Article
Niyaz Khafizov · Jul 19, 2018 4m read

K-Means clustering of the IRIS Dataset

Hi all. Today we are going to use k-means algorithm on the Iris Dataset.

Note: I have done the following on Ubuntu 18.04, Apache Zeppelin 0.8.0, python 3.6.5.

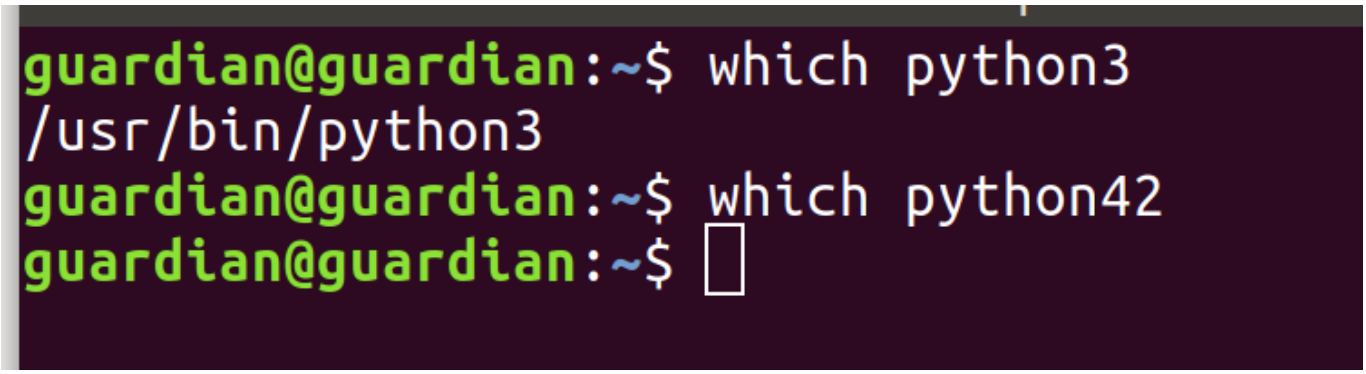
Introduction

K-Means is one of the simplest unsupervised learning algorithms that solves the clustering problem. It groups all the objects in such a way that objects in the same group (group is a cluster) are more similar (in some sense) to each other than to those in other groups. For example, assume you have an image with a red ball on the green grass. K-Means will split all pixels into two clusters. The first cluster will contain the pixels of the ball, the second cluster will contain the pixels of the grass.

IRIS Dataset is a table that contains several features of iris flowers of 3 species. Species can be "Iris-setosa", "Iris-versicolor", and "Iris-virginica". Each flower contains 5 features: **Petal Length**, **Petal Width**, **Sepal Length**, **Sepal Width**, and **Species**.

Check requirements

First of all, let us check all the requirements. Paste "**which python3**" in the terminal:



If python 3 is installed, you will see the first path. If it is empty (like "**which python42**"'s output), paste the following in the terminal:

```
sudo apt-get update

sudo apt-get -y upgrade

sudo apt install python3

sudo apt-get install -y python3-pip
```

Next, install pyspark:

```
pip3 install pyspark
```

Change **zeppelin.pyspark.python** in Spark interpreter settings to path to your "**which python3**"

zeppelin.pyspark.python	/usr/bin/python3

And finally, create new note with spark interpreter and paste this in a new paragraph and run:

```
%pyspark
import sys
print(sys.version)
```

If everything is fine, you will see the python version:

```
%pyspark
import sys
print(sys.version)

3.6.5 (default, Apr  1 2018, 05:46:30)
[GCC 7.3.0]
```

Clustering

Setup your InterSystems IRIS to let it work with Zeppelin and Spark. E.g. see my previous article about InterSystems IRIS, Apache Zeppelin, and Apache Spark connection.

First, load the IRIS dataset:

```
%pyspark
dataFrame=spark.read.format("com.intersystems.spark").\
option("url", "IRIS://localhost:51773/NEWSAMPLE").option("user", "dev").\
option("password", "123").\
option("dbtable", "DataMining.IrisDataset").load()

dataFrame.show()
```

```
+---+-----------+----------+-----------+----------+-----------+
| ID|PetalLength|PetalWidth|SepalLength|SepalWidth|    Species|
+---+-----------+----------+-----------+----------+-----------+
|  1|        1.4|       0.2|        5.1|       3.5|Iris-setosa|
|  2|        1.4|       0.2|        4.9|       3.0|Iris-setosa|
|  3|        1.3|       0.2|        4.7|       3.2|Iris-setosa|
|  4|        1.5|       0.2|        4.6|       3.1|Iris-setosa|
|  5|        1.4|       0.2|        5.0|       3.6|Iris-setosa|
|  6|        1.7|       0.4|        5.4|       3.9|Iris-setosa|
|  7|        1.4|       0.3|        4.6|       3.4|Iris-setosa|
|  8|        1.5|       0.2|        5.0|       3.4|Iris-setosa|
|  9|        1.4|       0.2|        4.4|       2.9|Iris-setosa|
| 10|        1.5|       0.1|        4.9|       3.1|Iris-setosa|
| 11|        1.5|       0.2|        5.4|       3.7|Iris-setosa|
| 12|        1.6|       0.2|        4.8|       3.4|Iris-setosa|
| 13|        1.4|       0.1|        4.8|       3.0|Iris-setosa|
| 14|        1.1|       0.1|        4.3|       3.0|Iris-setosa|
| 15|        1.2|       0.2|        5.8|       4.0|Iris-setosa|
```

If you want to see what Species does it have, paste this in a new paragraph and run:

```
%pyspark
dataFrame.select("Species").show(150)
```

Before we will continue, it will be a good idea to consider what data do we have. Add a new paragraph and paste this and run:

```
%pyspark

z.show(dataFrame)
```

As you can see below, the table has 5 features: **PetalLength**, **PetalWidth**, **SepalLength**, **SepalWidth**, and **Species**.

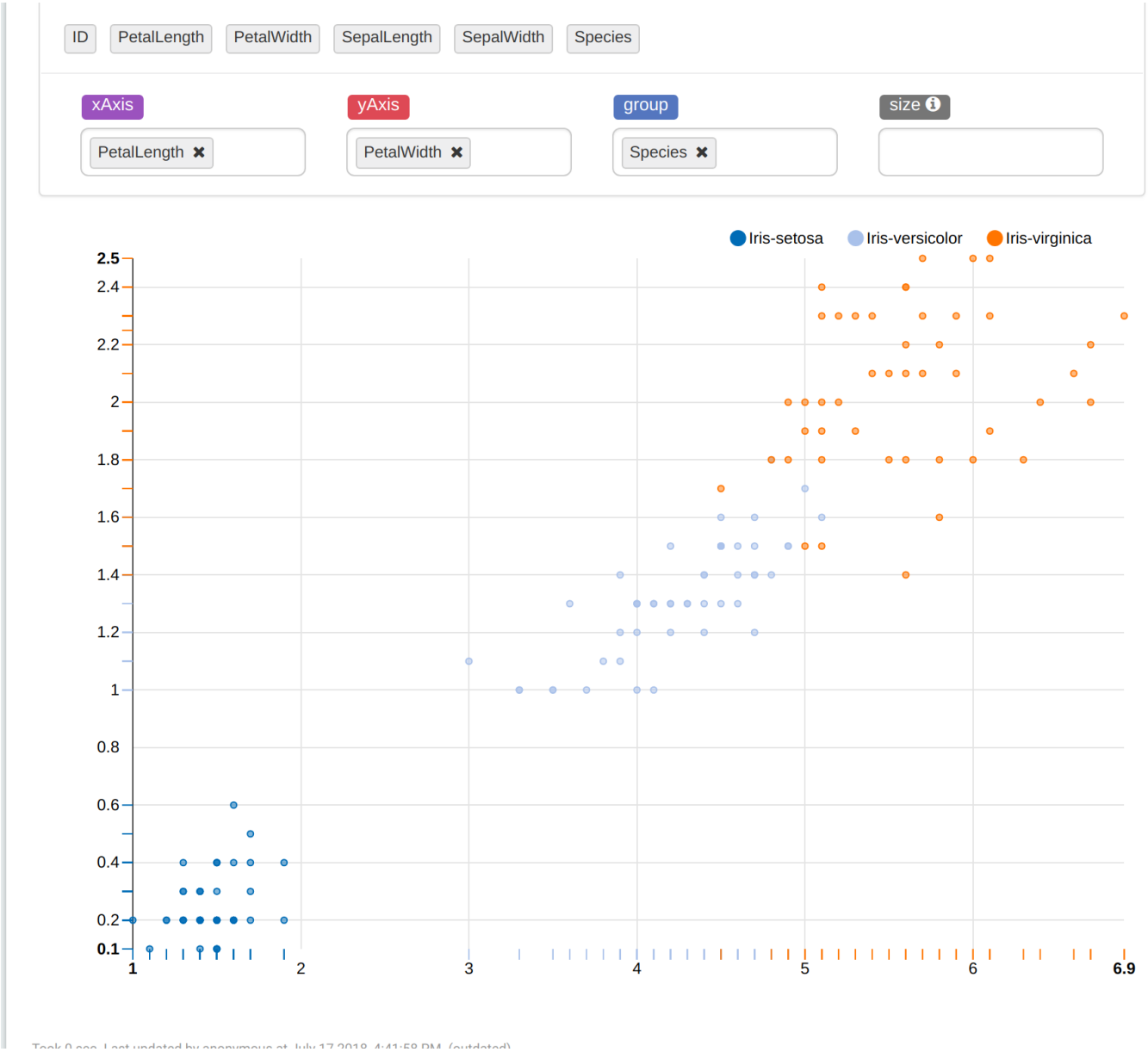
```
%pyspark
z.show(dataFrame)
```

FINISHED

settings

ID	PetalLength	PetalWidth	SepalLength	SepalWidth	Species
1	1.4	0.2	5.1	3.5	Iris-setosa
2	1.4	0.2	4.9	3	Iris-setosa
3	1.3	0.2	4.7	3.2	Iris-setosa
4	1.5	0.2	4.6	3.1	Iris-setosa
5	1.4	0.2	5	3.6	Iris-setosa
6	1.7	0.4	5.4	3.9	Iris-setosa
7	1.4	0.3	4.6	3.4	Iris-setosa
8	1.5	0.2	5	3.4	Iris-setosa

Choose **Scatter Chart**. Here we can see clusters in the image below (we cannot see a completely image because of 4d vectors). If you want to look at clusters from an another perspective, change the parameters of **xAxis** and **yAxis**.



So, our purpose is to predict the species of flowers using features. Add a new paragraph and paste the following in there and run:

```
%pyspark
```

```
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler

assembler = VectorAssembler(inputCols = ["PetalLength", "PetalWidth", "SepalLength", "SepalWidth"],
outputCol="features") // it makes a vector with 4 parameters mentioned in inputCols and name it as
outputCol.

irisFeatures = assembler.transform(dataFrame) // this will add to the table outputCol column with vectors.
irisFeatures.show(5)
```

Next, paste this in a new paragraph and run:

```
%pyspark
from pyspark.ml.clustering import KMeans

(trainingData, testData) = irisFeatures.randomSplit([0.7, 0.3]) // split data into two parts randomly
kmeans = KMeans().setK(3).setSeed(101010) // KMeans model with 3 clusters. setSeed
makes reproducible results.
model = kmeans.fit(trainingData) // train kmeans model

transformed = model.transform(trainingData) // add a new column to the table with predicted results
transformed.show(150)
```

```
%pyspark
from pyspark.ml.clustering import KMeans

(trainingData, testData) = irisFeatures.randomSplit([0.7, 0.3])
kmeans = KMeans().setK(3).setSeed(101010)
model = kmeans.fit(trainingData)

transformed = model.transform(trainingData)
transformed.show(150)
```

```
+---+-----------+----------+-----------+----------+--------------+-----------------+----------+
| ID|PetalLength|PetalWidth|SepalLength|SepalWidth|       Species|         features|prediction|
+---+-----------+----------+-----------+----------+--------------+-----------------+----------+
|  3|        1.3|       0.2|        4.7|       3.2|   Iris-setosa|[1.3,0.2,4.7,3.2]|         0|
|  4|        1.5|       0.2|        4.6|       3.1|   Iris-setosa|[1.5,0.2,4.6,3.1]|         0|
|  5|        1.4|       0.2|        5.0|       3.6|   Iris-setosa|[1.4,0.2,5.0,3.6]|         0|
|  7|        1.4|       0.3|        4.6|       3.4|   Iris-setosa|[1.4,0.3,4.6,3.4]|         0|
|  9|        1.4|       0.2|        4.4|       2.9|   Iris-setosa|[1.4,0.2,4.4,2.9]|         0|
| 10|        1.5|       0.1|        4.9|       3.1|   Iris-setosa|[1.5,0.1,4.9,3.1]|         0|
| 11|        1.5|       0.2|        5.4|       3.7|   Iris-setosa|[1.5,0.2,5.4,3.7]|         0|
| 12|        1.6|       0.2|        4.8|       3.4|   Iris-setosa|[1.6,0.2,4.8,3.4]|         0|
| 15|        1.2|       0.2|        5.8|       4.0|   Iris-setosa|[1.2,0.2,5.8,4.0]|         0|
| 16|        1.5|       0.4|        5.7|       4.4|   Iris-setosa|[1.5,0.4,5.7,4.4]|         0|
| 17|        1.3|       0.4|        5.4|       3.9|   Iris-setosa|[1.3,0.4,5.4,3.9]|         0|
| 18|        1.4|       0.3|        5.1|       3.5|   Iris-setosa|[1.4,0.3,5.1,3.5]|         0|
| 19|        1.7|       0.3|        5.7|       3.8|   Iris-setosa|[1.7,0.3,5.7,3.8]|         0|
| 22|        1.5|       0.4|        5.1|       3.7|   Iris-setosa|[1.5,0.4,5.1,3.7]|         0|
| 24|        1.7|       0.5|        5.1|       3.3|   Iris-setosa|[1.7,0.5,5.1,3.3]|         0|
```

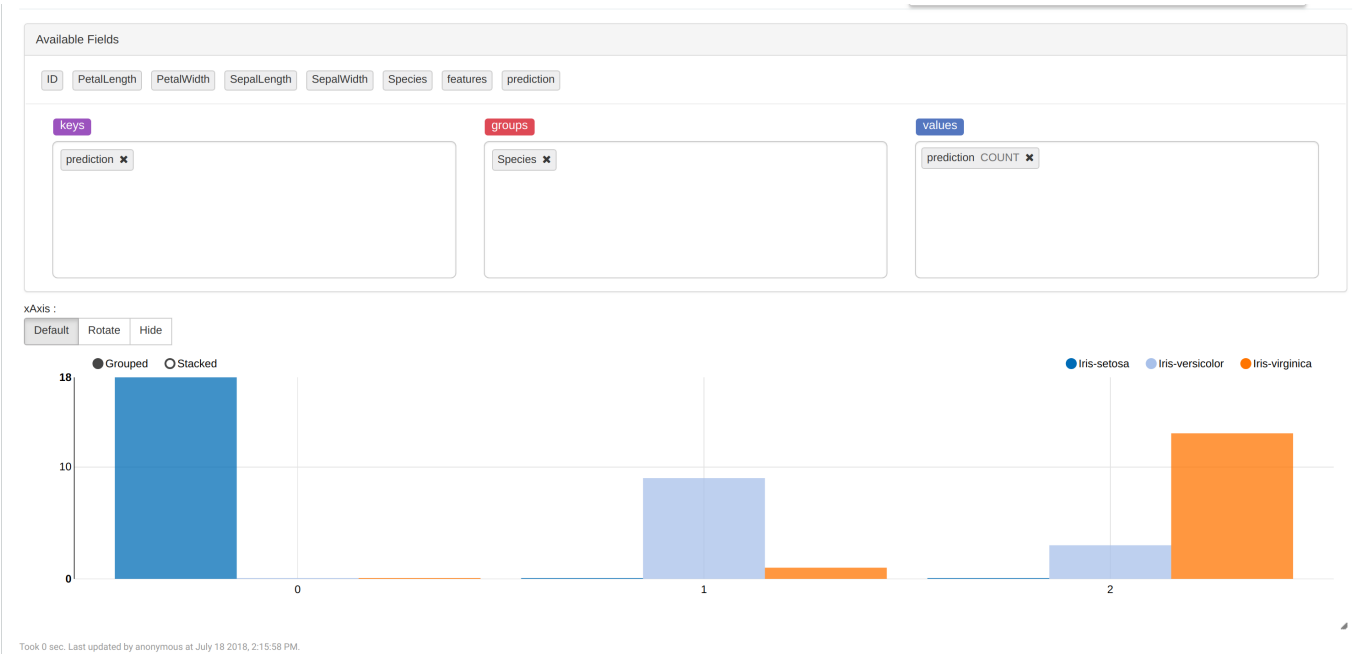
Use model on our testData:

```
%pyspark

predictions = model.transform(testData)
predictions.show(151)
```

And count the accuracy of model:

```
%pyspark

SpeciesAndPreds = predictions.select("Species", "prediction").collect()

def getCluster(specie):
    if specie == "Iris-setosa":
        return 0
    elif specie == "Iris-versicolor":
        return 1
    else:
        return 2

def getAccuracy(flowers):
    counter = 0;
    for flower in flowers:
        if getCluster(flower[0]) == flower[1]:
            counter += 1
    return counter / len(flowers)

accuracy = getAccuracy(SpeciesAndPreds)
print("accuracy is " + str(accuracy))

// My result is 0.9090909090909091
```

To see how many flowers are in each cluster, use "**z.show(predictions)**" and choose **Bar Chart**:



Conclusion

We made the model that predicts species of iris flowers in InterSystems IRIS pretty accurately (accuracy > 0.9). Also we noticed that the "Iris-setosa" is separable and "Iris-virginica" and "Iris-versicolor" are not separable on K-Means algorithm. So, it would be a good idea to use something else to increase accuracy.

Links

The way to launch Apache Spark + Apache Zeppelin + InterSystems IRIS

Spark SQL, DataFrames and Datasets Guide

Machine Learning library guide

K-means

Clustering (but this API for RDDs, so just look at the relevant information about algorithms)

Iris flower data set

#AI #API #Beginner #Machine Learning #Python #InterSystems IRIS