ALTIMETRIK

Data Science Bootcamp Documentation

DOCUMENTED BY

Varsha S

MARCH 2022

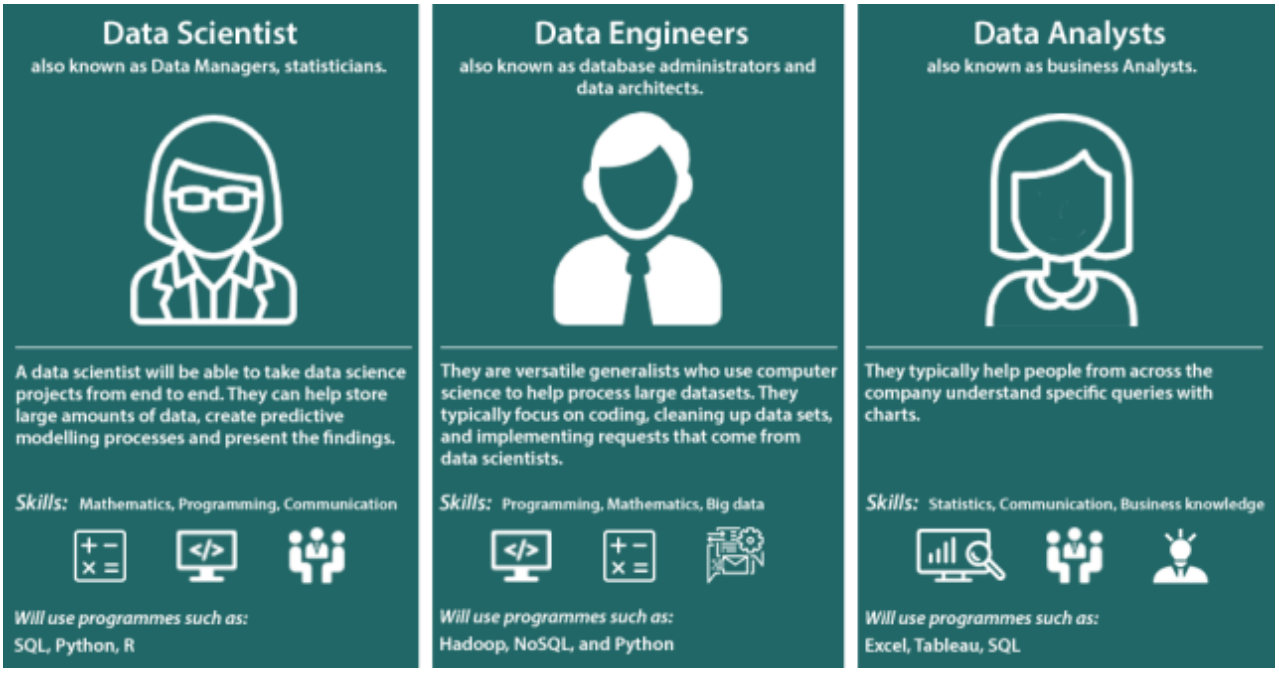
Table of Contents

DOCUMENTED BY : VARSHA S

Data Engineer and a Data Scientist

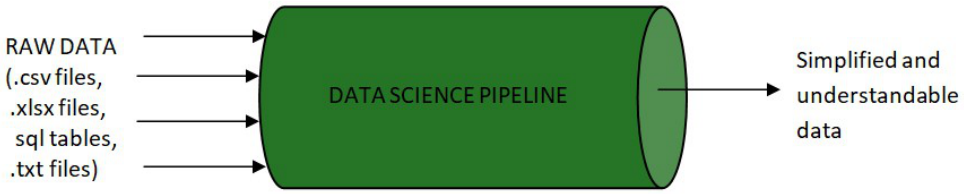
Who's who on the data analytics team

A data scientist	A data engineer
■ Identifies relevant data sets needed for data analytics applications.	■ Builds data pipelines to pull together data from different systems.
■ Works with others to collect, integrate and prepare data for analysis.	■ Aids data scientists in integrating, consolidating and cleansing data.
■ Develops, "trains" and runs analytical models and assesses their findings.	■ Structures and organizes data for use in specific analytics applications.
■ Communicates the results to business executives and other end users.	



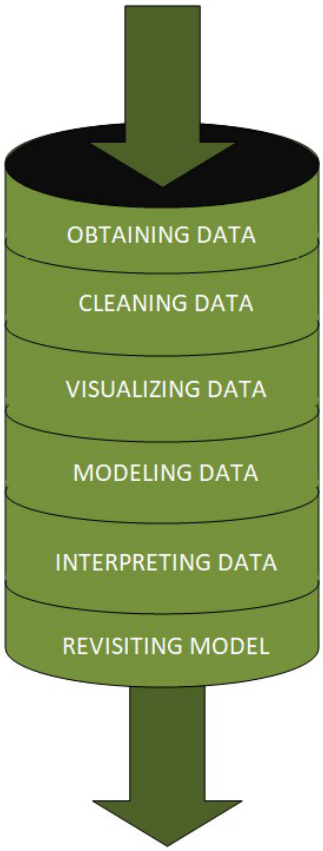
Data Science Pipeline

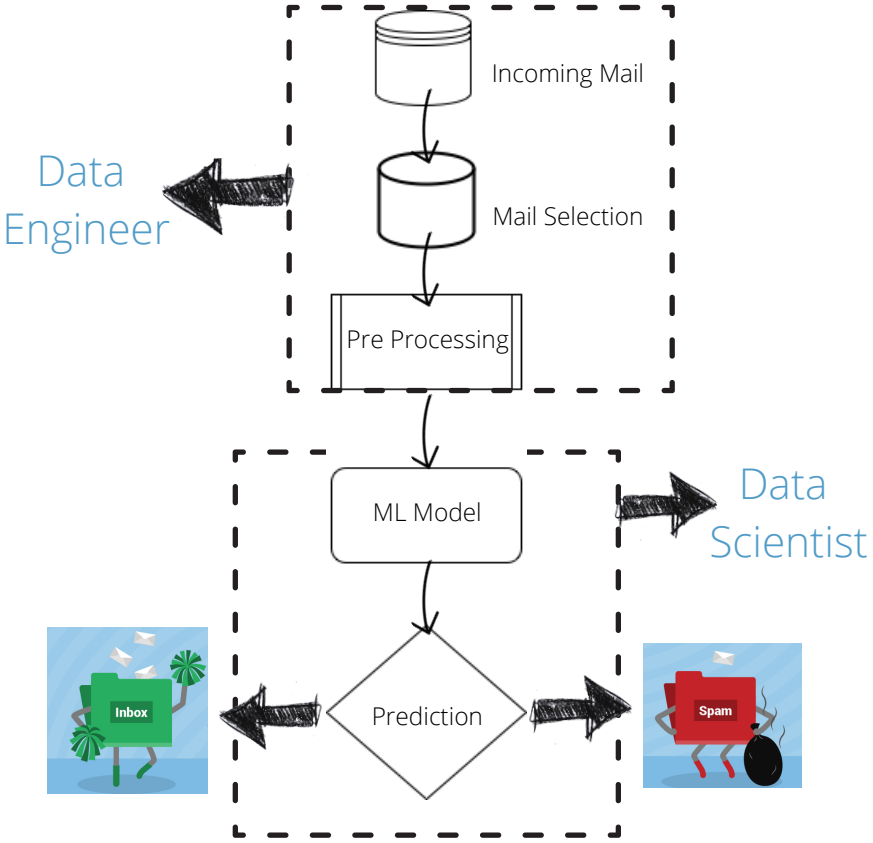
A pipeline in data science is "a set of actions which changes the raw (and confusing) data from various sources (surveys, feedback, list of purchases, votes, etc.), to an understandable format so that we can store it and use it for analysis."



Data science pipeline in a simplified way

The raw data undergoes different stages within a pipeline:





This Flowchart explains the flow of the pipeline in an email spam detection process.

Data Cleaning

The three popular data cleansing steps are,

1. Handling missing data
2. Scaling and Normalization
3. Parsing Dates

- **Handling missing data**

How many missing data points do we have?

 # get the number of missing data points per column

>> missing_values_count = data.isnull().sum()

Figure out why the data is missing.
Is this value missing because it wasn't recorded or because it doesn't exist?

Drop missing values

remove all the rows that contain a missing value

>> data.dropna()

remove all columns with at least one missing value

>> columns_with_na_dropped = data.dropna(axis=1)

Filling in missing values automatically

replace all NA's with 0.

>> data.fillna(0)

replace all NA's the value that comes directly after it in the same column,
then replace all the remaining na's with 0.

>> data.fillna(method='bfill', axis=0).fillna(0)

Data Cleaning

HANDLING MISSING DATA

1) Take a first look at the data

```
# To print first five rows of the data.To check if the data contains any NaN values.
sf_permits.head()
```

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Street Suffix	...	Existing Construction Type	Existing Construction Type Description	Proposed Construction Type	Pr Const Desc
0	201505065519	4	sign - erect	05/06/2015	0326	023	140	NaN	Ellis	St	...	3.0	constr type 3	NaN	
1	201604195146	4	sign - erect	04/19/2016	0306	007	440	NaN	Geary	St	...	3.0	constr type 3	NaN	
2	201605278609	3	additions alterations or repairs	05/27/2016	0595	203	1647	NaN	Pacific	Av	...	1.0	constr type 1	1.0	const
3	201611072166	8	otc alterations permit	11/07/2016	0156	011	1230	NaN	Pacific	Av	...	5.0	wood frame (5)	5.0	woo
4	201611283529	6	demolitions	11/28/2016	0342	001	950	NaN	Market	St	...	3.0	constr type 3	NaN	

5 rows × 43 columns

The first five rows of the data does show that several columns have missing values.

2) How many missing data points do we have?

```
# How many missing data points do we have? What percentage of the values in the dataset are missing?
missing_values = sf_permits.isnull().sum()
print("Total number of missing valus: ",missing_values.sum())
total_cells = np.product(sf_permits.shape) #to find total number of cells
total_missing = missing_values.sum() #to find total number of missing data

percent= (total_missing/total_cells)*100
print("Percentage of total values missing: ",percent)
percent_missing = 26.26002315058403
# Check your answer
q2.check()
```

```
Total number of missing valus:  2245941
Percentage of total values missing:  26.26002315058403
```

Correct

Data Cleaning

HANDLING MISSING DATA

3) Figure out why the data is missing

Look at the columns **"Street Number Suffix"** and **"Zipcode"** from the San Francisco Building Permits dataset. Both of these contain missing values.

- Which, if either, are missing because they don't exist?
- Which, if either, are missing because they weren't recorded?

Once you have an answer, run the code cell below.

```
sf_permits[['Street Number Suffix','Zipcode']]
```

	Street Number Suffix	Zipcode
0	NaN	94102.0
1	NaN	94102.0
2	NaN	94109.0
3	NaN	94109.0
4	NaN	94102.0
...	...	...
198895	NaN	NaN
198896	NaN	NaN
198897	NaN	NaN
198898	NaN	NaN
198899	NaN	NaN

198900 rows × 2 columns

If a value in the "Street Number Suffix" column is missing, it is likely because it does not exist. If a value in the "Zipcode" column is missing, it was not recorded.

4) Drop missing values: rows

```
sf_permits.dropna() #drops all the row with missing values
```

[18...

Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Street Suffix	...	Existing Construction Type

0 rows × 43 columns

+ Code + Markdown

Data Cleaning

HANDLING MISSING DATA

5) Drop missing values: columns

```
sf_permits_with_na_dropped = sf_permits.dropna(axis=1)
dropped_columns = sf_permits.shape[1]-sf_permits_with_na_dropped.shape[1]

print("Columns in original dataset: %d \n" % sf_permits.shape[1])
print("No of Columns dropped: %d" % dropped_columns )
print("Columns in new dataset:%d " % sf_permits_with_na_dropped.shape[1])

# Check your answer
q5.check()
```

```
Columns in original dataset: 43

No of Columns dropped: 31
Columns in new dataset:12
```

6) Fill in missing values automatically

```
sf_permits.fillna(0) #fills all the Na values in original data with 0
sf_permits_with_na_imputed = sf_permits.fillna(method='bfill', axis=0).fillna(0)
#the result is not set to a new dataframe
```

Data Cleaning

- Why Scaling and Normalization of Data? - (REAL-TIME EXAMPLE)

In real life,
If we take an example of **purchasing apples from a bunch of apples**, we go close to the shop, examine various apples and pick various apples of the **same attributes**.
Because we have learned about the attributes of apples and we know which are better and which are not good also we know which attributes can be compromised and which can not.
So if most of the apples consist of pretty similar attributes we will take less time in the selection of the apples which directly affect the time of purchasing taken by us.
The moral of the example is if the apples every apple in the shop is good we will take less time to purchase or if the apples are not good enough we will take more time in the selection process which means that if the values of attributes are closer we will work faster and the chances of selecting good apples also strong.

Similarly in the machine learning algorithms if the values of the features are closer to each other there are chances for the algorithm to get trained well and faster instead of the data set where the data points or features values have high differences with each other will take more time to understand the data and the accuracy will be lower.

Data Cleaning

- Scaling and Normalization Data

They are very similar! In both cases, you're transforming the values of numeric variables so that the transformed data points have specific helpful properties.

In scaling, you're changing the range of your data (transforms the data between 0 and 1)

In normalization, you're changing the shape of the distribution of your data. (transforms under a distribution)

When do scale a data ?

It is done when we're using methods based on measures of how far apart data points are, like *support vector machines (SVM)* or *k-nearest neighbors (KNN)*.

When do normalise a data ?

Data is made to undergo normalisation if we're going to be using a machine learning or statistics technique that assumes our data is normally distributed.
When you don't know the distribution of your data, normalization is a smart approach to apply.

The goal of normalization is to make every datapoint have the same scale so each feature is equally important.

Data Cleaning

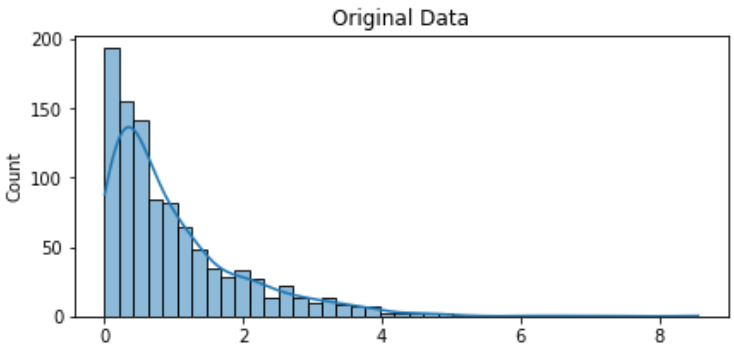
The most widely used types of normalization in machine learning are:

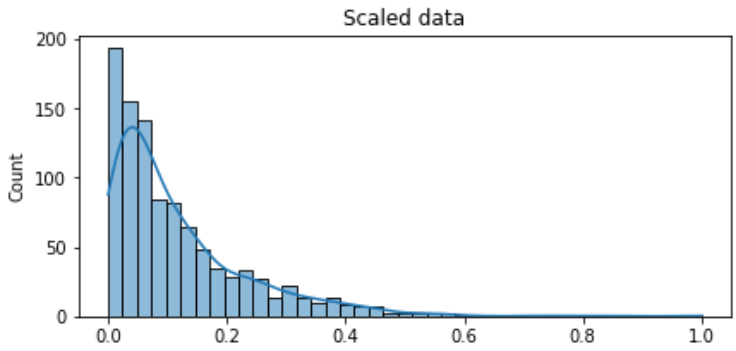
Min-Max Scaling -> Subtract the minimum value from each column's highest value and divide by the range. Each new column has a minimum value of 0 and a maximum value of 1.

Standardization -> It refers to setting the mean to zero and the standard deviation to one.

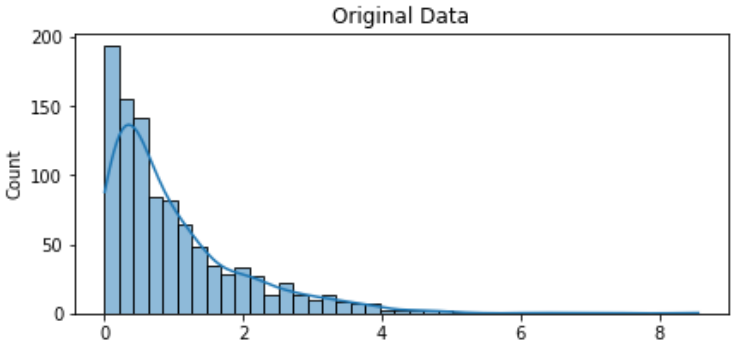
When your data has variable dimensions and the technique you're using (like logistic regression, linear regression, linear discriminant analysis) standardization is useful.

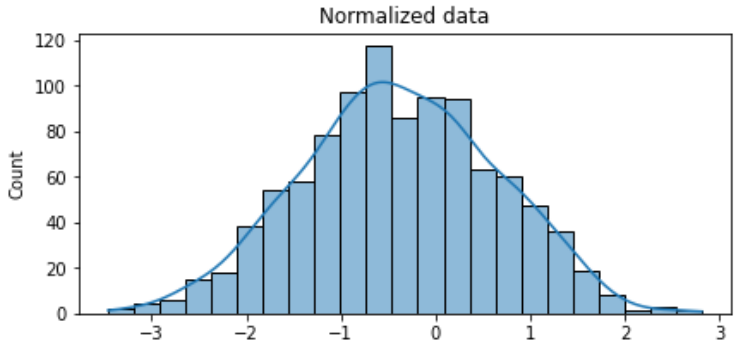
Data after Scaling





Data after Normalization





Data Cleaning

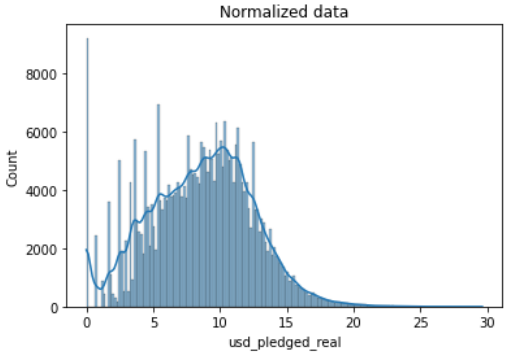
A dataset of Kickstarter campaigns (Kickstarter is a website where people can ask people to invest in various projects and concept products.) is scaled here.

```
# scale the goals from 0 to 1
scaled_data = minmax_scaling(original_data, columns=['usd_goal_real'])

print('Original data\nPreview:\n', original_data.head())
print('Minimum value:', float(original_data.min()),
      '\nMaximum value:', float(original_data.max()))
print('_'*30)

print('\nScaled data\nPreview:\n', scaled_data.head())
print('Minimum value:', float(scaled_data.min()),
      '\nMaximum value:', float(scaled_data.max()))
```

```
Original data
Preview:
    usd_goal_real
0         1533.95
1        30000.00
2        45000.00
3         5000.00
4        19500.00
Minimum value: 0.01
Maximum value: 1663613.71
______________________________

Scaled data
Preview:
    usd_goal_real
0        0.000009
1        0.000180
2        0.000270
3        0.000030
4        0.000117
Minimum value: 0.0
Maximum value: 1.0
```

Here we can clearly see that the Min and Max values are scaled between 0 and 1

```
# get the index of all positive pledges (Box-Cox only takes positive values)
index_of_positive_pledges = kickstarters_2017.pledged > 0

# get only positive pledges (using their indexes)
positive_pledges = kickstarters_2017.pledged.loc[index_of_positive_pledges]

# normalize the pledges (w/ Box-Cox)
normalized_pledges = pd.Series(stats.boxcox(positive_pledges)[0],
                               name='usd_pledged_real', index=positive_pledges.index)
ax = sns.histplot(normalized_pledges, kde=True)
ax.set_title("Normalized data")
plt.show()
```



After normalisation of data

Data Cleaning

- **Parsing Dates**

1. Check the data type of our date column.
2. Convert our date columns to DateTime.
3. Select the day of the month.
4. Plot the day of the month to check the date parsing.

Here I am using earthquake-related data to parse dates.

- Checking the date type of the column.

```
[4]: earthquakes['Date'].head()
```

```
[4]: 0    01/02/1965
     1    01/04/1965
     2    01/05/1965
     3    01/08/1965
     4    01/09/1965
     Name: Date, dtype: object
```

we can clearly infer that the datatype is object and not datetime64 dtypes.

- converting the date type of the column.

```
earthquakes['date_parsed'] = pd.to_datetime(earthquakes['Date'], format="%m/%d/%Y")
earthquakes['date_parsed'].head()
```

```
0   1965-01-02
1   1965-01-04
2   1965-01-05
3   1965-01-08
4   1965-01-09
Name: date_parsed, dtype: datetime64[ns]
```

A new column "date_parsed"is created in datetime64 dtype format.

Data Cleaning

- Select the day of the month.

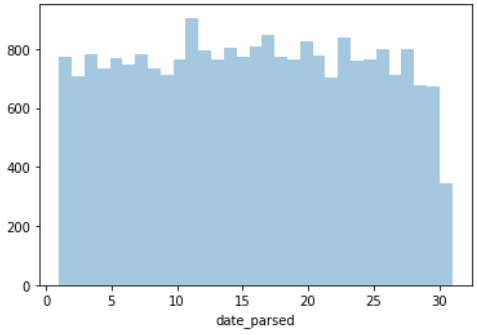
```
#to get the day of the month from the date column
day_of_month_earthquakes = earthquakes['date_parsed'].dt.day
print(day_of_month_earthquakes)
```

```
0         2
1         4
2         5
3         8
4         9
         ..
23407    28
23408    28
23409    28
23410    29
23411    30
Name: date_parsed, Length: 23412, dtype: int64
```

- Plot the date of the month to check date parsing.

```
# remove na's
day_of_month_earthquakes = day_of_month_earthquakes.dropna()

# plot the day of the month
sns.distplot(day_of_month_earthquakes, kde=False, bins=31)
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function a
nd will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with simil
ar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
<AxesSubplot:xlabel='date_parsed'>
```



Data Cleaning

- **Creating Dummy Variables and Converting Ordinal Variables to Numbers**

The picture given below shows our data, which has five observations and two columns. The first column is 'Place' which is a nominal variable and the other is 'Population' which is an ordinal variable.

	Place	Population
0	New Delhi	High
1	Mumbai	Medium
2	New Delhi	High
3	Bengaluru	Medium
4	Xyz	Low

Since machine learning algorithms can't handle string or text values such as 'New Delhi' or 'Mumbai', we need to convert them into numerical values such as 0 or 1. But we can not just replace New Delhi with 0, Mumbai with 1 and Bengaluru with 2. Why? Because it is nominal data, not ordinal data. In the case of nominal data, we have to create separate dummy variables for each city.

```
import pandas as pd

#data import
data = pd.read_csv("C:/Users/Admin/Desktop/Blog/Preprocessing/data.csv")

#create (n-1) dummies for 'Place' column with City as prefix in new dummy column names
data=pd.get_dummies(data, prefix=['City'], drop_first=True, columns=['Place'])
```

```
data #view data
```

	Population	City_Mumbai	City_New Delhi	City_Xyz
0	High	0	1	0
1	Medium	1	0	0
2	High	0	1	0
3	Medium	0	0	0
4	Low	0	0	1

Data Cleaning

An important part of Data analysis is analyzing Duplicate Values and removing them.
Pandas drop_duplicates() method helps in removing duplicates from the data frame.

syntax: DataFrame.drop_duplicates(subset=None, keep='first', inplace=False)
Parameters:
subset: Subset takes a column or list of column labels. Its default value is none. After passing columns, it will consider them only for duplicates.
keep: keep is to control how to consider duplicate value. It has only three distinct values and the default is 'first'.

- If 'first', it considers the first value as unique and the rest of the same values as duplicate.
- If 'last', it considers the last value as unique and the rest of the same values as duplicate.
- If False, it considers all of the same values as duplicates

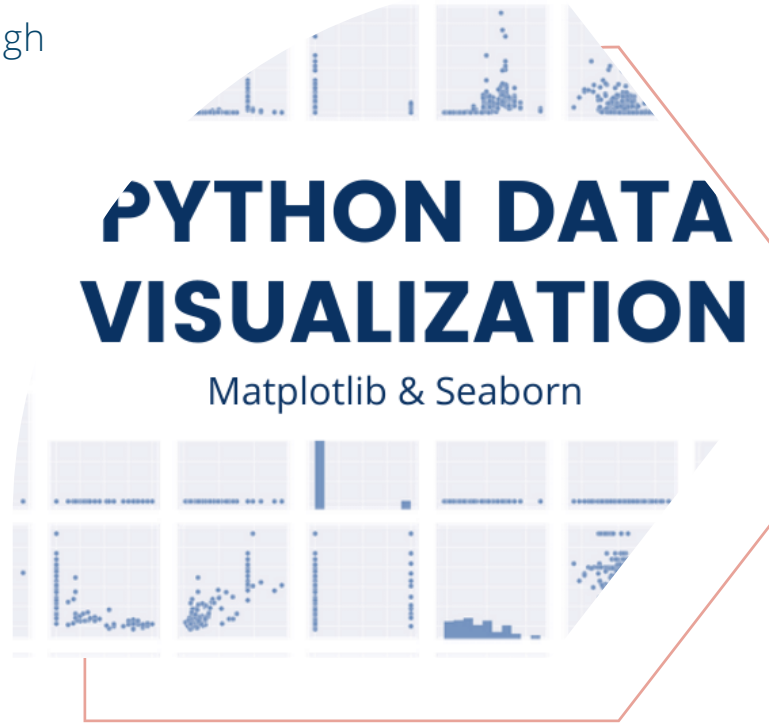
In place: Boolean values removes rows with duplicates if True.
Return type: DataFrame with removed duplicate rows depending on Arguments passed.

For Example, if we have a **CSV file** that has details of employees, we can first sort the names of the employees and **drop the duplicate values** in a row using a data frame.drop_duplicates function.

Data Visualisation

USING MATPLOTLIB AND SEABORN

In the process of data analysis, after completing data cleaning and data manipulation, the next step is to bring meaningful insights and conclusions from
the data which can be achieved through graphs and charts. Python provides several libraries for this purpose.



Some of the most common Libraries are
seaborn, MatplotLib, Plotly and Cufflinks.

Commands to import seaborn and matplotlib libraries.

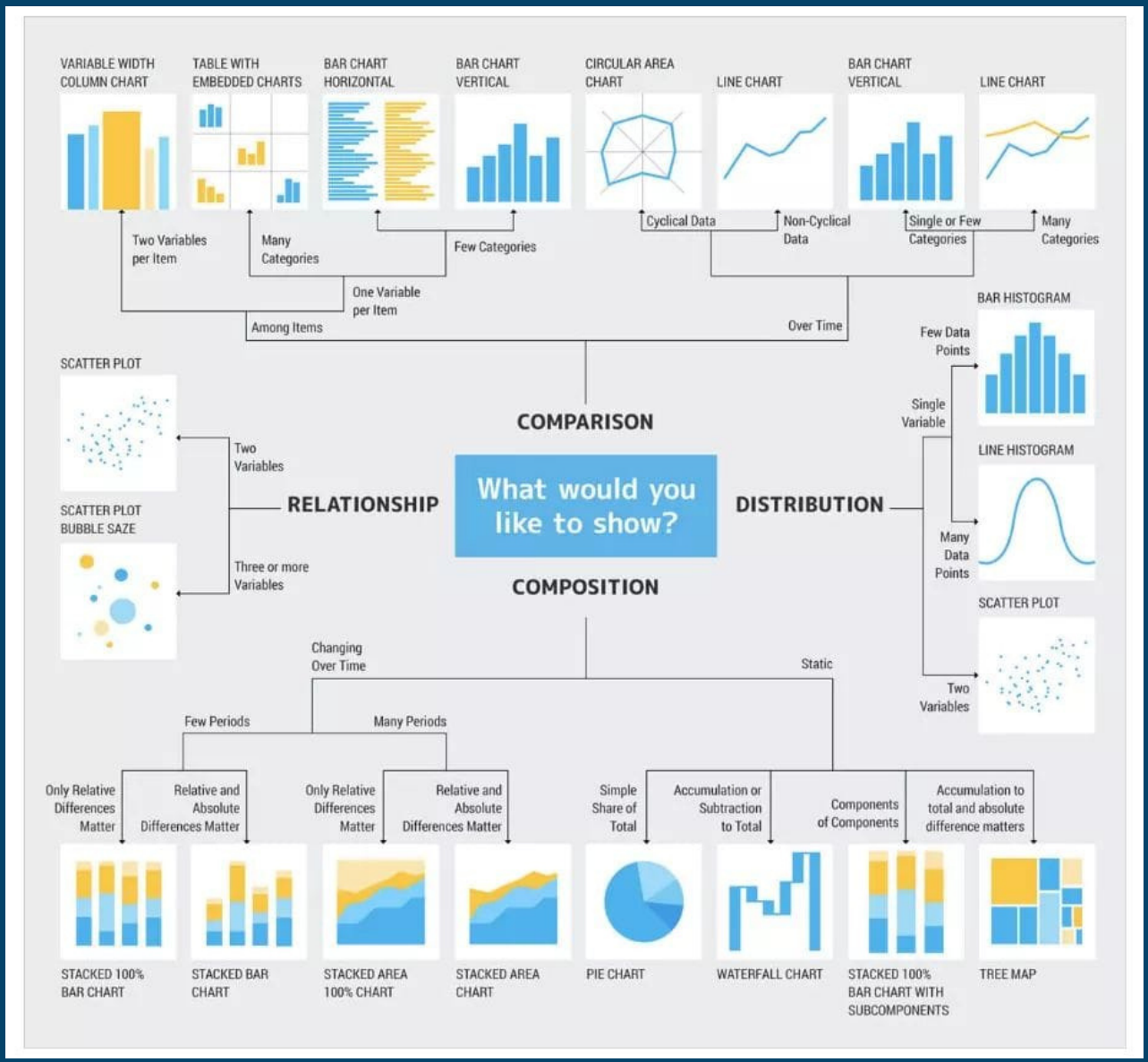
```
import seaborn as sns
import matplotlib.pyplot as plt
```

The most commonly used Data visualization types are :

-> Line charts -> Barcharts. -> Histograms -> Heatmaps

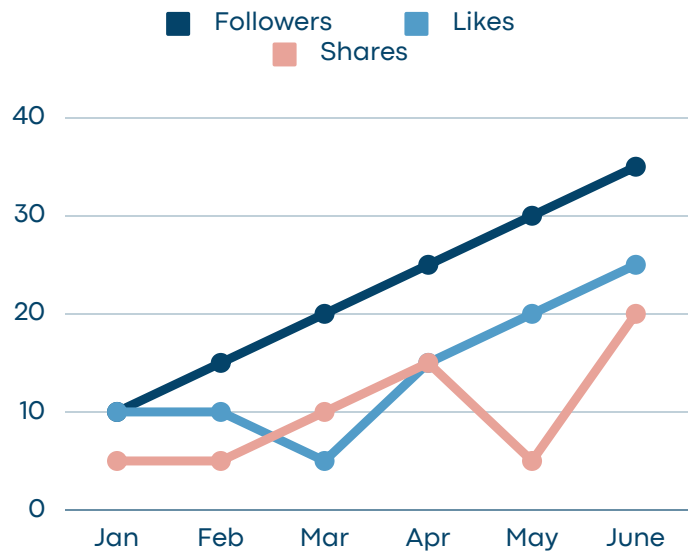
-> ScatterPlots

Data Visualization Cheat Sheet



Line Charts

1. Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded.
2. They can also be used to display several dependent variables against one independent variable.



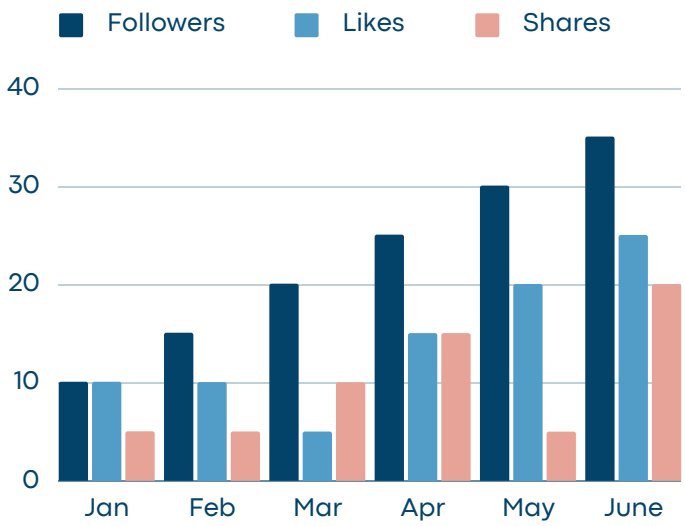
Social Media Engagements

WHEN TO USE A LINE GRAPH	WHEN NOT TO USE A LINE GRAPH
line graphs are only useful if the axes follow the same scales. Line graphs are usually used to represent changes over time.	Some experts recommend no more than 4 lines on a single graph; any more than that and it becomes difficult to interpret.

As best as possible, label the line with its name rather than using a legend. Highlight important points in the line with the exact value, such as the highest and lowest points or those points where actual data collection occured.

Bar Charts

A bar chart plots the number of times a particular value or category occurs in a data set, with the length of the bar representing the number of observations with that score or in that category.

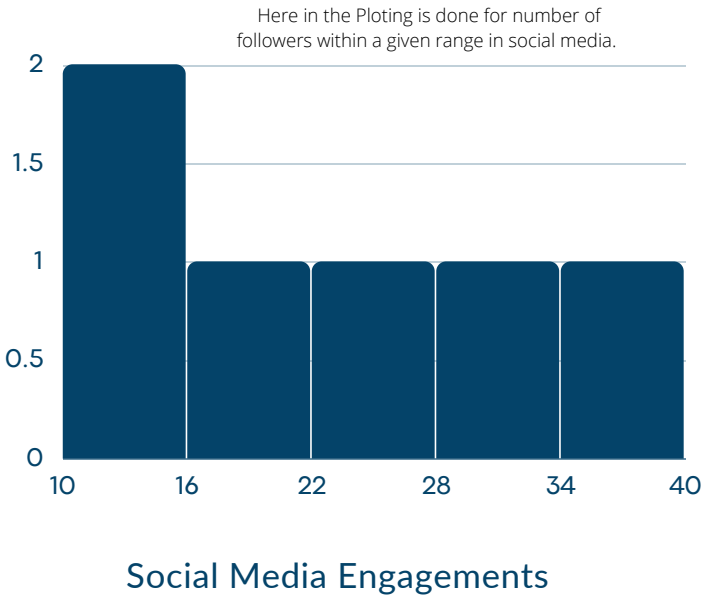


Social Media Engagements

WHEN TO USE A BAR GRAPH	WHEN NOT TO USE A BAR GRAPH
People are most accurate at judging length, thus making bar charts one of the best choices for communicating data. They are fairly well understood and easily interpreted.	Too many columns or bars can oversaturate the chart and become confusing. To solve this, prepare multiple comparative charts for different variables.

Histogram

The key difference between a bar chart and histogram is,
Bar charts can be displayed horizontally or vertically (which are sometimes referred to as column charts) and they are drawn with a gap between the bars,
whereas the bars of a histogram are drawn immediately next to each other.



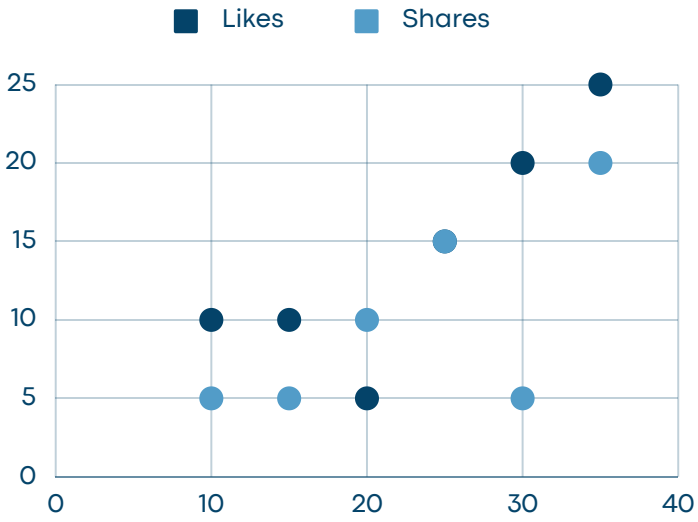
Social Media Engagements

WHEN TO USE A HISTOGRAM	WHEN NOT TO USE A HISTOGRAM
Comes handy when we have to compair sizes of different group (values in range).	Too many columns or bars can oversaturate the chart and become confusing. To solve this, prepare multiple comparative charts for different variables.

Scatter Plot

A Scatterplot is used to display the relationship between two quantitative variables plotted along two axes.

A series of dots represent the position of observations from the data set.



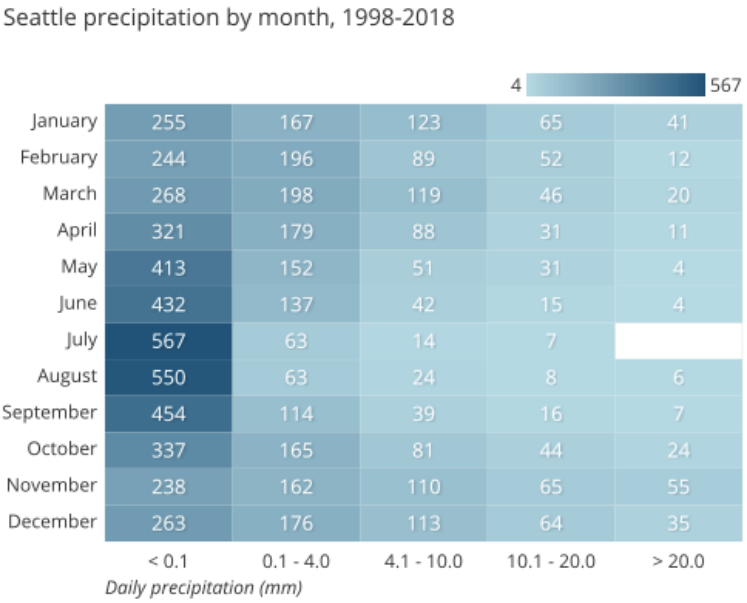
Social Media Engagements

WHEN TO USE A SCATTER PLOT	WHEN NOT TO USE A SCATTER PLOT
Scatterplots are appropriate when you want to graph two continuous quantitative variables, like height and weight.	The major cause of problems with scatterplots is discretization of values. This happens when decimal places are rounded off, measurements are not accurate enough, or a data field is categorical.

Avoid a scatter plot when your data is not at all related.

Heatmap

Heatmap is one good visualization technique used to compare any 2 variables/features with respective to the values.
The heatmap from seaborn library will create a grid like plot along with an optional color bar.



WHEN TO USE A HEATMAP	WHEN NOT TO USE A HEATMAP
A heatmap is very useful in visualizing the concentration of values between two dimensions of a matrix. This helps in finding patterns and gives a perspective of depth.	It is difficult to map color onto a continuous scale. Thus heat maps are not well-suited for seeing individual results, even in small data sets. When the clustering is done in a way that it doesnt bring out any trends.

Plotly and cufflinks : Advanced Python Data Visualization Libraries

Plotly is an open-source and charting library that provides the facility of interactive plotting. The library is available for several programming languages such as Python, R, MATLAB, and REST, among others.

Cufflink is also a python library that connects plotly with pandas so that we can create charts directly on data frames. It basically acts as a plugin.

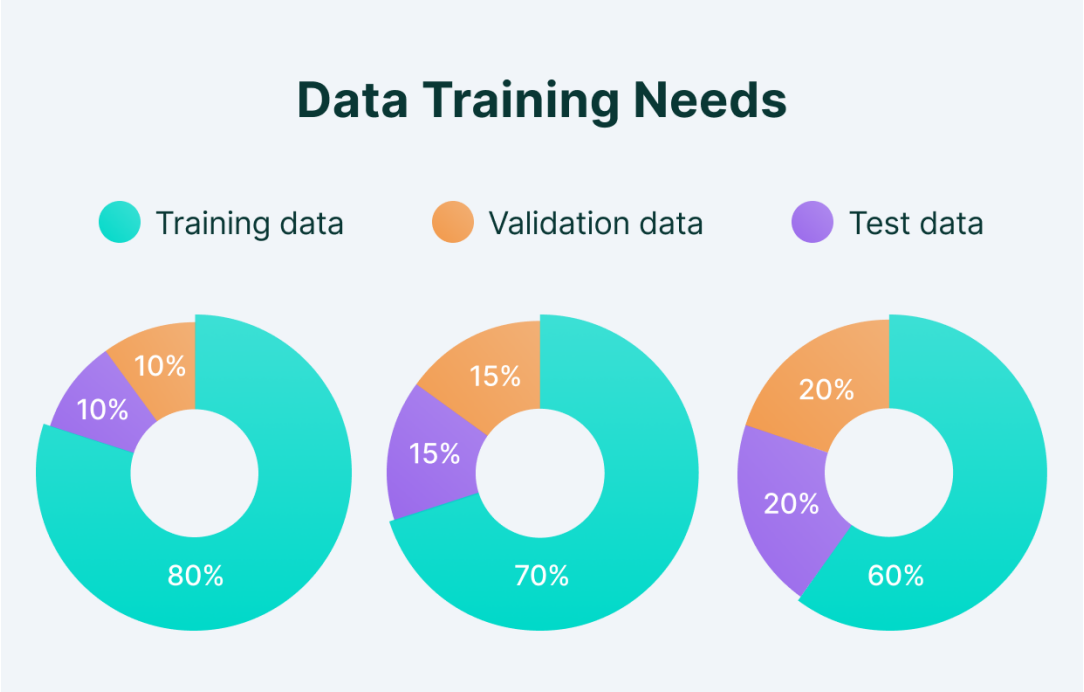
ADVANTAGES OF USING PLOTLY AND CUFFLINKS

1. Plotly charts are dynamic in nature, it allows us to hover over the values, zoom in zoom out of graphs, identifies outliers in the dataset.
2. On the other hand, matplotlib and seaborn charts are static, we can't zoom in or zoom-out, it does not provide detail of every value on the chart.
3. The most important feature of plotly is that it allows us to build dynamic charts for the web directly from python which is not possible in the case of matplotlib.
4. With plotly, we can also create animations and interactive graphs on geographical, scientific, statistical, and financial data.

Introduction to Machine Learning

What ML does, depends on the user input or a query requested by the client, the framework checks whether it is available in the knowledge base or not. If it is available, it will restore the outcome to the user related to that query, however, if it isn't stored initially, the machine will take in the user input and will enhance its knowledge base, to give a better value to the end-user.

After cleansing and manipulation of data, the main question is : How do we split data in Machine Learning for training ?



Introduction to Machine Learning

The data is basically split using the Train-Test Split algorithm to evaluate the performance of the model

It can be used for classification or regression problems and can be used for any supervised learning algorithm.

- Train Dataset: Used to fit the machine learning model.
- Test Dataset: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.

How to Configure the Train-Test Split
This is most commonly expressed as a percentage between 0 and 1 for either the train or test datasets. For example, a training set with the size of 0.67 (67 per cent) means that the remaining percentage of 0.33 (33 per cent) is assigned to the test set.
There is no optimal split percentage.
You must choose a split percentage that meets your project's objectives with considerations that include:

- Computational cost in training the model.
- Computational cost in evaluating the model.
- Training set representativeness.
- Test set representativeness.

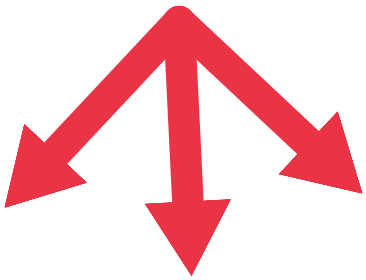
Nevertheless, common split percentages include:

- Train: 80%, Test: 20%
- Train: 67%, Test: 33%
- Train: 50%, Test: 50%

sample syntax << X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)

Introduction to Machine Learning

Types of learning



Supervised Learning

Unsupervised Learning

Reinforcement Learning

Supervised learning algorithms are trained using labled examples, such as desired output is known.

Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance.

Supervised and Unsupervised Learning

Supervised learning is classified into two categories of algorithms:

- Classification: A classification problem is when the output variable is a category, such as “Red” or “blue” or “disease” and “no disease”.
- Regression: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.
- Techniques of Supervised Machine Learning algorithms include linear and logistic regression, multi-class classification, Decision Trees and support vector machines.

SUPERVISED LEARNING
Types:- **Regression** **Logistic Regression** **Classification** **K-NN (k nearest neighbors)** **Decision Trees** **Support Vector Machine**

What is the difference between classification and regression?

Classification is used to produce discrete results, classification is used to classify data into some specific categories.
For example, **classifying e-mails into spam and non-spam categories.**

Whereas, We use regression analysis when we are dealing with continuous data, for example **predicting stock prices at a certain point of time.**

Supervised and Unsupervised Learning

Unsupervised learning is classified into two categories of algorithms:

- Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

UNSUPERVISED LEARNING

Types of Unsupervised Learning:-
Clustering

1. **Exclusive (partitioning)**
2. **Agglomerative**
3. **Overlapping**
4. **Probabilistic**

Clustering Types:-

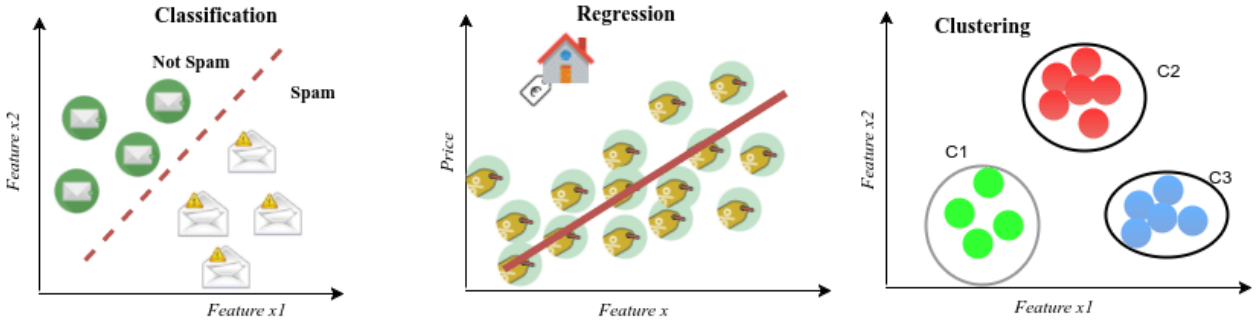
1. **Hierarchical clustering**
2. **K-means clustering**
3. **Principal Component Analysis**
4. **Singular Value Decomposition**
5. **Independent Component Analysis**

When to use Supervised and Unsupervised learning?

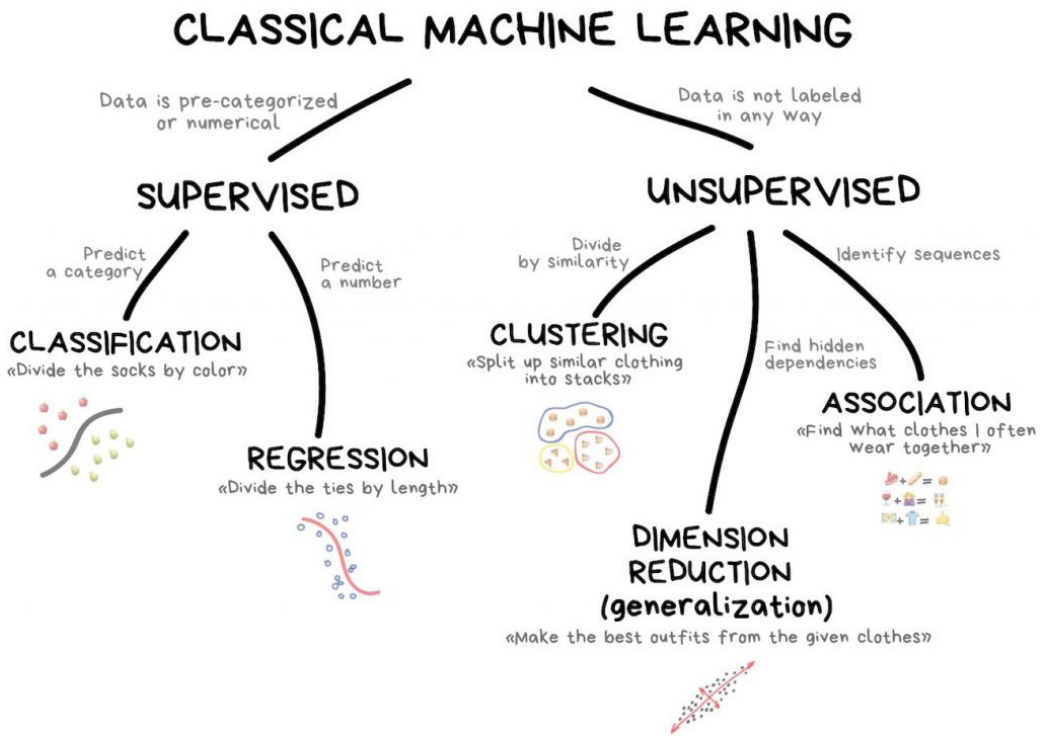
"We choose supervised learning for applications when labelled data is available and the goal is to predict or classify future observations."

"We use unsupervised learning when labelled data is not available and the goal is to build strategies by identifying patterns or segments from the data."

Supervised and Unsupervised Learning



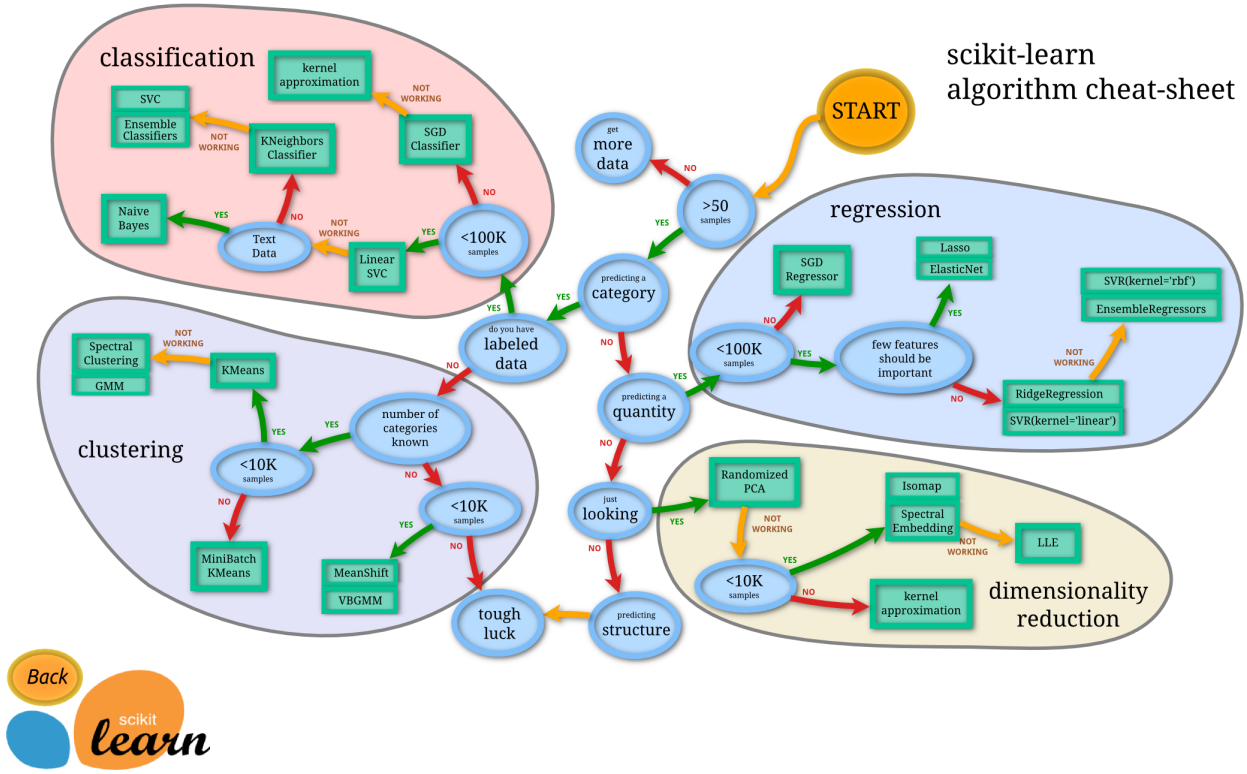
Example of Classification Regression and Clustering



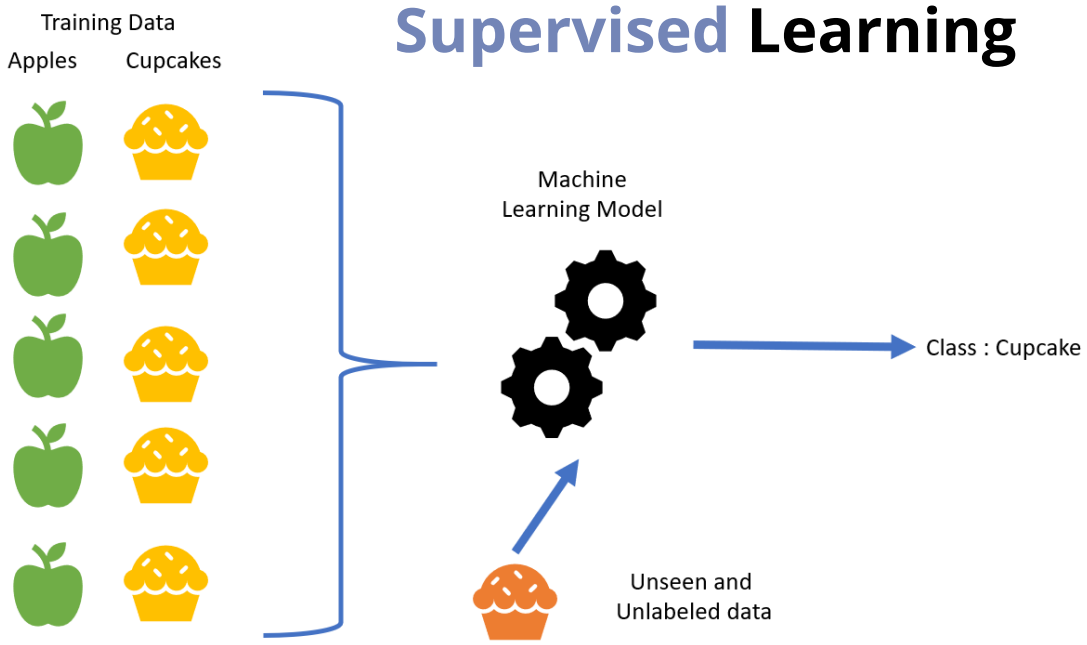
The above diagram gives an idea of when to choose which algorithm.

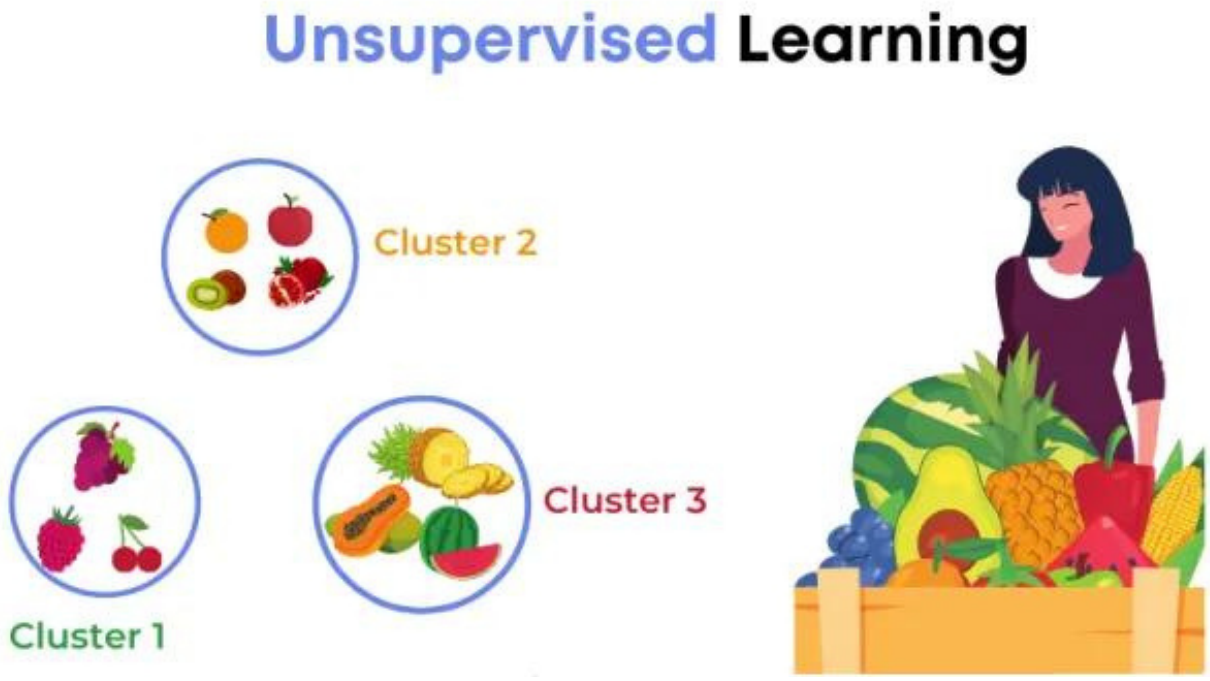
Supervised and Unsupervised Learning

Scikit Learn Cheat Sheet for machine learning algorithm:



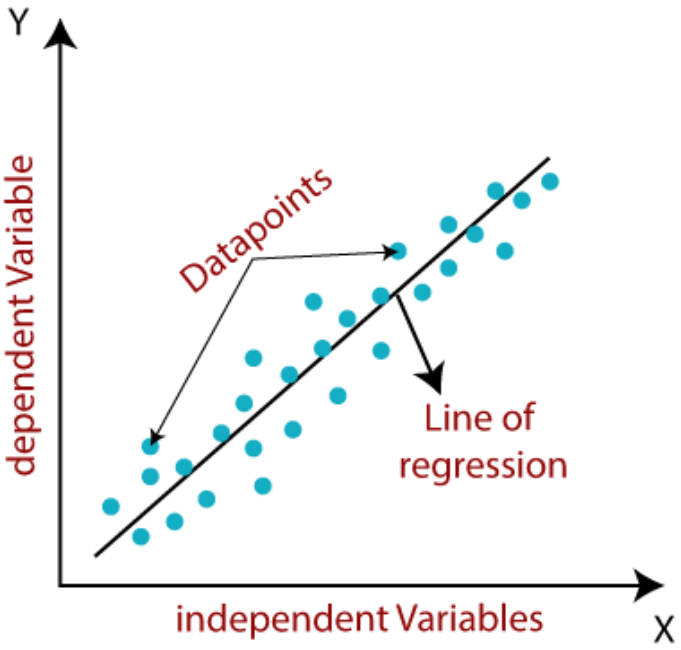
Real-time example of Supervised and Unsupervised Learning





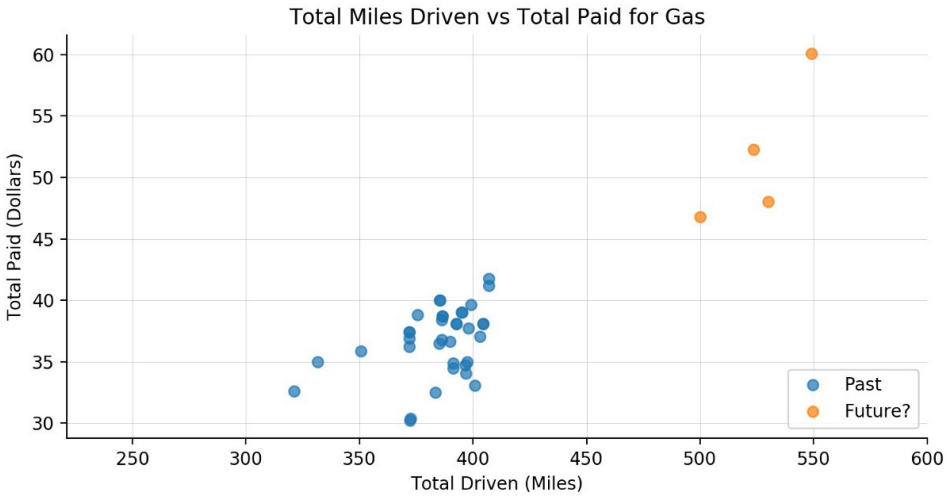
Linear Regression

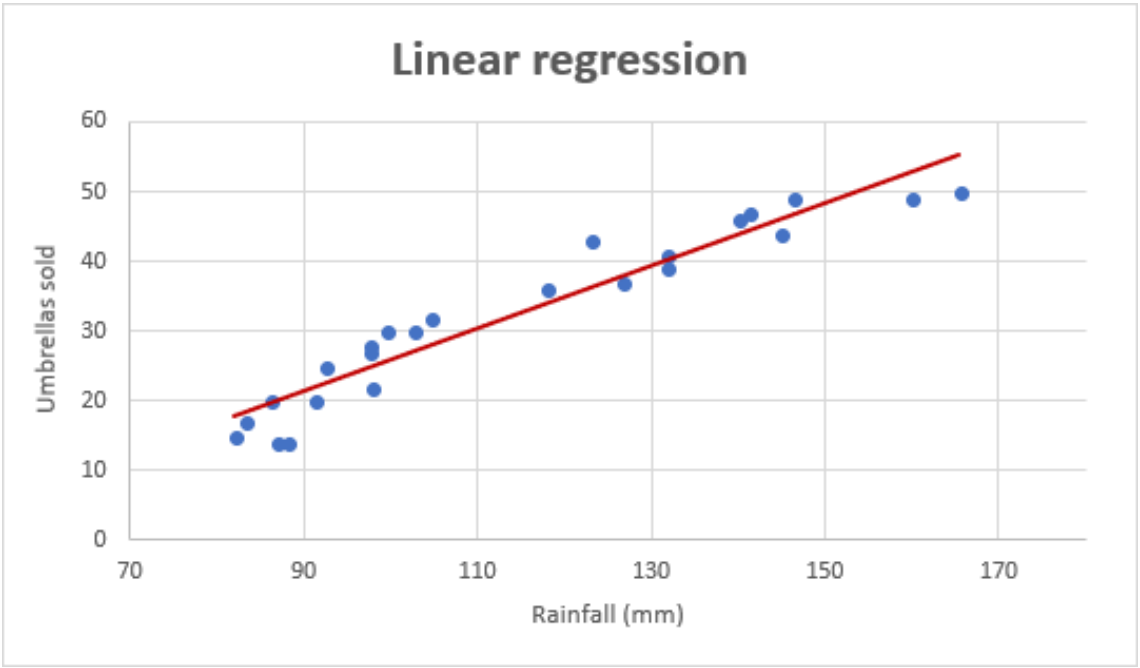
Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called linear regression.
Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.



Mathematically, we can represent a linear regression as:
$y= a0+a1x+ \varepsilon$
Here,
Y= Dependent Variable (Target Variable)
X= Independent Variable (predictor Variable)
a0= intercept of the line (Gives an additional degree of freedom)
a1 = Linear regression coefficient (scale factor to each input value).
ε = random error
The values for x and y variables are training datasets for Linear Regression model representation.

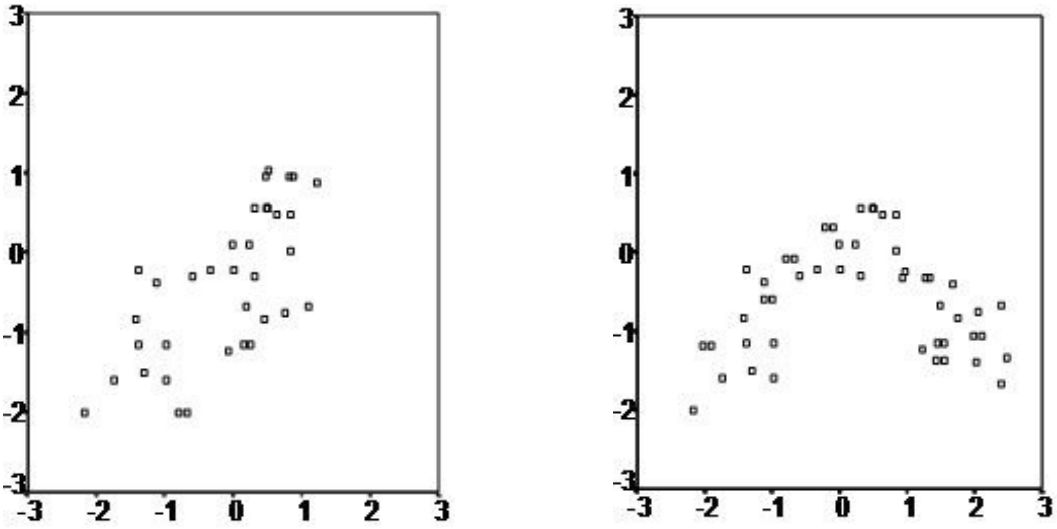
Real Time Example of Linear Regression





How to Conduct Linear Regression

Linear Regression Analysis consists of more than just fitting a linear line through a cloud of data points.
It consists of 3 stages –
(1) analyzing the correlation and directionality of the data,
(2) estimating the model, i.e., fitting the line, and
(3) evaluating the validity and usefulness of the model.



- First, a scatter plot should be used to *analyze the data* and *check for the directionality and correlation of data*. The first scatter plot indicates *a positive relationship* between the two variables. The data *is fit to run a regression analysis*.

- The second scatter plot seems to *have an inverse U-shape* this indicates that a regression line *might not be the best way* to explain the data, even if a correlation analysis establishes a positive link between the two variables.

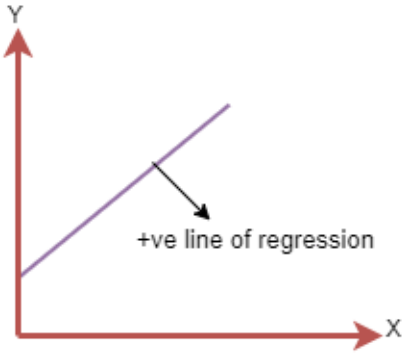
1. Analyzing the correlation and directionality of the data
Linear Regression Line

A linear line showing the relationship between the dependent and independent variables is called a *regression line.*

A regression line can show two types of relationship:

- **Positive Linear Relationship:**

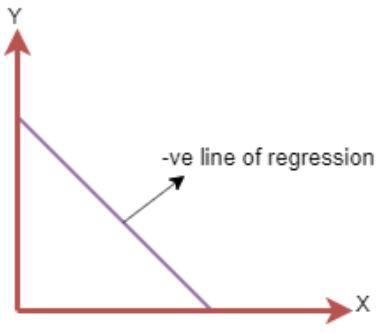
If the dependent variable increases on the Y-axis and the independent variable increases on X-axis, then such a relationship is termed a positive linear relationship.



The line equation will be: $Y= a_0+a_1x$

- **Negative Linear Relationship:**

If the dependent variable decreases on the Y-axis and the independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.



The line of equation will be: $Y= -a_0+a_1x$

2. Finding the best fit line:

When working with linear regression, our main goal is to find the *best fit line* which means the error between predicted values and actual values should be minimized.
The best fit line will have the least error.

The different values for weights or the coefficient of lines (a0, a1) gives a different line of regression, so we need to calculate the best values for a0 and a1 to find the best fit line, so to calculate this we use the **cost function.**

Cost function-
What is COST FUNCTION?

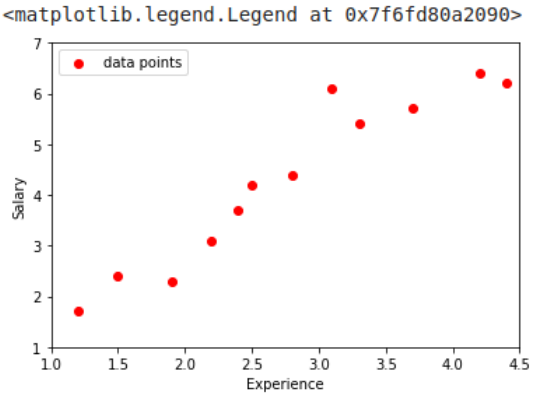
- The different values for weights or coefficient of lines (a0, a1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line.

What does cost function do?

- Cost function optimizes the regression coefficients or weights. It measures how a linear regression model is performing.

EXAMPLE:-
here is some data and a graph

	salary	experience
0	1.7	1.2
1	2.4	1.5
2	2.3	1.9
3	3.1	2.2
4	3.7	2.4



Understanding the Cost Function

Let's do an analysis using the squared error cost function.

$$\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

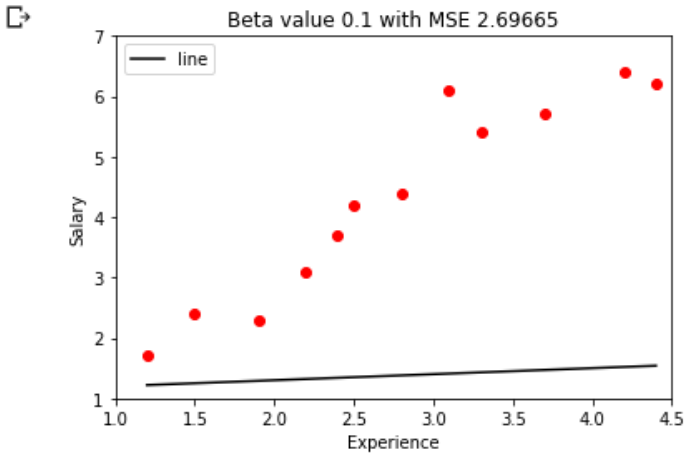
Starting the Line using small values of parameters(beta and b)

To start with let me take β = 0.1 and b = 1.1
Where b is the Intercept constant and β is the Coefficient.

Best fit 1<< is calculated by forming a line with the data points generated using **"data.experience[i]*beta + b".**
Then mse (mean squared error) of that line is found using

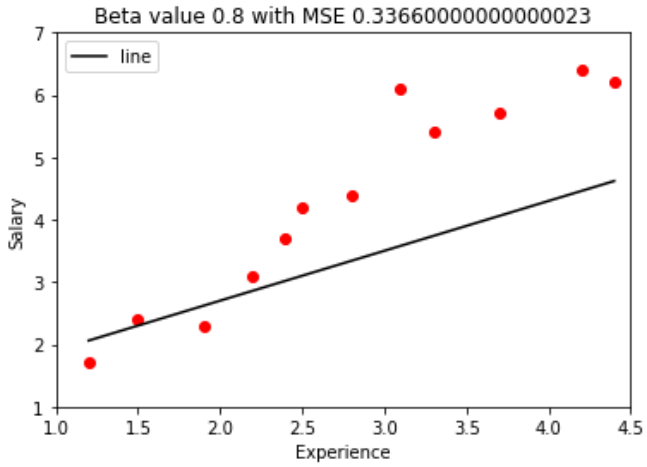
$$MSE = 1\frac{1}{N}\sum_{i=1}^{n}(y_i - (a_1x_i + a_0))^2$$

And a graph is generated for the original data, the best fit line and mse is calculated for that line.



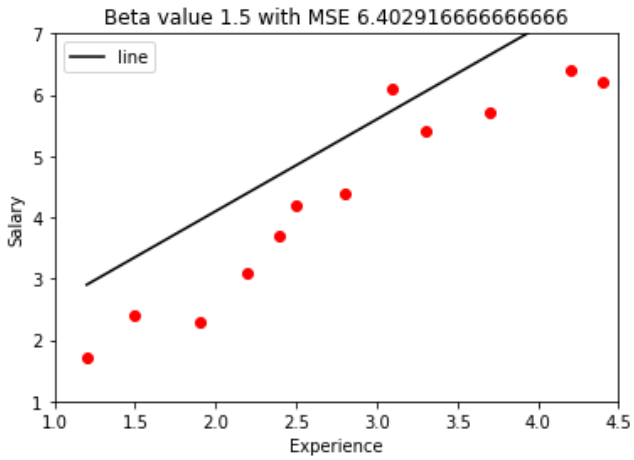
We have this line for beta = 0.1 and b = 1.1 and the MSE for this line is 2.69. the slope of the line is very less so, we try out a higher slope instead of beta = 0.1 it is changed to beta = 1.5.

Best fit 2 <<



This is the line for beta = 1.5 and b = 1.1 and the MSE for this line is 6.40. You can see a better slope but it is probably more than what we actually want. So the right value might be somewhere in between 0.1 and1.5, so let's try beta = 0.8

Best fit 3 <<



This is the line with beta = 0.8 and b = 1.1 and we can see that the mean squared error has come down to 0.336

We have tried 3 different lines with each of these having different MSE; the first one had 2.69, the second had 6.4 and the third one had 0.33.

Computing Cost Function over a range of values of Beta

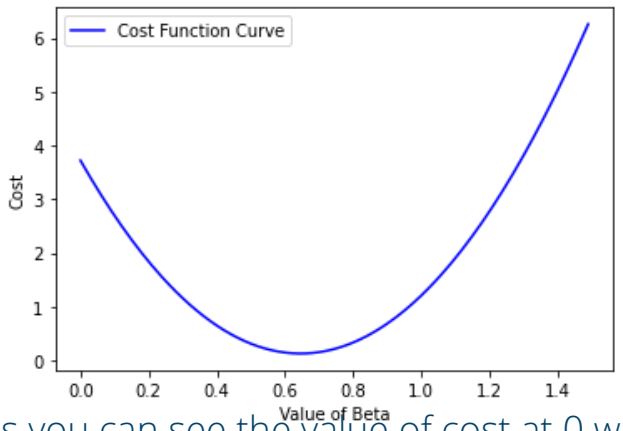
Now I am trying all values of Beta between 0 to 1.5 with an increment of 0.01. and a data frame is created against beta and cost function/mse.

Visualizing cost with respect to Beta

	Beta	Cost
0	0.00	3.721667
1	0.01	3.611426
2	0.02	3.502906
3	0.03	3.396105
4	0.04	3.291024

This data frame is showing that for a value of Beta which is 0.00 the cost or MSE we're getting is 3.72, similarly for beta = 0.04, we are getting cost = 3.29. Let's quickly visualize this:

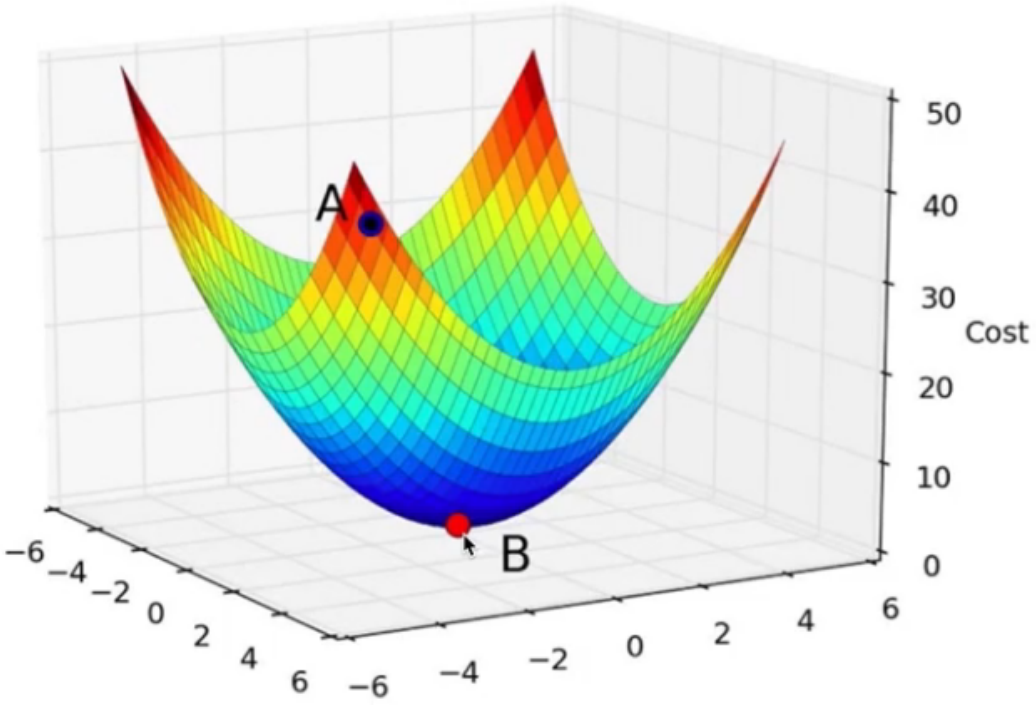
```
<matplotlib.legend.Legend at 0x7f6fd81050d0>
```



So as you can see the value of cost at 0 was around 3.72, so that is the starting value. After that the error goes down with the increasing value of Beta, reaches a minimum, and then it starts increasing.

Now the question is given that we know this relationship, what are the values of beta and b for which we can find out this particular location where my cost is minimum.

Now what happens when you have two parameters, I mean right now we assumed b to be 0.1 but actually what we want to do is change it with respect to b as well as beta and in this particular situation this is the curve which we get:



This is just a 3D plot, so you have two dimensions but it will again have a minimum value and the idea would be to find the minimum value. But you cannot do this iteratively in the way we did so far. So what we need to find is a smarter way in which we can find this minimum value and that is where the Gradient Descent technique comes into play.

3. Evaluating the validity and usefulness of the model.

Residuals: The distance between the actual value and predicted values is called residual. If the observed points are far from the regression line, then the residual will be high, and so the cost function will be high. If the scatter points are close to the regression line, then the residual will be small and hence the cost function.

Gradient Descent:

- Gradient descent is used to minimize the MSE by calculating the gradient of the cost function.
- A regression model uses gradient descent to update the coefficients of the line by reducing the cost function.
- It is done by a random selection of values of coefficient and then iteratively updating the values to reach the minimum cost function.

When can a Gradient Descent be used?

The gradient descent algorithm does not work for all functions. There are two specific requirements.

A function has to be:

- differentiable
- convex

GRADIENT DESCENT METHOD'S STEPS ARE:

1. choose a starting point (initialisation)
2. calculate the gradient at this point
3. make a scaled step in the opposite direction to the gradient (objective: minimise)
4. repeat points 2 and 3 until one of the criteria is met:

- maximum number of iterations reached
- step size is smaller than the tolerance.

It can be implemented using the function gradient_descent().

This function takes *5 parameters*:
1. starting point - in our case, we define it manually but in practice, it is often a random initialisation
2. gradient function - has to be specified before-hand
3. learning rate - scaling factor for step sizes
4. maximum number of iterations
5. tolerance to conditionally stop the algorithm (in this case a default value is 0.01)

Types of gradient Descent:

1. **Batch Gradient Descent:** This is a type of gradient descent that processes all the training examples for each iteration of gradient descent. But if the number of training examples is large, then batch gradient descent is computationally very expensive.
2. **Stochastic Gradient Descent:** This is a type of gradient descent that processes 1 training example per iteration. Hence, the parameters are being updated even after one iteration in which only a single example has been processed. Hence this is quite faster than batch gradient descent.
3. **Mini Batch gradient descent:** This is a type of gradient descent that works faster than both batch gradient descent and stochastic gradient descent. Here b, examples where b<m are processed per iteration. So even if the number of training examples is large, it is processed in batches of b training examples in one go. Thus, it works for larger training examples and that too with a lesser number of iterations.

What is Classification Algorithm?

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data.

The best example of an ML classification algorithm is **Email Spam Detector**.

- **Binary Classifier:** If the classification problem has only two possible outcomes, then it is called a Binary Classifier.

Examples: YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.

- **Multi-class Classifier:** If a classification problem has more than two outcomes, then it is called a Multi-class Classifier.

Example: Classifications of types of crops, Classification of types of music.

Types of ML Classification Algorithms:

- Linear Classifiers: Logistic Regression
- Tree-Based Classifiers: Decision Tree Classifier
- Support Vector Machines
- Artificial Neural Networks
- Bayesian Regression
- Gaussian Naive Bayes Classifiers
- Stochastic Gradient Descent (SGD) Classifier
- Ensemble Methods: Random Forests, AdaBoost, Bagging Classifier, Voting Classifier, ExtraTrees Classifier

Evaluating a Classification model:

Once our model is completed, it is necessary to evaluate its performance; either it is a Classification or Regression model. So for evaluating a Classification model, we have the following ways:

1. Confusion matrix
2. AUC/ROC curve

1. Confusion Matrix:

- The confusion matrix provides us with a matrix/table as output and describes the performance of the model.
- It is also known as the error matrix.
- The matrix consists of predictions resulting in a summarized form, which has a total number of correct predictions and incorrect predictions. The matrix looks as below table:

	Actual Positive	Actual Negative
Predicted Positive	True Positive	False Positive
Predicted Negative	False Negative	True Negative

Accuracy:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{Total Population}}$$

Precision:

Precision tells us how many of the correctly predicted cases actually turned out to be positive.

$$Precision = \frac{TP}{TP + FP}$$

Recall: Recall tells us how many of the actual positive cases we were able to predict correctly with our model.

$$Recall = \frac{TP}{TP + FN}$$

F1-score: It's the harmonic mean of Precision and Recall

$$F1 - score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

- Sklearn confusion_matrix() returns the values of the Confusion matrix.
- Sklearn classification_report() outputs precision, recall and f1-score for each target class. In addition to this, it also has some extra values: micro avg, macro avg, and weighted avg

Mirco average is the precision/recall/f1-score calculated for all the classes.
Macro average is the average of precision/recall/f1-score.
The weighted average is just the weighted average of precision/recall/f1-score.

Sensitivity: Sensitivity tells us what proportion of the positive class got correctly classified.

$$Sensitivity = \frac{TP}{TP + FN}$$

False Negative Rate:
False Negative Rate (FNR) tells us what proportion of the positive class got incorrectly classified by the classifier.

$$FNR = \frac{FN}{TP + FN}$$

True Negative Rate: Specificity tells us what proportion of the negative class got correctly classified.

$$Specificity = \frac{TN}{TN + FP}$$

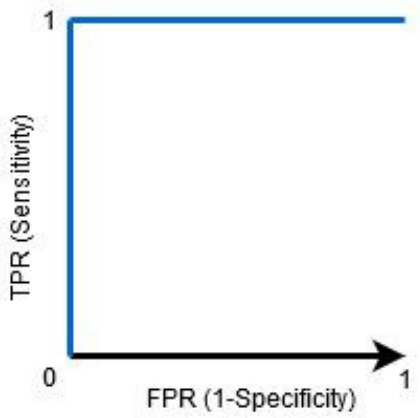
False Positive Rate: FPR tells us what proportion of the negative class got incorrectly classified by the classifier.

$$FPR = \frac{FP}{TN + FP} = 1 - Specificity$$

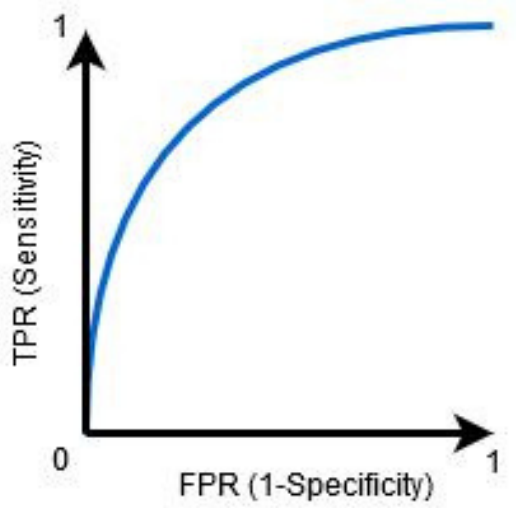
2. AUC-ROC curve:

- ROC curve stands for Receiver Operating Characteristics Curve and AUC stands for Area Under the Curve.
- It is a graph that shows the performance of the classification model at different thresholds.
- To visualize the performance of the multi-class classification model, we use the AUC-ROC Curve.
- The ROC curve is plotted with TPR and FPR, where TPR (True Positive Rate) on Y-axis and FPR(False Positive Rate) on X-axis.

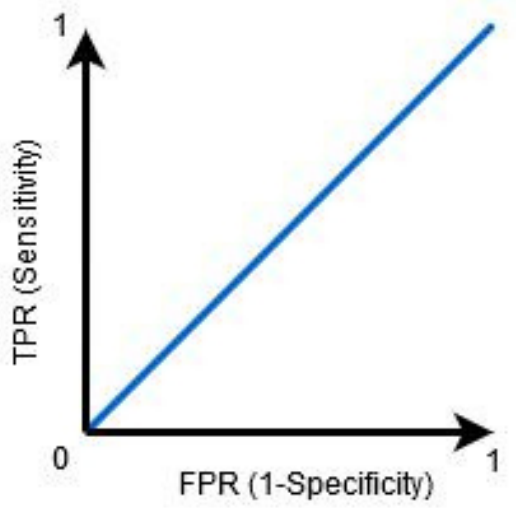
The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.



When AUC = 1, then the classifier is able to perfectly distinguish between all the Positive and the Negative class points correctly. If, however, the AUC had been 0, then the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.



When 0.5<AUC<1, there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values. This is so because the classifier is able to detect more numbers of True positives and True negatives than False negatives and False positives.



When AUC=0.5, then the classifier is not able to distinguish between Positive and Negative class points. Meaning either the classifier is predicting a random class or a constant class for all the data points.

So, the higher the AUC value for a classifier, the better its ability to distinguish between positive and negative classes.

In a ROC curve,

- A higher X-axis value indicates a higher number of False positives than True negatives.
- A higher Y-axis value indicates a higher number of True positives than False negatives.

So, the choice of the threshold depends on the ability to balance between False positives and False negatives.

Difference between Confusion matrix and AUC/ROC curve?

A confusion matrix evaluates one particular classifier with a **fixed threshold**, while the AUC evaluates that classifier over **all possible thresholds**.
In the Confusion matrix, Given a set of input cases, the classifier scores each one, and scores above the threshold are labelled Class 1 and scores below the threshold are labelled Class 2.

in AUC/ROC curve, Given a set of input cases, the classifier scores each one. The ROC curve is then generated by testing every possible threshold and plotting each result as a point on the curve.

What is Logistic Regression?

The classification algorithm Logistic Regression is used to predict the likelihood of a categorical dependent variable.
The dependant variable in logistic regression is a binary variable with data coded as 1 (yes, True, normal, success, etc.) or 0 (no, False, abnormal, failure, etc.).
The goal of Logistic Regression is to discover a link between characteristics and the likelihood of a specific outcome.

A Logistic Regression model is similar to a Linear Regression model, except that the Logistic Regression utilizes a more sophisticated cost function, which is known as the “Sigmoid function” or “logistic function” instead of a linear function.

Why do we use Logistic Regression?

Sometimes you will want to predict the "risk" of something happening, i.e. 0 (absolutely no risk of the thing happening) or 1 (100% chance guaranteed the thing will happen). However, the predictions from the model will not be 0 or 1; it will be a number ranging from 0 to 1. So, if the prediction from your model is 0.5, this is saying the risk of the thing happening is 50% chance. The closer the prediction is to 0, the less risk of the thing happening; the closer to 1, the greater the risk of the thing happening.

After training data in the Logistic Regression model, we can evaluate it using a confusion matrix.

Real-Time Examples for Logistic Regression

Text editing (Eg: Grammarly, Siri)

As we talked about texts, it is worth mentioning that logistic regression is a popular choice in many natural language processing tasks.
First, the text preprocessing is performed, then features are extracted, and finally, logistic regression is used to make some claim about a text fragment. Toxic speech detection, topic classification for questions to support, and email sorting are examples where logistic regression shows good results. Other popular algorithms for making a decision in these fields are support vector machines and random forests.

KNN - K Nearest Neighbour

What is KNN?

K-nearest neighbors (KNN) is a type of **supervised learning algorithm** used for both regression and classification. KNN tries to predict the correct class for the test data by ***calculating the distance between the test data and all the training points***. Then select the K number of points that is close to the test data.

When do we use KNN algorithm?

- KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry.
- The KNN algorithm can compete with the most accurate models because it makes highly accurate predictions.
- Although this method increases the costs of computation compared to other algorithms, KNN is still the better choice for applications where *predictions are not requested frequently but where accuracy is important*.

The KNN algorithm is implemented in the KNN and PREDICT_KNN stored procedures. To print the KNN model, use the PRINT_MODEL stored procedure.

In real-life problems where we have many points the question that arises is how to select the value of K?

Choosing the right value of K is called parameter tuning and it's necessary for better results. By choosing the value of K we square root the total number of data points available in the dataset.
a. K = sqrt (total number of data points).
b. Odd value of K is always selected to avoid confusion between 2 classes.

KNN - K Nearest Neighbour

When do we use KNN?

a. We have properly labelled data. For example, if we are predicting someone is having diabetes or not the final label can be 1 or 0. It cannot be NaN or -1.

b. Data is noise-free. For the diabetes data set, we cannot have a Glucose level as 0 or 10000. It's practically impossible.

c. Small dataset.

How does KNN work?

We usually use Euclidean distance to calculate the nearest neighbor. If we have two points (x, y) and (a, b). The formula for Euclidean distance (d) will be $d = sqrt((x-a)^2+(y-b)^2)$

Decision Trees

Imagine that I play Tennis every Saturday and I always invite a friend to come with me.

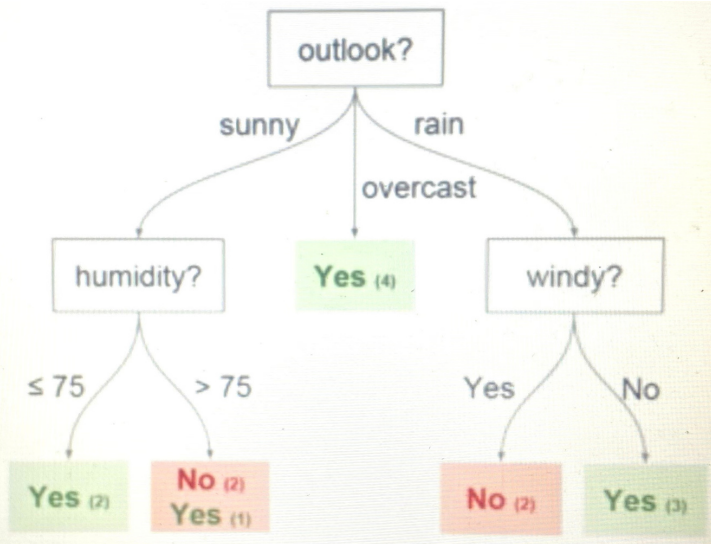
Sometimes my friend shows up, sometimes not.

For him, it depends on a variety of factors, such as weather, temperature, humidity, wind etc.

I start keeping track of these features and whether or not he showed up to play with me.

I want to use this data to predict whether or not he will show up to play.

An intuitive way to do this is through a Decision Tree



What is a Decision Tree and Random Forest?

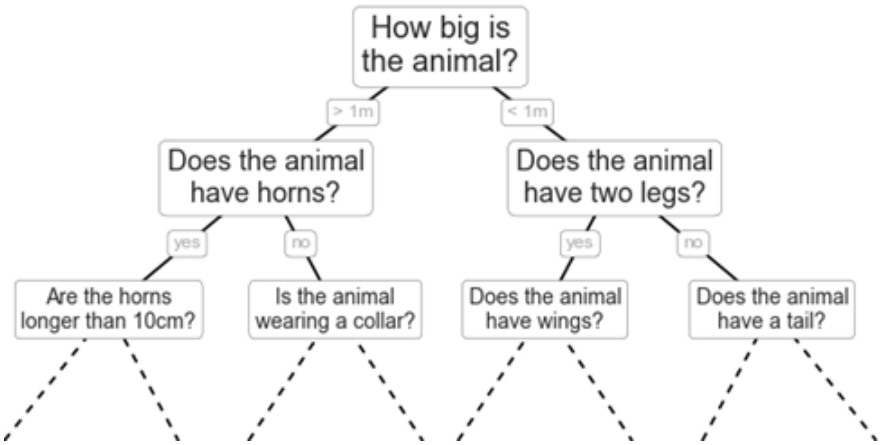
Decision trees and random forests are *supervised learning algorithms* used for both classification and regression problems.

These two algorithms are best explained together because *random forests are a bunch of decision trees combined*.

There are of course certain dynamics and parameters to consider when creating and combining decision trees.

The decision tree has:

- Nodes: Split for the value of a certain attribute.
- Edges: The outcome of a split to the next node.
- Root: The node that performs the first split.
- Leaves: Terminal nodes that predict the outcome.



The first split is based on the size of the animal. Although the question seems to be “How big is the animal?”, it is asked in the form of “Is the animal bigger than 1m?” because we want to split the data points in two groups at each step. The questions get more specific as the tree gets deeper.
What you ask at each step is the most critical part and greatly influences the performance of decision trees

A Decision Tree tries to achieve the following :

When choosing a feature to split, a decision tree algorithm tries to achieve

- More predictiveness
- Less impurity
- Lower entropy

There are two ways to measure the quality of a split: **Gini Impurity** and **Entropy.**

Gini Impurity:

Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labelled if it was randomly labelled according to the distribution of labels in the subset.

Entropy:

Entropy is a measure of uncertainty or randomness. The more randomness a variable has, the higher the entropy is.

Limitations of Decision Tree:

- The model can keep asking questions until all the nodes are pure. Pure nodes include data points from only one class.

- However, this would be a too specific model and would not generalize well. It achieves high accuracy with the training set but performs poorly on new, previously unseen data points.
- It is very important to control or **limit the depth** of a tree to prevent **overfitting**.

The list of all hyperparameters of DecisionTreeClassifier() is specified here

Random Forests

Random forest is an ensemble of many decision trees. Random forests are built using a method called bagging in which each decision tree are used as a parallel estimator.

If used for a classification problem, the result is based on a majority vote of the results received from each decision tree.

For regression, the prediction of a leaf node is the mean value of the target values in that leaf.

Random forest regression takes the mean value of the results from decision trees.

Advantages of Random Forest

- Random forests reduce the risk of overfitting and accuracy is much higher than a single decision tree.
- The success of a random forest highly depends on using uncorrelated decision trees. If we use the same or very similar trees, the overall result will not be much different from the result of a single decision tree.

Random forests achieve to have uncorrelated decision trees by **bootstrapping** and **feature randomness**.

- Bootstrapping is randomly selecting samples from training data with replacement. They are called bootstrap samples.

- Feature randomness is achieved by selecting features randomly for each decision tree in a random forest. The number of features used for each tree in a random forest can be controlled with the **max_features** parameter.

Pros and Cons

Decision Trees

Pros:

- It is usually not needed to normalize or scale features
- Suitable to work on a mixture of feature data types (continuous, categorical, binary)
- Easy to interpret

Cons:

- Prone to overfitting and need to be ensembled in order to generalize well

Random Forests

Pros:

- A powerful, highly accurate model on many different problems
- Like decision trees, does not require normalization or scaling
- Like decision trees, can handle different feature types together
- Runs the trees in parallel so the performance is not effected

Cons:

- Not a good choice for high-dimensional data sets (i.e. text classification) compared fast linear models (i.e. Naive Bayes)

K Means Clustering

K-means clustering is one of the simplest and most popular unsupervised machine learning algorithms.
Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known or labelled outcomes.

You'll define a target number k, which refers to the number of centroids you need in the dataset. A centroid is the imaginary or real location representing the centre of the cluster.

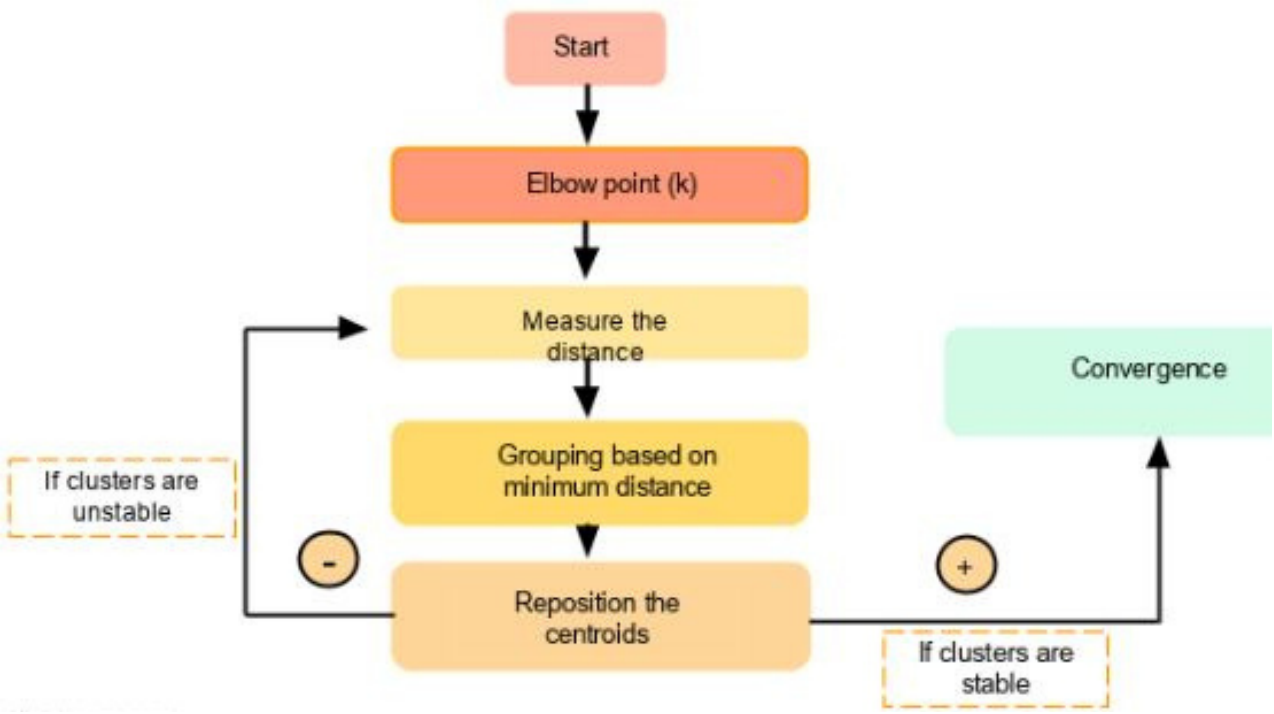
Distance Measure

Distance measure determines the similarity between two elements and influences the shape of clusters.

K-Means clustering supports various kinds of distance measures, such as:

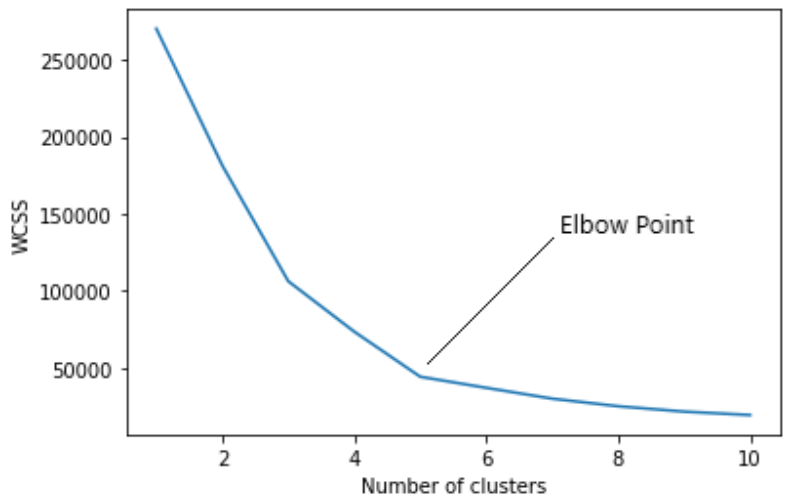
- Euclidean distance measure
- Manhattan distance measure
- A squared euclidean distance measure
- Cosine distance measure

Flowchart to execute K means Cluster Algorithm



The **Elbow method** is used to find the K value

- WCSS is the sum of squared distance between each point and the centroid in a cluster.
- When we plot the WCSS with the K value, the plot looks like an Elbow. As the number of clusters increases, the WCSS value will start to decrease. WCSS value is largest when K = 1.
- When we analyze the graph we can see that the graph will rapidly change at a point and thus creating an elbow shape. From this point, the graph starts to move almost parallel to the X-axis. The K value corresponding to this point is the optimal K value or an optimal number of clusters.



K-Means clustering is used in a variety of examples or business cases in real life, like:

- Academic performance
- Diagnostic systems
- Search engines

What is Principal Component Analysis?

The Principal Component Analysis is a popular unsupervised learning technique for reducing the dimensionality of data. It increases interpretability yet, at the same time, it minimizes information loss. It helps to find the most significant features in a dataset and makes the data easy for plotting in 2D and 3D. PCA helps in finding a sequence of linear combinations of variables.

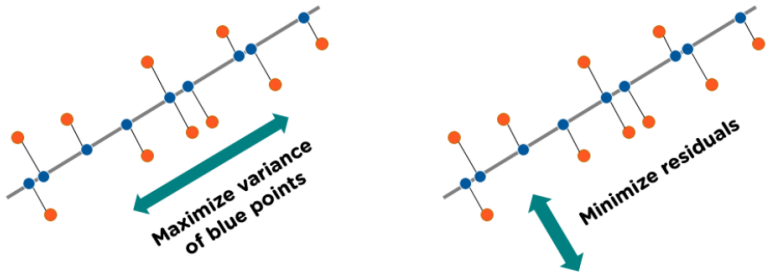
What is a Principal Component?

The Principal Components are a straight line that captures most of the variance of the data. They have a direction and magnitude. Principal components are orthogonal projections (perpendicular) of data onto lower-dimensional space.

Application of PCA

- PCA is used to visualize multidimensional data.
- It is used to reduce the number of dimensions in healthcare data.
- PCA can help resize an image.
- It can be used in finance to analyze stock data and forecast returns.
- PCA helps to find patterns in high-dimensional datasets.

How does Principal Component Analysis Work?

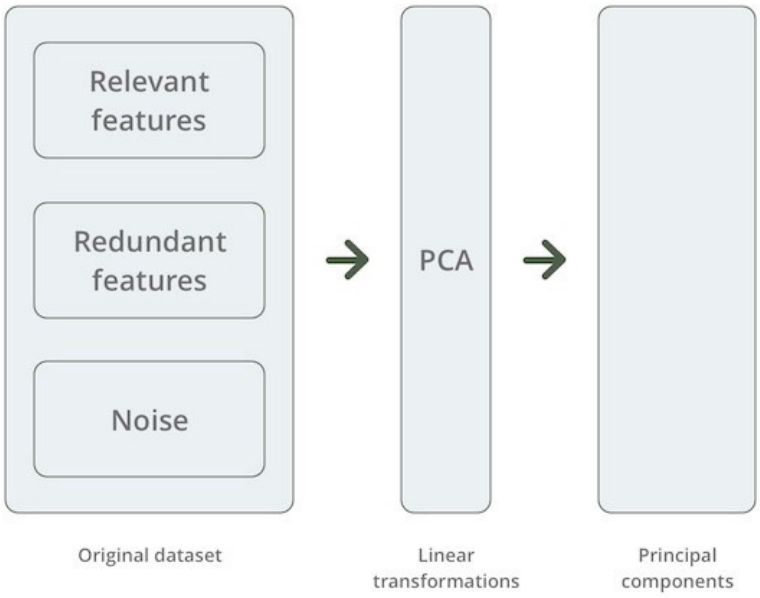


1. Normalize the data
2. Build the covariance matrix: Construct a square matrix to express the correlation between two or more features in a multidimensional dataset.
3. Find the Eigenvectors and Eigenvalues: Calculate the eigenvectors/unit vectors and eigenvalues. Eigenvalues are scalars by which we multiply the eigenvector of the covariance matrix.
4. Sort the eigenvectors in the highest to lowest order and select the number of principal components.

For instance, there is a dataset whose features are:

- Total number of doors in a house,
- Total number of bedrooms,
- Total number of bathrooms.

Having a feature for the total number of doors doesn't seem like it could reveal something critical about the data since we have the other two features.
Putting it in statistical terms, the total number of doors in a house is most likely a redundant feature and **doesn't explain the variance in the data.**
If the total number of doors in a house is a redundant feature, we can remove it from the dataset. This makes sense in our case, but in a dataset with **hundreds or even thousands of features**, we need to be more **cautious**.
In a dataset with a large number of features, we need to be sure that a feature is redundant or noise before removing it.
We might be removing important information about the data without knowing.



Natural Language Processing

Imagine you work for Google News and you want to group news articles by topic Or you work for a legal firm and you need to sift through thousands of pages of legal documents to find relevant ones. This is where NLP can help!
We will want to:

- Compile Documents
- Feature Them
- Compare their features

Simple Example:
You have 2 documents:
"Blue House"
"Red House"
Featurize based on word count:
"Blue House" -> (red,blue,house) -> (0,1,1)
"Red House" -> (red,blue,house) -> (1,0,1)

A document represented as a count of words is called "Bag of Words".
We can use Cosine similarity on those vectors to determine similarity.

We can improve on Bag of Words by adjusting word counts based on their frequency in the corpus (the group of all the documents)
We can use TF-IDF (Term Frequency - Inverse Document Frequency)

Term Frequency - Importance of the term within that document

- TF(d,t) = Number of occurrences of term t in document d

Inverse Document Frequency - Importance of the term in the corpus

- IDF(t) = log(D/t) where

D = total number of documents
t= number of documents with the term

Natural Language Processing or NLP is a field of Artificial Intelligence that gives machines the ability to read, understand and derive meaning from human languages.

Use Cases of NLP

NLP enables the recognition and prediction of diseases based on electronic health records and patients' own speech.

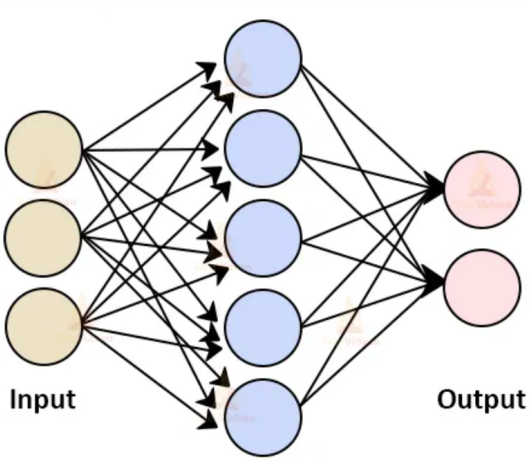
- For example, Amazon Comprehend Medical is a service that uses NLP to extract disease conditions, medications and treatment outcomes from patient notes, clinical trial reports and other electronic health records.

- companies like Yahoo and Google filter and classify your emails with NLP by analyzing text in emails that flow through their servers and stopping spam before they even enter your inbox.

- Amazon's Alexa and Apple's Siri are examples of intelligent voice-driven interfaces that use NLP to respond to vocal prompts and do everything like find a particular shop, tell us the weather forecast, suggest the best route to the office or turn on the lights at home.

What is Neural Nets and Deep Learning?

Neural Network is a series of algorithms that are trying to mimic the human brain and find the relationship between the sets of data. It is being used in various use-cases like in regression, classification, Image Recognition and many more.



Neural networks are a class of machine learning algorithms. The neuron part of the neural is the computational component, and the network part is how the neurons are connected. Neural networks pass data among themselves, gathering more and more meaning as the data moves along. Because the networks are interconnected, more complex data can be processed more efficiently.
A neural network has many layers and each layer performs a specific function, and as the complexity of the model increases, the number of layers also increases that why it is known as the multi-layer perceptron.

The purest form of a neural network has three layers :

- an input layer, the hidden layer, and the output layer.

The input layer picks up the input signals and transfers them to the next layer and finally, the output layer gives the final prediction and these neural networks have to be trained with some training data as well like machine learning algorithms before providing a particular problem.

Advantages

1. ANN has the ability to learn and model non-linear and complex relationships as many relationships between input and output are non-linear.
2. After training, ANN can infer unseen relationships from unseen data, and hence it is generalized.
3. Unlike many machine learning models, ANN does not have restrictions on datasets like data should be Gaussian distributed or nay other distribution.

Applications

There are many applications of ANN. Some of them are :

1. Image Preprocessing and Character Recognition.
2. Forecasting.
3. Credit rating.
4. Fraud Detection.
5. Portfolio Management.

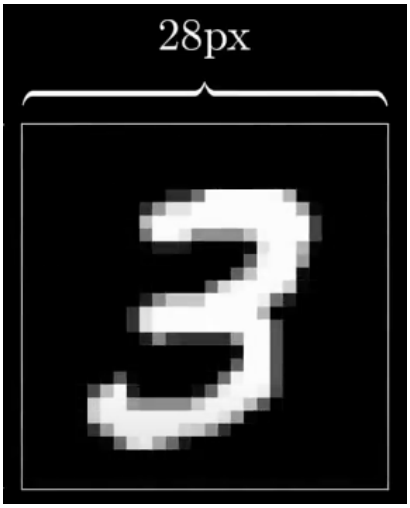
Why Deep Learning is Game Changing

Deep Learning algorithms use Artificial Neural Networks as their main structure. What sets them apart from other algorithms is that they don't require expert input during the feature design and engineering phase. Neural Networks can learn the characteristics of the data.

Neural Networks are inspired by, but not necessarily an exact model of, the structure of the brain. There's a lot we still don't know about the brain and how it works, but it has been serving as inspiration in many scientific areas due to its ability to develop intelligence.

To have a more deep understanding of Neural networks let's look into this illustration

What is this?



This is a three which is written sloppily and rendered an extremely low resolution of 28 by 28 pixels.

But your brain has no trouble recognizing it as a three.

And how can we do the same with a computer?

This is where **neural networks** come into play and they create multiple hidden layers which when we feed data to the machine tries to replicate functions of neurons of our brain.

Keras

It is an Open Source Neural Network library that runs on top of Theano or Tensorflow. It is designed to be fast and easy for the user to use. It is a useful library to construct any deep learning algorithm of whatever choice we want.

TensorFlow

TensorFlow is an open-source platform for machine learning and a symbolic math library that is used for machine learning applications.

The creation of framework can be of the following two types –

- Sequential API
- Functional API

When to use a Sequential model :
A Sequential model is appropriate for **a plain stack of layers** where each layer has **exactly one input tensor and one output tensor**.

A Sequential model is **not appropriate** when:

- Your model has multiple inputs or multiple outputs
- Any of your layers have multiple inputs or multiple outputs
- You need to do layer sharing
- You want non-linear topology (e.g. a residual connection, a multi-branch model)

Consider the following eight steps to create a deep learning model in Keras –

- Loading the data
- Preprocess the loaded data
- Definition of model
- Compiling the model
- Fit the specified model
- Evaluate it
- Make the required predictions
- Save the model

Creating a Sequential model
-> You can create a Sequential model by passing a list of layers to the Sequential constructor << model= keras.Sequential

-> Its layers are accessible via the layers attribute << model.layers()

-> You can also create a Sequential model incrementally via the add() method << model.add()

-> pop() method to remove layers: a Sequential model behaves very much like a list of layers.

->Then the model is compiled using model.compile() function with the following as parameters.

->Loss function, metrics and optimizers. (whose parameter documentation is available here)

Since Data Science is a vast topic, And this being a documentation of data science Bootcamp there are a few topics that are left such as topics mentioned below.

Topics to be covered in future:

- Perceptron
- Tensorflow and Keras in detail
- Computational Graph
- How to write a custom loss function
- How to create a model without using sequential
- How to access weights of a layer
- How to train a model without using .fit()
- how to write a custom training loop
- how to visualise the TensorFlow model
- how to load and save the model
- understand TensorFlow callback
- and try writing a custom call back