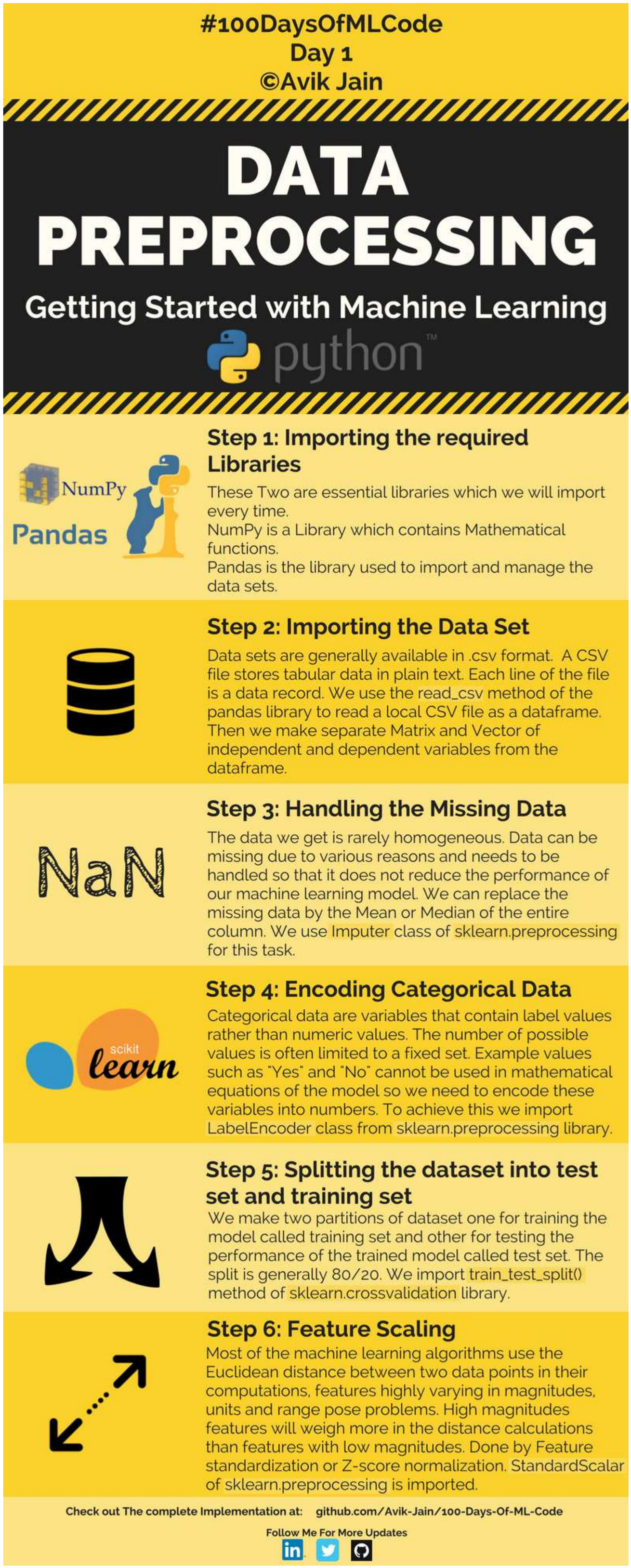
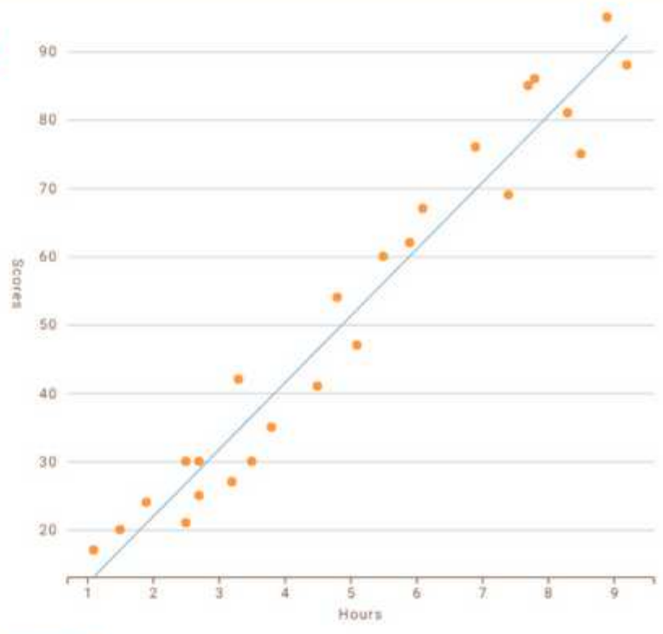


Simple Linear Regression | Day 2

Check out the code from here.

#100DaysOfMLCode
Day 2
©Avik Jain

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).

Dependent Variable

$$y = b_0 + b_1 x_1$$

Independent Variable

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).

Predicted Value

Observed Value

$$y_i \quad y_p$$

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

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.

Slope

$$Score = b_0 + b_1 * hours$$

Y - intercept

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

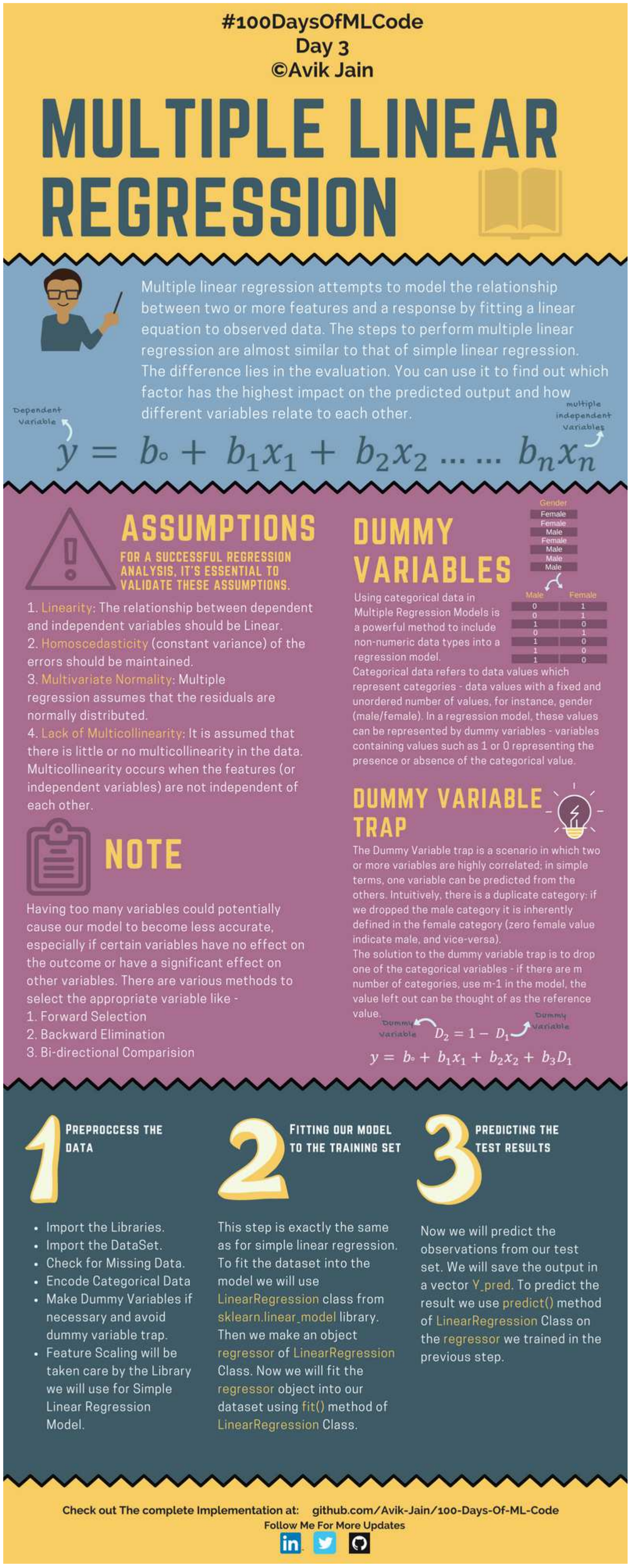
The final step is to visualize our results. We will use `matplotlib.pyplot` library to make Scatter Plots of our Training set results and Test set results to see how close our model predicted the Values

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

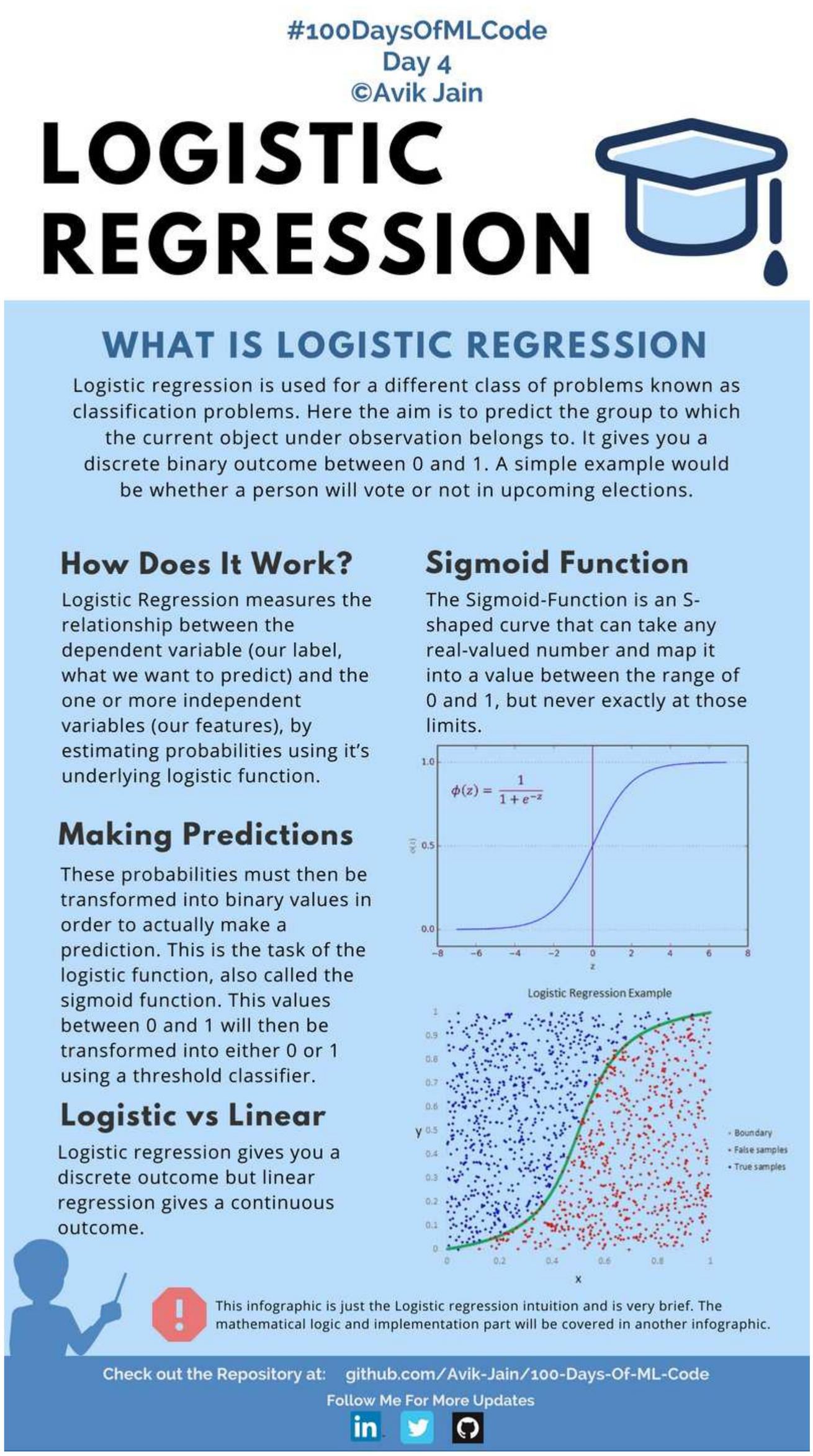
Follow Me For More Updates

Multiple Linear Regression | Day 3

Check out the code from here.



Logistic Regression | Day 4



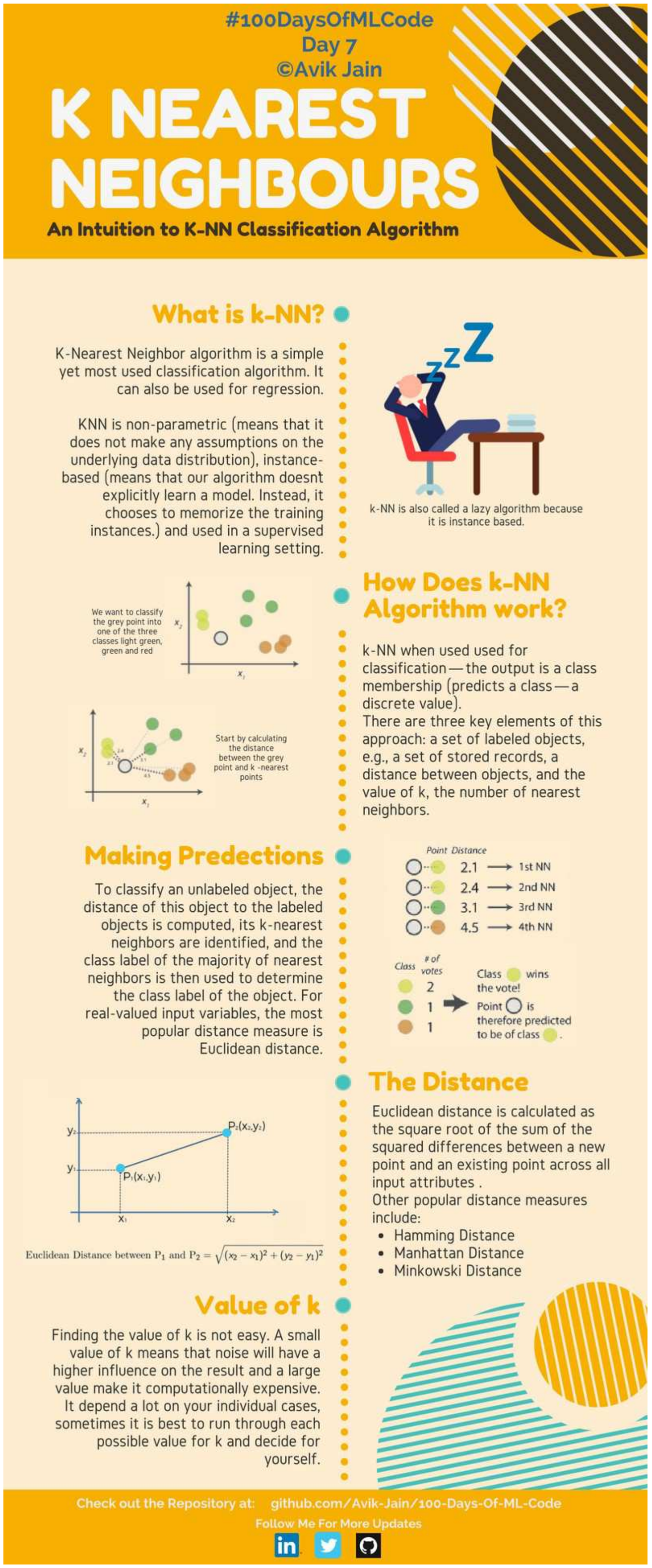
Logistic Regression | Day 5

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



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 intution 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

#100DaysOfMLCode
Day 12
©Avik Jain

SUPPORT VECTOR MACHINES

What is SVM?

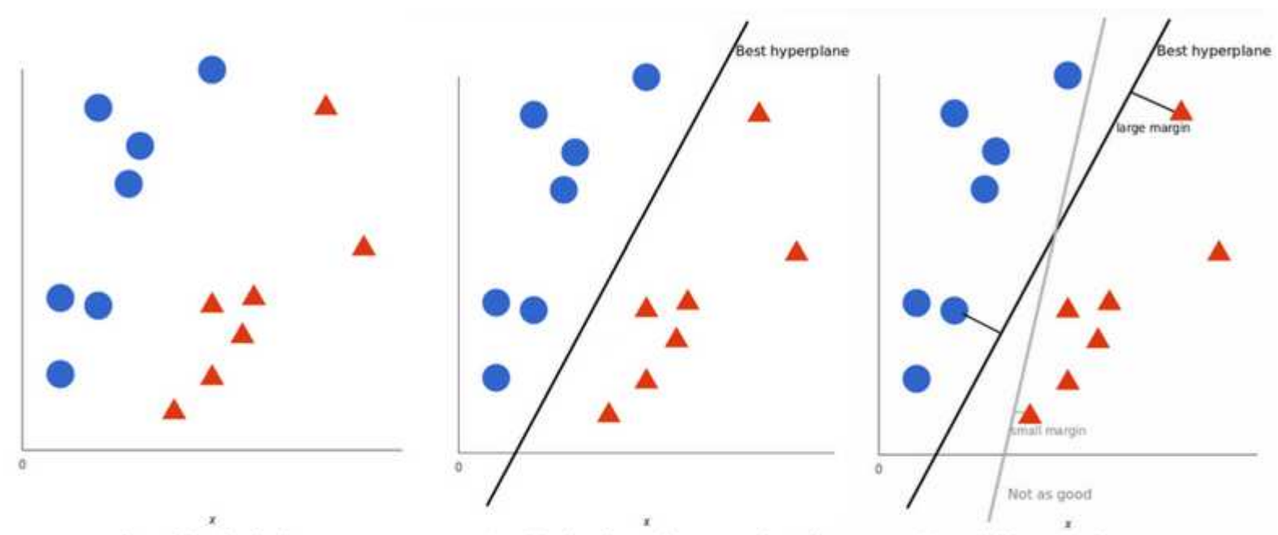
Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression. However, it is mostly used in classification problems.
In this algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features) with 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.

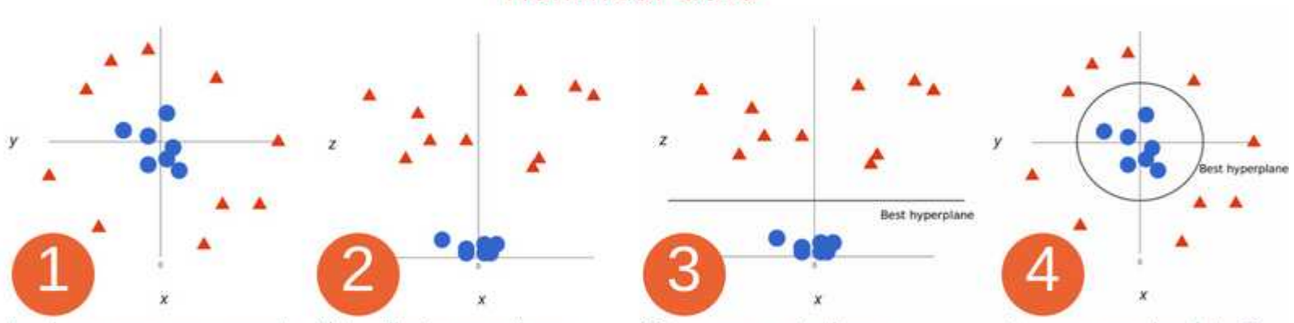
What is a optimal Hyper-Plane?

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.



Our labeled data | In 2D, the best hyper-plane is simply a line | Not all hyper-planes are created equal

Nonlinear data



In above case we cannot draw a linear boundary.We will now add a third dimension. We create a new z dimension, and we rule that it be calculated a certain way that is convenient for us: $z = x^2 + y^2$ (equation for a circle).

This will give us a three-dimensional space. From a different perspective, the data is now in two linearly separated groups. All values for z would be always positive because z is the squared sum of both x and y.

Since we are in three dimensions now, the hyper-plane is a plane parallel to the x axis at a certain z. We choose the hyperplane, one that maximizes the margins from both of the Classes.

Now we map back to 2 dimensions.Our decision boundary is a circumference which separates both tags using SVM. We got a circle as a hyper-plane.

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 calculates separation line in higher dimension. This is called kernel trick.

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.

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.

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

#100DAYSOFMLCODE
DAY 23
©AVIK JAIN

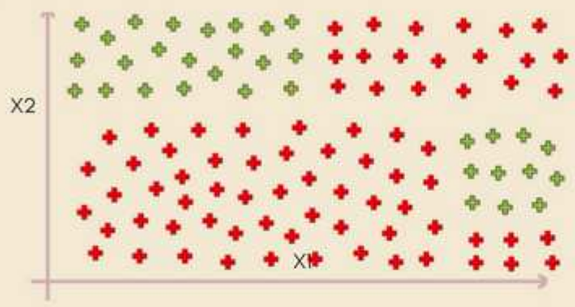
DECISION TREE

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.

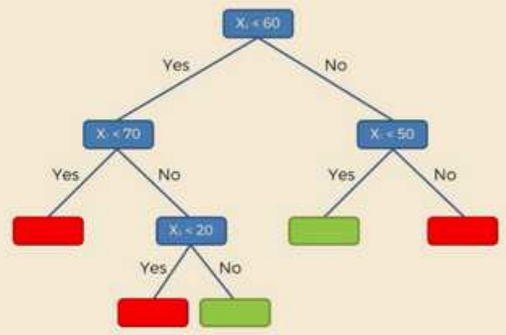
A decision tree is a tree in which each branch node represents a choice between a number of alternatives and each leaf node represents a decision.



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 us 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:

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

S = Collection of training examples
A = Particular attribute
|Sv| = Number of elements in Sv
|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.

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

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

Introduction To Statistical Learning Theory | Day 24

Lec 3 of Bloomberg ML course introduced some of the core concepts like input space, action space, outcome space, prediction functions, loss functions, and hypothesis spaces.

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.

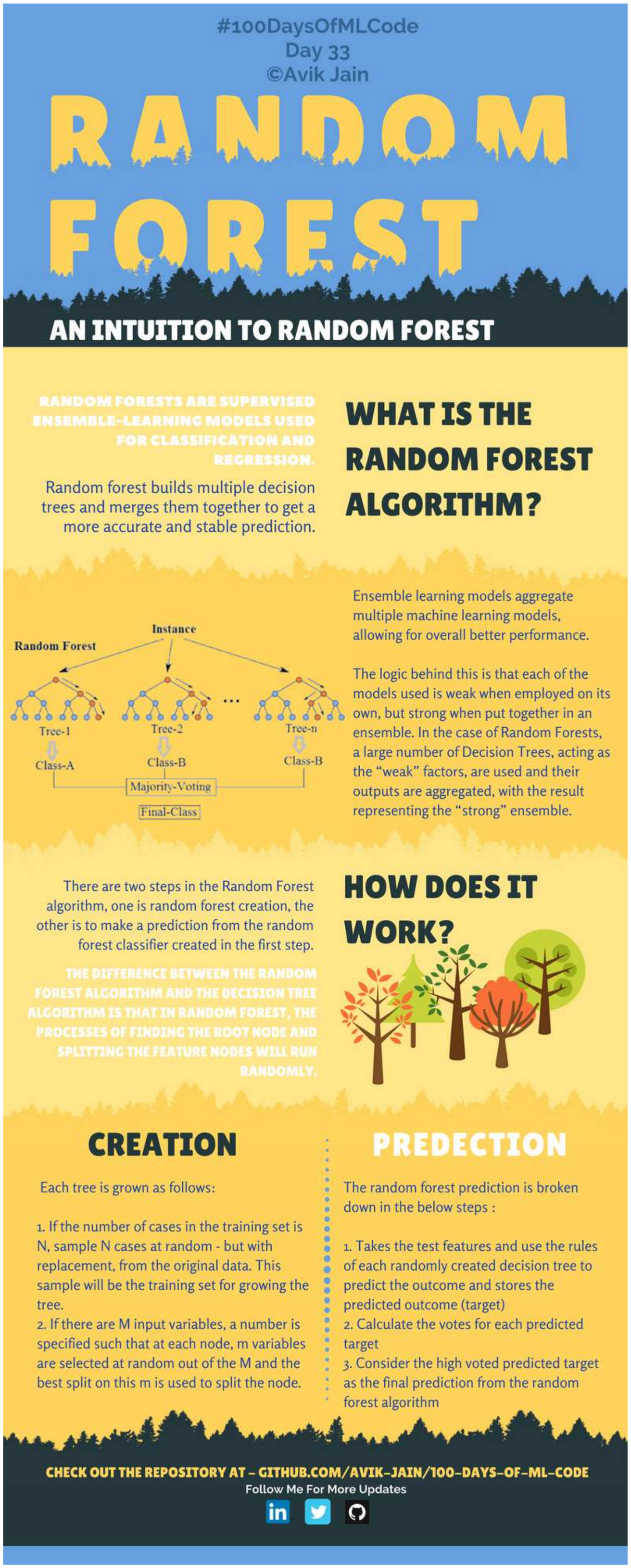
Link to the playlist here.

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



Implementing Random Forests | Day 34

Check the code here.

But what *is* a Neural Network? | Deep learning, chapter 1 | Day 35

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

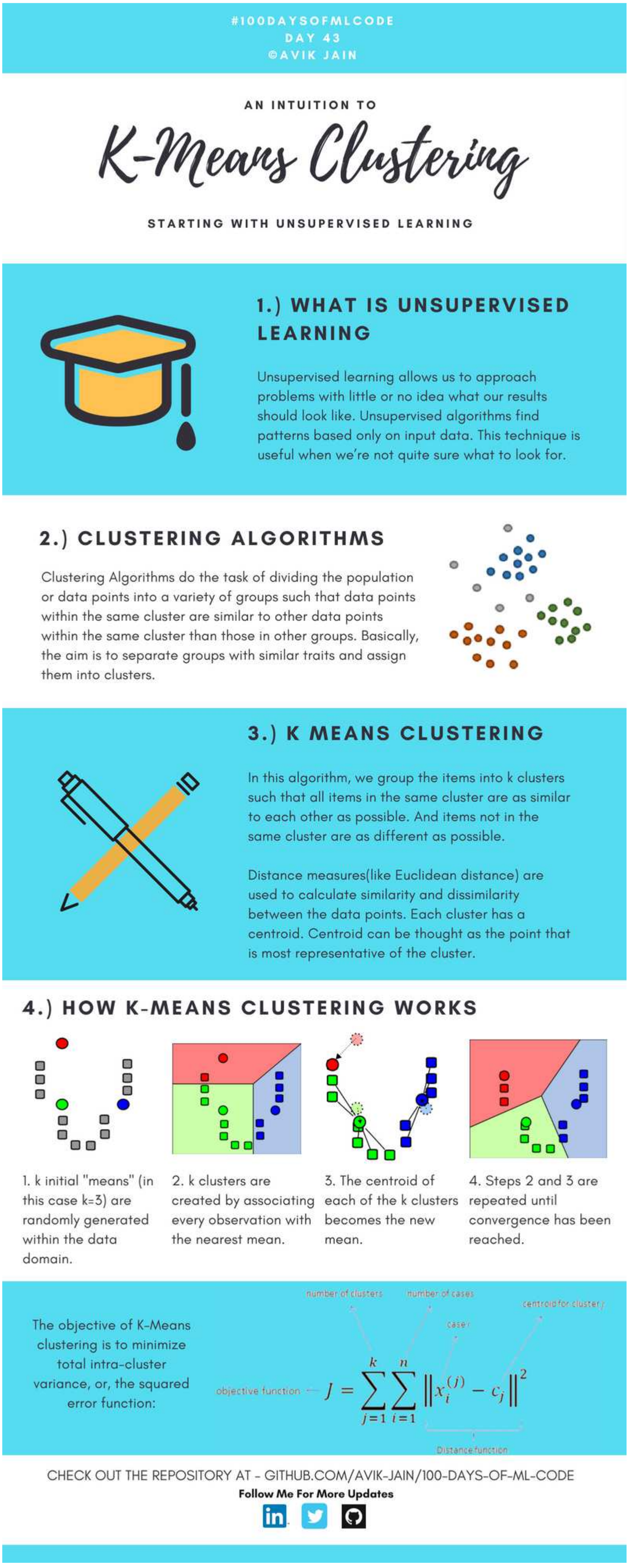
Link To the video.

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



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, Comparisions 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

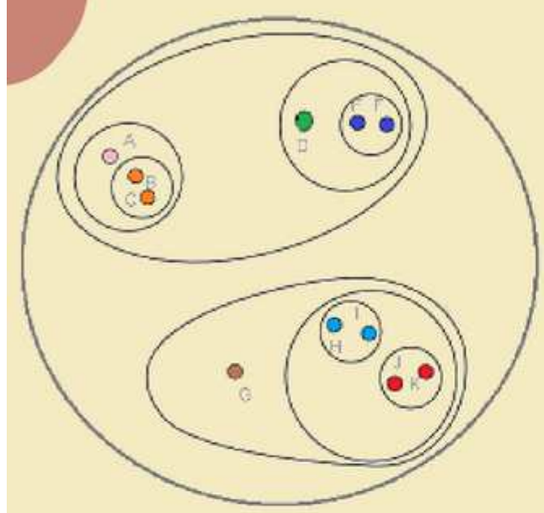
Chapter 4: Covered Three Dimensional Plotting in Mathplotlib.
Links to Notebooks:
Three-Dimensional Plotting in Matplotlib

Hierarchical Clustering | Day 54

Studied about Hierarchical Clustering. Check out this amazing Visualization.

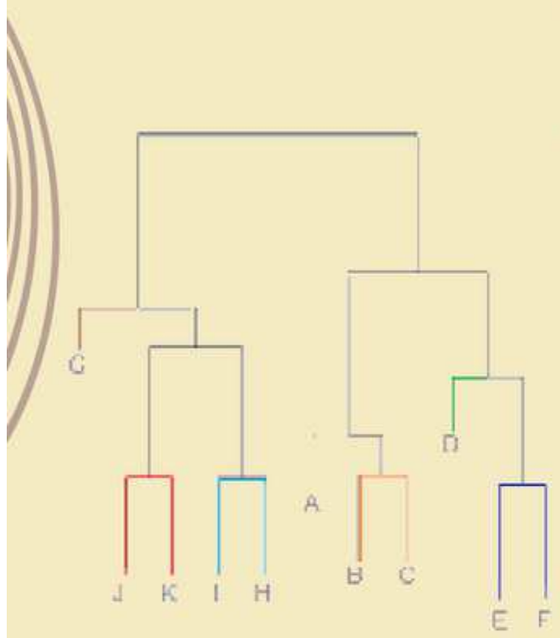
#100DAYSOFMLCODE
DAY 54
©AVIK JAIN

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.

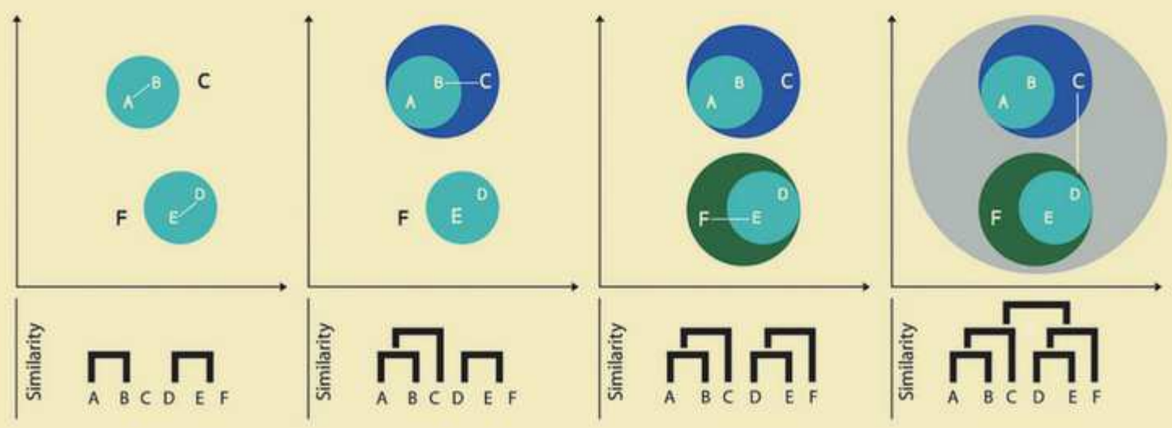


AGGLOMERATIVE HIERARCHICAL CLUSTERING

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.



- Combine A & B based on similarity
- Combine D & E based on similarity
- Combination of A & B is combined with C
- Combination of D & E is combined with F
- Final tree contains all clusters combined into a single cluster

1 2 3 4

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

Follow Me For More Updates