****

Department of Electric Power and Energy Systems

**EH2745 – Computer Applications in Power Systems**

**Assignment II**

**Power System State Labeling & Results**

Georgios Papadopoulos

931001-1995

Francisco de Lima

920427-6191

Stockholm, 2018-06-08

# Labeling

## High Load

In order to assess if a given power system state can be classified as *“High Load”*, it is precise to verify the voltage magnitudes at buses 5, 7 and 9 (the buses with connected loads). If the sum of all these three voltages if beneath a certain threshold (in our case we selected 2.93 p.u.) then we can establish that the present scenario is one of high load.

## Shut Down

To determine if any of the generators is shut down, we need to verify the injected power into the buses where the generators are located. A generator will be labeled as *“Shut-Down”* if the power flow through the line where it is connected is very small (beneath 0.02 rad).

If one of these power flows is too low, it means that the generator is offline.

## Low Load

In order to assess if a given power system state can be classified as *“Low Load”*, it is precise to verify the voltage magnitudes in buses 6, 7 and 8. Whenever a low load condition takes place in bus 7, it affects the most nearby buses (6 and 8) in such a way that all these buses exhibit voltage magnitudes above 1.01 p.u. We can establish a criteria where, if the sum of the all these three voltages is above a certain threshold (in our case we selected 3.02 p.u.) then we can establish that the present scenario is one of low load.

## Disconnect

A straight-forward way to determine if a line disconnection has taken place in the power system is to use the voltage magnitudes of the buses. Consider that, if a failure (or maintenance) occurs, and the line has to be tripped, the current will have to find alternate ways to reach the loads, which means that voltage drops will heavily increase in the remaining lines, which in turn means that one or more buses will have very low voltages (we consider this to be beneath 0.85 p.u.)

# Results

Figure 1 shows the execution results of the program.

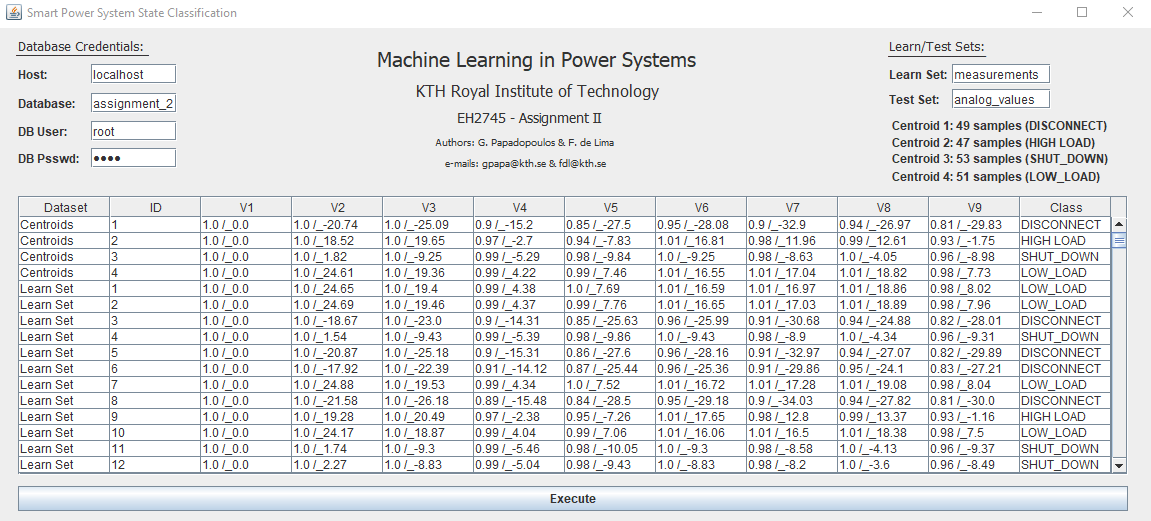


Figure 1 Execution Results

The solution obtained from the program shows the following results.

Table 1 Learn Set Samples Classification

|  |  |
| --- | --- |
| **Class** | **Number of Samples (out of 200)** |
| *High Load* | 47 |
| *Shut Down* | 53 |
| *Low Load* | 51 |
| *Disconnect* | 49 |

The performed *kNN* classification algorithm shows the following results.

The query: SELECT \* FROM assignment\_2.test\_set; was executed so as to acquire the test set classification results.

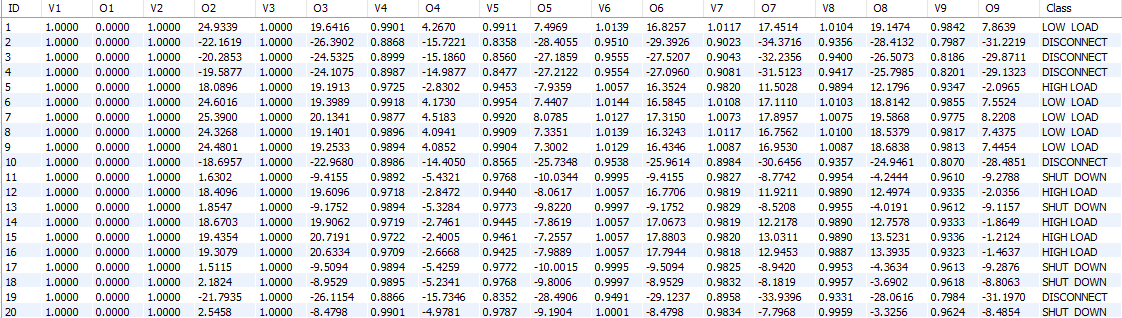


Figure 2 Execution Results

It has been found that there are 5 samples of each class among the test set objects.

Table 2 Learn Set Samples Classification

|  |  |  |
| --- | --- | --- |
| **Class** | **Number of Samples (out of 20)** | **Executed Query** |
| *High Load* | 5 | SELECT COUNT(\*) FROM assignment\_2.test\_set WHERE Class='HIGH LOAD'; |
| *Shut Down* | 5 | SELECT COUNT(\*) FROM assignment\_2.test\_set WHERE Class='SHUT\_DOWN'; |
| *Low Load* | 5 | SELECT COUNT(\*) FROM assignment\_2.test\_set WHERE Class='LOW\_LOAD'; |
| *Disconnect* | 5 | SELECT COUNT(\*) FROM assignment\_2.test\_set WHERE Class='DISCONNECT'; |

# Appendix

The next algorithm shows the *KLabel* class implemented in the *Java* program to label the different power system states.

**package** assignment2;

**import** java.util.ArrayList;

**public** **class** KLabel {

//\*\*\* LABEL CENTROIDS \*\*\*

**public** **static** **void** LabelCentroids(ArrayList<Sample> centroids) {

**for** (Sample centroid : centroids) {

*isHighLoad*(centroid);

*isShutDown*(centroid);

*isLowLoad*(centroid);

*isDisconnect*(centroid);

centroids.set(centroid.cluster, centroid);

}

}

//\*\*\* DETERMINE IF CENTROID IS HIGH LOAD \*\*\*

**private** **static** **void** isHighLoad(Sample centroid) {

**int** Nbus = centroid.attribute.length/2;

**double**[] v = **new** **double**[Nbus];

**for** (**int** n=0; n < Nbus; n++) {

v[n] = centroid.attribute[2\*n];

}

**if** (v[4] + v[6] + v[8] < 2.93) {

//buses 5, 7 and 9 have low voltages

centroid.state = Sample.***HIGH\_LOAD***;

}

}

//\*\*\* DETERMINE IF CENTROID IS SHUT DOWN \*\*\*

**private** **static** **void** isShutDown(Sample centroid) {

**int** Nbus = centroid.attribute.length/2;

**double**[] o = **new** **double**[Nbus];

**for** (**int** n=0; n < Nbus; n++) {

o[n] = centroid.attribute[2\*n+1]\*Math.***PI***/180;

}

**double** pmin = 0.02; //minimum power to assume generator is online

**double** p14 = Math.*abs*(o[0]-o[3]); //flow through line 1-4

**double** p28 = Math.*abs*(o[1]-o[7]); //flow through line 2-8

**double** p36 = Math.*abs*(o[2]-o[5]); //flow through line 3-6

**if** (p14 < pmin || p28 < pmin || p36 < pmin) {

//if one of these flows is too low, it means that the generator is offline

centroid.state = Sample.***SHUT\_DOWN***;

}

}

//\*\*\* DETERMINE IF CENTROID IS LOW LOAD \*\*\*

**private** **static** **void** isLowLoad(Sample centroid) {

**int** Nbus = centroid.attribute.length/2;

**double**[] v = **new** **double**[Nbus];

**for** (**int** n=0; n < Nbus; n++) {

v[n] = centroid.attribute[2\*n];

}

**if** (v[5] + v[6] + v[7] > 3.02) {

//buses 6, 7 and 8 have high voltages

centroid.state = Sample.***LOW\_LOAD***;

}

}

//\*\*\* DETERMINE IF CENTROID IS DISCONNECT \*\*\*

**private** **static** **void** isDisconnect(Sample centroid) {

**int** Nbus = centroid.attribute.length/2;

**double**[] v = **new** **double**[Nbus];

**for** (**int** n=0; n < Nbus; n++) {

v[n] = centroid.attribute[2\*n];

}

**for** (**int** i=0; i < Nbus; i++) {

**if** (v[i] < 0.85) {

//if one of the voltages is too low, it is because of line disconnection

centroid.state = Sample.***DISCONNECT***;

**break**;

}

}

}

//\*\*\* LABEL SAMPLES BASED ON CENTROIDS \*\*\*

**public** **static** **void** LabelSamples(ArrayList<Sample> centroids, ArrayList<Sample> samples) {

**for** (**int** m=0; m < samples.size(); m++) {

Sample sample = samples.get(m);

**for** (Sample centroid : centroids) {

**if** (sample.cluster == centroid.cluster) {

sample.state = centroid.state;

}

}

samples.set(m, sample);

}

}

}