (2016)
2010s -
DEEP LEARNING
Deep multilayered neural networks that learn from big data
• CNN, LSTM, RNN, GANs
Data Science blends data analytics, computer science and business domain expertise to solve business problems.
Data Analytics is the practice of using Machine Learning algorithms and visualization to derive insights.
Spam Filters, Forecasting, Expert systems
Fraud and anomaly detection, customer segmentation
Chatbots, Image diagnostics, language translation, speech to text
1st AI winter 1974-1980
2nd AI winter 1987-1993
3rd AI winter ???

1.1 Artificial Intelligence:

AI is purely math and scientific exercise, but when it became computational, it started to solve human problems formalized into a subset of computer science. Artificial intelligence has changed the original computational statistics paradigm to the modern idea that machines could mimic actual human capabilities, such as decision making and performing more "human" tasks. Modern AI into two categories 1. General AI - Planning, decision making, identifying objects, recognizing sounds, social & business transactions 2. Applied AI - driver-less Autonomous car or machine smartly trade stocks

1.2 Machine Learning:

Instead of engineers *teaching* or programming computers to have what they need to carry out tasks, that perhaps computers could teach themselves – learn something without being explicitly programmed to do so. ML is a form of AI where based on more data, and they can change actions and response, which will make more efficient, adaptable and scalable. e.g., navigation apps and recommendation engines.



1.3 Data Science:

Data science has many tools, techniques, and algorithms called from these fields, plus others –to handle big data

The goal of data science, somewhat similar to machine learning, is to make accurate predictions and to automate and perform transactions in real-time, such as purchasing internet traffic or automatically generating content. on math and coding and more on data and building new systems to process the data. Relying on the fields of data integration, distributed architecture, automated machine learning, data visualization, data engineering, and automated data-driven decisions, data science can cover an entire spectrum of data processing, not only the algorithms or statistics related to data.

Deep Learning: It is a technique for implementing ML.

ML provides the desired output from a given input, but DL reads the input and applies it to another data. In ML, we can easily classify the flower based upon the features. Suppose you want a machine to look at an image and determine what it represents to the human eye, whether a face, flower, landscape, truck, building, etc.

Machine learning is not sufficient for this task because machine learning can only produce an output from a data set – whether according to a known algorithm or based on the inherent structure of the data. You might be able to use machine learning to determine whether an image was of an "X" – a flower, say – and it would learn and get more accurate. But that output is binary (yes/no) and is dependent on the algorithm, not the data. In the image recognition case, the outcome is not binary and not dependent on the algorithm.

The neural network performs MICRO calculations with computational on many layers. Neural networks also support weighting data for 'confidence'. These results in a probabilistic system, vs. deterministic, and can handle tasks that we think of as requiring more 'human-like' judgment.

2 difference between supervised , unsupervised and Reinforcement learning?

Ans 2: Machine learning is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead.

Building a model by learning the patterns of historical data with some relationship between data to make a data-driven prediction.

Types of Machine Learning:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

2.1 Supervised learning

In a supervised learning model, the algorithm learns on **labeled dataset**, to generate reasonable predictions for the response to new data. (Forecasting outcome of new data)

- Regression
- Classification

2.2 Unsupervised learning

An unsupervised model, in contrast, provides **unlabelled data** that the algorithm tries to make sense of by extracting features, co-occurrence and underlying patterns on its own. We use unsupervised learning for

- Clustering
- Anomaly detection
- Association
- Autoencoders
- Reinforcement Learning

2.3 Reinforcement learning

Reinforcement learning is less supervised and depends on the learning agent in determining the output solutions by arriving at different possible ways to achieve the best possible solution.

What is Machine Learning? Two definitions of Machine Learning are offered. Arthur Samuel described it as: "the field of study that gives computers the ability to learn without being explicitly programmed." This is an older, informal definition.

Tom Mitchell provides a more modern definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Example: playing checkers.

E = the experience of playing many games of checkers

T = the task of playing checkers.

P = the probability that the program will win the next game.

In general, any machine learning problem can be assigned to one of two broad classifications:

Supervised learning and Unsupervised learning.

In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output.

Supervised learning problems are categorized into "regression" and "classification" problems. In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories.

Example 1:

Given data about the size of houses on the real estate market, try to predict their price. Price as a function of size is a continuous output, so this is a regression problem.

We could turn this example into a classification problem by instead making our output about whether the house "sells for more or less than the asking price." Here we are classifying the houses based on price into two discrete categories.

Example 2:

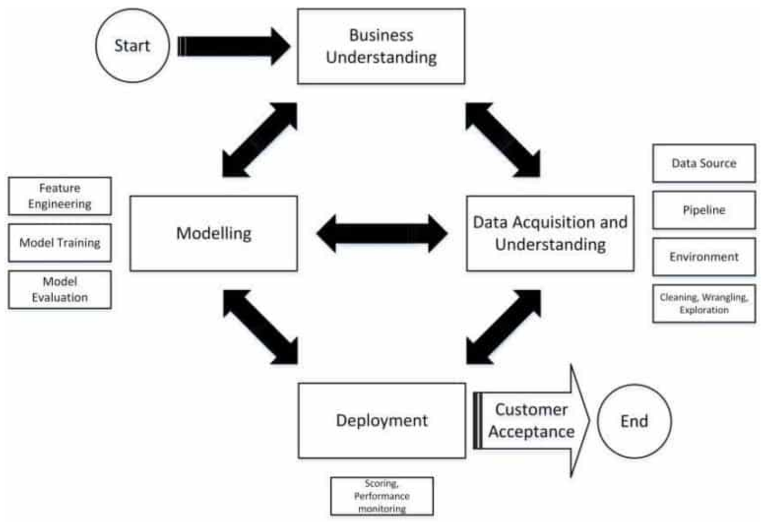
(a) Regression - Given a picture of a person, we have to predict their age on the basis of the given picture

(b) Classification - Given a patient with a tumor, we have to predict whether the tumor is malignant or benign.

Unsupervised Learning Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

We can derive this structure by clustering the data based on relationships among the variables in the data.

3 Q3. Describe the general architecture of Machine learning.



3.1 Business understanding:

Understand the give use case, and also, it's good to know more about the domain for which the use cases are built.

3.2 Data Acquisition and Understanding:

Data gathering from different sources and understanding the data. Cleaning the data, handling the missing data if any, data wrangling, and EDA(Exploratory data analysis).

3.3 Modeling:

Feature engineering – scaling the data, feature selection – not all features are important. We use the backward elimination method, correlation factors, PCA and domain knowledge to select the features.

Model Training based on trial and error method or by experience, we select the algorithm and train with the selected features.

Model evaluation Accuracy of the model, confusion matrix and cross-validation.

If accuracy is not high, to achieve higher accuracy, we tune the model…either by changing the algorithm used or by feature selection or by gathering more data, etc.

3.4 Deployment:

Once the model has good accuracy, we deploy the model either in the cloud or Rasberry py or any other place. Once we deploy, we monitor the performance of the model.if its good…we go live with the model or reiterate the all process until our model performance is good.

It's not done yet!!!

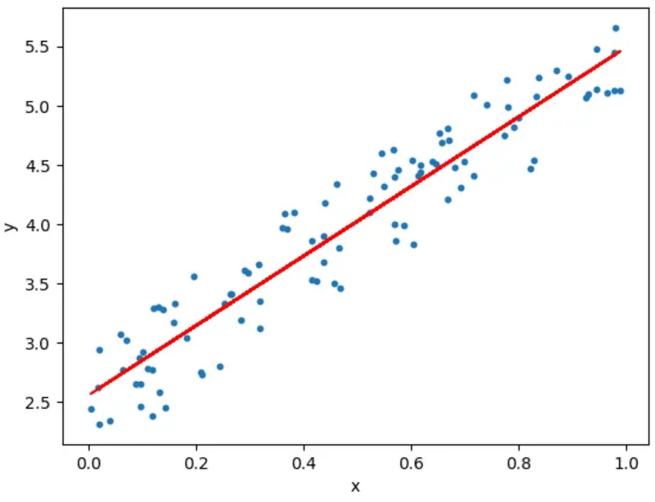
What if, after a few days, our model performs badly because of new data. In that case, we do all the process again by collecting new data and redeploy the model.

4 What is Linear Regression?

Linear Regression tends to establish a relationship between a dependent variable(Y) and one or more independent variable(X) by finding the best fit of the straight line.

The equation for the Linear model is

$$\overset{\text{Dependent Variable}}{Y_i} = \underbrace{\overset{\text{Population Y intercept}}{\beta_0} + \overset{\text{Population Slope Coefficient}}{\beta_1} \overset{\text{Independent Variable}}{X_i}}_{\text{Linear component}} + \underbrace{\overset{\text{Random Error term}}{\varepsilon_i}}_{\text{Random Error component}}$$



In the above diagram, the blue dots we see are the distribution of 'y' w.r.t 'x.' There is no straight line that runs through all the data points. So, the objective here is to fit the best fit of a straight line that will try to minimize the error between the expected and actual value.

Learn more youtobe

The higher the t-value for the feature, the more significant the feature is to the output variable. And also, the p-value plays a rule in rejecting the Null hypothesis(Null hypothesis stating the features has zero significance on the target variable.). If the p-value is less than 0.05(95% confidence interval) for a feature, then we can consider the feature to be significant.

5 What is L1 Regularization (L1 = lasso) ?)

The main objective of creating a model(training data) is making sure it fits the data properly and reduce the loss. Sometimes the model that is trained which will fit the data but it may fail and give a poor performance during analyzing of data (test data). This leads to overfitting. Regularization came to overcome overfitting.

Lasso Regression (Least Absolute Shrinkage and Selection Operator) adds "Absolute value of magnitude" of coefficient, as penalty term to the loss function.

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.

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2}_{\text{Loss function}} + \lambda \underbrace{\sum_{j=0}^{M} W_j^2}_{\substack{\text{Regularization} \\ \text{Term}}}$$

Methods like Cross-validation, Stepwise Regression are there to handle overfitting and perform feature selection work well with a small set of features. These techniques are good when we are dealing with a large set of features.

Along with shrinking coefficients, the lasso performs feature selection, as well. (Remember the 'selection' in the lasso full-form?) Because some of the coefficients become exactly zero, which is equivalent to the particular feature being excluded from the model.

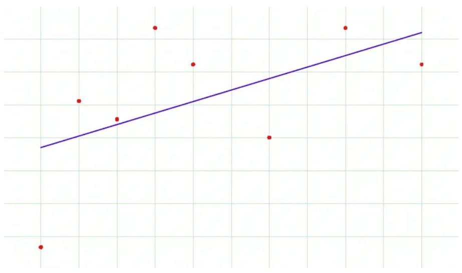
6 L2 Regularization(L2 = Ridge Regression)

$$Cost\,function = Loss + \frac{\lambda}{2m} * \sum \|w\|^2$$

Overfitting happens when the model learns signal as well as noise in the training data and wouldn't perform well on new/unseen data on which model wasn't trained on.

To avoid overfitting your model on training data like cross-validation sampling, reducing the number of features, pruning, regularization, etc.

So to avoid overfitting, we perform Regularization.



The Regression model that uses L2 regularization is called Ridge Regression.

The formula for Ridge Regression:

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \boxed{\lambda \sum_{j=1}^{n} \theta_j^2}\right]$$

$$\min_\theta J(\theta)$$

Regularization adds the penalty as model complexity increases. The regularization parameter (lambda) penalizes all the parameters except intercept so that the model generalizes the data and won't overfit.

Ridge regression adds "squared magnitude of the coefficient" as penalty term to the loss function. Here the box part in the above image represents the L2 regularization element/term.

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

Lambda is a hyper-parameter.

If lambda is zero, then it is equivalent to OLS. But if lambda is very large, then it will add too much weight, and it will lead to under-fitting.

Ridge regularization forces the weights to be small but does not make them zero and does not give the sparse solution.

Ridge is not robust to outliers as square terms blow up the error differences of the outliers, and the regularization term tries to fix it by penalizing the weights

Ridge regression performs better when all the input features influence the output, and all with weights are of roughly equal size.

6.1 Similarity and Difference L1 and L2 Regularization

L2 regularization can learn complex data patterns.

Similarity:

- Regularization does prevent the model from overfitting.

- Regularization is good for models with high sampling variance. High Sampling Variance means the model parameters fluctuate significantly with different training sets.
- Regularization limits space of learnable models, which reduces variance.
- However, it introduces bias - the learned model isn't the "best" possible according to the training error.

Difference: Lasso (l1) Regularization as reducing the Quantity (number) of features: By reducing the sum of absolute values of the coefficients, what Lasso Regularization (L1 Norm) does is to reduce the number of features in the model altogether to predict the target variable.

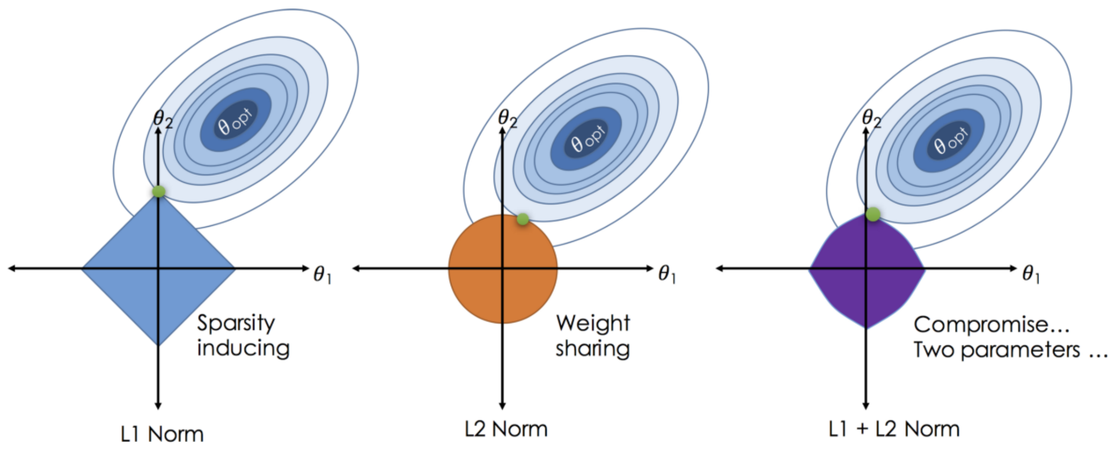
L1 regularized regression does perform feature selection but does not always have a unique solution.

Ridge (l2) Regularization as reducing the Quality (attribute) of features: by reducing the sum of square of coefficients, Ridge Regularization (L2 Norm) doesn't necessarily reduce the number of features selection, but rather reduces the magnitude/impact that each features has on the model by reducing the coefficient value.

L2 regularization does not perform feature selection but always has a unique solution.

6.2 Combination of Lasso and Ridge Regularization:

ElasticNet would be the ideal type of regularization to perform on a model.



7 What is R square(where to use and where not)?

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

The definition of R-squared is the percentage of the response variable variation that is explained by a linear model.

R-squared = Explained variation / Total variation

R-squared is always between 0 and 100%.

0% indicates that the model explains none of the variability of the response data around its mean.

100% indicates that the model explains all the variability of the response data around its mean.

In general, the higher the R-squared, the better the model fits your data.

Sum Squared Regression Error

$$R^2 = 1 - \frac{SS_{Regression}}{SS_{Total}}$$

Sum Squared Total Error

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

There is a problem with the R-Square. The problem arises when we ask this question to ourselves.** Is it good to help as many independent variables as possible?**

The answer is No because we understood that each independent variable should have a meaningful impact. But, even** if we add independent variables which are not meaningful**, will it improve R-Square value?

Yes, this is the basic problem with R-Square. How many junk independent variables or important independent variable or impactful independent variable you add to your model, the R-Squared value will always increase. It will never decrease with the addition of a newly independent variable, whether it could be an impactful, non-impactful, or bad variable, so we need another way to measure equivalent R-Square, which penalizes our model with any junk independent variable.

So, we calculate the Adjusted R-Square with a better adjustment in the formula of generic R-square.

$$R^2_{adjusted} = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

where

R^2 = sample R-square

p = Number of predictors

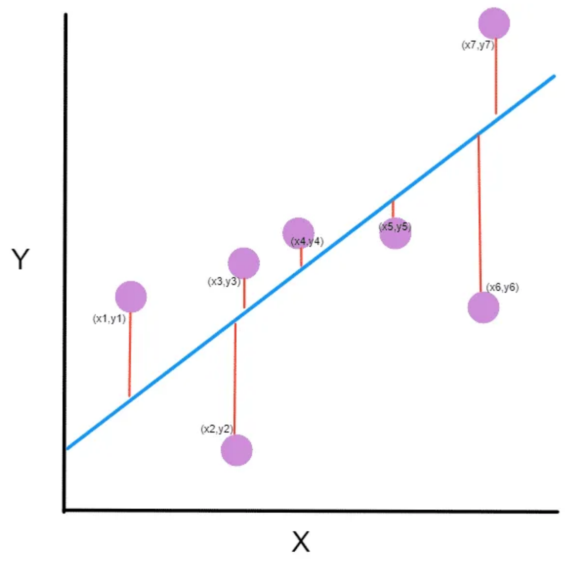
N = Total sample size.

In general, the higher the R-squared, the better the model fits your data. However, there are important conditions for this guideline that I'll talk about both in this post and my next post.

8 What is Mean Square Error?

The mean squared error tells you how close a regression line is to a set of points. It does this by taking the distances from the points to the regression line (these distances are the "errors") and squaring them.

Giving an intuition:



The line equation is y=Mx+B. We want to find M (slope) and B (y-intercept) that minimizes the squared error.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

9 Support Vector Regression?

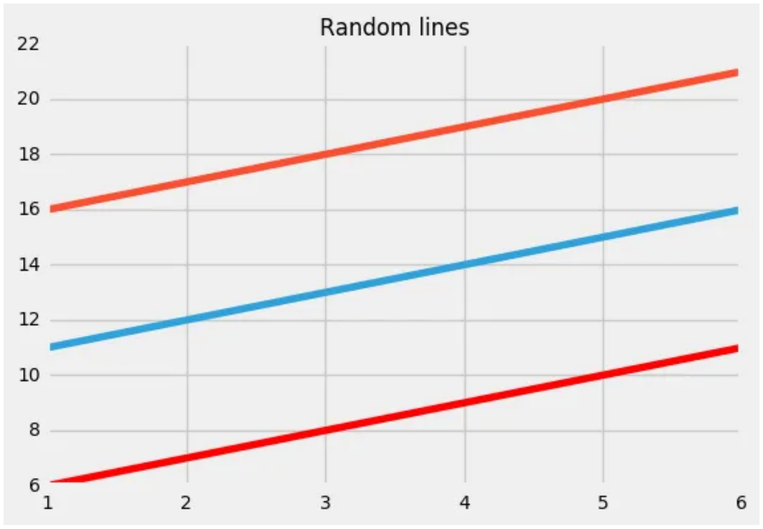
Why Difference between SVR and a simple regression model?

Ans 10:

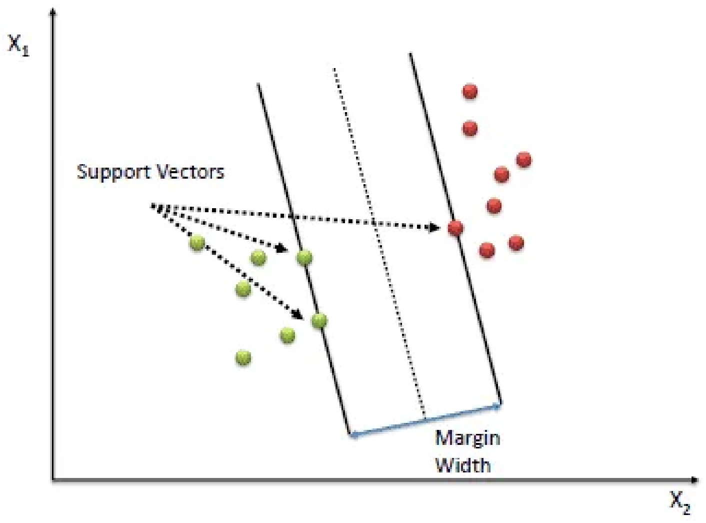
In simple linear regression, try to minimize the error rate. But in SVR, we try to fit the error within a certain threshold.

Main Concepts:

- Boundary
- Kernel
- Support Vector
- Hyper Plane

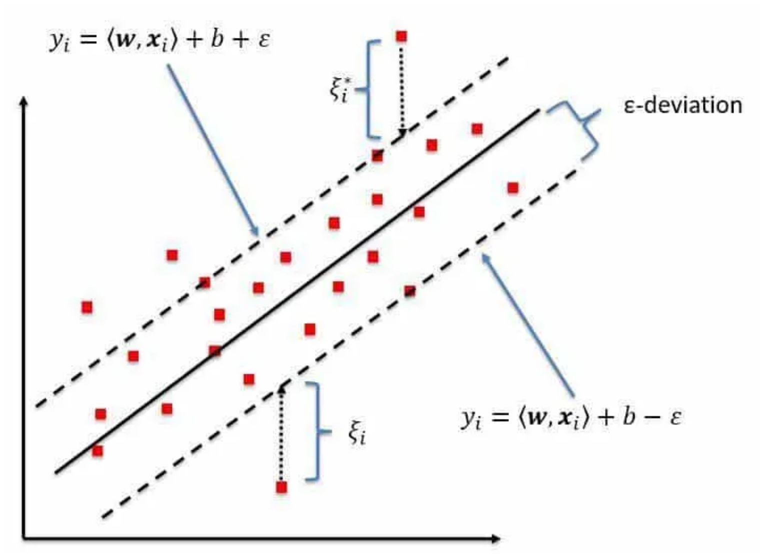


Blueline: Hyper Plane; Red Line: Boundary-Line



Our best fit line is the one where the hyperplane has the maximum number of points.

We are trying to do here is trying to decide a decision boundary at 'e' distance from the original hyperplane such that data points closest to the hyperplane or the support vectors are within that boundary line.

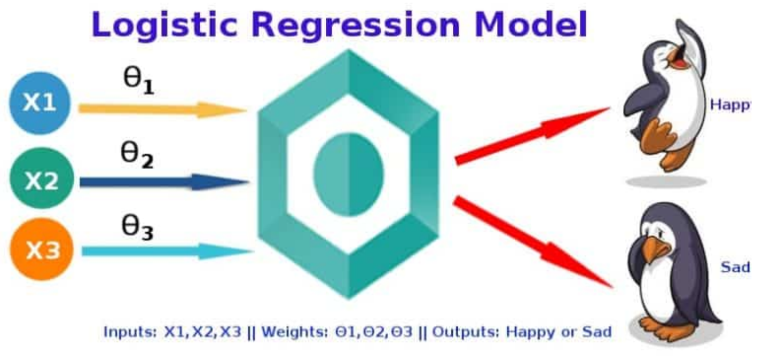

$y_i = \langle \mathbf{w}, \mathbf{x}_i \rangle + b + \varepsilon$
ξ_i^*
ε-deviation
ξ_i
$y_i = \langle \mathbf{w}, \mathbf{x}_i \rangle + b - \varepsilon$

Part I

Day 2

1 What is Logistic Regression?

The logistic regression technique involves the dependent variable, which can be represented in the binary (0 or 1, true or false, yes or no) values, which means that the outcome could only be in either one form of two. For example, it can be utilized when we need to find the probability of a successful or fail event.

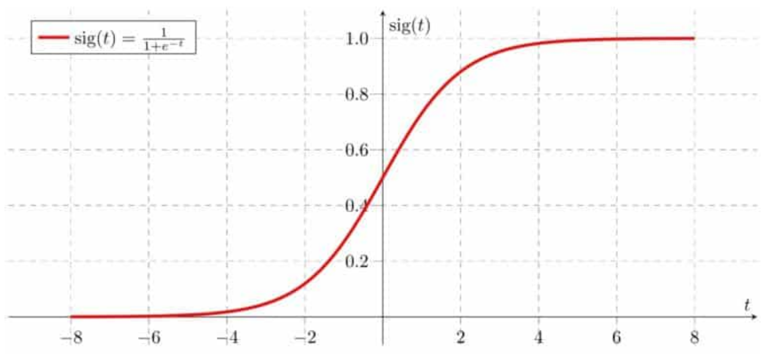

Logistic Regression Model
X1
X2
X3
θ1
θ2
θ3
Happ
Sad
Inputs: X1,X2,X3 || Weights: θ1,θ2,θ3 || Outputs: Happy or Sad

Logistic Regression is used when the dependent variable (target) is categorical.

Model

Output = 0 or 1

Z=WX+B



If ‘Z’ goes to infinity, Y(predicted) will become 1, and if ‘Z’ goes to negative infinity, Y(predicted) will become 0.

The output from the hypothesis is the estimated probability. This is used to infer how confident can predicted value be actual value when given an input X.

Cost Function

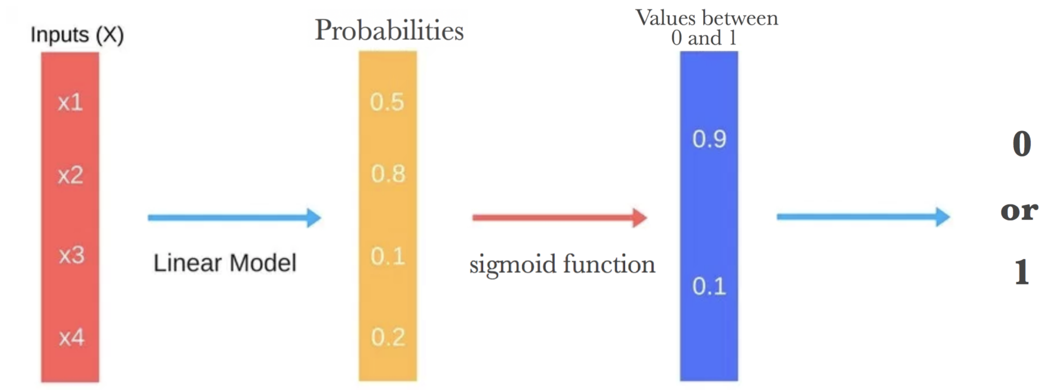
Cost($h_\Theta(x)$, y) = -y log($h_\Theta(x)$) – (1-y) log (1- $h_\Theta(x)$)

If y = 1, (1-y) term will become zero, therefore – log ($h_\Theta(x)$) alone will be present

If y = 0, (y) term will become zero, therefore – log (1- $h_\Theta(x)$) alone will be present

2 Difference between logistic and linear regression?

Linear and Logistic regression are the most basic form of regression which are commonly used. The essential difference between these two is that Logistic regression is used when the dependent variable is binary. In contrast, Linear regression is used when the dependent variable is continuous, and the nature of the regression line is linear.



2.1 Key Differences between Linear and Logistic Regression

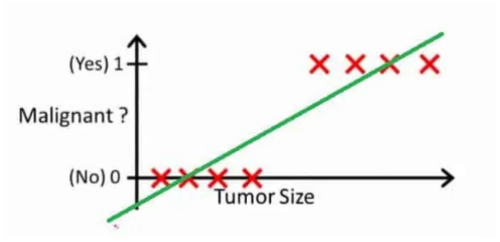
Linear regression models data using continuous numeric value. As against, logistic regression models the data in the binary values.

Linear regression requires to establish the linear relationship among dependent and independent variables, whereas it is not necessary for logistic regression.

In linear regression, the independent variable can be correlated with each other. On the contrary, in the logistic regression, the variable must not be correlated with each other.

3 Why we can't do a classification problem using Regression?

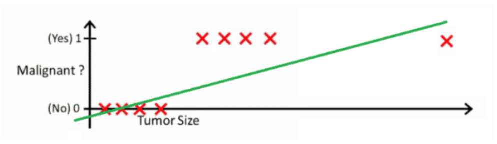
With linear regression you fit a polynomial through the data – say, like on the example below, we fit a straight line through tumor size, tumor type sample set:



Above, malignant tumors get 1, and non-malignant ones get 0, and the green line is our hypothesis h(x). To make predictions, we may say that for any given tumor size x, if h(x) gets bigger than 0.5, we predict malignant tumors. Otherwise, we predict benignly.

It looks like this way, we could correctly predict every single training set sample, but now let's change the task a bit.

Intuitively it's clear that all tumors larger certain threshold are malignant. So let's add another sample with huge tumor size, and run linear regression again:



Now our h(x)>0.5 -> malignant doesn't work anymore. To keep making correct predictions, we need to change it to h(x)>0.2 or something, but that not how the algorithm should work.

We cannot change the hypothesis each time a new sample arrives. Instead, we should learn it off the training set data, and then (using the hypothesis we've learned) make correct predictions for the data we haven't seen before.

Linear regression is unbounded.

3.1 Conclusion

Linear regression is suitable for predicting output that is continuous value, such as predicting the price of a property. Its prediction output can be any real number, range from negative infinity to infinity. The regression line is a straight line.

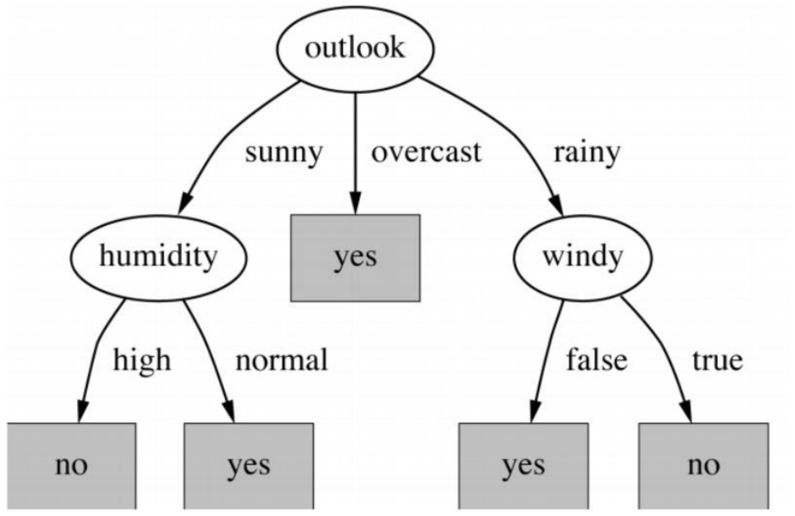
Whereas logistic regression is for classification problems, which predicts a probability range between 0 to 1. For example, predict whether a customer will make a purchase or not. The regression line is a sigmoid curve.

Jupyter Notebook: Why Linear Regression is not suitable for Classification

4 What is Decision Tree?

Video Decision Tree

Final decision tree



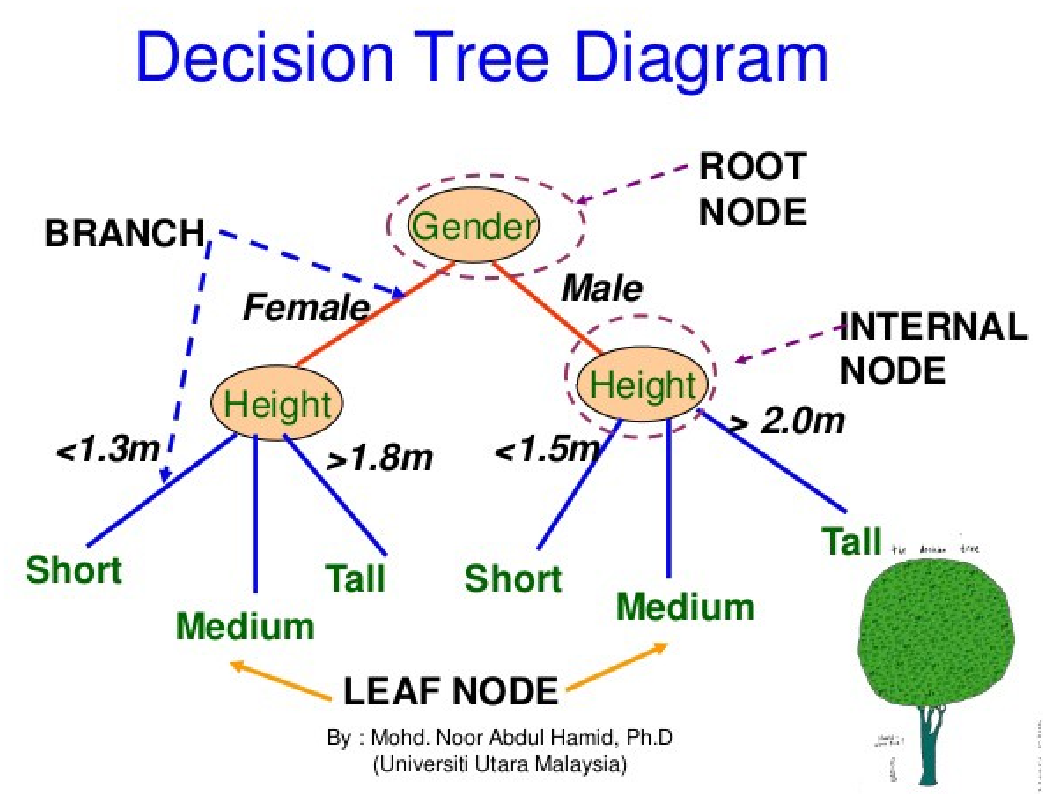
A decision tree is a type of supervised learning algorithm that can be used in classification as well as regressor problems. The input to a decision tree can be both continuous as well as categorical. The decision tree works on an if-then statement. Decision tree tries to solve a problem by using tree representation (Node and Leaf)

Assumptions while creating a decision tree:

1) Initially all the training set is considered as a root

2) Feature values are preferred to be categorical, if continuous then they are discretized

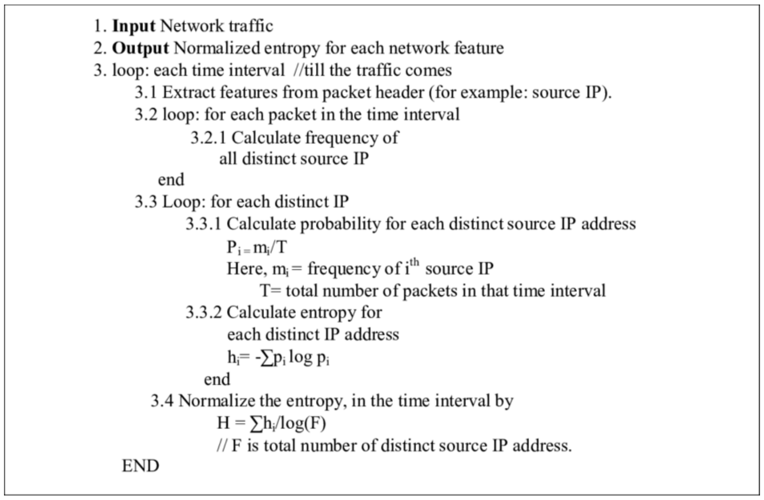
3) Records are distributed recursively on the basis of attribute values 4) Which attributes are considered to be in root node or internal node is done by using a statistical approach.



4.1 Entropy, Information Gain, Gini Index, Reducing Impurity?

There are different attributes which define the split of nodes in a decision tree. There are few algorithms to find the optimal split.

ID3(Iterative Dichotomiser 3): This solution uses Entropy and Information gain as metrics to form a better decision tree. The attribute with the highest information gain is used as a root node, and a similar approach is followed after that. Entropy is the measure that characterizes the impurity of an arbitrary collection of examples.



```
1. Input Network traffic
2. Output Normalized entropy for each network feature
3. loop: each time interval  //till the traffic comes
      3.1 Extract features from packet header (for example: source IP).
      3.2 loop: for each packet in the time interval
              3.2.1 Calculate frequency of
                    all distinct source IP
         end
      3.3 Loop: for each distinct IP
              3.3.1 Calculate probability for each distinct source IP address
                    $P_i = m_i/T$
                    Here, $m_i$ = frequency of $i^{th}$ source IP
                          T= total number of packets in that time interval
              3.3.2 Calculate entropy for
                    each distinct IP address
                    $h_i = -\sum p_i \log p_i$
                 end
        3.4 Normalize the entropy, in the time interval by
                 $H = \sum h_i/\log(F)$
                 // F is total number of distinct source IP address.
   END
```

Entropy varies from 0 to 1. 0 if all the data belong to a single class and 1 if the class distribution is equal. In this way, entropy will give a measure of impurity in the dataset. Steps to decide which attribute to split:

1. Compute the entropy for the dataset

2. For every attribute:

2.1 Calculate entropy for all categorical values.

2.2 Take average information entropy for the attribute.

2.3 Calculate gain for the current attribute.

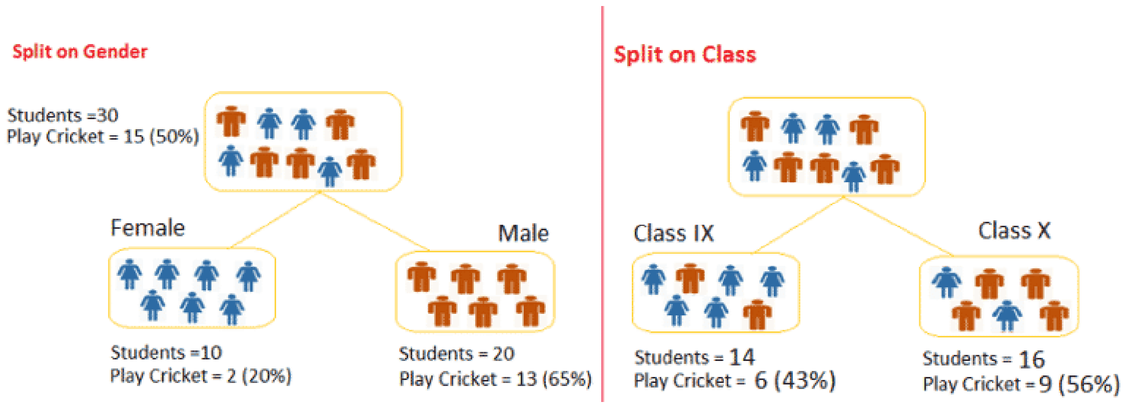
Pick the attribute with the highest information gain. Repeat until we get the desired tree. A leaf node is decided when entropy is zero

Information Gain = 1 – Σ (Sb/S)*Entropy (Sb)

Sb – Subset, S – entire data

CART Algorithm (Classification and Regression trees): In CART, we use the GINI index as a metric. Gini index is used as a cost function to evaluate split in a dataset Steps to calculate Gini for a split:

Calculate Gini for subnodes, using formula sum of the square of probability for success and failure (p2+q2). Calculate Gini for split using weighted Gini score of each node of that split. Choose the split based on higher Gini value



Split on Gender:

Gini for sub-node Female = (0.2)*(0.2)+(0.8)*(0.8)=0.68 Gini for sub-node Male = (0.65)*(0.65)+(0.35)*(0.35)=0.55

Weighted Gini for Split Gender = (10/30)*0.68+(20/30)*0.55 = 0.59

Similar for Split on Class:

Gini for sub-node Class IX = (0.43)*(0.43)+(0.57)*(0.57)=0.51

Gini for sub-node Class X = (0.56)*(0.56)+(0.44)*(0.44)=0.51

Weighted Gini for Split Class = (14/30)*0.51+(16/30)*0.51 = 0.51

Here Weighted Gini is high for gender, so we consider splitting based on gender

4.2 How to control leaf height and Pruning?

To control the leaf size, we can set the parameters:-

4.2.1 Maximum depth :

Maximum tree depth is a limit to stop the further splitting of nodes when the specified tree depth has been reached during the building of the initial decision tree.

NEVER use maximum depth to limit the further splitting of nodes. In other words: use the largest possible value.

4.3 Minimum split size:

Minimum split size is a limit to stop the further splitting of nodes when the number of observations in the node is lower than the minimum split size.

This is a good way to limit the growth of the tree. When a leaf contains too few observations, further splitting will result in overfitting (modeling of noise in the data).

4.4 Minimum leaf size

Minimum leaf size is a limit to split a node when the number of observations in one of the child nodes is lower than the minimum leaf size.

4.5 Example:

Leaf size = number of cases or observations in that leaf.

Consider this simplified example for illustration purpose. We start with 1000 rows/observations and are building a decision tree to predict yes/no.

Split 1: variable 1 split 1000 into 700Y, 300N.

Split 2: variable 2 split the above 700 into 550Y, 150N

Split 3: variable 3 split the above 550 into 300Y,150N

question... how long does this go on. If our data set is kind to us, maybe at split 4 we may get all Y or all N and end of story. But if not, how much more splitting will be needed. This is where the concept of minimum size of the leaf to attempt a split comes in. If we choose too small a leaf size, say 20, you can see it may take us 10 or more splits to reach upto 20. Too deep a tree means overfitting! On the flip side, if we choose too large a leaf size, say in the above example, 500, the tree will stop growing after the second split itself. Meaning poor predictive performance. Hope that helps you see the reason why we need to discover the optimal minimum leaf size.

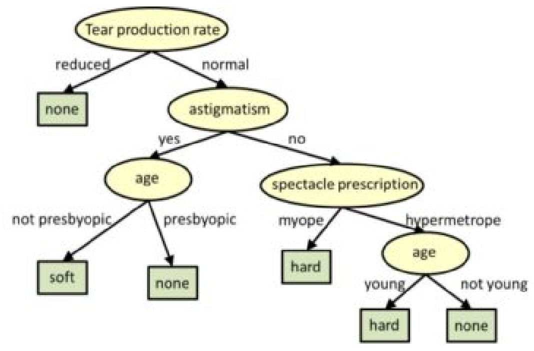
4.6 Pruning

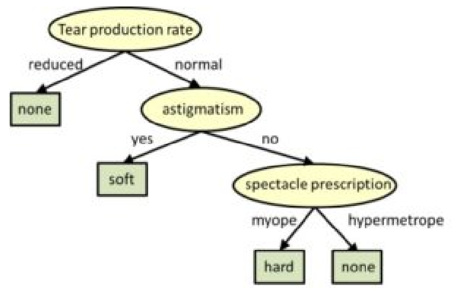
is mostly done to reduce the chances of overfitting the tree to the training data and reduce the overall complexity of the tree.

There are two types of pruning: Pre-pruning and Post-pruning.

1. Pre-pruning is also known as the early stopping criteria. As the name suggests, the criteria are set as parameter values while building the model. The tree stops growing when it meets any of these pre-pruning criteria, or it discovers the pure classes.

2. In Post-pruning, the idea is to allow the decision tree to grow fully and observe the CP value. Next, we prune/cut the tree with the optimal CP(Complexity Parameter) value as the parameter. The CP (complexity parameter) is used to control tree growth. If the cost of adding a variable is higher, then the value of CP, tree growth stops.





5 How to handle a decision tree for numerical and categorical data?

Decision trees can handle both categorical and numerical variables at the same time as features. There is not any problem in doing that.

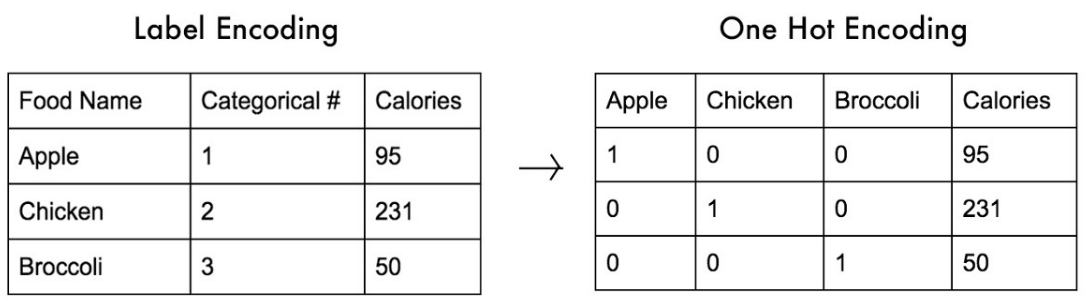
Every split in a decision tree is based on a feature.

1. If the feature is categorical, the split is done with the elements belonging to a particular class.

2. If the feature is continuous, the split is done with the elements higher than a threshold.

At every split, the decision tree will take the best variable at that moment. This will be done according to an impurity measure with the split branches. And the fact that the variable used to do split is categorical or continuous is irrelevant (in fact, decision trees categorize continuous variables by creating binary regions with the threshold).

At last, the good approach is to always convert your categoricals to continuous using **LabelEncoder or OneHotEncoding.**



Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

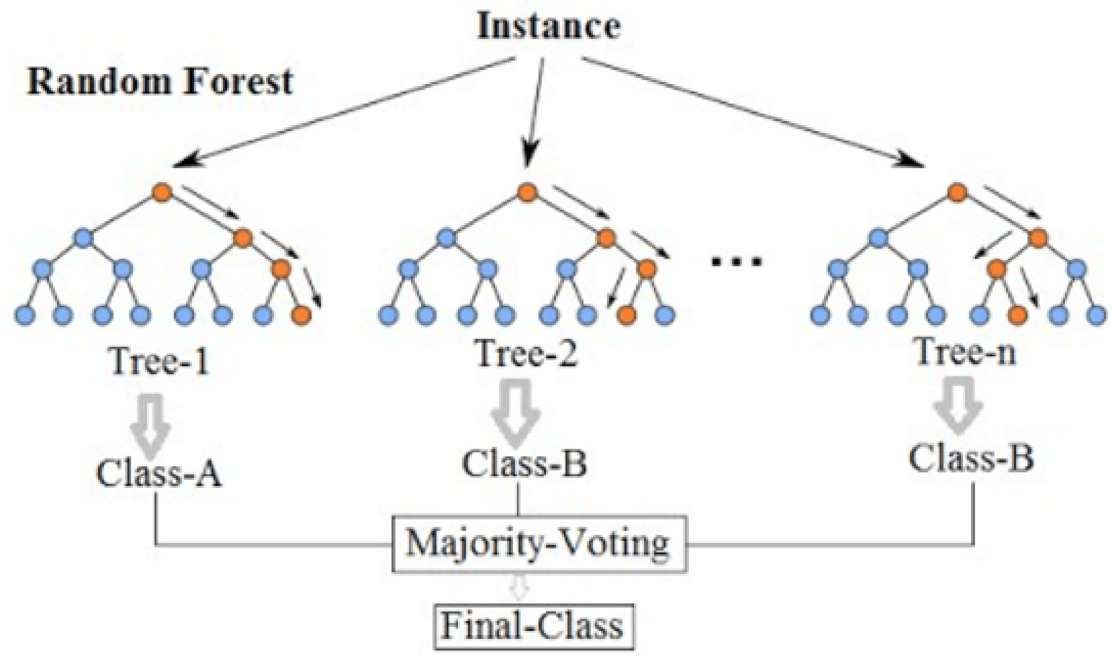
$\rightarrow$

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

6 What is the Random Forest Algorithm?

Video Random Forest - Fun and Easy Machine Learning



Random Forest is an ensemble machine learning algorithm that follows the bagging technique. The base estimators in the random forest are decision trees. Random forest randomly selects a set of features that are used to decide the best split at each node of the decision tree. Looking at it step-by-step, this is what a random forest model does:

1. Random subsets are created from the original dataset (bootstrapping).

2. At each node in the decision tree, only a random set of features are considered to decide the best split.

3. A decision tree model is fitted on each of the subsets.

4. The final prediction is calculated by averaging the predictions from all decision trees.

To sum up, the Random forest randomly selects data points and features and builds multiple trees (Forest).

Random Forest is used for feature importance selection. The attribute (.feature importances) is used to find feature importance.

Some Important Parameters:-

6.0.1 n estimators:

It defines the number of decision trees to be created in a random forest.

6.0.2 criterion:

Gini as default

6.0.3 min samples split:

Used to define the minimum number of samples required in a leaf node before a split is attempted

6.0.4 max features:

It defines the maximum number of features allowed for the split in each decision tree.

6.0.5 n jobs:

The number of jobs to run in parallel for both fit and predict. Always keep (-1) to use all the cores for parallel processing.

7 What is Variance and Bias tradeoff?

Machine Learning Fundamentals: Bias and Variance

In predicting models, the prediction error is composed of two different errors

1. Bias

2. Variance

It is important to understand the variance and bias trade-off which tells about to minimize the Bias and Variance in the prediction and avoids overfitting & under fitting of the model.

Bias: The inability for machine learning method(like linear regression) to capture the true relationship is called Bias. Model with high bias pays very little attention to the training data and oversimplifies the model causing leading to underfitting.

Variance: Variability of a model prediction for a given data point. Model with high variance pays too much attention to training data and does not generalize on the data which it hasn't seen before and gives inaccurate prediction. As a result, such models perform very well on training data but has high error rates on test data.So model with high Variance leads to overfitting.

bias: Pays very little attention to the training data and oversimplifies the model

Variance: Pays too much attention to training data and does not generalize on the data

Best: low Variance, low bias

Underfitting : high bias , low Variance

overfitting : low bias , high Variance

Underfitting: happens when a model unable to capture the underlying pattern of the data. These models usually have high bias and low variance. It happens when we have very less amount of data to build an accurate model or when we try to build a linear model with a nonlinear data. Also, these kind of models are very simple to capture the complex patterns in data like Linear Regression, Linear Discriminant Analysis and Logistic Regression.

Overfitting happens when our model captures the noise along with the underlying pattern in data. It happens when we train our model a lot over noisy dataset taking too many features to consider. These models have low bias and high variance. These models are very complex like Decision Trees, k-Nearest Neighbors and Support Vector Machines which are prone to overfitting.

Three commonly used methods for finding the sweet spot between simple and complicated models are:

1. Regularization

2. Boosing

3. Bagging

7.1 Summary of Bias and Variance

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict

Variance: Variance is the variability of model prediction for a given data point or a value which tells us spread of our data.

High bias: Less understanding for training data High Variance: pays a lot of attention to training data and does not generalize on the data

Low bias, low variance: Aiming at the target and hitting it with good precision. Low bias, high variance: Aiming at the target, but not hitting it consistently. High bias, low variance: Aiming off the target, but being consistent. High bias, high variance: Aiming off the target and being inconsistent.

Total Error = Bias + Variance high bias & high Variance -> Underfitting Low bias & high Variance -> overfitting

8 What are Ensemble Methods?

1. Bagging

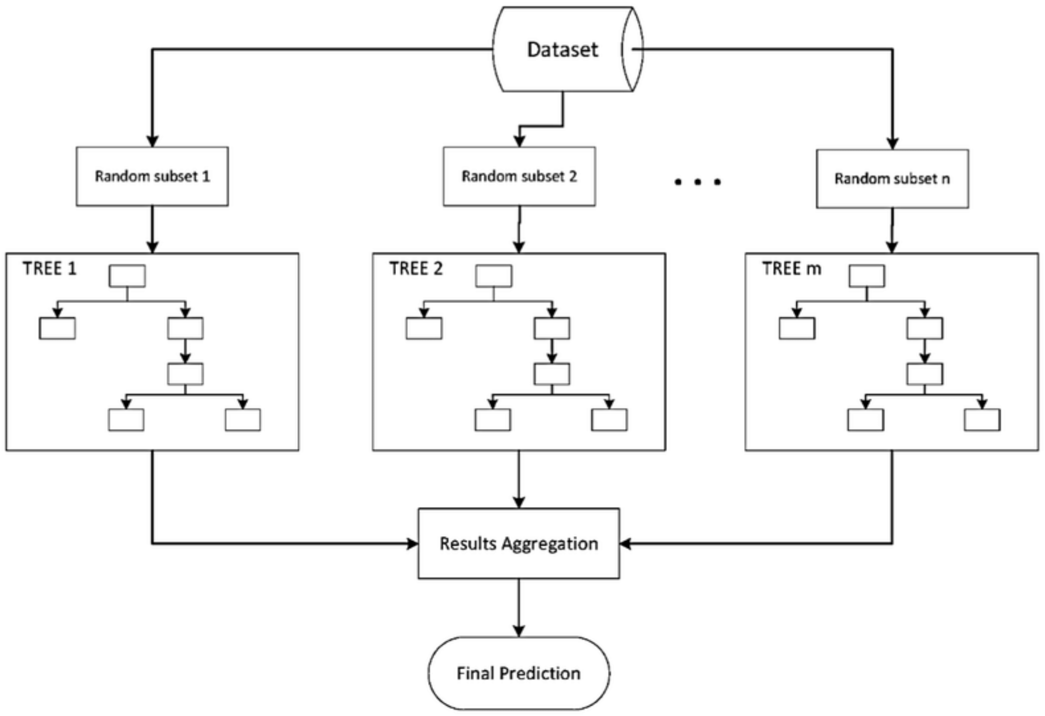
2. Boosting

Decision trees have been around for a long time and also known to suffer from bias and variance.

You will have a large bias with simple trees and a large variance with complex trees.

Ensemble methods – which combines several decision trees to produce better predictive performance than utilizing a single decision tree. The main principle behind the ensemble model is that a group of weak learners come together to form a strong learner.

Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model.



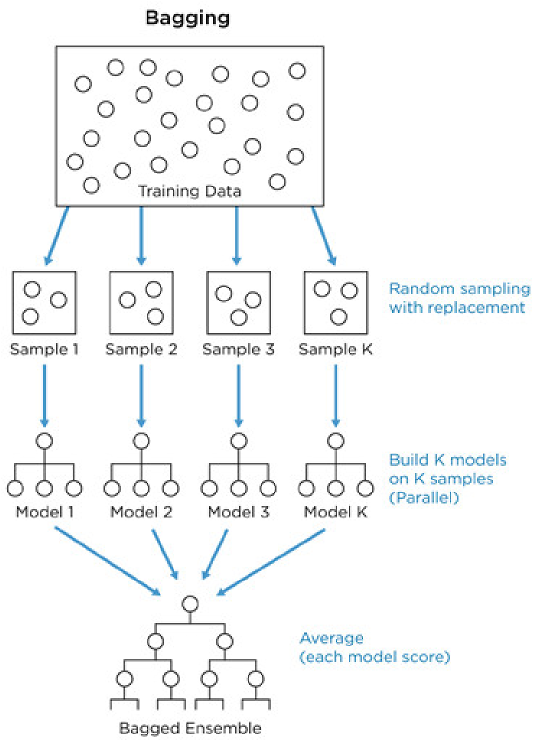
Two techniques to perform ensemble decision trees:

1. Bagging

2. Boosting

Bagging (Bootstrap Aggregation) is used when our goal is to reduce the variance of a decision tree. Here the idea is to create several subsets of data from the training sample chosen randomly with replacement. Now, each collection of subset data is used to train their decision trees. As a result, we end up with an ensemble of different models. Average of all the predictions from different trees are used which is more robust than a single decision tree.

Video Bootstrap aggregating bagging


Bagging
Training Data
Sample 1
Sample 2
Sample 3
Sample K
Random sampling with replacement
Model 1
Model 2
Model 3
Model K
Build K models on K samples (Parallel)
Average (each model score)
Bagged Ensemble

Boosting is another ensemble technique to create a collection of predictors. In this technique, learners are learned sequentially with early learners fitting simple models to the data and then analyzing data

for errors. In other words, we fit consecutive trees (random sample), and at every step, the goal is to solve for net error from the prior tree.

When a hypothesis misclassifies an input, its weight is increased, so that the next hypothesis is more likely to classify it correctly. By combining the whole set at the end converts weak learners into a better performing model.

The different types of boosting algorithms are:

1. AdaBoost

What is AdaBoost (BOOSTING TECHNIQUES)

2. Gradient Boosting

3. XGBoost

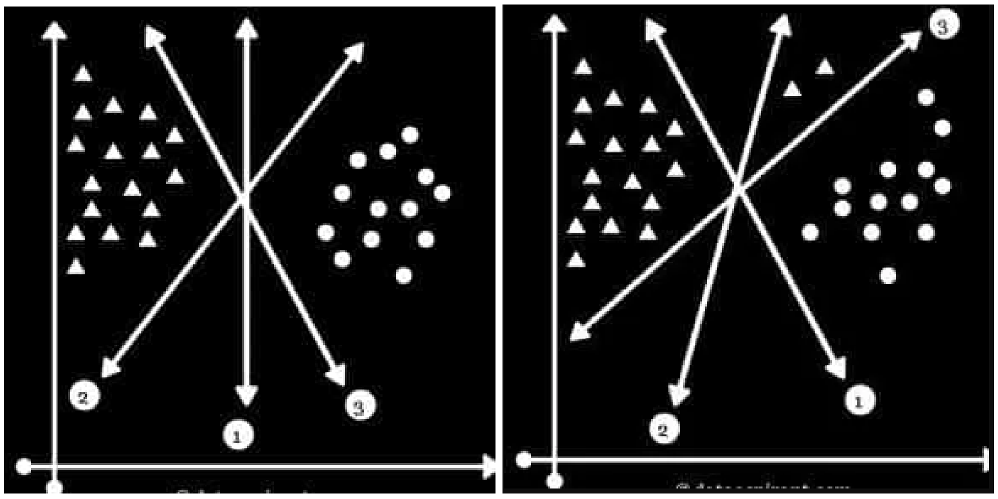
9 What is SVM Classification?

SVM or Large margin classifier is a supervised learning algorithm that uses a powerful technique called SVM for classification.

9.1 We have two types of SVM classifiers:

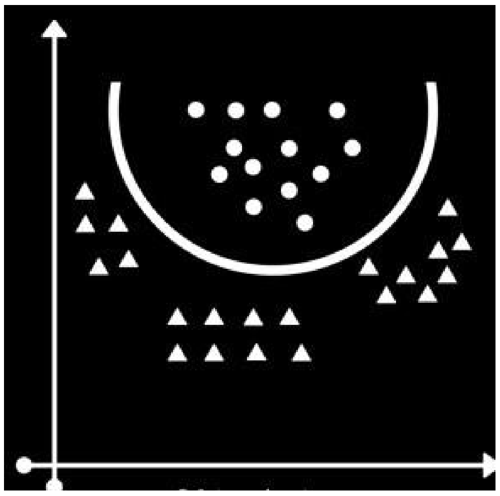
9.1.1 Linear SVM:

In Linear SVM, the data points are expected to be separated by some apparent Therefore, the SVM algorithm predicts a straight hyperplane dividing the two classes. The hyperplane is also called as maximum margin hyperplane.



9.1.2 Non-Linear SVM:

It is possible that our data points are not linearly separable in a p-dimensional space, but can be linearly separable in a higher dimension. Kernel tricks make it possible to draw nonlinear hyperplanes. Some standard kernels are a) Polynomial Kernel b) RBF kernel(mostly used).



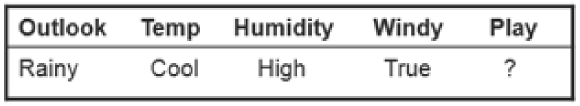
9.2 Advantages of SVM classifier:

SVMs are effective when the number of features is quite large. It works effectively even if the number of features is greater than the number of samples. Non-Linear data can also be classified using customized hyperplanes built by using kernel trick. It is a robust model to solve prediction problems since it maximizes margin.

9.3 Disadvantages of SVM classifier:

The biggest limitation of the Support Vector Machine is the choice of the kernel. The wrong choice of the kernel can lead to an increase in error percentage. With a greater number of samples, it starts giving poor performances. SVMs have good generalization performance, but they can be extremely slow in the test phase. SVMs have high algorithmic complexity and extensive memory requirements due to the use of quadratic programming.

10 What is Naive Bayes Classification and Gaussian Naive Bayes



Outlook	Temp	Humidity	Windy	Play
Rainy	Cool	High	True	?

$$P(Yes \mid X) = P(Rainy \mid Yes) \times P(Cool \mid Yes) \times P(High \mid Yes) \times P(True \mid Yes) \times P(Yes)$$

$$P(Yes \mid X) = 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = \boxed{0.00529} \longrightarrow 0.2 = \frac{0.00529}{0.02057 + 0.00529}$$

$$P(No \mid X) = P(Rainy \mid No) \times P(Cool \mid No) \times P(High \mid No) \times P(True \mid No) \times P(No)$$

$$P(No \mid X) = 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = \boxed{0.02057} \longrightarrow 0.8 = \frac{0.02057}{0.02057 + 0.00529}$$

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

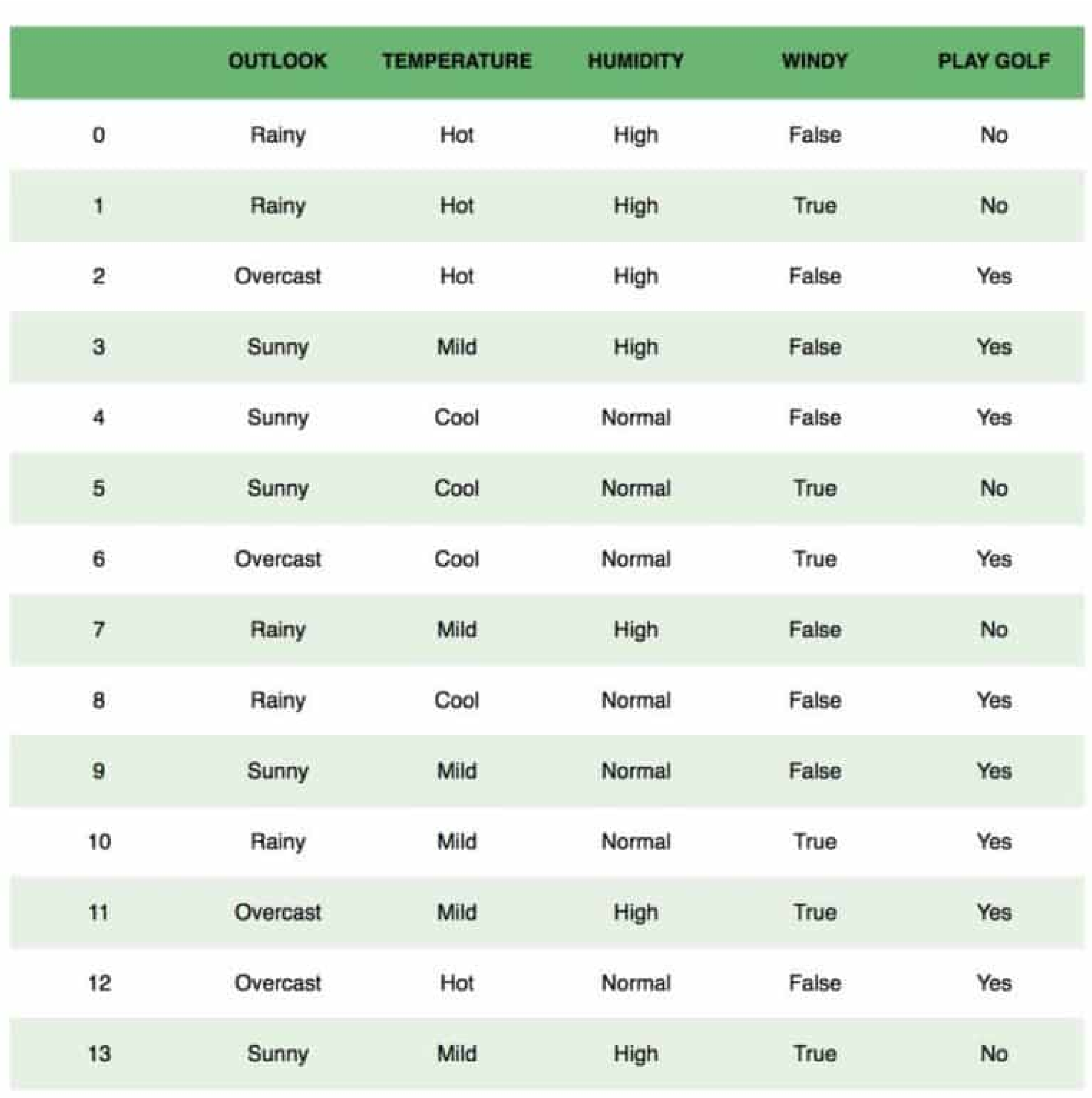
Now, with regards to our dataset, we can apply Bayes' theorem in following way:

P(y | X) = P(X | y) P(y)/P(X)

where, y is class variable and X is a dependent feature vector (of size n) where:

X =

$$(x_1, x_2, x_3, \ldots.., x_n)$$



	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

To clear, an example of a feature vector and corresponding class variable can be: (refer 1st row of the dataset)

X = (Rainy, Hot, High, False) y = No So basically, P(X|y) here means, the probability of "Not playing golf" given that the weather conditions are "Rainy outlook", "Temperature is hot", "high humidity" and "no wind".

10.1 Naive Bayes Classification:

We assume that no pair of features are dependent. For example, the temperature being 'Hot' has nothing to do with the humidity, or the outlook being 'Rainy' does not affect the winds. Hence, the features are assumed to be independent.

Secondly, each feature is given the same weight (or importance). For example, knowing the only temperature and humidity alone can't predict the outcome accurately. None of the attributes is irrelevant and assumed to be contributing equally to the outcome.

10.2 Gaussian Naive Bayes

Continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called Normal distribution. When plotted, it gives a bell-shaped curve which is symmetric about the mean of the feature values as shown below:

This is as simple as calculating the mean and standard deviation values of each input variable (x) for each class value.

Mean (x) =

$$1/n * sum(x)$$

Where n is the number of instances, and x is the values for an input variable in your training data.

We can calculate the standard deviation using the following equation:

Standard deviation(x) =

$$\sigma = \sqrt{\mu_2}$$

When to use what? Standard Naive Bayes only supports categorical features, while Gaussian Naive Bayes only supports continuously valued features.

11 What is RandomizedSearchCV?

Randomized search CV is used to perform a random search on hyperparameters. Randomized search CV uses a fit and score method, predict proba, decision func, transform, etc..,

The parameters of the estimator used to apply these methods are optimized by cross-validated search over parameter settings.

In contrast to GridSearchCV, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The number of parameter settings that are tried is given by n iter.

Code Example :

[1]:
```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform
iris = load_iris()
logistic = LogisticRegression(solver='saga', tol=1e-2, max_iter=200,
                              random_state=0)
distributions = dict(C=uniform(loc=0, scale=4),
                     penalty=['l2', 'l1'])
clf = RandomizedSearchCV(logistic, distributions, random_state=0)
search = clf.fit(iris.data, iris.target)
search.best_params_
```

[1]:
```
{'C': 2.195254015709299, 'penalty': 'l1'}
```

12 What is GridSearchCV?

Grid search is the process of performing hyperparameter tuning to determine the optimal values for a given model.

Exhaustive search over specified parameter values for an estimator.

12.1 Code Example:

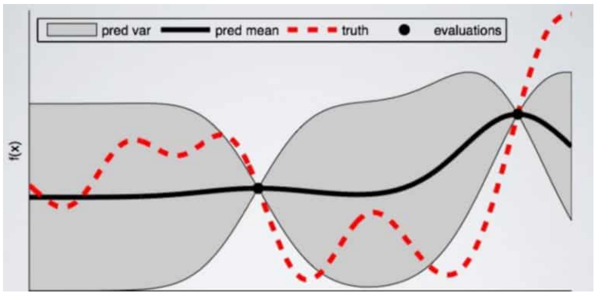
[7]:
```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.svm import SVC

print(__doc__)

# Loading the Digits dataset
digits = datasets.load_digits()

# To apply an classifier on this data, we need to flatten the image, to
# turn the data in a (samples, feature) matrix:
n_samples = len(digits.images)
X = digits.images.reshape((n_samples, -1))
y = digits.target

# Split the dataset in two equal parts
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.5, random_state=0)

# Set the parameters by cross-validation
tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
                     'C': [1, 10, 100, 1000]},
                    {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]

scores = ['precision', 'recall']

for score in scores:
    print("# Tuning hyper-parameters for %s" % score)
    print()

    clf = GridSearchCV(
        SVC(), tuned_parameters, scoring='%s_macro' % score
    )
    clf.fit(X_train, y_train)

    print("Best parameters set found on development set:")
    print()
    print("Best Paramaters is " , clf.best_params_)
    print()
    print("Grid scores on development set:")
    print()
    means = clf.cv_results_['mean_test_score']
    stds = clf.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds, clf.cv_results_['params']):
        print("%0.3f (+/-%0.03f) for %r"
              % (mean, std * 2, params))
    print()

    print("Detailed classification report:")
    print()
    print("The model is trained on the full development set.")
    print("The scores are computed on the full evaluation set.")
    print()
    y_true, y_pred = y_test, clf.predict(X_test)
    print(classification_report(y_true, y_pred))
    print()

# Note the problem is too easy: the hyperparameter plateau is too flat and the
# output model is the same for precision and recall with ties in quality.
```

```
Automatically created module for IPython interactive environment
# Tuning hyper-parameters for precision

Best parameters set found on development set:

Best Paramaters is  {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}

Grid scores on development set:

0.986 (+/-0.016) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
0.959 (+/-0.028) for {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
0.988 (+/-0.017) for {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
0.982 (+/-0.026) for {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
0.988 (+/-0.017) for {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
0.983 (+/-0.026) for {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
0.988 (+/-0.017) for {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
0.983 (+/-0.026) for {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}
0.974 (+/-0.012) for {'C': 1, 'kernel': 'linear'}
0.974 (+/-0.012) for {'C': 10, 'kernel': 'linear'}
0.974 (+/-0.012) for {'C': 100, 'kernel': 'linear'}
0.974 (+/-0.012) for {'C': 1000, 'kernel': 'linear'}

Detailed classification report:
```

```
The model is trained on the full development set.
The scores are computed on the full evaluation set.

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        89
           1       0.97      1.00      0.98        90
           2       0.99      0.98      0.98        92
           3       1.00      0.99      0.99        93
           4       1.00      1.00      1.00        76
           5       0.99      0.98      0.99       108
           6       0.99      1.00      0.99        89
           7       0.99      1.00      0.99        78
           8       1.00      0.98      0.99        92
           9       0.99      0.99      0.99        92

    accuracy                           0.99       899
   macro avg       0.99      0.99      0.99       899
weighted avg       0.99      0.99      0.99       899


# Tuning hyper-parameters for recall

Best parameters set found on development set:

Best Paramaters is  {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}

Grid scores on development set:

0.986 (+/-0.019) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
0.957 (+/-0.028) for {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
0.987 (+/-0.019) for {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
0.981 (+/-0.028) for {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
0.987 (+/-0.019) for {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
0.982 (+/-0.026) for {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
0.987 (+/-0.019) for {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
0.982 (+/-0.026) for {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}
0.971 (+/-0.010) for {'C': 1, 'kernel': 'linear'}
0.971 (+/-0.010) for {'C': 10, 'kernel': 'linear'}
0.971 (+/-0.010) for {'C': 100, 'kernel': 'linear'}
0.971 (+/-0.010) for {'C': 1000, 'kernel': 'linear'}

Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

              precision    recall  f1-score   support
           0       1.00      1.00      1.00        89
           1       0.97      1.00      0.98        90
           2       0.99      0.98      0.98        92
           3       1.00      0.99      0.99        93
           4       1.00      1.00      1.00        76
           5       0.99      0.98      0.99       108
           6       0.99      1.00      0.99        89
           7       0.99      1.00      0.99        78
           8       1.00      0.98      0.99        92
           9       0.99      0.99      0.99        92

    accuracy                           0.99       899
   macro avg       0.99      0.99      0.99       899
weighted avg       0.99      0.99      0.99       899
```

13 What is BaysianSearchCV?

Bayesian search, in contrast to the grid and random search, keeps track of past evaluation results, which they use to form a probabilistic model mapping hyperparameters to a probability of a score on the objective function.



The parameters of the estimator used to apply these methods are optimized by cross-validated search over parameter settings.

In contrast to GridSearchCV, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The number of parameter settings that are tried is given by n iter.

Parameters are presented as a list of skopt.space.Dimension objects.

Code Example:

[12]:
```
from skopt import BayesSearchCV
# parameter ranges are specified by one of below
from skopt.space import Real, Categorical, Integer

from sklearn.datasets import load_iris
```

```python
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split

X, y = load_iris(True)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    train_size=0.75,
                                                    random_state=0)

# log-uniform: understand as search over p = exp(x) by varying x
opt = BayesSearchCV(
    SVC(),
    {
        'C': Real(1e-6, 1e+6, prior='log-uniform'),
        'gamma': Real(1e-6, 1e+1, prior='log-uniform'),
        'degree': Integer(1,8),
        'kernel': Categorical(['linear', 'poly', 'rbf']),
    },
    n_iter=32,
    random_state=0
)

# executes bayesian optimization
_ = opt.fit(X_train, y_train)

# model can be saved, used for predictions or scoring
print(opt.score(X_test, y_test))
```

```
0.9736842105263158
```

14 What is ZCA Whitening?

ZCA = Zero-phase Component Analysis

Video Prewhitening with zero-phase component analysis (ZCA)

Making the co-variance matrix as the Identity matrix is called whitening. This will remove the first and second-order statistical structure ZCA transforms the data to zero means and makes the features linearly independent of each other In some image analysis applications, especially when working with images of the color and tiny typ e, it is frequently interesting to apply some whitening to the data before, e.g. training a classifier.