Data Preprocessing and Predictive Models for Tabular Datasets

Table of Content

Overview of both workshops

The goal of these workshops is to expose different types of data for high-school students, and teach them how to preprocess the data to make it ready and suitable to be used. In addition, they will learn about different types of Machine Learning models and how to develop them, such as Linear Regression, Decision Trees, K-Nearest Neighbors (KNN), and Support Vector Machines (SVM). At the end, students will apply their knowledge on a project of their choice using Python.
Note that the following is a detailed guide for enthusiastic students who want to learn more about what has been covered during the workshops.

Duration

Five hours (i.e. Two hours and a half for each workshop).

Name of workshop facilitator

Rim Achour
Nataly Dalal

Learning Objectives

1- Learn different types of data: their characteristics, features and methods of visualization with their importance.
2- Learn how to preprocess tabular data.
3- Learn different model categories and types.
4- Learn how to how to develop a Machine Learning model and how to tune it.

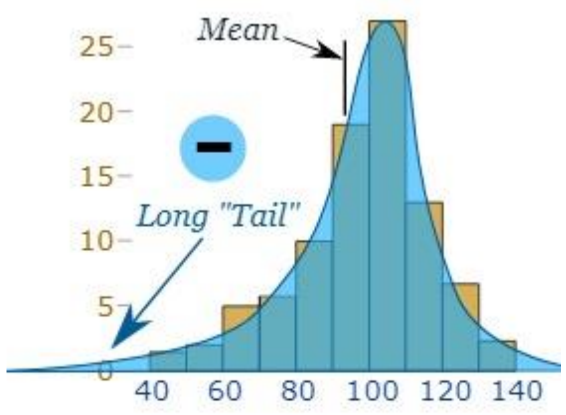
Workshop1

A- Data Preprocessing:

1- Importance

Data preprocessing is an essential first step before developing a model.
Data usually suffers from several issues like:

- ***Incompleteness*****:** missing values e.g. address = “ “.
- ***Noisiness*****:** illogical information e.g. age = “-8”, misspelling: e.g. car = “otomobile”.
- ***Inconsistency*****:** e.g. age = “21” - birthday “09/07/2000”.
- ***Skewed*****:** occurs when the majority of data are located towards a certain range, causing an asymmetrical distribution as shown in the figure below.



Usually what causes the above issues is errors in data entry, data collection, and data transmission.

2- Types of Data

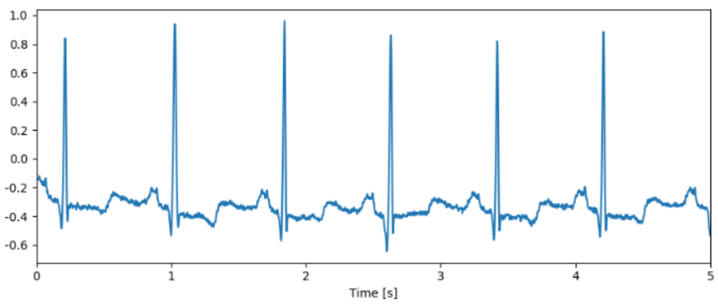
i. *Types of attribute values*

Other words for attribute: feature, independent variable, predictor

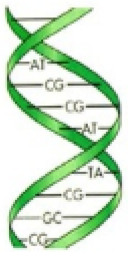
- *Numerical:* the values are numbers and it can be:
  - *Discrete:* can have only certain set of numbers e.g. dice faces
  - *Continuous:* can take any numeric value e.g. age or grade, can be:
    - *Interval-Scaled:* values are divided into intervals e.g. age
- *Categorical*: has certain options or classes
  - *Nominal*: there's no certain order for the classes e.g. color can be red/blue/purple …
  - *Ordinal*: has certain order or levels e.g. high/medium/low
  - *Binary*: present or not present e.g. has disease/doesn't have disease
- *Vector Objects*: e.g. text
- *Variables of Mixed types*

ii. *Data categories:*

- *Numerical*
- *Images*
- *Text*
- *Time-Series*: variable that varies with time e.g. stock change, monthly/yearly income change, ECG signal (figure below)



- *Sequence*: has a certain repeated pattern e.g. DNA sequence:



iii. *Feature extraction*

Feature extraction is a method of dimensionality reduction where certain features or aspects are extracted from data in the raw form (example images or time-series). These features are selected to best capture the data while reducing it and hence its processing. [1]

So, in general, data can be dealt with as raw or as extracted features. If we need the data to be in tabular form, feature extraction is a must.

A. **How to deal with images as numerical data**

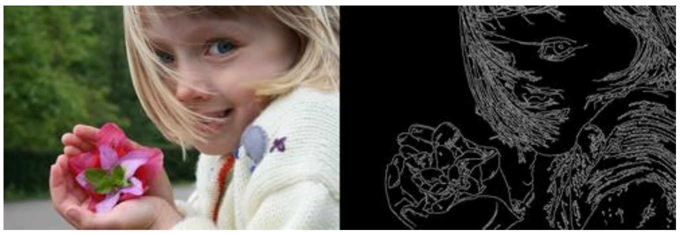
a. As raw image

To deal with raw images, methods of deep learning are used that involve neural networks which take the raw image and perform feature extraction themselves inside the network. Some pre-processing can be done like rescaling, changing resolution, image augmentation, loading the image as a matrix and normalizing its pixels, changing it into an array… [2]

b. Feature Extraction

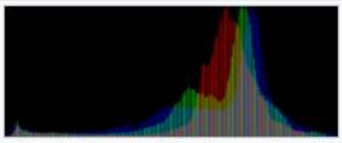
Some important image features can be:

- *Edges*: they basically represent boundaries between different regions of the image. They are considered points where the brightness experiences a sharp change or discontinuity. For that reason, they form an important feature in image processing and computer vision. [3] An example is shown below.

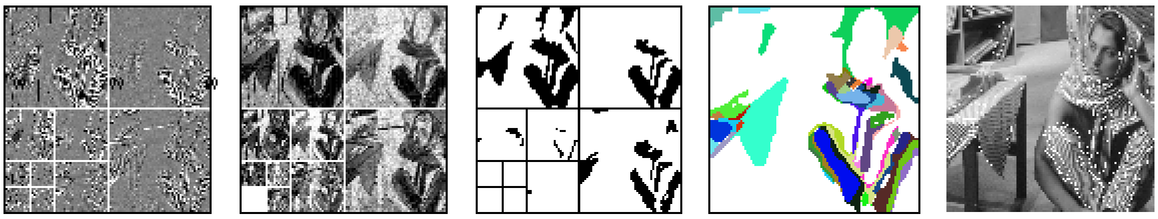


- *Color:* it is one of the first image aspects perceived by humans. In computer vision, color is represented by color histograms, which show the distribution of color over the image pixels as the example shown below [4]:

A picture of a cat


Color histogram of the above cat picture with x-axis being RGB and y-axis being the frequency.

- *Texture:* represents a group of metrics that quantify the image's perceived texture, which represents the spatial organization or distribution of the image's color or intensities, and hence forms an important feature. [5] The figure below shows an example of texture extraction [6].

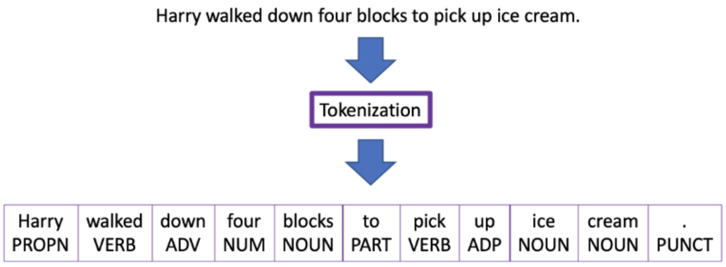


More features will be covered in the Computer Vision session.

B. How to deal with text as numerical data

a. Tokenization

Dividing a text into its individual words.



Harry	walked	down	four	blocks	to	pick	up	ice	cream	.
PROPN	VERB	ADV	NUM	NOUN	PART	VERB	ADP	NOUN	NOUN	PUNCT

b. Embeddings

Word embeddings are numerical representations of words in the aim of mapping the human understanding of these words to that of the machine. This representation is essential since machine learning algorithms can't process strings or text but need numbers instead. An example of these representations, called word vectors, can be one-hot encoding which is a binary representation of each word in a certain sentence with a 1 for the position of this word as shown below for the sentence "Go Intelligent Bot Service Artificial Intelligence, Oracle" [7]:

word	go	intelligent	bot	service	artificial	intelligence	oracle
Go	1	0	0	0	0	0	0
Intelligent	0	1	0	0	0	0	0
...	...	...	...	...	...	...	...

3- Descriptive Statistics

Before performing Data preprocessing, one has to explore the data in order to understand it and make sense of it.

Some descriptive statistics to be performed are the following:

i. *Central tendency characteristics represented by:*

a. *Average* (i.e. Weighted arithmetic mean):

➢ $\bar{x} = \mu = \frac{1}{n}\sum_{i=1}^{n} x_i$

b. *Median*: the middle value of the feature if total number of values is odd, or the average of the middle two values if the total number of values is even.

c. *Mode*: the value that occurs most frequently in the data.
A feature can be unimodal (has one mode), bimodal (has two modes) or trimodal (has three modes).

ii. *Data Dispersion measures represented by:*

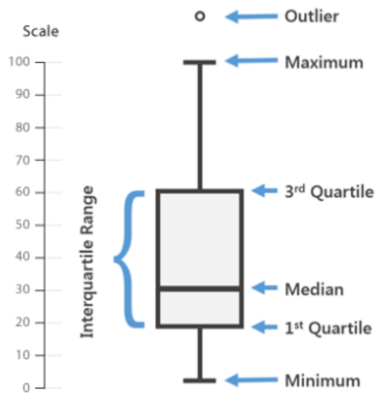
a. *Quartiles*:

1. First quartile Q_1 (i.e 25th percentile) represents 25% of the data
2. Third quartile Q_3 (i.e. 75th percentile) represents 75% of the data

b. *Inter-quartile range*:

➢ IQR = $Q_3 - Q_1$

c. *Boxplot:* a graphical representation of statistics information of the data. It shows Q_1, median, Q_3 and outliers. (Note that the figure below doesn't show the actual scale of the boxplot)



d. *Outlier:* can be considered as a value that is far located from most of the value in the dataset, one can claim that an outlier is a value higher or lower than 1.5 × IQR (above Q3 or below Q1).

e. *Variance*: measures the dispersion of the values from the mean. (or we take a statistical definition?)

➢ $Var = \sigma^2 = \frac{1}{N}\sum_{i=1}^{n}(x_i - \mu)^2$

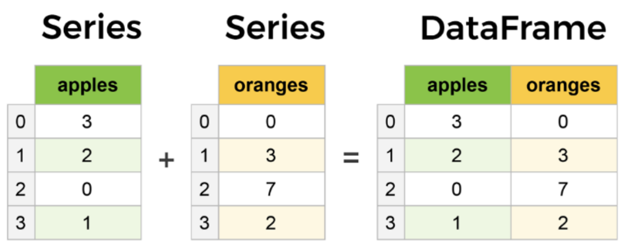
High variance means that most values are far away from the mean. Low variance indicates that most values cluster tightly about the mean.

f. *Standard deviation:* measures the amount of variation of a set of values.

➢ $\sigma = \sqrt{Var}$

4- Introduction to Pandas library in Python

Pandas is a high-level data manipulation tool built on top of NumPy and is mostly used to handle and work with datasets (storing tabular data, reading, writing, reshaping, slicing …). It has two main components: Series and DataFrame. A Series consists of one column, whereas a DataFrame is a collection of Series forming a multi-dimensional table as shown in the figure below:



Series

	apples
0	3
1	2
2	0
3	1

+

Series

	oranges
0	0
1	3
2	7
3	2

=

DataFrame

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

To get started, refer to “Introduction to Numpy and Pandas Libraries” Colab notebook.

5- Methods of Data Preprocessing

This part is applied with details in the “Data Preprocessing” Colab notebook. It covers the following:

i. Data Cleaning

a. Missing values
b. Duplicates
c. Outliers

ii. Data Transformation

a. Normalization
b. Encoding

iii. Data Visualization

a. Histogram
b. Scatter plot
c. Pie chart
d. Boxplot

Workshop2

B- Predictive Models

1- Model Categories

Recap:

- *Supervised Learning:* the training data is labelled (we have values for the targeted attribute to be predicted)
- *Unsupervised Learning:* no labels, the model divides and groups data according to similarity measures (clustering)
- *Reinforcement Learning:* the agent learns from experience where it performs actions (that move it from one state to another) which maximize the reward (best path to reach the target)

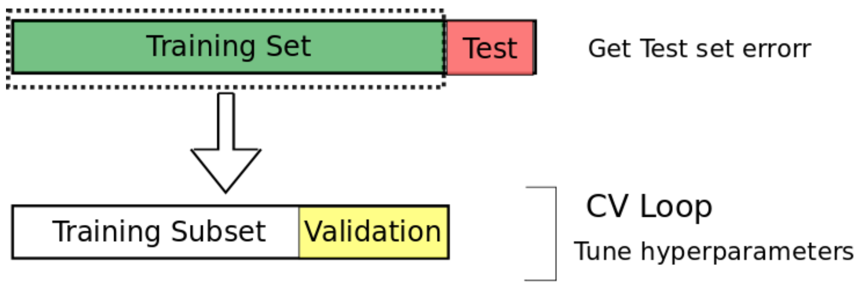
2- Model Development and Tuning

i. Data Splitting

Data should be split into the following:

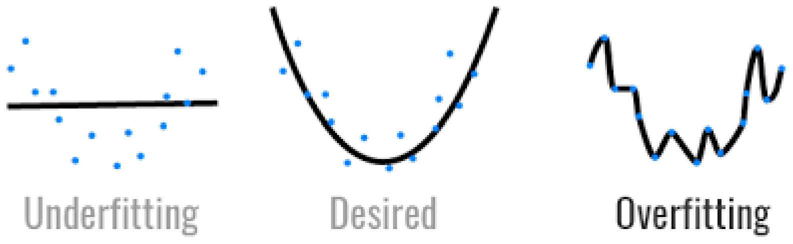
- *Training Set:* the set used to train and fit the model
- *Validation Set:* Used to tune the model (i.e. its parameters). The model doesn't use this data for learning, but we use this data to evaluate the model and improve it or tune it accordingly. This set should be unbiased (contains balanced data) so that the resulting model won't be overfitting. This can be done by continuously resampling the data (the training/validation splits) throughout the tuning process. Resampling approaches include:
  - <u>K-Fold Cross-Validation</u>: divide the data into k groups (random or balanced) then in each round, use one different group for validation while the others for training.
  - <u>Holdout</u>: repeatedly split the data randomly into 80% training and 20% validation (holdout set).
  - <u>Bootstrap</u>: random sampling but with replacement. It is similar to k-fold with k = 3.
- *Test Set:* used to test the *final* complete model (after training and tuning). It is also important when comparing different models' performances. It should also be unbiased to provide a correct evaluation of the model.

It is important to split the data so that different sets can be used for training and evaluation to give unbiased evaluation and avoid overfitting the model to the data present. The splitting ratio can be 67%-80% for training (train + validate) versus 33%-20% for testing (rule of thumb). This splitting can be random (not recommended for imbalanced data) or stratified (data is partitioned into sub-groups each having similar attribute classes, then the training and test sets are formed by picking random samples from each sub-group to result in balanced sets).

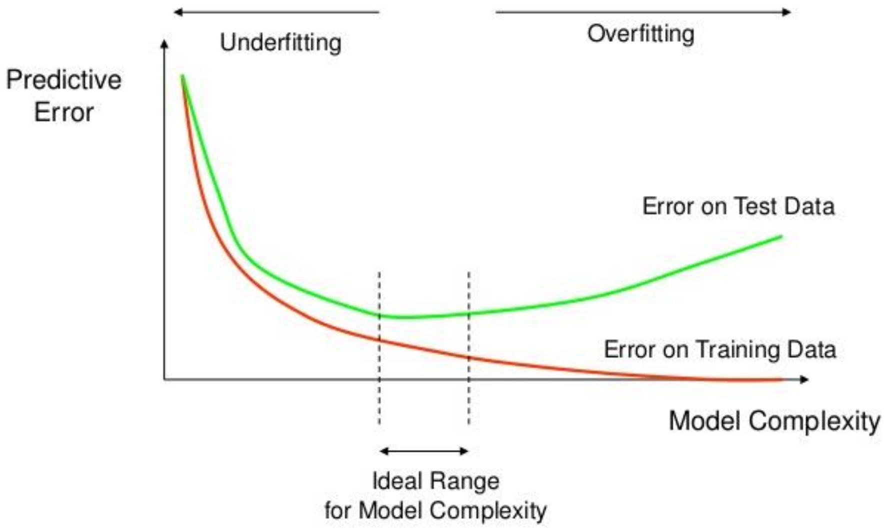


ii. *Overfitting*

Overfitting is when the model fits the training data too well. It'll then be very specific to this data, giving suspiciously high accuracy on it, but much lower accuracy on new unseen data. The opposite is underfitting where the model is way too general or simple. Below is an example demonstrating these concepts.



The below graph shows the bias-variance trade-off where underfitting will give a very simple model (low variance) hence lower accuracy (more errors) but with lower bias to the training data, whereas overfitting will give a more complex model with high variance (shown clearly in the figure above) hence high accuracy on the training data but a low accuracy on the test data (new unseen data) due to the high bias to the training set. We want a model that is in between with a bias-variance balance.



Overfitting occurs when the training data is too small or when it's very unbalanced or biased. It also occurs when we use the same set for training and validation/testing. Hence, the solution is to split the data carefully and use a relatively large training set that consists of balanced data (among the different attribute classes). Resampling approaches mentioned before in model tuning are very important in avoiding overfitting.

iii. *Comparing Models*

It is important when comparing different models to make sure that this comparison is valid. For that, you should use the same data, same preprocessing steps, same splits …

iv. *Evaluation metrics:*

a. **Classification:**

True Class/Predicted Class	C_1	C_2
C_1 (P)	True Positive (TP)	False Negative (FN)
C_2 (N)	False Positive (FP)	True Negative (TN)

Where for example C_1 is Survived = Yes and C_2 is Survived = No.

$$P = TP + FN \text{ and } N = FP + TN$$

i. Accuracy

$$sensitivity\ or\ recall = \frac{\mathrm{TP}}{\mathrm{P}}$$

$$specificity = \frac{TN}{N}$$

$$accuracy = sensitivity * \frac{P}{P+N} + specificity * \frac{N}{P+N}$$

Accuracy is the percentage of test entities that are classified correctly by the model.

$$error\ rate = 1 - accuracy$$

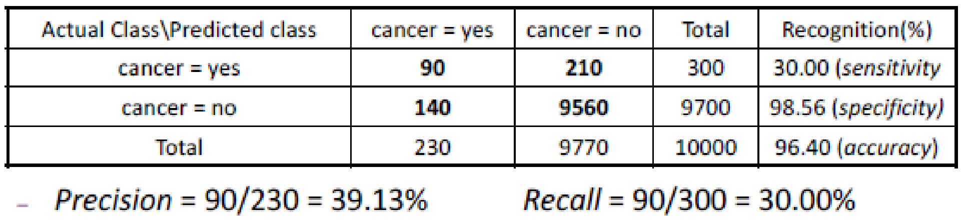
ii. F-Score (introduce balance in data)

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

$$F\ score = \frac{2 * precision * sensitivity}{precision + sensitivity}$$

F-score is the harmonic mean of precision and sensitivity. It’s similar to accuracy but better used when the data is not balanced.

Below is an example of the mentioned equations:



Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity*
cancer = no	**140**	**9560**	9700	98.56 (*specificity*)
Total	230	9770	10000	96.40 (*accuracy*)

- *Precision* = 90/230 = 39.13% *Recall* = 90/300 = 30.00%

b. **Regression**

i. MSE

MSE is the mean-square error which is the average of the squared error (the error is the difference between the actual value and the predicted one) as shown below:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Where:

- n is the number of data points
- Y_i is the actual value
- $\hat{Y}_i$ is the predicted value

Sometimes the root of MSE is taken and called RMSE (root of mean-squared error).

ii. MAE

Mean absolute error takes the absolute error instead of the squared one.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$$

iii. R^2

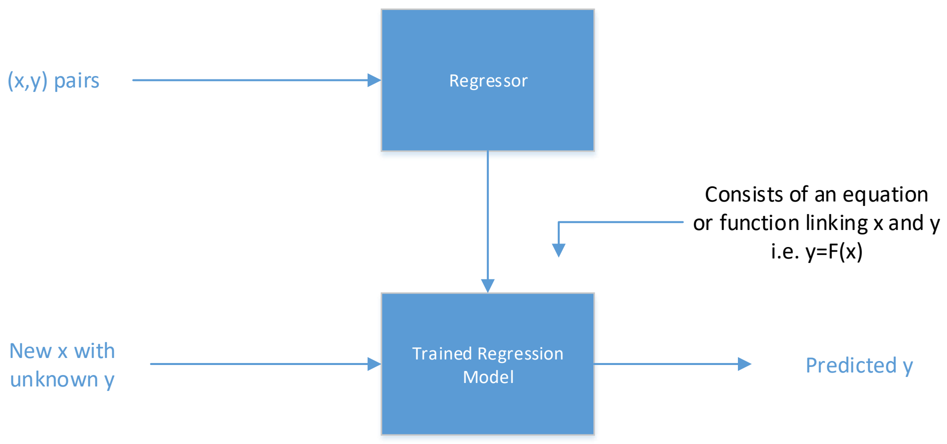
It is a value between 0 and 1 that shows how well the model fits the data (0 for poor fit and 1 for very good fit)

$$R^2 = 1 - \frac{Sum\ of\ error\ squared}{Variance\ of\ observed\ response} = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - Y_{mean_i})^2}$$

3- Regression Model Development

i. *Introduction*

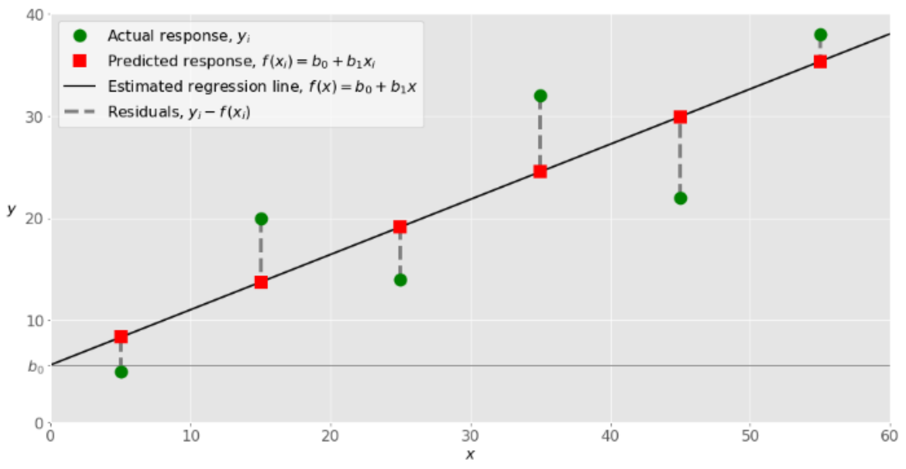
Regression is used for numerical prediction, that is, modelling continuous variables instead of categorical ones; for example: predicting the price or sales of a certain item. It usually looks for a relationship between variables, and how much the predicted variable depends on the features inputted. The below block diagram shows the basic concept of regression, with x being the inputted features and y the outcome or targeted variable.



ii. *Linear Regression*

A. Problem formulation

In linear regression, the trained model will be a linear equation or relation between x and y (y=ax + b) as shown in the figure below [8]. The green dots are the actual data or output, and the line is the model trying to fit the actual data. This will give an estimated or predicted value (red square) for each actual one (green dot) and the difference between the two is the residual or error.



Example of simple linear regression

B. Example in python

Five basic steps make up the process of executing linear regression in python [8]:

- Import the needed packages and functions
- Find a suitable dataset and apply pre-processing
- Fit a regression model to the data
- Check the performance of the model with known data
- Predict new data using the model

Check “Model Development” notebook [9].

C. Testing

In notebook.

D. Metrics

In notebook.

4- Classification Model Development

i. Introduction:

A. Goal:

Classification is a function that assigns features in a collection to target categories or classes.

Its aim is to predict the target class for each feature in the most accurate way.

Hence, in classification, the attribute to be predicted is categorical (has certain classes). For example, predicting if a passenger survives or not and predicting nationality of a person are classification problems.

B. Input and Output of the Learning model:

The model takes an input X_train and Y_train that represents usually around 80% of the numerical features and label in the dataset. It outputs the predicted set of Y which is our label. Therefore, it has the same general architecture as regression (block diagram in section 3i) but the predicted output is categorical not numerical.

ii. Algorithms:

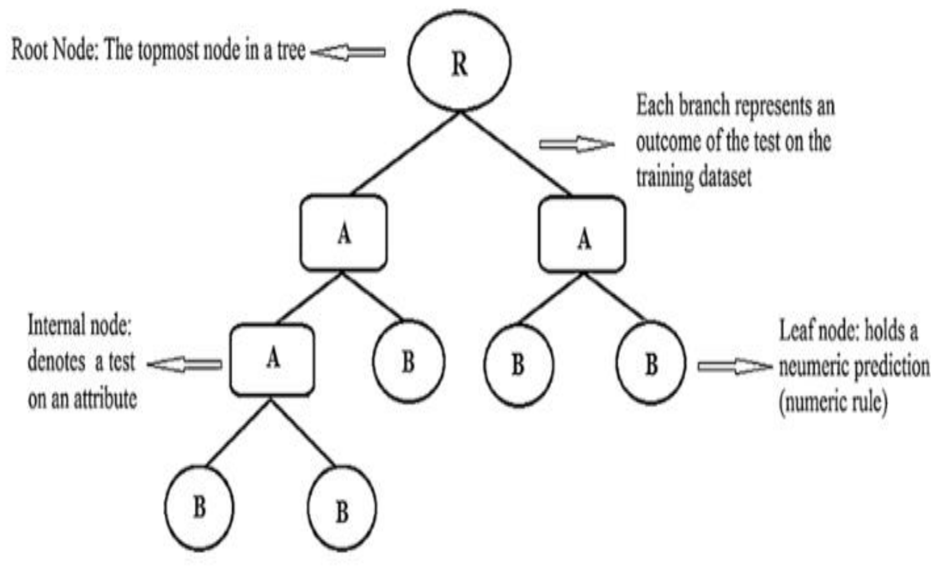
A. Decision Tree

i. Model

The decision tree is used in order to present the data in a structured way and predict results by using certain patterns.

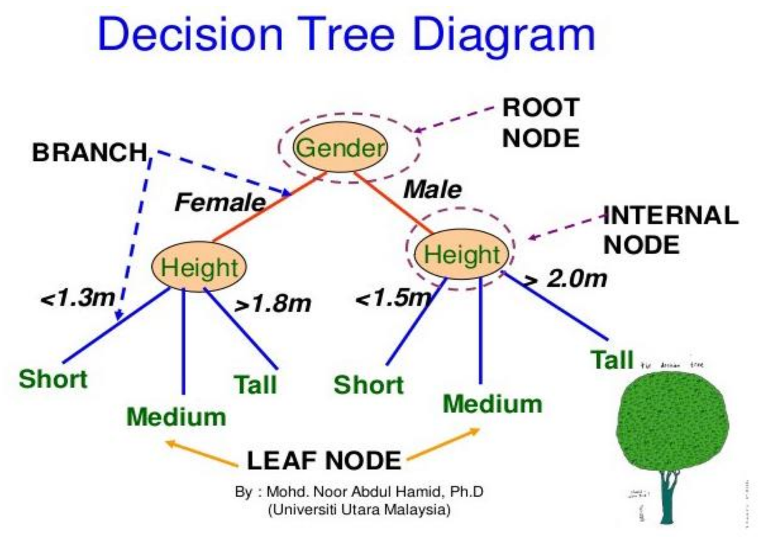
The model of the decision tree is based on the below:

- A top-down recursive divide-and-conquer method: This method consists in dividing the data into smaller shares. The tree is constituted of different types of nodes which represent each a set of data. The tree must start with a *root node*. Then, two types of nodes will be encountered. The *decision nodes* or *internal nodes* which show the decision to be taken. These nodes are fragmented into branches that show the choices leading to the corresponding decision. *The leaf nodes* constitute the end of the tree and show the result obtained due to the decisions made previously.
- Attributes are categorical.
- Samples (or tuples) are divided in a recursive manner according to the correspondant attributes.
- Below is a figure showing a basic decision:


Root Node: The topmost node in a tree
R
Each branch represents an outcome of the test on the training dataset
A
A
Internal node: denotes a test on an attribute
A
B
B
B
Leaf node: holds a neumeric prediction (numeric rule)
B
B

Basic Decision Tree [10]

Moreover, we can find below a figure showing an example of this basic decision tree. The example portrays the division of a group of students by gender and by height:



Decision Tree Students' Example [10]

ii. Info gain

To build a decision tree, we need to calculate the Entropy or the Gini Index.
The entropy checks for the homogeneity of the dataset. In fact, if the dataset is homogeneous, the entropy will be equal to zero and if it is equally divided, the entropy will be equal to 1.

The information gain is based on the decrease in entropy after a dataset is split on a feature. The aim is to find the feature that returns the highest information gain (i.e. the most homogeneous branches).

- Entropy (Target) = $\sum_{i=1}^{c} -p_i \, log_2 \, p_i$
- Entropy (Target, Feature) = $\sum_{c \in X} P(c) Entropy(c)$
- Gain (Target) = Entropy(Target) – Entropy(Target, Feature)

A branch with entropy of 0 is a leaf node.
A branch with entropy more than 0 needs further splitting.

The Gini Index is a measure of how often a randomly chosen feature from the dataset would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset.
• Gini index can be computed by summing (the probability of each item being chosen from a certain category) times (the probability of a mistake in categorizing that item).

- Gini (df) = $\sum_{i=1}^{n} p_i \, (1 - \, p_i)$

• Its minimum is equal to zero when all possible cases in the node are of the same target category.

• If a dataset (df) contains examples from n classes, gini index, Gini (df) is defined as:

➢ Gini (df) = $1 - \sum_{i=1}^{n} p_i^2$

where p_i is the relative frequency of class i in df.

• If a dataset (df) is split on A into two subsets df1 and df2, the gini index Gini (df) is defined as:

➢ $Gini_A$(df) = $\frac{|df1|}{|df|}$ Gini(df1) + $\frac{|df2|}{|df|}$ Gini(df2)

iii. Code in python

In notebook.

iv. Testing

In notebook.

v. Metrics

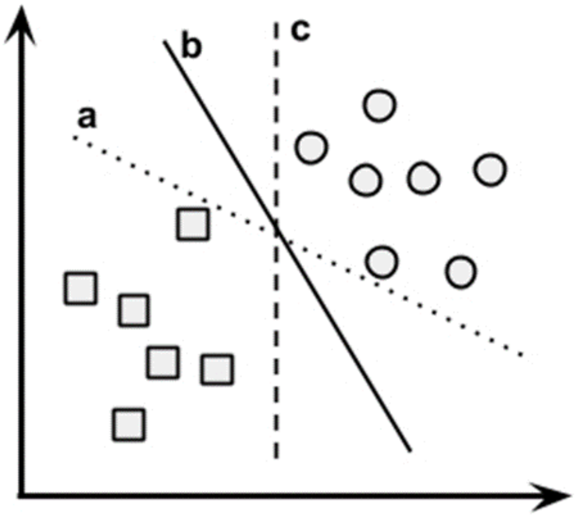
In notebook.

B. Support Vector Machine (SVM)

i. Linear SVM

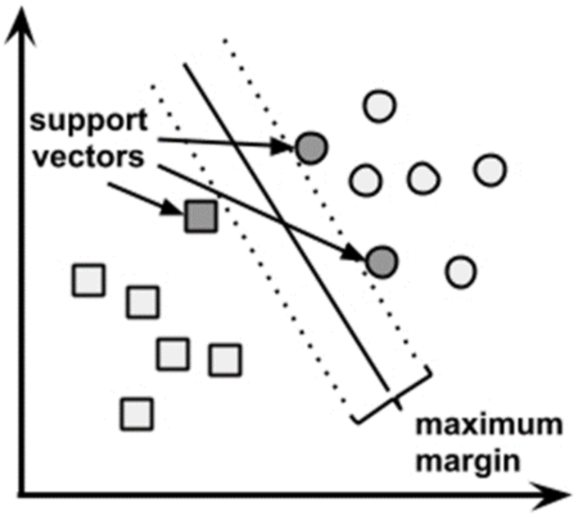
1. Model

- The aim of the SVM algorithm is to identify a line that separates two classes of data.
- One can think of more than one choice of dividing line between the two classes of data (i.e. circles and squares in the below figure [11]).
- Three such possibilities are labeled a, b, and c.
- In this case, the data is linearly separable.
- So how does the algorithm choose?



- This involves a search for the **Maximum Margin Hyperplane (MMH)**
- **The MMH** creates the greatest separation between the two classes.
- Now, any of the three lines (a, b, c) separating the circles and squares would correctly classify all the data points. However, it is more likely that the line that shows the greatest separation will generalize better for future data.

- Note that a slight variation in the positions of the points near the boundary of a line might cause one of them to drop over the line by chance.
- **The support vectors** are the points from each class that are the closest to the MMH.
- They are indicated by arrows in the figure below [11].



- An alternative (but equivalent) approach involves a search through the space of every possible hyperplane.
- The search is to find a set of two parallel planes which:
  - divide the points into homogeneous groups
  - yet themselves are as far apart as possible.
- In n-dimensional space, the following equation is used to define a **hyperplane:**
  - $\vec{w} \,.\, \vec{x}$ + b = 0
- w is a vector of n weights w = $\{w_1, w_2, \ldots, w_n\}$.
- b is a single number known as the bias.
- Using this formula, **the aim of the approach** is to find a set of weights that specify two hyperplanes, as follows:
  - $\vec{w} \,.\, \vec{x}$ + b $\geq +1$
  - $\vec{w} \,.\, \vec{x}$ + b ≤ -1
- This requires that all the points of one class fall above the first hyperplane and all the points of the other class fall beneath the second hyperplane.
- The distance between these two planes is defined as followed:
  - $\frac{2}{||\vec{w}||}$, where $||\vec{w}||$ represents the Euclidian distance.
- Recall, the goal was to maximize the distance (i.e. increase the margins). Therefore, **we need to minimize $||\vec{w}||$.**
- In fact, the task is reformulated as a constrained problem as followed:
  - min $\frac{1}{2}||\vec{w}||^2$
    subject to $y_i(\vec{w} \,.\, \vec{x}$ - b) ≥ 1, $\forall\, x_i$.

2. *Tuning Parameters: Kernel, Regularization, Gamma and Margin*

• Kernel: Linear Kernel

It is represented by the dot product of the features:

- $K(\vec{x_i}, \vec{x_j}) = \vec{x_i} \cdot \vec{x_j}$

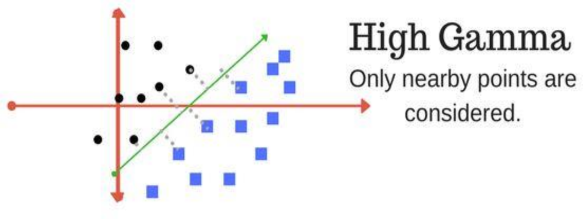
• Regularization C

It determines how much one is avoiding misclassification.

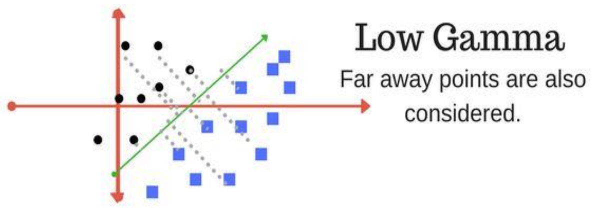
Large values of C will lead to smaller-margin hyperplane with a higher chance of getting all the training points classified correctly than choosing a very small value of C, which will cause a larger-margin separating hyperplane.

• Gamma

It defines how far the influence of a single training example reaches, with low values (i.e. 'far') and high values (i.e. 'close'). This is shown in the below figures taken from [12].



High Gamma

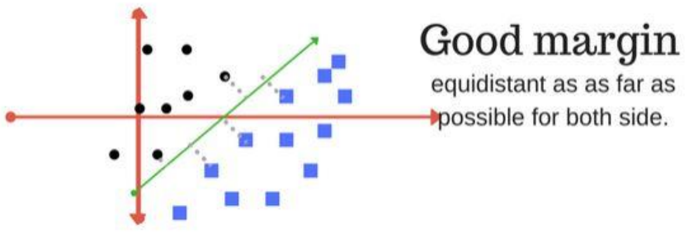


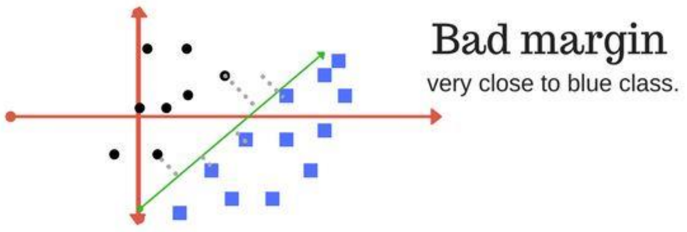
Low Gamma

• Margin

Recall: A good margin is one where the separation of both classes is larger for both the classes. Hence, a good margin allows the points to be in their respective classes without crossing to other class [12].

The figures below give represents an example of good and bad margin [12].





3. Model in python

In notebook.

4. Testing

In notebook.

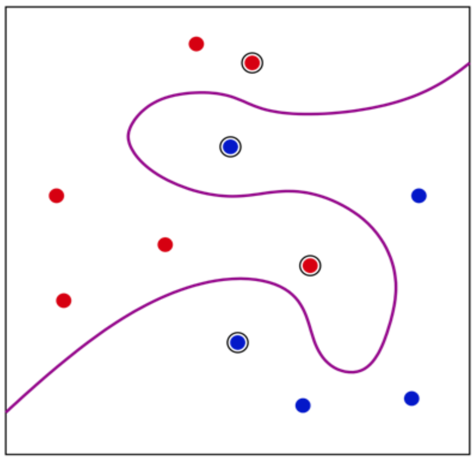
5. Evaluation metrics

In notebook.

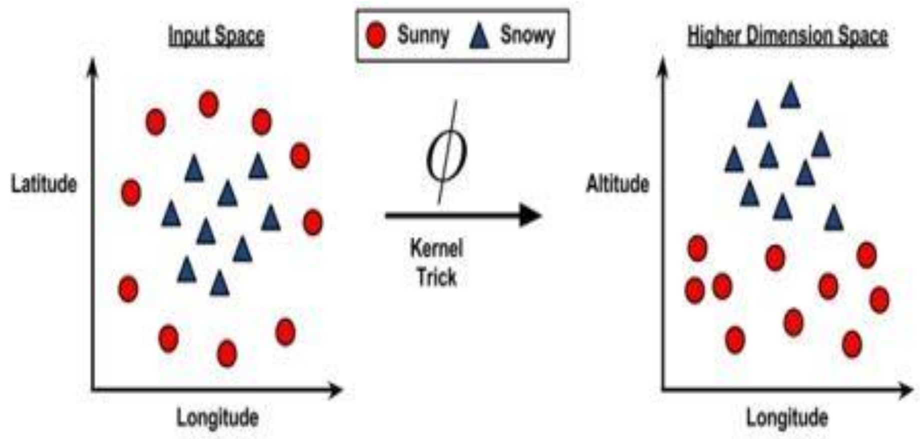
ii. Non-Linear SVM

1. Motivating example

- What if the data is not linearly separable?
- An example of a non-linearly separable data is represented in the figure below [13]:



- To approach the problem of non-linearity, SVMs can map the problem into a higher dimension space using a process known as the **kernel trick**.
- An example is represented in the figure below [11]:



2. Kernels

Kernel functions are generally of the following form:

➢ $K(\vec{x_i}, \vec{x_j}) = \emptyset(\vec{x_i}) . \emptyset(\vec{x_j})$

Where ϕ (x) is a mapping of the data into another space.

• Polynomial Kernel

The polynomial kernel of degree d adds a simple non-linear transformation of the data where a is a constant:

➢ $K(\vec{x_i}, \vec{x_j}) = (\vec{x_i} . \vec{x_j} + a)^d$

• Sigmoid Kernel

The sigmoid kernel uses the sigmoid activation function as follows:

➢ $K(\vec{x_i}, \vec{x_j}) = tanh(k\, \vec{x_i} . \vec{x_j} - \delta)$

• Gaussian RBF Kernel

The gaussian rbf kernel is a function for which its values depend on the distance from the origin or from some point. It follows the gaussian distribution as followed:

➢ $K(\vec{x_i}, \vec{x_j}) = e^{\frac{-\|\vec{x_i} - \vec{x_j}\|^2}{2\sigma^2}}$

3. Coding example

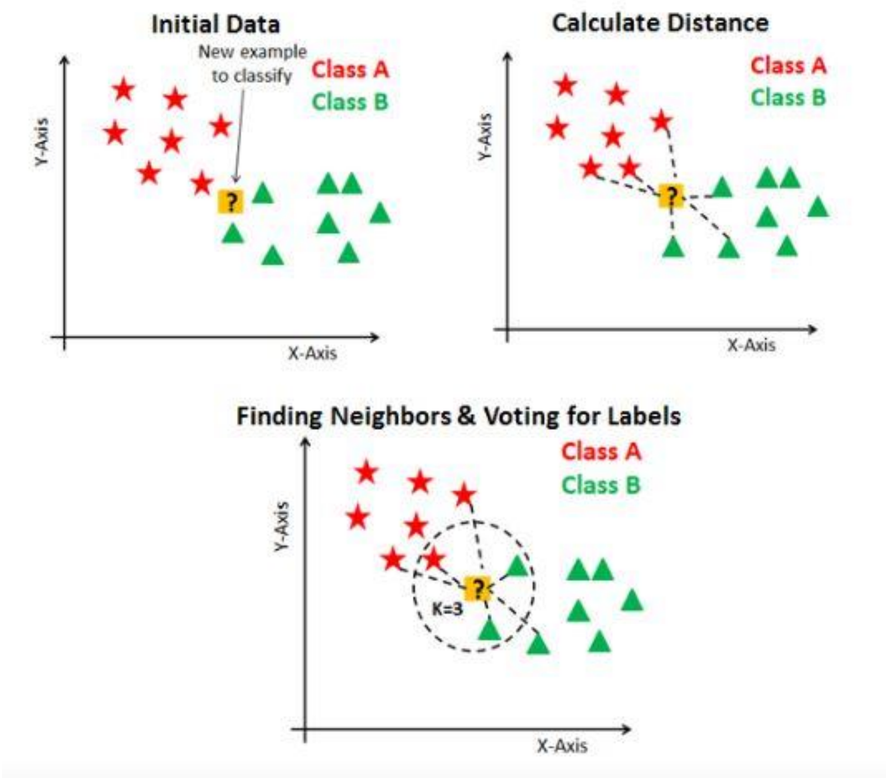
In notebook.

C. K-Nearest Neighbor (KNN)

i. How does it work

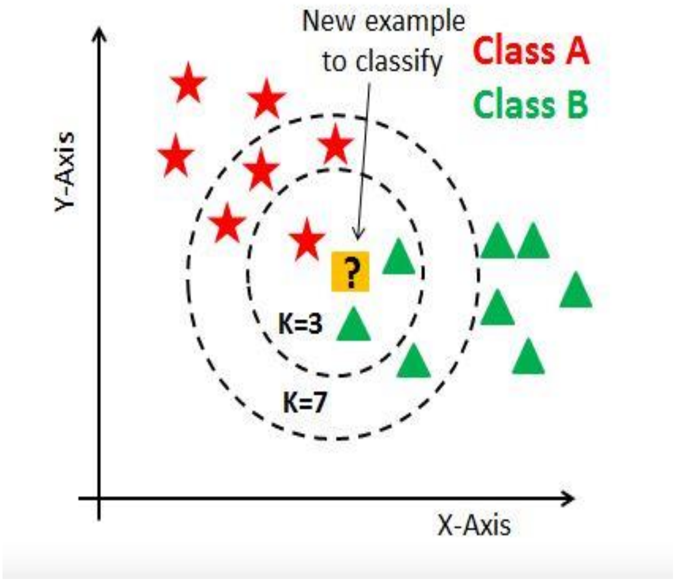
- k represents the number of nearest neighbors.
- The simplest case holds for k=1, and the algorithm is called Nearest Neighbor (NN).
- Suppose one needs to predict the label of a point p using KNN.
  - First one needs to find the k closest neighbors to p by calculating the distance between each point and p.
  - Then the classification of the label is predicted by the majority of votes of its k neighbors. Each object/point votes for their class and the class with the most votes is taken as the prediction.

- The figure below clarifies the above explanation [14]:



ii. Choosing an appropriate parameter (k)

- Deciding how many neighbors to use for kNN determines how well the model will generalize to future data.
- In practice, choosing k depends on the difficulty of the concept to be learned and the number of records in the training data.
- Research has shown that no optimal number of neighbors suits all kind of data sets. Each dataset has its own requirements [14].
- One common practice is to set k equal to the square root of the number of training examples [11].
- Choosing a large k reduces the impact or variance caused by noisy data. However, it can bias the learner such that it runs the risk of ignoring small, but important patterns and it is computationally expensive [11].
- In the case of a small number of neighbors, the noise will have a higher influence on the result. However, it is the most flexible fit for having low bias but high variance [14].
- Generally, data scientists choose k as an odd number if the number of classes is even as shown in the figure below [14].



- One can also decide on k by generating the model while examining the performance on different values of k.

iii. Model in python

In notebook.

iv. Testing

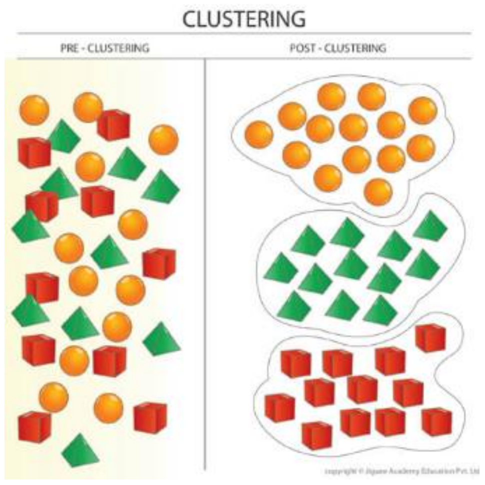
In notebook.

v. Evaluating metrics

In notebook.

5- Clustering

Clustering works on un-labeled training data by grouping them into clusters or sub-groups having certain similarities.



6- Table of Comparison

Model	Regression	Classification	Clustering	Advantages [11]	Disadvantages [11]

Linear Regression	✓			• By far the most common approach for modeling numeric data. • Can be adapted to model almost any data. • Provides estimates of the strength and size of the relationships among features and the outcome.	• Makes strong assumptions about the data. • The model's form must be specified by the user in advance. • Does not do well with missing data. • Only works with numeric features, so categorical data require extra processing. • Requires some knowledge of statistics to understand the model.
Decision Tree	✓	✓	✓	• An all-purpose classifier that does well on most problems. • Can handle numeric or nominal features, missing data. • Uses only the most important features. • Can be used on data with relatively few training examples.	• Often biased toward splits on features having a large number of levels. • It is easy to overfit or underfit the model. • Can have trouble modeling some relationships due to reliance on axis-parallel splits. • Small changes in training data can result in large changes to decision logic. • Large trees can be difficult to interpret and the decisions they make may seem counterintuitive.

SVM	✓	✓		• Not influenced by noisy data and not very prone to overfitting. • Gaining popularity due to its high accuracy and high-profile wins in data mining competitions.	• Finding the best model requires testing of various combinations of kernels and model parameters. • Can be slow to train, particularly if the input dataset has a large number of features or examples. • Results in a complex black box model that is difficult if not impossible to interpret.
KNN	✓	✓	✓	• Simple and effective. • Makes no assumptions about the underlying data distribution. • Fast training phase.	• Does not produce a model, which limits the ability to find novel insights in relationships among features. • Slow classification phase. • Requires a large amount of memory.
Neural Networks	✓	✓	✓	• Can be adapted to classification or numeric prediction problems. • Among the most accurate modeling approaches. • Makes few assumptions about the data's underlying relationships.	• Reputation of being computationally intensive and slow to train, particularly if the network topology is complex. • Easy to overfit or underfit training data. • Results in a complex black

					box model that is difficult if not impossible to interpret.
K-means	✓	✓	✓	• Uses simple principles for identifying clusters which can be explained in non-statistical terms. • Highly flexible and can be adapted to address nearly all of its shortcomings with simple adjustments. • Fairly efficient and performs well at dividing the data into useful clusters.	• Less sophisticated than more recent clustering algorithms. • Because it uses an element of random chance, it is not guaranteed to find the optimal set of clusters. • Requires a reasonable guess as to how many clusters naturally exist in the data.

References

[1]: https://deepai.org/machine-learning-glossary-and-terms/feature-extraction
[2]:https://medium.com/nybles/create-your-first-image-recognition-classifier-using-cnn-keras-and-tensorflow-backend-6eaab98d14dd
[3]: https://en.wikipedia.org/wiki/Edge_detection
[4]: https://en.wikipedia.org/wiki/Color_histogram
[5]: https://en.wikipedia.org/wiki/Image_texture
[6]: http://www.ee.columbia.edu/~jrsmith/html/pubs/abtfsfir/node7.html
[7]: https://www.datascience.com/blog/embedding-in-natural-language-processing
[8]: https://realpython.com/linear-regression-in-python/
[9]: https://becominghuman.ai/implementing-and-visualizing-linear-regression-in-python-with-scikit-learn-a073768dc688
[10]:https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb
[11]: CMPS 297O Introduction to Data Science Lecture Slides (Spring16-17) – Professor Fatima Abu Salem
[12]:https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72
[13]: CMPS 396v Machine Learning Lecture Slides (Fall17-18) – Professor Shady Elbassuoni
[14]: https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

Most of the info were also taken from EECE 633 Data Mining slides (Dr. Hazem Hajj – Fall18-19)