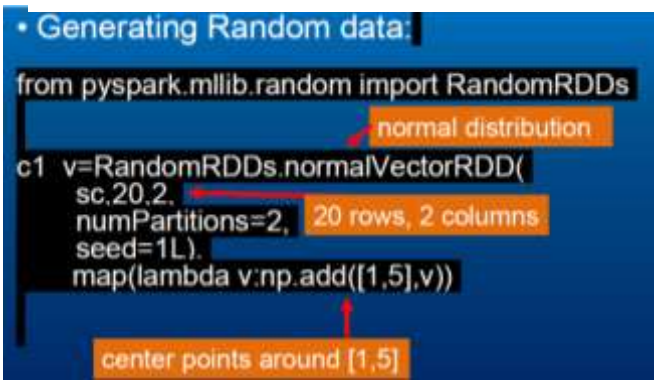
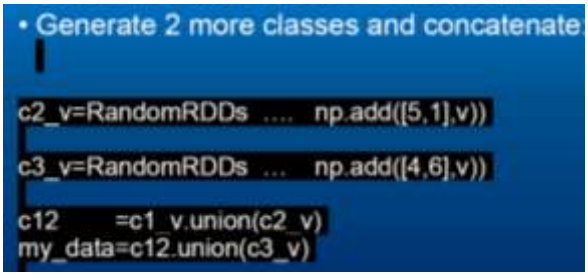
Clustering & Classification in SPARK-Due Date 1398-3-7

Clustering part1: Run the K-means clustering in Spark, and make some observations to answer the following questions:

Create the dokmeans.py file. This script should create some normally distributed random 2D data points, centered around 3 different coordinates. Use the provided slides (SparkMLlibClustering_Slides) as sample to create random data (my_data), then use my_data for clustering using K-means and prints out the sum-squared-error (SSE). Then answer the following questions:





dokmeans.py

(note that the random data is generated with **seed values set** so it should be the same across runs)

1. Execute the kmeans script as follows:

```
>>> exec(open('dokmeans.py').read())
```

The script is set up to run Kmeans for k=1 clusters. **Compute the SSE.**

(Note: In fact if you look at the cluster centers, using my_kmmodel.clusterCenters, you get almost the same values used to create the data. You can use the cluster center coordinate points as a kind of summary of the data. In some cases the cluster centers could be labeled to serve as descriptions of the data.)

2. Change the number of clusters to k=4 and rerun the training command and get the SSE (or rerun the whole script).

Compute the approximate SSE for k=4.

3. Try getting summary statistics on the RDD of random data.

```
>>> my_data.stats()
```

Take note of standard deviation values. If you square each value, you get the variance for each dimension. Now compare these stats results to Kmeans SSE when k=3.

Observe the relationship between SSE when k=3 and the variance of each dimension.

Clustering Questions to be answered for part 1 and included in your final report:

1. After running the Kmeans pyspark script, with the number of clusters k=3: What is the SSE?

2. Change the number of clusters to k=4 and re-run the training command and get the SSE (or rerun the whole script).What is the approximate SSE for k=4?

(note that the random data is generated with seed values set, so it should be the same across runs)

3. What is the relationship between SSE when k=1 and the variance of each dimension?

Clustering Part2: Use the weather data file:///home/cloudera/Downloads/big-data-4/minute_weather.csv'.

Perform cluster analysis on the minute-weather.csv dataset using the k-means algorithm. This dataset contains weather measurements such as temperature, relative humidity, etc., from a weather station in San Diego, California, collected at one-minute intervals. The goal of cluster analysis on this data is to identify different weather patterns for this weather station. Perform the following steps:

Step1: Download the **minute_weather.csv and make your data frame for later analysis.**

Step 2: Subset and filter the data in the data farme. There are over 1.5 million rows in the DataFrame (you can count them using your_df.count()). Clustering this data on your computer in the Cloudera VM can take a long time, so use only one-tenth of the data. We can subset by calling filter() and using the rowID column:

filteredDF =your_ df.filter((df.rowID%10) == 0),

check filteredDF.count()

Step3: compute the following summary statistics and drop zero and missing values:

summary	count	mean	stddev	min	max

The weather measurements in this dataset were collected during a drought in San Diego. count the how many values of rain accumulation and duration are 0 (Hint: rain_accumulation==0 and rain_duration==0). Report each of them in your final report for clustering part2.

Since most the values for these columns are 0, let's drop them from the DataFrame to speed up our analyses. We can also drop the hpwren_timestamp column since we do not use it. In addition, drop rows with missing values and count how many rows were dropped. Report this count in your final report for clustering part2.

Report the statistics after these pre-processing in your final report.

Step 4. Scale the data. Since the features are on different scales (e.g., air pressure values are in the 900's, while relative humidifies range from 0 to 100), they need to be scaled. Scale them so that each feature will have a value of 0 for the mean, and a value of 1 for the standard deviation.

Step 5. Create elbow plot. The k-means algorithm requires that the value of k, the number of clusters, to be specified. To determine a good value for k, we will use **the "elbow" method**. This method involves applying k-means, using different values for k, and calculating the within-cluster sum-of-squared error (WSSE). Since this means applying k-means multiple times, this process can be very compute-intensive. To speed up the process, use only a subset of the dataset like every third sample from the dataset to create this subset. You can use something like :

elbowset = scaledData.filter((scaledData.rowID % 3)==0

elbowset.persist() : (or cache) this method tells Spark to keep the data in memory (if possible), which will speed up the computations.

Then compute k-means clusters for k = 2 to 30 to create an elbow plot, (Hint : Train for cluster size 2 to 30 and compute WSSE for each) .Plot the results to find the elbow. The values for k are plotted against the WSSE values, and the elbow, or bend in the curve, provides a good estimate for **the value for k**. Report k in your final report for clustering part2.

Step 6. Cluster using the selected k and Report the computed centers for k clusters in your final report for clustering part 2.

What to submit for Clustering Part2 assignment: **All the highlighted red text above**.

Part3: Classification using Naïve Bayes and Decision tree.

Using the script in doweathclass.py do classification using Naive Bayes and Decision Trees. Use sample codes explained in SparkMLlibClassification_slides or the provided python files to create the classifiers. Then answer the following questions:

1. Execute NaiveBayes script in Spark and identify: What is the approximate percent correct?

2. After running Naive Bayes, look at the confusion matrix that is printed out. In the confusion matrix the rows are the true labels (0 or 1) and columns are the predicted labels (0 or 1). If 0 ='False' and 1='True' what is the number of False Positives?

3. After you execute the decision tree classification script, look at the confusion matrix that is output. What is the number of False Negative Predictions?

4. The decision tree function returns a decision tree object. How are the number of nodes counted?

5. After adding a useless variable: What is the percent correct in the new models?

6. After adding a new test point in the programming assignment instructions: Which variables are used to get the prediction node for this test point?

Part 4: Classification using Naïve Bayes and Decision tree on real data set.

Now using the provided daily_weather.csv data set create Naïve Bayes and decision tree classifiers to predict days with low humidity. The class label is 0 when the humidity value is less than 25%, otherwise it should be 1. (You need to first compute the class label for the provided data)

Make sure to:

- Remove rows with missing values
- Split the data as 70 % for train and 30 % for test
- Report confusion matrix for both classifiers.