F1-Score of Clustering Algorithms on IRIS Classification

Praveen Kumar
Machine learning Intern
AI-Technology & Systems
www.ai-techsystems.com
inbox.praveen.kumar17@gmail.com

***Abstract*—Iris dataset is the best known database to be found on pattern recognition literature. We present different clustering solutions for IRIS classification. We use and compare f1-scores of each clustering algorithm and find the possible reasons for dominance of one clustering algorithm over the others with evidences to support our claim.**

Keywords—Clustering, Affinity propagation, Gaussian mixture model, K-means, IRIS

I. INTRODUCTION

IRIS is a classification dataset which is like hello world to the Machine Learning Classification algorithm. We try to classify 3 different IRIS species using clustering algorithms. Clustering algorithms are generally not preferred for the simplest reason being they fall under unsupervised learning. Such unsupervised algorithms does not perform well on less data. We try to use clustering algorithms for supporting the same hypothesis of unsupervised learning. We use 3 clustering algorithms namely Affinity propagation, Means and Gaussian mixture model and produce the f1 scores for each. We then compare and conclude and end the report by stating future scope as well as concluding statements. Section II consists of information about the dataset and some basic exploratory data analysis using graphical python libraries. All the code and results produced are not reproducible and python 3 is used throughout the project. Section III consists of feature engineering. Section IV consists of splitting of datasets and building model and training on the data. Section V consists of comparison study of the different models. Section VI consists of future scope and conclusion.

II. DATASET & EDA

A. Dataset

The *Iris* flower data set or Fisher's *Iris* data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper. *The use of multiple measurements in taxonomic problems* as an example of linear discriminant analysis. It is sometimes called Anderson's *Iris* data set because Edgar Anderson collected the data to quantify the morphologic variation of *Iris* flowers of three related species. Two of the three species were collected in the Gaspé Peninsula "all from the same pasture, and picked on the same day and measured at the same time by the same person with the same apparatus".

The data set consists of 50 samples from each of three species of *Iris* (*Iris setosa*, *Iris virginica* and *Iris versicolor*). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other [1].

B. Exploratory Data Analysis

We plot different graphs for petals and sepals and see each type of iris flower have different shapes. Let's look at each of these one by one.

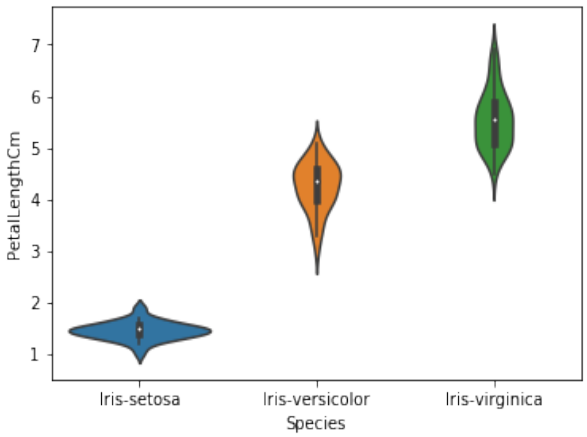

Fig 1.0

In the above figure we see that iris setosa have the lowest petal length while other have roughly the same length range. When it comes to width, it's no different.

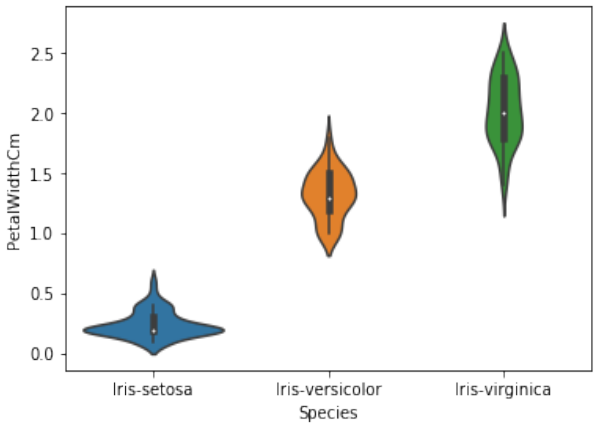

Fig 2.0

So we conclude that the iris-setosa is the most distinct flower and easily forms a single cluster while not the same case for Versicolor and Virginica. Let's see sepal attribute now.

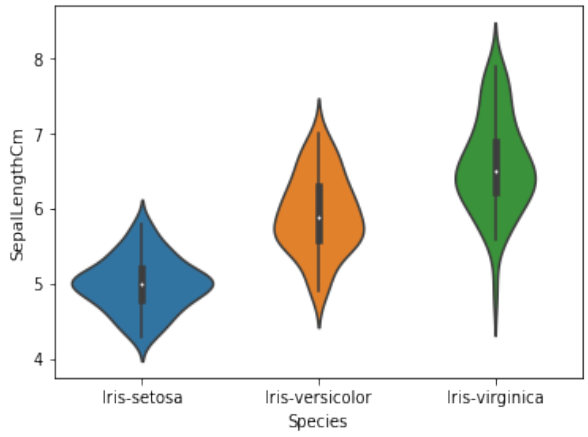

Fig 3.0

From fig 3.0, we again tend to see the same pattern. Iris-Setosa has many training examples at 5cm sepal length and hence we see a broader clustered graph. Other iris category becomes more difficult to cluster since they have almost same cluster.

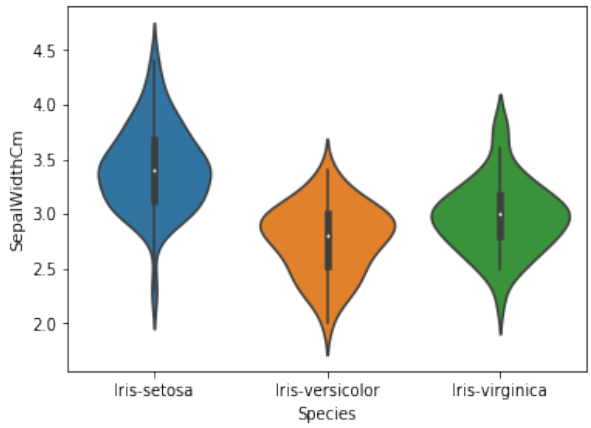

Fig 4.0

However, unlike the case of petals, setosa have much larger petal width when compared to the other iris flowers. Hence we can conclude that setosa iris is the one which has small petals and sepal length but wider sepal.

III. Feature Engineering

Considering the dataset [5], we just have 150 examples and just 4 features to train our model. When we trained the model on just 4 features, we see gradual decrease and low clustering performance since we do not have many training example. This is one of the most difficult problem in unsupervised algorithms. Unless we have more data, we need to adjust the approach to train our model otherwise it will lead very low performances. We use feature engineering to create 6 new features. We obtain 2 features by taking element wise multiplication of sepal length and width & petal length and width. Hence we form 2 new features by calculating total area of the sepal and petal. We also form 2 new features by simply taking average of length and width of sepal and width. Finally, we form other 2 features by multiplying the newly formed sepal area and petal area. Our new dataset formed have 8 features now which increases the model performance by providing much more details than the initial data. The names of the all features are:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	SepalArea	PetalArea	SepalAvg	PetalAvg	FlowerArea	AverageFlowerArea
0	5.1	3.5	1.4	0.2	Iris-setosa	17.85	0.28	4.30	0.80	4.9980	3.4400
1	4.9	3.0	1.4	0.2	Iris-setosa	14.70	0.28	3.95	0.80	4.1160	3.1600
2	4.7	3.2	1.3	0.2	Iris-setosa	15.04	0.26	3.95	0.75	3.9104	2.9625
3	4.6	3.1	1.5	0.2	Iris-setosa	14.26	0.30	3.85	0.85	4.2780	3.2725
4	5.0	3.6	1.4	0.2	Iris-setosa	18.00	0.28	4.30	0.80	5.0400	3.4400

Fig 5.0

IV. Splitting Dataset & Model Building

We split the dataset by ¾ where 75% is chosen for training and 25% for testing. So, out of 150 data examples, we have 112 for training and 38 for testing. Hence shape of training data is (112,8) & (38,) for test. After splitting, we choose our clustering algorithms and build the models and fit on the training data. The hyper parameters for each model are shown below.

KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
n_clusters=3, n_init=15, n_jobs=None, precompute_distances=True,
random_state=None, tol=0.01, verbose=0) [6]

k-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. *k*-means clustering aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells [2]. For KMeans, we choose number of cluster by using the elbow plot. We test number of clusters from 1 to 15 and choose the optimal k value. The graph is shown below.

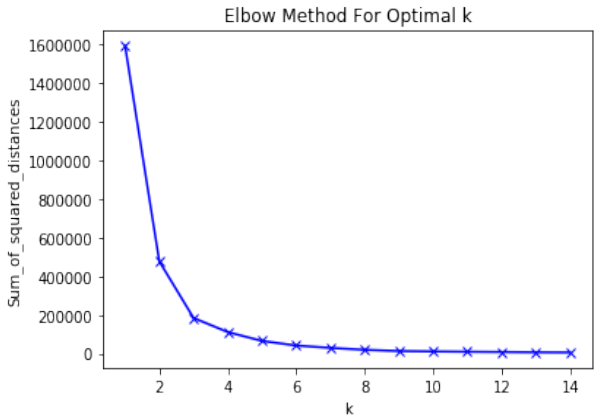

Fig 6.0

A Gaussian mixture model (GMM) is a category of probabilistic model which states that all generated data points are derived from a mixture of a finite Gaussian distributions that has no known parameters. The parameters for Gaussian mixture models are derived either from maximum a posteriori estimation or an iterative expectation-maximization algorithm from a prior model which is well trained. Gaussian mixture models are very useful when it comes to modeling data, especially data which comes from several groups [3]. The hyper parameters used for GMM during training are

GaussianMixture(covariance_type='spherical',
init_params='kmeans', max_iter=100,

means_init=None, n_components=3, n_init=3, precisions_init=None,
random_state=None, reg_covar=0.01, tol=0.01, verbose=0,
verbose_interval=10, warm_start=False, weights_init=None) [7]

In statistics and data mining, affinity propagation (AP) is a clustering algorithm based on the concept of "message passing" between data points. Unlike clustering algorithms such as *k*-means or *k*-medoids, affinity propagation does not require the number of clusters to be determined or estimated before running the algorithm. Similar to *k*-medoids, affinity propagation finds "exemplars", members of the input set that are representative of clusters [4]. The hyper parameters used during training are

AffinityPropagation(affinity='euclidean', convergence_iter=15, copy=True,
damping=0.5, max_iter=300, preference=None, verbose=False) [8]

V. Comparision of F1 scores

After training, we do prediction on the test dataset and we obtain the following classification reports on each model.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	12
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	0
3	0.27	1.00	0.43	3
4	0.00	0.00	0.00	8
5	0.00	0.00	0.00	14
accuracy			0.08	38
macro avg	0.05	0.17	0.07	38
weighted avg	0.02	0.08	0.03	38

Fig 7.0 F1 Score of KMeans

	precision	recall	f1-score	support
0	0.00	0.00	0.00	10
1	0.00	0.00	0.00	7
2	1.00	0.81	0.89	21
3	0.00	0.00	0.00	0
accuracy			0.45	38
macro avg	0.25	0.20	0.22	38
weighted avg	0.55	0.45	0.49	38

Fig 8.0 F1 Score of Gaussian mixture model

	precision	recall	f1-score	support
0	0.00	0.00	0.00	13
1	0.00	0.00	0.00	3
2	0.82	0.64	0.72	22
3	0.00	0.00	0.00	0
accuracy			0.37	38
macro avg	0.21	0.16	0.18	38
weighted avg	0.48	0.37	0.42	38

Fig 9.0 F1 Score of Affinity Propagation

Upon comparing, we obtain mean f1 score of 0.3684 on kmeans, 0.4473 on Gaussian mixture model & 0.0789 on affinity propagation. Hence the best model is produced by Gaussian mixture model, yet far away from general classification algorithms.

VI. Conclusion & Future Scope

We conclude that clustering algorithm are not a good approach when the data is too less. For iris, the best fit would be produced by classification algorithms rather than clustering based algorithms. Clustering based algorithms can only be used when the amount of data is enough so that it can help our model to training with much high variance in data.

References

[1] Wikipedia "https://en.wikipedia.org/wiki/Iris_flower_data_set"

[2] Wikipedia "https://en.wikipedia.org/wiki/K-means_clustering".

[3] Technoedia "https://www.techopedia.com/definition/30331/gaussian-mixture-model-gmm"

[4] WikiPedia "https://en.wikipedia.org/wiki/Affinity_propagation"

[5] Kaggle.com "https://www.kaggle.com/ashishs0ni/iris-dataset"

[6] Sklearn "https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html"

[7] Sklearn "https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html"

[8] Sklearn "https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AffinityPropagation.html"