**What Is Data Clustering?**

**Clustering** is a classic data mining technique based on machine learning that divides ​groups of abstract objects into classes of similar objects.

Clustering helps to split data into several subsets. Each of these clusters consists of data objects with high inter-similarity and low intra-similarity.

Clustering methods can be classified into the following categories:

* Partitioning method
* Hierarchical method
* Density-based method
* Grid-based method
* Model-based method
* Constraint-based method

**What Is TF-IDF Representation?**

TF-IDF (term frequency-inverse document frequency) was invented for document search and information retrieval. It works by increasing proportionally to the number of times a word appears in a document, but is offset by the number of documents that contain the word. So, words that are common in every document, such as this, what, and if, rank low even though they may appear many times, since they don’t mean much to that document in particular.

However, if the word *Bug* appears many times in a document, while not appearing many times in others, it probably means that it’s very relevant. For example, if what we’re doing is trying to find out which topics some NPS responses belong to, the word *Bug* would probably end up being tied to the topic Reliability, since most responses containing that word would be about that topic.

**\*\*\***

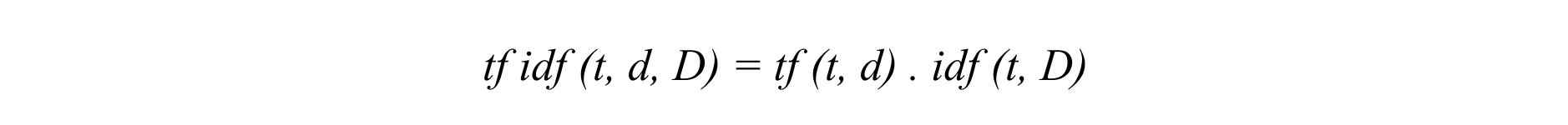
**How is TF-IDF calculated?**

TF-IDF for a word in a document is calculated by multiplying two different metrics:

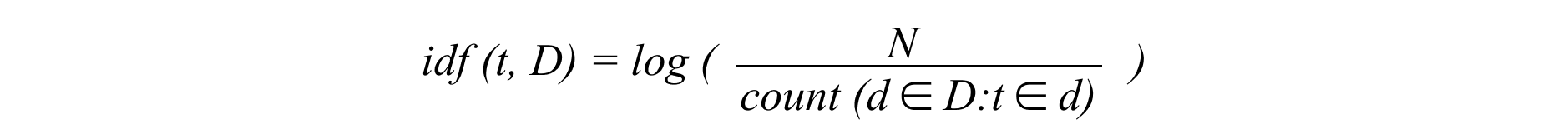
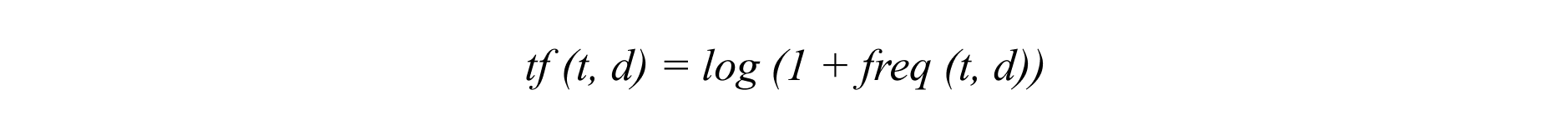
* The **term frequency** of a word in a document. There are several ways of calculating this frequency, with the simplest being a raw count of instances a word appears in a document. Then, there are ways to adjust the frequency, by length of a document, or by the raw frequency of the most frequent word in a document.
* The **inverse document frequency** of the word across a set of documents. This means, how common or rare a word is in the entire document set. The closer it is to 0, the more common a word is. This metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm.
* So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1.

Multiplying these two numbers results in the TF-IDF score of a word in a document. The higher the score, the more relevant that word is in that particular document.

To put it in more formal mathematical terms, the TF-IDF score for the word t in the document d from the document set D is calculated as follows:



Where :



**\*\*\*\***

**What is k-means Clustering Algorithm?**

**K-means algorithm** is an iterative **algorithm** that tries to partition the dataset into Kpre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. ... Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.

**Streaming K-means Algorithm?**

“Streaming K-means” typically refers to what is also known as " K-Means". Which  refer to computations that are performed iteratively, with data arriving during the computation in single observations or in batches, in contrast to "offline" algorithms, where all the data are available when the computation starts. Typically, some intermediate result is calculated based on initial data and then modified as new data arrive. For instance, exponential smoothing is a forecasting algorithm that is very naturally performed online.

In the specific case of k-means, we would first apply a standard k-means algorithm to cluster an initial dataset. Then, the cluster centers would be updated as new data arrive. Such algorithms often keep and update clusterings with different numbers of clusters, because the optimal number of clusters may change over time as data arrives.

Alternative names for algorithms include "sequential algorithms", and a number of other synonyms as per.

**\*\*\*\***

**Why We Use Sparse Representation?**

**In Sparse Vector representations (Latent Semantic Analysis)**concept of Bag of words is used where words are represented in the form of encoded vectors. It is a sparse vector representation where the dimension is equal to the size of vocabulary. If the word occurs in the dictionary, it is counted, else not. But Disadvantages of Bag of Words method is

* It ignores the order of the word, for example, **‘this is bad ‘= ‘bad is this’**.
* It ignores the context of words. Suppose If I write the sentence “He loved books. Education is best found in books”. It would create two vectors one for “He loved books” and other for “Education is best found in books.” It would treat both of them orthogonal which makes them independent, but in reality, they are related to each other

To overcome these limitations, Word Embeddings is developed.

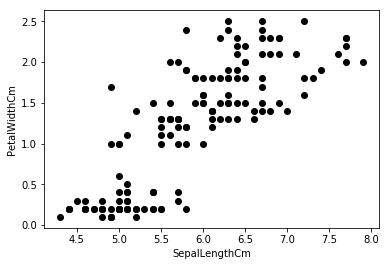
Word2Vec is an approach to implement such.

**Implementing K-means Algorithm As A Method In The Paper**

**The Objective Of The Project As Below:**

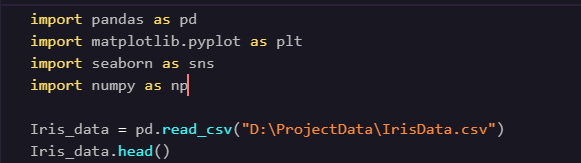


..

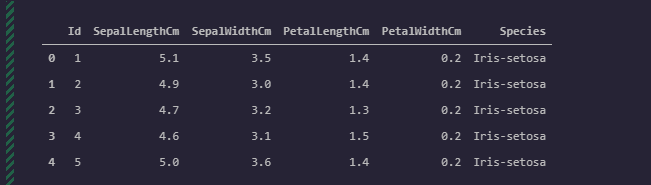


**\*\*\***

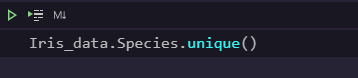
# **Step1)** ImportingData:



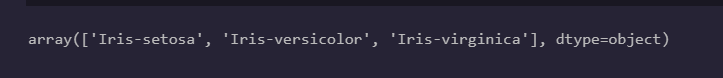
**Output**)



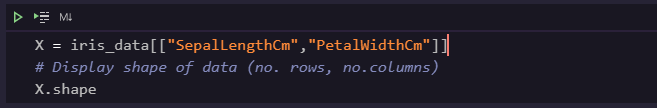
**Step 2)** Identifying the species of plants in our dataset



**Output)**

****

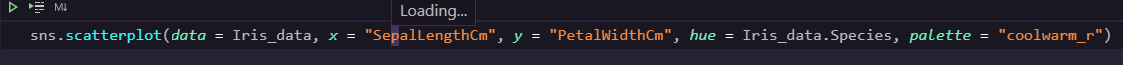
**Step 3)** Store selected data: Sepal length and Petal width into variable X

****

# **Plotting our Data**

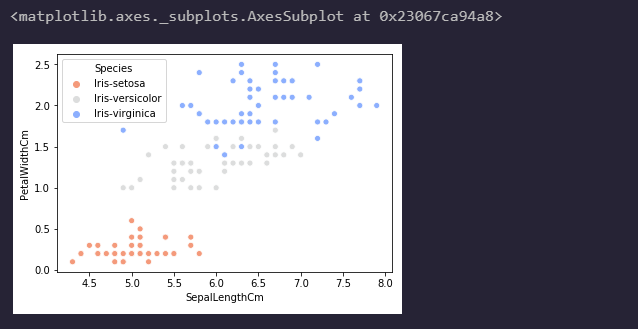
Using seaborn library

**Step4)**



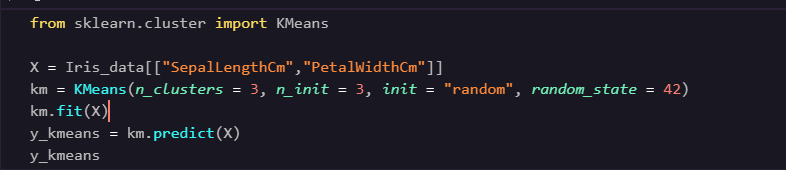
\*\*\*

**Output)**

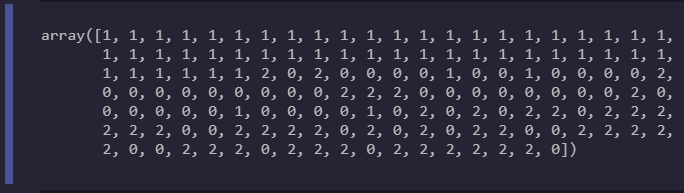
****

**\*\*\*\***

# **Step5)** Implementing K-means Algorithm

****

**Output)**

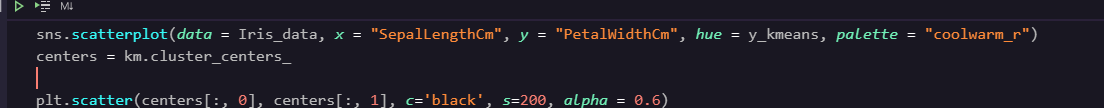
****

* *y\_kmeans gives an array of values which show which cluster each data point belongs to*

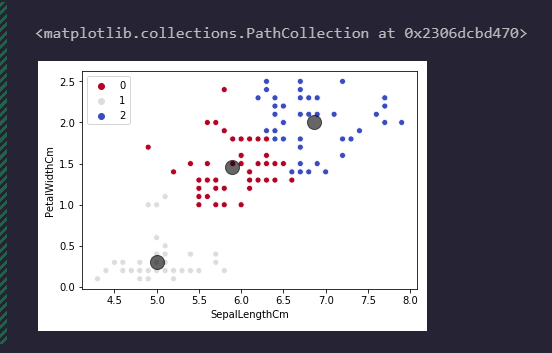
## Plotting our Clusters and Centroids

* To plot our clusters we will use the same code for the scatter plot before but simply change the **hue** to y\_kmeans and plot the centres of each cluster.

**Step 6)**

****

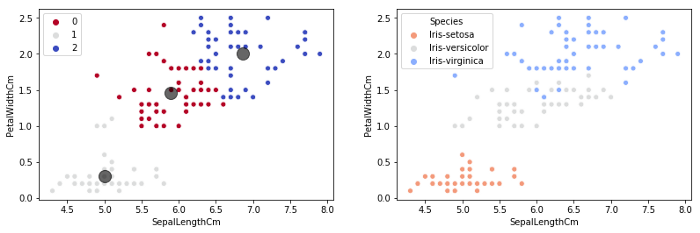
**Output)**

****

**Cluster Irritations :**

****

* **Note :** We can see above that our k-mean clustering algorithm has produced 3 clusters fairly similar to our previous plot. We can now use these clusters and centroids produced to make predictions for new flower data. Comparing clusters 0, 1 and 2 to our previous plot:

****

***Cluster 0****most likely refers to****Iris-versicolor******Cluster 1****most likely refers to****Iris-setosa******Cluster 2****most likely refers to****Iris-virginica***

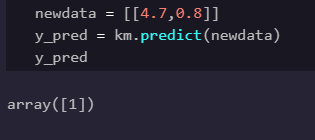
# **Making Predictions**

The clusters and centroids produced from our k-mean algorithm can be used to place any new petal width and sepal length data collected from new flowers into a cluster, essentially giving us a prediction of the flower type.

Let us say for example we recorded a flower to have a petal width of 0.8cm and sepal length of 4.8cm — what type is this flower?

Using our **model**:

**Step 7)**

****

*We expect this flower to belong to cluster or centroid 1, our middle cluster, which when comparing our two plots most likely belongs to the species****iris-setosa.***

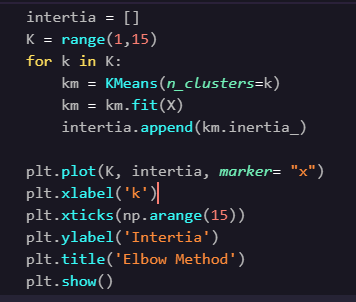
# **Selecting number of clusters K**

## ****The Elbow Method****

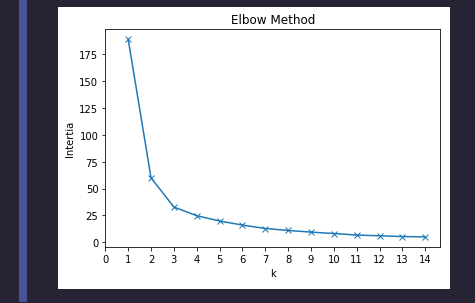
To evaluate the performance of our k-means algorithm we can take a look at the Inertia or objective function value. This is essentially the sum of squared distances our data points are away from their cluster centroid.

By looking at different Inertia values for different numbers of clusters (K):

**Step8)**

****

**Output)**

****

he “elbow” of the above graph gives the optimum number of clusters for our data. This is the point before a roughly linear decrease in Inertia — which in this case is**k = 3**. This helpfully matches our number of Iris species.