

DATA PREPROCESSING

Getting Started with Machine Learning



Step 1: Importing the required Libraries

These Two are essential libraries which we will import every time.

NumPy is a Library which contains Mathematical functions.

Pandas is the library used to import and manage the data sets.



Step 2: Importing the Data Set

Data sets are generally available in .csv format. A CSV file stores tabular data in plain text. Each line of the file is a data record. We use the read_csv method of the pandas library to read a local CSV file as a dataframe. Then we make separate Matrix and Vector of independent and dependent variables from the dataframe.

NaN

Step 3: Handling the Missing Data

The data we get is rarely homogeneous. Data can be missing due to various reasons and needs to be handled so that it does not reduce the performance of our machine learning model. We can replace the missing data by the Mean or Median of the entire column. We use Imputer class of sklearn.preprocessing for this task.



Step 4: Encoding Categorical Data

Categorical data are variables that contain label values rather than numeric values. The number of possible values is often limited to a fixed set. Example values such as "Yes" and "No" cannot be used in mathematical equations of the model so we need to encode these variables into numbers. To achieve this we import LabelEncoder class from sklearn.preprocessing library.



Step 5: Splitting the dataset into test set and training set

We make two partitions of dataset one for training the model called training set and other for testing the performance of the trained model called test set. The split is generally 80/20. We import train_test_split() method of sklearn.crossvalidation library.



Step 6: Feature Scaling

Most of the machine learning algorithms use the Euclidean distance between two data points in their computations, features highly varying in magnitudes, units and range pose problems. High magnitudes features will weigh more in the distance calculations than features with low magnitudes. Done by Feature standardization or Z-score normalization. StandardScaler of sklearn.preprocessing is imported.

Check out The complete Implementation at: github.com/Avik-Jain/100-Days-Of-ML-Code

Follow Me For More Updates



SIMPLE LINEAR REGRESSION

Predicting a response using a single feature.

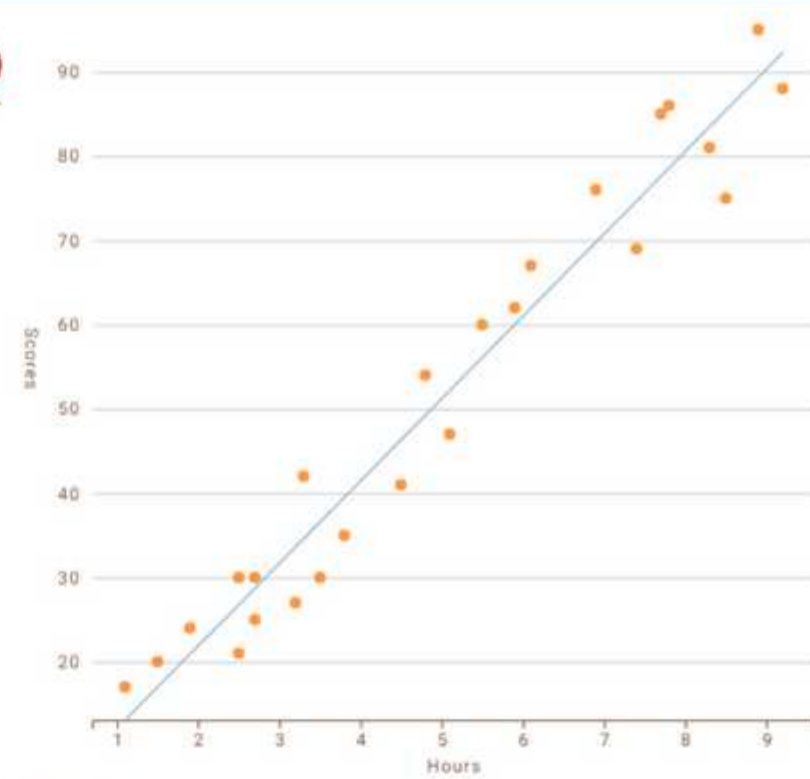
It is a method to predict dependent variable (Y) based on values of independent variables (X). It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value(y) as accurately as possible as a function of the feature or independent variable(x).

How to find the best fit line?

In this regression model, we are trying to minimize the errors in prediction by finding the "line of best fit" – the regression line from the errors would be minimal. We are trying to minimize the length between the observed value (Yi) and the predicted value from our model (Yp).

$$\min \{ \text{SUM}(y_i - y_p)^2 \}$$

Observed Value y_i Predicted Value y_p



$$y = b_0 + b_1 x_1$$

Dependent Variable y Independent Variable x_1

In this regression task, we will predict the percentage of marks that a student is expected to score based upon the number of hours they studied.

$$\text{Score} = b_0 + b_1 * \text{hours}$$

Slope b_1 y-intercept b_0

STEP 1: PREPROCESS THE DATA

We will follow the same steps as in my previous infographic of Data Preprocessing.

- Import the Libraries.
- Import the DataSet.
- Check for Missing Data.
- Split the DataSet.
- Feature Scaling will be taken care by the Library we will use for Simple Linear Regression Model.



STEP 2: FITTING SIMPLE LINEAR REGRESSION MODEL TO THE TRAINING SET

To fit the dataset into the model we will use `LinearRegression` class from `sklearn.linear_model` library. Then we make an object `regressor` of `LinearRegression` Class. Now we will fit the regressor object into our dataset using `fit()` method of `LinearRegression` Class.



STEP 3: PREDICTING THE RESULT

Now we will predict the observations from our test set. We will save the output in a vector `Y_pred`. To predict the result we use `predict` method of `LinearRegression` Class on the regressor we trained in the previous step.



STEP 4: VISUALIZATION

The final step is to visualize our results. We will use `matplotlib.pyplot` library to make Scatter Plots of our Training set results and Test set results to see how close our model predicted the Values



Check out The complete Implementation at: github.com/Avik-Jain/100-Days-Of-ML-Code

Follow Me For More Updates



MULTIPLE LINEAR REGRESSION



Multiple linear regression attempts to model the relationship between two or more features and a response by fitting a linear equation to observed data. The steps to perform multiple linear regression are almost similar to that of simple linear regression. The difference lies in the evaluation. You can use it to find out which factor has the highest impact on the predicted output and how different variables relate to each other.

Dependent Variable

$$y = b_0 + b_1x_1 + b_2x_2 \dots \dots b_nx_n$$

multiple independent Variables



ASSUMPTIONS

FOR A SUCCESSFUL REGRESSION ANALYSIS, IT'S ESSENTIAL TO VALIDATE THESE ASSUMPTIONS.

1. **Linearity:** The relationship between dependent and independent variables should be Linear.
2. **Homoscedasticity** (constant variance) of the errors should be maintained.
3. **Multivariate Normality:** Multiple regression assumes that the residuals are normally distributed.
4. **Lack of Multicollinearity:** It is assumed that there is little or no multicollinearity in the data. Multicollinearity occurs when the features (or independent variables) are not independent of each other.



NOTE

Having too many variables could potentially cause our model to become less accurate, especially if certain variables have no effect on the outcome or have a significant effect on other variables. There are various methods to select the appropriate variable like -

1. Forward Selection
2. Backward Elimination
3. Bi-directional Comparison

DUMMY VARIABLES

Using categorical data in Multiple Regression Models is a powerful method to include non-numeric data types into a regression model.

Categorical data refers to data values which represent categories - data values with a fixed and unordered number of values, for instance, gender (male/female). In a regression model, these values can be represented by dummy variables - variables containing values such as 1 or 0 representing the presence or absence of the categorical value.

Gender

Female

Female

Male

Female

Male

Male

Male

Male	Female
0	1
0	1
1	0
0	1
1	0
1	0
1	0

DUMMY VARIABLE TRAP



The Dummy Variable trap is a scenario in which two or more variables are highly correlated; in simple terms, one variable can be predicted from the others. Intuitively, there is a duplicate category: if we dropped the male category it is inherently defined in the female category (zero female value indicate male, and vice-versa).

The solution to the dummy variable trap is to drop one of the categorical variables - if there are m number of categories, use m-1 in the model, the value left out can be thought of as the reference value.

$$D_2 = 1 - D_1$$

$$y = b_0 + b_1x_1 + b_2x_2 + b_3D_1$$

1 PREPROCESS THE DATA

- Import the Libraries.
- Import the DataSet.
- Check for Missing Data.
- Encode Categorical Data
- Make Dummy Variables if necessary and avoid dummy variable trap.
- Feature Scaling will be taken care by the Library we will use for Simple Linear Regression Model.

2 FITTING OUR MODEL TO THE TRAINING SET

This step is exactly the same as for simple linear regression. To fit the dataset into the model we will use `LinearRegression` class from `sklearn.linear_model` library. Then we make an object `regressor` of `LinearRegression` Class. Now we will fit the `regressor` object into our dataset using `fit()` method of `LinearRegression` Class.

3 PREDICTING THE TEST RESULTS

Now we will predict the observations from our test set. We will save the output in a vector `Y_pred`. To predict the result we use `predict()` method of `LinearRegression` Class on the `regressor` we trained in the previous step.

Check out The complete Implementation at: github.com/Avik-Jain/100-Days-Of-ML-Code

Follow Me For More Updates



LOGISTIC REGRESSION



WHAT IS LOGISTIC REGRESSION

Logistic regression is used for a different class of problems known as classification problems. Here the aim is to predict the group to which the current object under observation belongs to. It gives you a discrete binary outcome between 0 and 1. A simple example would be whether a person will vote or not in upcoming elections.

How Does It Work?

Logistic Regression measures the relationship between the dependent variable (our label, what we want to predict) and the one or more independent variables (our features), by estimating probabilities using it's underlying logistic function.

Making Predictions

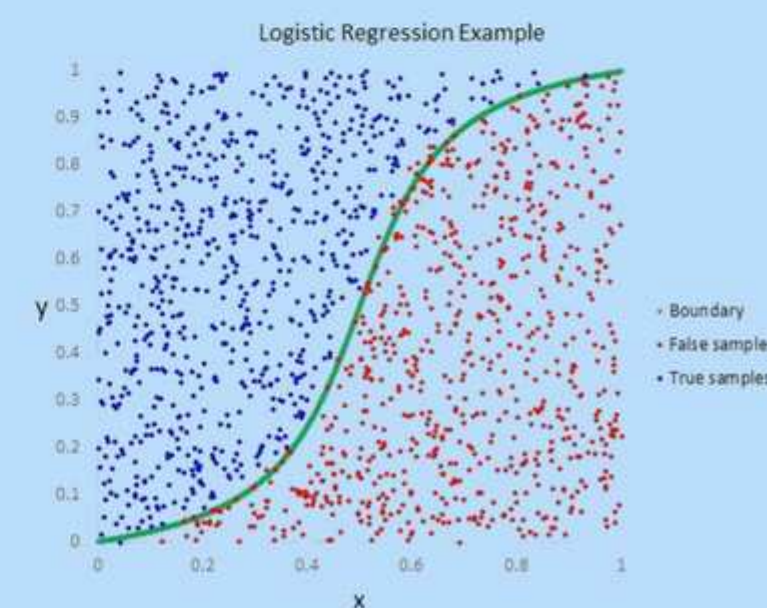
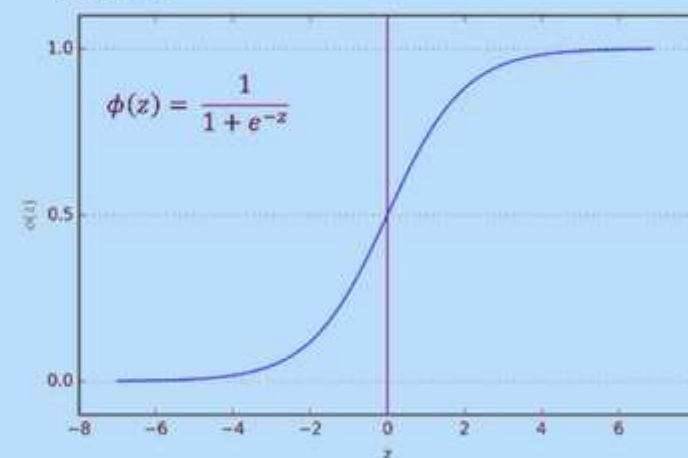
These probabilities must then be transformed into binary values in order to actually make a prediction. This is the task of the logistic function, also called the sigmoid function. This values between 0 and 1 will then be transformed into either 0 or 1 using a threshold classifier.

Logistic vs Linear

Logistic regression gives you a discrete outcome but linear regression gives a continuous outcome.

Sigmoid Function

The Sigmoid-Function is an S-shaped curve that can take any real-valued number and map it into a value between the range of 0 and 1, but never exactly at those limits.



This infographic is just the Logistic regression intuition and is very brief. The mathematical logic and implementation part will be covered in another infographic.

Check out the Repository at: github.com/Avik-Jain/100-Days-Of-ML-Code

Follow Me For More Updates



Logistic Regression | Day 5

Moving forward into #100DaysOfMLCode today I dived into the deeper depth of what Logistic Regression actually is and what is the math involved behind it. Learned how cost function is calculated and then how to apply gradient descent algorithm to cost function to minimize the error in prediction.

Due to less time I will now be posting an infographic on alternate days. Also if someone wants to help me out in documentaion of code and already has some experince in the field and knows Markdown for github please contact me on LinkedIn :).

Implementing Logistic Regression | Day 6

Check out the Code [here](#)

K Nearest Neighbours | Day 7

K NEAREST NEIGHBOURS

An Intuition to K-NN Classification Algorithm

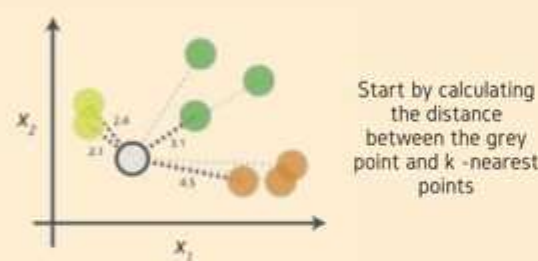
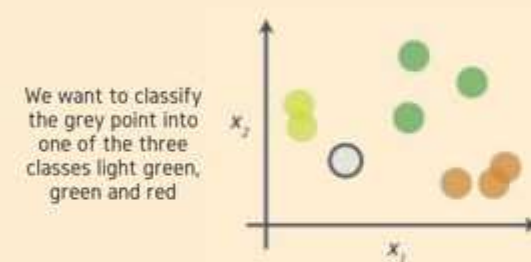
What is k-NN?

K-Nearest Neighbor algorithm is a simple yet most used classification algorithm. It can also be used for regression.

KNN is non-parametric (means that it does not make any assumptions on the underlying data distribution), instance-based (means that our algorithm doesn't explicitly learn a model. Instead, it chooses to memorize the training instances.) and used in a supervised learning setting.



k-NN is also called a lazy algorithm because it is instance based.



How Does k-NN Algorithm work?

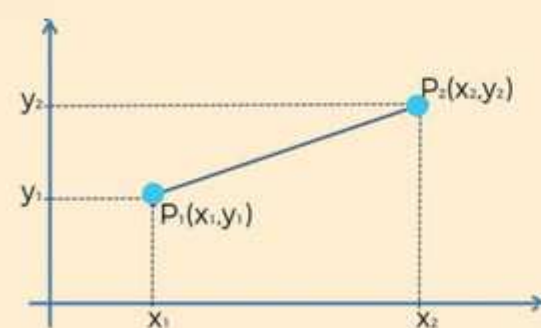
k-NN when used for classification—the output is a class membership (predicts a class—a discrete value). There are three key elements of this approach: a set of labeled objects, e.g., a set of stored records, a distance between objects, and the value of k, the number of nearest neighbors.

Making Predictions

To classify an unlabeled object, the distance of this object to the labeled objects is computed, its k-nearest neighbors are identified, and the class label of the majority of nearest neighbors is then used to determine the class label of the object. For real-valued input variables, the most popular distance measure is Euclidean distance.

Point	Distance	
Light Green	2.1	→ 1st NN
Light Green	2.4	→ 2nd NN
Green	3.1	→ 3rd NN
Red	4.5	→ 4th NN

Class	# of votes	
Light Green	2	Class Light Green wins the vote! Point is therefore predicted to be of class Light Green.
Green	1	
Red	1	



$$\text{Euclidean Distance between } P_1 \text{ and } P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Value of k

Finding the value of k is not easy. A small value of k means that noise will have a higher influence on the result and a large value make it computationally expensive. It depends a lot on your individual cases, sometimes it is best to run through each possible value for k and decide for yourself.

The Distance

Euclidean distance is calculated as the square root of the sum of the squared differences between a new point and an existing point across all input attributes.

Other popular distance measures include:

- Hamming Distance
- Manhattan Distance
- Minkowski Distance

Check out the Repository at: github.com/Avik-Jain/100-Days-Of-ML-Code

Follow Me For More Updates



Math Behind Logistic Regression | Day 8

#100DaysOfMLCode To clear my insights on logistic regression I was searching on the internet for some resource or article and I came across this article (<https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc>) by Saishruthi Swaminathan.

It gives a detailed description of Logistic Regression. Do check it out.

Support Vector Machines | Day 9

Got an intuition on what SVM is and how it is used to solve Classification problem.

SVM and KNN | Day 10

Learned more about how SVM works and implementing the K-NN algorithm.

Implementation of K-NN | Day 11

Implemented the K-NN algorithm for classification. #100DaysOfMLCode Support Vector Machine Infographic is halfway complete. Will update it tomorrow.

Support Vector Machines | Day 12

SUPPORT VECTOR MACHINES



What is SVM?

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression. However, it is mostly used in classification problems.

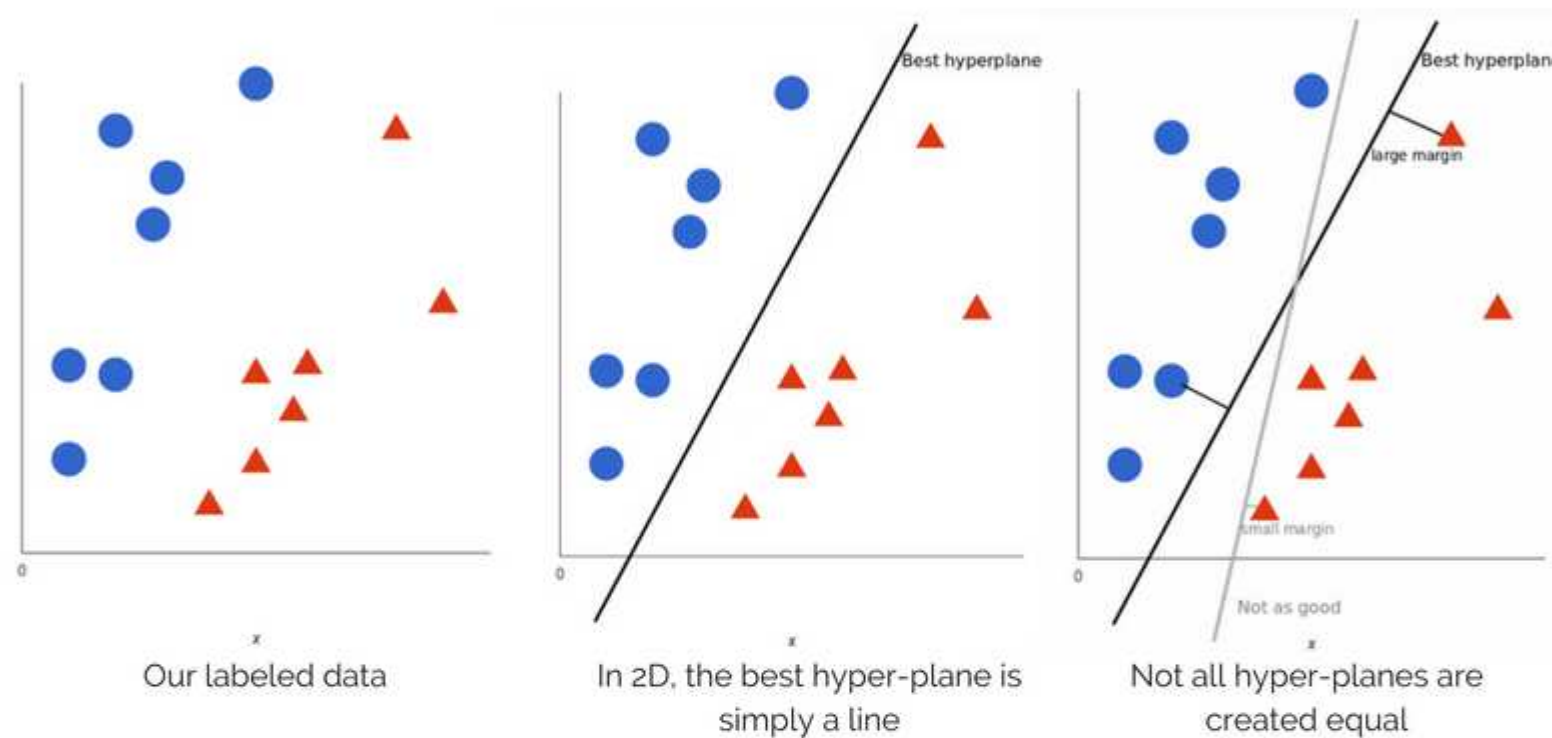
In this algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features) with the value of each feature being the value of a particular coordinate.

How is the data classified?

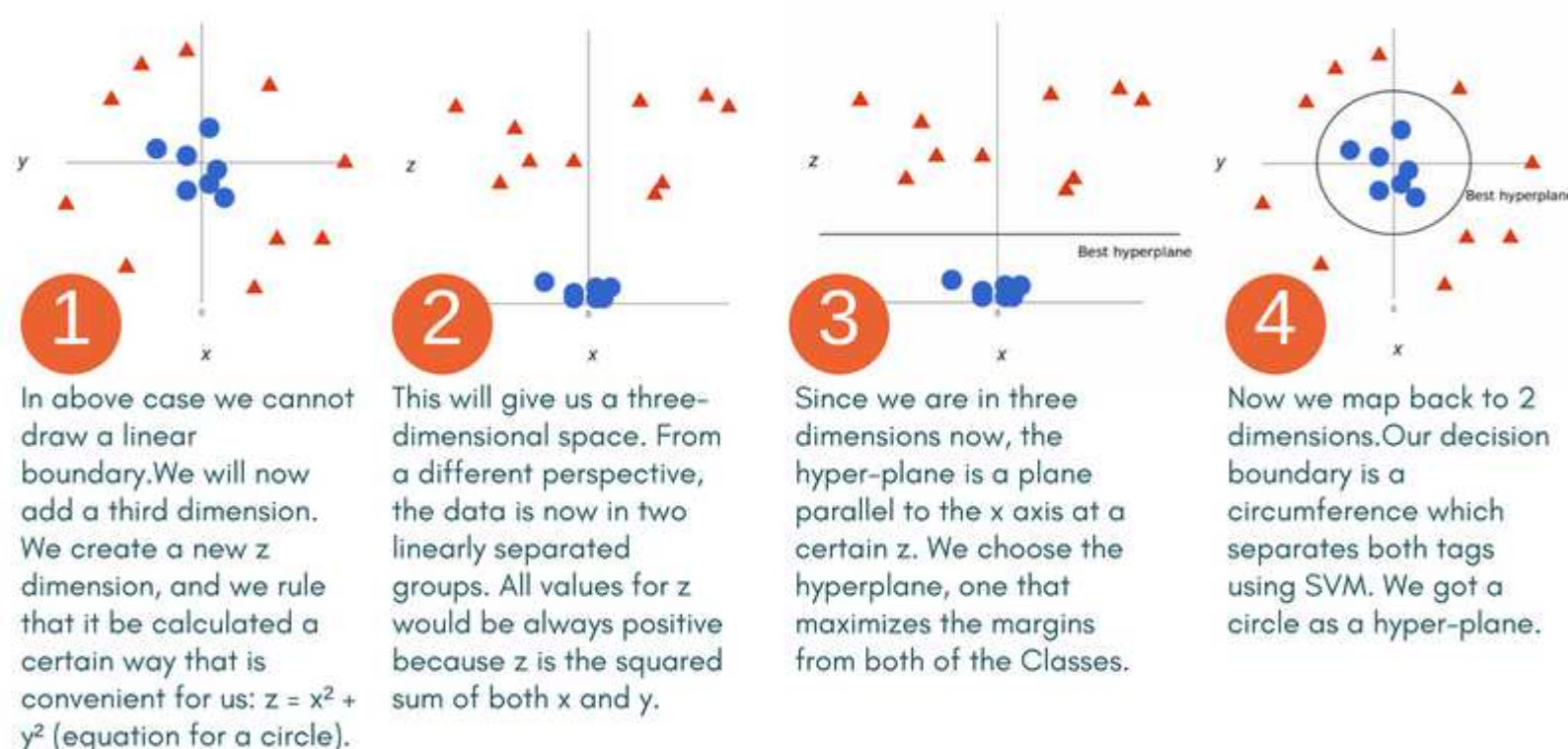
We perform classification by finding the hyperplane that differentiates the two classes very well. In other words the algorithm outputs an optimal hyperplane which categorizes new examples.

What is an optimal Hyper-Plane?

For SVM, it's the one that maximizes the margins from both tags. In other words: the hyperplane whose distance to the nearest element of each tag is the largest.



Nonlinear data



TUNING PARAMETERS

KERNEL

The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role. Polynomial and exponential kernels calculate separation line in higher dimension. This is called kernel trick.

GAMMA

The gamma parameter defines how far the influence of a single training set reaches. With low gamma, points far away from the possible separation line are considered in calculation for the separation line. Where as high gamma means the points close to possible line are considered in calculation.

REGULARIZATION

For large values of this parameter, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of it will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

MARGIN

A margin is a separation of line to the closest class points.

A good margin is one where this separation is larger for both the classes. A good margin allows the points to be in their respective classes without crossing to other class.

Check out The complete Implementation at: github.com/Avik-Jain/100-Days-Of-ML-Code

Follow Me For More Updates



Naive Bayes Classifier | Day 13

Continuing with #100DaysOfMLCode today I went through the Naive Bayes classifier. I am also implementing the SVM in python using scikit-learn. Will update the code soon.

Implementation of SVM | Day 14

Today I implemented SVM on linearly related data. Used Scikit-Learn library. In Scikit-Learn we have SVC classifier which we use to achieve this task. Will be using kernel-trick on next implementation. Check the code [here](#).

Naive Bayes Classifier and Black Box Machine Learning | Day 15

Learned about different types of naive bayes classifiers. Also started the lectures by [Bloomberg](#). First one in the playlist was Black Box Machine Learning. It gives the whole overview about prediction functions, feature extraction, learning algorithms, performance evaluation, cross-validation, sample bias, nonstationarity, overfitting, and hyperparameter tuning.

Implemented SVM using Kernel Trick | Day 16

Using Scikit-Learn library implemented SVM algorithm along with kernel function which maps our data points into higher dimension to find optimal hyperplane.

Started Deep learning Specialization on Coursera | Day 17

Completed the whole Week 1 and Week 2 on a single day. Learned Logistic regression as Neural Network.

Deep learning Specialization on Coursera | Day 18

Completed the Course 1 of the deep learning specialization. Implemented a neural net in python.

The Learning Problem , Professor Yaser Abu-Mostafa | Day 19

Started Lecture 1 of 18 of Caltech's Machine Learning Course - CS 156 by Professor Yaser Abu-Mostafa. It was basically an introduction to the upcoming lectures. He also explained Perceptron Algorithm.

Started Deep learning Specialization Course 2 | Day 20

Completed the Week 1 of Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization.

Web Scraping | Day 21

Watched some tutorials on how to do web scraping using Beautiful Soup in order to collect data for building a model.

Is Learning Feasible? | Day 22

Lecture 2 of 18 of Caltech's Machine Learning Course - CS 156 by Professor Yaser Abu-Mostafa. Learned about Hoeffding Inequality.

Decision Trees | Day 23

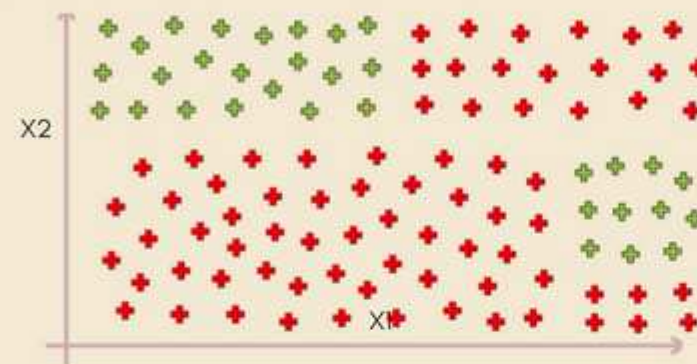
DECISION TREE

AN INTUITION ON DECISION TREE FOR CLASSIFICATION

1 WHAT IS A DECISION TREE?

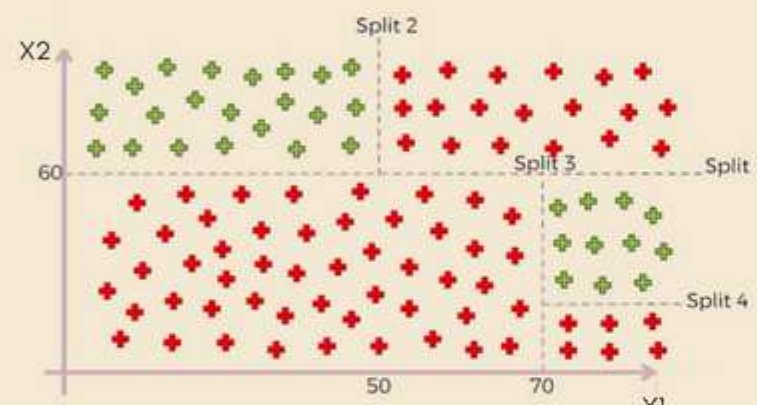
It is a type of supervised learning algorithm that is mostly used in classification problems and works for both categorical and continuous input and output variables.

A decision tree is a tree in which each branch node represents a choice between a number of alternatives and each leaf node represents a decision.

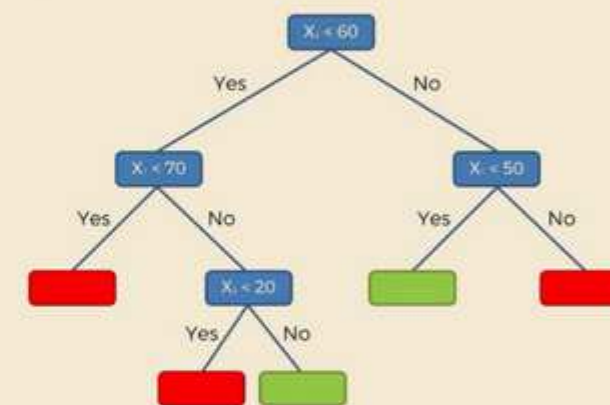


Here we've got an example with lots of points on our two dimensional scatter plot. Now how does a decision tree work.

So what it is going to do is cut it up into slices in several iterations.



We split the data and construct a decision tree side by side which we will use later. This very task is achieved by using various algorithms. It builds a decision tree from a fixed set of examples and the resulting tree is used to classify future samples.



The resulting Tree (obtained by applying algorithms like CART, ID3) which will be later used to predict the outcomes

3 DECISION TREE ALGORITHM: ID3

ID3 stands for Iterative Dichotomizer 3. The basic idea is to construct the decision tree by employing a top-down, greedy search through the given sets to test each attribute at every tree node.

Sounds simple – but which node should we select to build the correct and most precise decision tree? How would we decide that? Well, we have some measures that can help us in selecting the best choice!

Loop:
A → Best Attribute
Assign A as decision attribute for node.
For each value of A, create a descendant of node.
Sort training examples to leaves.
If examples perfectly classified:
STOP
Else:
Iterate over leaves

INFORMATION GAIN

The best attribute is the one which gives us maximum Information Gain. Broadly speaking, it is a mathematical way to capture the amount of information we want by picking a particular attribute. But what it really speaks about is the reduction in the randomness, over the tables that we have with the set of data, based upon knowing the value of a particular attribute. Information gain is defined by:

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_v \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

S = Collection of training examples

A = Particular attribute

|S_v| = Number of elements in S_v

|S| = Number of elements in S

v = All the possible values of the attribute

ENTROPY

Entropy in machine learning also carries almost the same meaning as it does in Thermodynamics. It is a measure of randomness.

$$\text{Entropy} = - \sum_v p(v) \log_2 p(v)$$

Where v = possible values for the attribute.

Steps :

1. compute the entropy for data-set

2. for every attribute/feature:

1. calculate entropy for all categorical values

2. take average information entropy for the current attribute

3. calculate gain for the current attribute

3. pick the highest gain attribute.

4. Repeat until we get the tree we desired

Check out the Repository at: github.com/Avik-Jain/100-Days-Of-ML-Code

Follow Me For More Updates



Implementing Decision Trees | Day 25

Check the code [here](#).

Jumped To Brush up Linear Algebra | Day 26

Found an amazing [channel](#) on youtube 3Blue1Brown. It has a playlist called Essence of Linear Algebra. Started off by completing 4 videos which gave a complete overview of Vectors, Linear Combinations, Spans, Basis Vectors, Linear Transformations and Matrix Multiplication.

Link to the playlist [here](#).

Jumped To Brush up Linear Algebra | Day 27

Continuing with the playlist completed next 4 videos discussing topics 3D Transformations, Determinants, Inverse Matrix, Column Space, Null Space and Non-Square Matrices.

Link to the playlist [here](#).

Jumped To Brush up Linear Algebra | Day 28

In the playlist of 3Blue1Brown completed another 3 videos from the essence of linear algebra. Topics covered were Dot Product and Cross Product.

Link to the playlist [here](#).

Jumped To Brush up Linear Algebra | Day 29

Completed the whole playlist today, videos 12-14. Really an amazing playlist to refresh the concepts of Linear Algebra. Topics covered were the change of basis, Eigenvectors and Eigenvalues, and Abstract Vector Spaces.

Link to the playlist [here](#).

Essence of calculus | Day 30

Completing the playlist - Essence of Linear Algebra by 3blue1brown a suggestion popped up by youtube regarding a series of videos again by the same channel 3Blue1Brown. Being already impressed by the previous series on Linear algebra I dived straight into it. Completed about 5 videos on topics such as Derivatives, Chain Rule, Product Rule, and derivative of exponential.

Link to the playlist [here](#).

Essence of calculus | Day 31

Watched 2 Videos on topic Implicit Diffrentiation and Limits from the playlist Essence of Calculus.

Link to the playlist [here](#).

Essence of calculus | Day 32

Watched the remaining 4 videos covering topics Like Integration and Higher order derivatives.

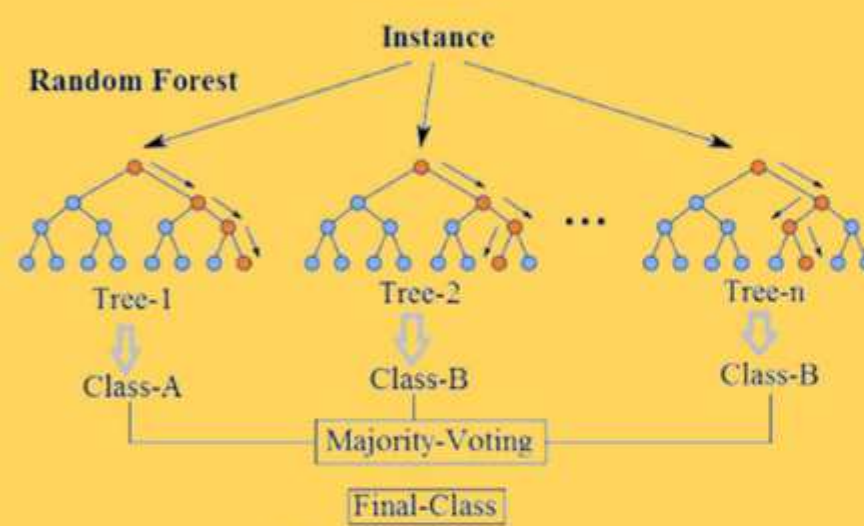
Link to the playlist [here](#).

Random Forests | Day 33

AN INTUITION TO RANDOM FOREST

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

The logic behind this is that each of the models used is weak when employed on its own, but strong when put together in an ensemble. In the case of Random Forests, a large number of Decision Trees, acting as the “weak” factors, are used and their outputs are aggregated, with the result representing the “strong” ensemble.



1. If the number of cases in the training set is N , sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.
2. If there are M input variables, a number is specified such that at each node, m variables are selected at random out of the M and the best split on this m is used to split the node.

1. Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target)
2. Calculate the votes for each predicted target
3. Consider the high voted predicted target as the final prediction from the random forest algorithm

Follow Me For More Updates



An Amazing Video on neural networks by 3Blue1Brown youtube channel. This video gives a good understanding of Neural Networks and uses Handwritten digit dataset to explain the concept. Link To the [video](#).

Gradient descent, how neural networks learn | Deep learning, chapter 2 | Day 36

Part two of neural networks by 3Blue1Brown youtube channel. This video explains the concepts of Gradient Descent in an interesting way. 169 must watch and highly recommended. Link To the [video](#).

What is backpropagation really doing? | Deep learning, chapter 3 | Day 37

Part three of neural networks by 3Blue1Brown youtube channel. This video mostly discusses the partial derivatives and backpropagation. Link To the [video](#).

Backpropagation calculus | Deep learning, chapter 4 | Day 38

Part four of neural networks by 3Blue1Brown youtube channel. The goal here is to represent, in somewhat more formal terms, the intuition for how backpropagation works and the video moslty discusses the partial derivatives and backpropagation. Link To the [video](#).

Deep Learning with Python, TensorFlow, and Keras tutorial | Day 39

Link To the [video](#).

Loading in your own data - Deep Learning basics with Python, TensorFlow and Keras p.2 | Day 40

Link To the [video](#).

Convolutional Neural Networks - Deep Learning basics with Python, TensorFlow and Keras p.3 | Day 41

Link To the [video](#).

Analyzing Models with TensorBoard - Deep Learning with Python, TensorFlow and Keras p.4 | Day 42

Link To the [video](#).

K Means Clustering | Day 43

Moved to Unsupervised Learning and studied about Clustering. Working on my website check it out [avikjain.me](#) Also found a wonderful animation that can help to easily understand K - Means Clustering [Link](#)

AN INTUITION TO K-Means Clustering

STARTING WITH UNSUPERVISED LEARNING

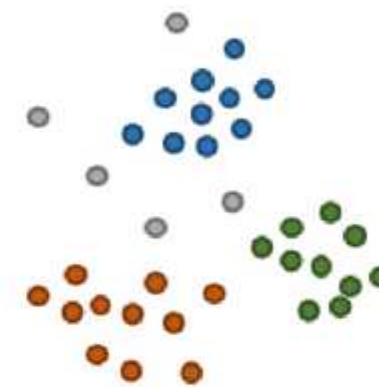


1.) WHAT IS UNSUPERVISED LEARNING

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. Unsupervised algorithms find patterns based only on input data. This technique is useful when we're not quite sure what to look for.

2.) CLUSTERING ALGORITHMS

Clustering Algorithms do the task of dividing the population or data points into a variety of groups such that data points within the same cluster are similar to other data points within the same cluster than those in other groups. Basically, the aim is to separate groups with similar traits and assign them into clusters.



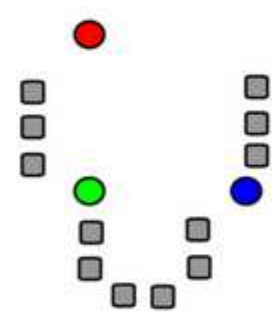
3.) K MEANS CLUSTERING



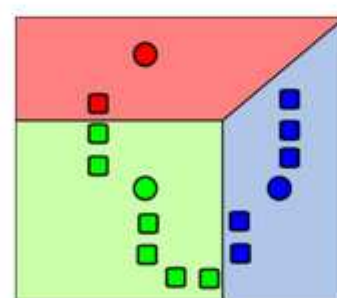
In this algorithm, we group the items into k clusters such that all items in the same cluster are as similar to each other as possible. And items not in the same cluster are as different as possible.

Distance measures (like Euclidean distance) are used to calculate similarity and dissimilarity between the data points. Each cluster has a centroid. Centroid can be thought as the point that is most representative of the cluster.

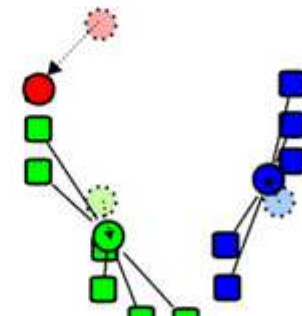
4.) HOW K-MEANS CLUSTERING WORKS



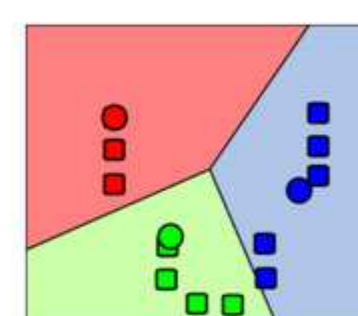
1. k initial "means" (in this case k=3) are randomly generated within the data domain.



2. k clusters are created by associating every observation with the nearest mean.



3. The centroid of each of the k clusters becomes the new mean.



4. Steps 2 and 3 are repeated until convergence has been reached.

The objective of K-Means clustering is to minimize total intra-cluster variance, or, the squared error function:

$$J = \sum_{j=1}^k \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2$$

Labels in the diagram: 'number of clusters' points to k, 'number of cases' points to n, 'case i' points to i, 'centroid for cluster j' points to c_j, 'Distance function' points to the norm symbol, and 'objective function' points to J.

CHECK OUT THE REPOSITORY AT - [GITHUB.COM/AVIK-JAIN/100-DAYS-OF-ML-CODE](https://github.com/Avik-Jain/100-Days-Of-ML-Code)

Follow Me For More Updates



K Means Clustering Implementation | Day 44

Implemented K Means Clustering. Check the code [here](#).

Digging Deeper | NUMPY | Day 45

Got a new book "Python Data Science HandBook" by JK VanderPlas Check the Jupyter notebooks [here](#).
Started with chapter 2 : Introduction to Numpy. Covered topics like Data Types, Numpy arrays and Computations on Numpy arrays.
Check the code -
[Introduction to NumPy](#)
[Understanding Data Types in Python](#)
[The Basics of NumPy Arrays](#)
[Computation on NumPy Arrays: Universal Functions](#)

Digging Deeper | NUMPY | Day 46

Chapter 2 : Aggregations, Comparisons and Broadcasting
Link to Notebook:
[Aggregations: Min, Max, and Everything In Between](#)
[Computation on Arrays: Broadcasting](#)
[Comparisons, Masks, and Boolean Logic](#)

Digging Deeper | NUMPY | Day 47

Chapter 2 : Fancy Indexing, sorting arrays, Struchered Data
Link to Notebook:
[Fancy Indexing](#)
[Sorting Arrays](#)
[Structured Data: NumPy's Structured Arrays](#)

Digging Deeper | PANDAS | Day 48

Chapter 3 : Data Manipulation with Pandas
Covered Various topics like Pandas Objects, Data Indexing and Selection, Operating on Data, Handling Missing Data, Hierarchical Indexing, ConCat and Append.
Link To the Notebooks:
[Data Manipulation with Pandas](#)
[Introducing Pandas Objects](#)
[Data Indexing and Selection](#)
[Operating on Data in Pandas](#)
[Handling Missing Data](#)
[Hierarchical Indexing](#)
[Combining Datasets: Concat and Append](#)

Digging Deeper | PANDAS | Day 49

Chapter 3: Completed following topics- Merge and Join, Aggregation and grouping and Pivot Tables.
[Combining Datasets: Merge and Join](#)
[Aggregation and Grouping](#)
[Pivot Tables](#)

Digging Deeper | PANDAS | Day 50

Chapter 3: Vectorized Strings Operations, Working with Time Series
Links to Notebooks:
[Vectorized String Operations](#)
[Working with Time Series](#)
[High-Performance Pandas: eval\(\) and query\(\)](#)

Digging Deeper | MATPLOTLIB | Day 51

Chapter 4: Visualization with Matplotlib Learned about Simple Line Plots, Simple Scatter Plotsand Density and Contour Plots.
Links to Notebooks:
[Visualization with Matplotlib](#)
[Simple Line Plots](#)
[Simple Scatter Plots](#)
[Visualizing Errors](#)
[Density and Contour Plots](#)

Digging Deeper | MATPLOTLIB | Day 52

Chapter 4: Visualization with Matplotlib Learned about Histograms, How to customize plot legends, colorbars, and buliding Multiple Subplots.
Links to Notebooks:
[Histograms, Binnings, and Density](#)
[Customizing Plot Legends](#)
[Customizing Colorbars](#)
[Multiple Subplots](#)
[Text and Annotation](#)

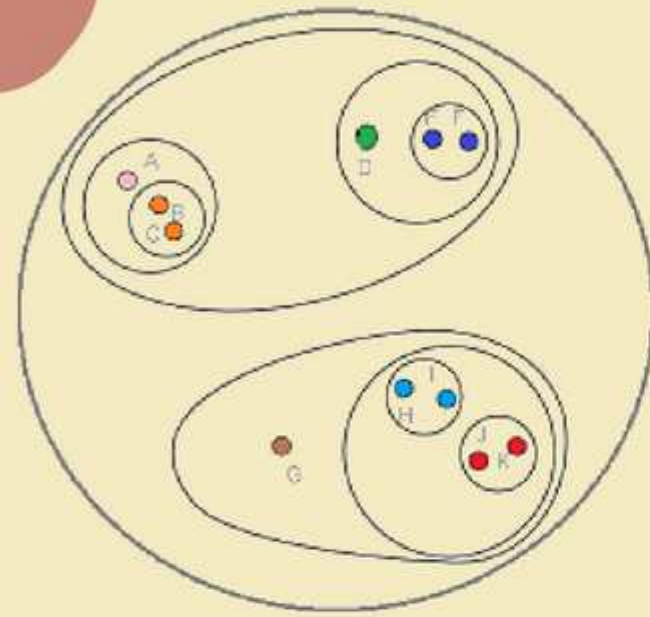
Digging Deeper | MATPLOTLIB | Day 53

Chapter 4: Covered Three Dimensional Plotting in Mathplotlib.
Links to Notebooks:
[Three-Dimensional Plotting in Matplotlib](#)

Hierarchical Clustering | Day 54

Studied about Hierarchical Clustering. Check out this amazing [Visualization](#).

HIERARCHICAL CLUSTERING

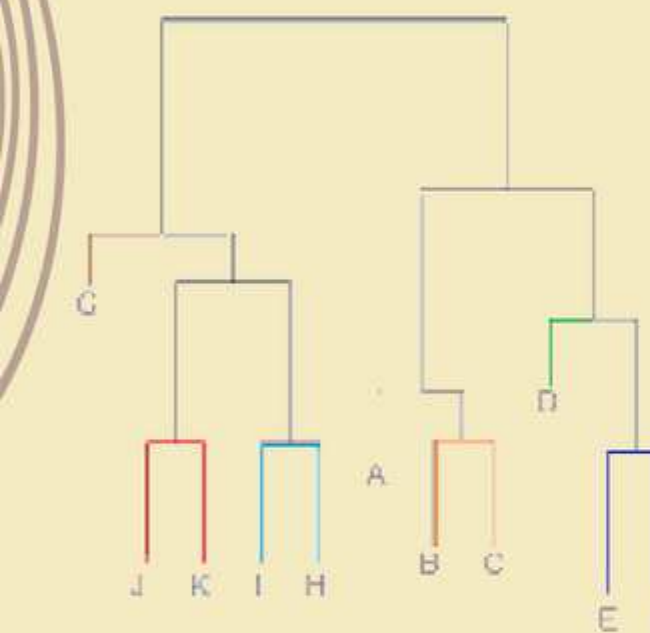


HIERARCHICAL CLUSTERING

Hierarchical clustering, as the name suggests is an algorithm that builds a hierarchy of clusters. This algorithm starts with all the data points assigned to a cluster of their own. Then two nearest clusters are merged into the same cluster. In the end, this algorithm terminates when there is only a single cluster left. There are two types of hierarchical clustering, Divisive and Agglomerative.

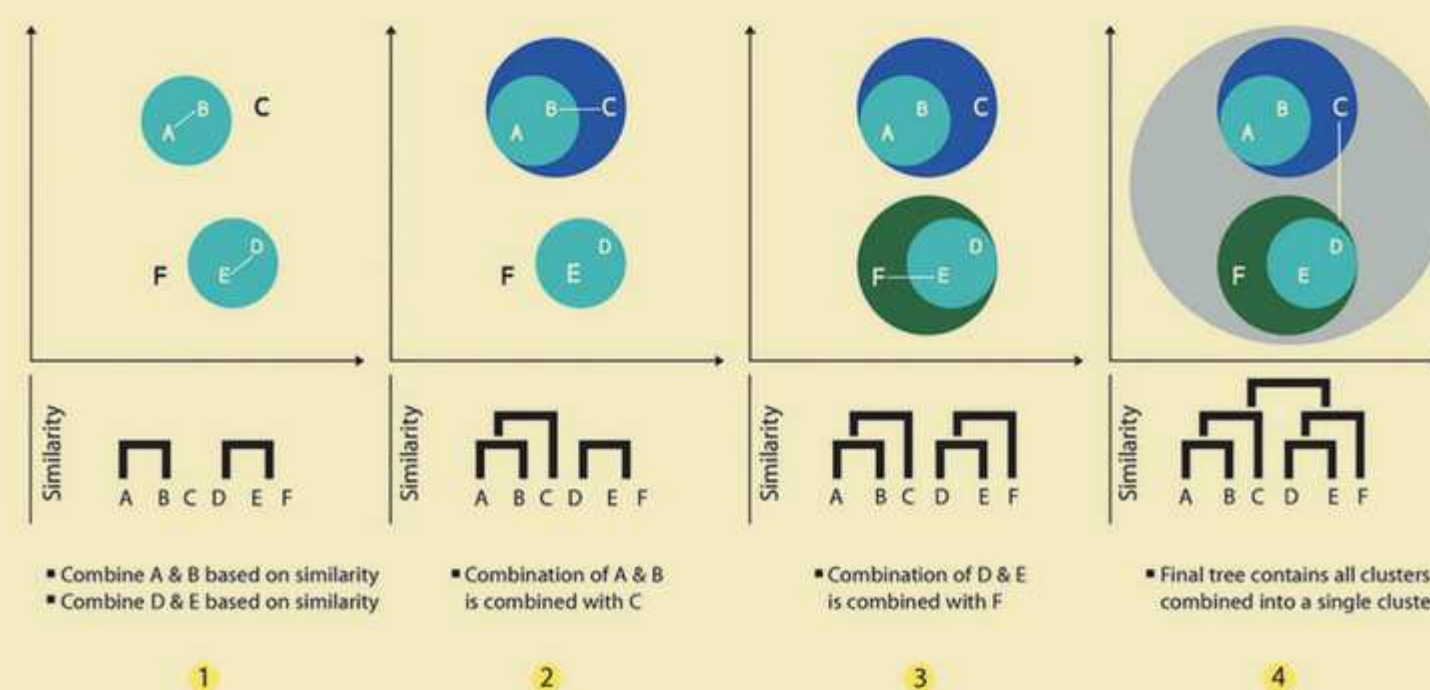
AGGLOMERATIVE HIERARCHICAL CLUSTERING

Here, each observation is initially considered as a cluster of its own (leaf). Then, the most similar clusters are successively merged until there is just one single big cluster (root). This hierarchy of clusters is represented as a tree (or dendrogram).



DENDROGRAM

The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample.



The decision of the no. of clusters that can best depict different groups can be chosen by observing the dendrogram. The best choice of the no. of clusters is the no. of vertical lines in the dendrogram cut by a horizontal line that can transverse the maximum distance vertically without intersecting a cluster.

CHECK OUT THE REPOSITORY AT - [GITHUB.COM/AVIK-JAIN/100-DAYS-OF-ML-CODE](https://github.com/AVIK-JAIN/100-DAYS-OF-ML-CODE)

Follow Me For More Updates

