# Assignment 2 : Using *k-means* to Detect Network Intrusion

**Operation system:** Mac OS (local) Hadoop(beacon)
**Platform:** Python 2.7.10(local) Hadoop2.5 and spark 1.5.2(beacon)

## Abstract

In this assignment , KDD cup data of 1999 are used as input data. *k-means*, a widely used clustering algorithm, aims to partition data into *k* clusters in which each datum belongs to the cluster with the nearest distance to the center of the cluster. Also, *k-means* is used to detect anomalous network connection in the KDD cup data of 1999. The steps to cluster the data are shown as follows.

1.Parsing KDD cup data and first pass at clustering with *k=2*.
2.Choosing *k* for the parsed KDD cup data.
3.Normalizing the feature space.
4.Improving the clustering with labels.
5.Interpreting the clusters.

## Our first pass at clustering

Before our first pass at clustering, *textFile* in *SparkContext* is used to load the CSV data as an RDD of String. Later, we split data into tuples which include the label and count. Get the total number of each label by using *countByValue* function and sort the tuple data by number in descending order.

```
labels =raw_data.map(lambda line: line.strip().split(",")[-1])
   label_counts =labels.countByValue()
   sorted_labels =OrderedDict(sorted(label_counts.items(), key=lambda t: t[1],
                                              reverse=True))
   print("label and count")
   for label, count in sorted_labels.items():
      print (label, count)
```

The results of label and count are shown as follows. There are 23 labels and label ”smurf” has the highest occurrence frequency.

```
smurf.      2807886
neptune.      1072017
normal.      972781
satan.    15892
ipsweep.      12481
portsweep.      10413
```

```
nmp.      2316
back.      2203
warezclient.      1020
pod.      264
guess_passwd.      53
buffer_overflow.      30
land.    21
warezmaster.      20
imap.      12
rootkit.      10
loadmodule.    9
ftp_write.    8
multihop.      7
phf.      4
perl.    3
spy.      2
```

Because there are nonnumeric features in the RDD data, we should remove the nonnumeric features (e.g. second column, third column, fourth column and the last column) to create numeric features for *k-means* to work correctly. A parsing function is defined as *parse_interaction()* which split every line by comma and later only select the numeric columns to form a clean list. This function returns a tuple in which the last column is set as key ,and the numeric columns is converted into a array as the value.

```
def parse_interaction(line):
   line_split =line.split(",")
   clean_line_split =[line_split[0]]+line_split[4:-1]
   return (line_split[-1], array([float(x) for x in clean_line_split]))
```

The first tuple is shown as follows.

```
(u'normal.', array
([ 0.00000000e+00, 2.15000000e+02, 4.50760000e+04, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 1.00000000e+00,
1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00]))
```

After getting the data, the *Kmeans.train* function is called to get the kmeans model. The values in the parsed data are used to train the model and the k is set as 2 which means the data will be clustered into two classes. Moreover, *model.clusterCenters* function in *Kmeans* is convenient to get a list of the cluster centers, and *model.predict* function is called to predict which cluster each array in the parsed data belongs to. Using the *model.predict* function, we return a tuple result which includes cluster and label. As before, we get the total number of each label in every cluster by using the *countByValue* function. Later, we use *sorted* function to sort the results by setting the cluster number as the first keyword to sort the result. If the results have the same cluster number, then the alphabets of the label is the second keyword to sort.

```
model =KMeans.train(parsed_data_values, k=2 )
  print("Cluster centers: " +str(model.clusterCenters))
  cluster_label=parsed_data.map(lambda(label,datum): (model.predict(datum),label))
  cluster_label_count=cluster_label.countByValue()
  cluster_label_count_sorted=OrderedDict(sorted(cluster_label_count.items(),
                                             key=(lambda t: (t[0][0], t[0][1]))))
  print("Cluster label count")
  for (a,b),cc in cluster_label_count_sorted.items():
     print(a,str(b),cc)
```

The two cluster centers are printed as follows.

```
[array([ 4.83401949e+01, 1.83462155e+03, 8.26203190e+02, 5.71611720e-06,
6.48779303e-04, 7.96173468e-06, 1.24376586e-02, 3.20510858e-05, 1.43529049e-01,
8.08830584e-03, 6.81851124e-05, 3.67464677e-05, 1.29349608e-02, 1.18874823e-03,
7.43095237e-05, 1.02114351e-03, 0.00000000e+00, 4.08294086e-07, 8.35165553e-04,
3.34973508e+02, 2.95267146e+02, 1.77970317e-01, 1.78036989e-01, 5.76648988e-02,
5.77299094e-02, 7.89884132e-01, 2.11796106e-02, 2.82608101e-02, 2.32981078e+02,
1.89214283e+02, 7.53713390e-01, 3.07109788e-02, 6.05051931e-01, 6.46410789e-03,
1.78091184e-01, 1.77885898e-01, 5.79276115e-02, 5.76592214e-02]), array([
1.09990000e+04, 0.00000000e+00, 1.30993740e+09, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 1.00000000e+00,
1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 1.00000000e+00,
1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 2.55000000e+02, 1.00000000e+00,
0.00000000e+00, 6.50000000e-01, 1.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 1.00000000e+00, 1.00000000e+00])]
```

The printed cluster, label and count are shown as follows.

```
0     back.      2230
```

```
0    buffer_overflow.     30
0    ftp_write.   8
0    guess_passwd.   53
0    imap.   12
0    jpsweep.   12481
0    land.   21
0    loadmodule.   9
0    multihop.   7
0    neptune.   1072017
0    nmap.   2316
0    normal.   972781
0    perl.   3
0    phf.   4
0    pod.   264
0    portsweep.   10412
0    rootkit.   10
0    satan.   15891
0    smurf.   2807886
0    spy.   2
0    teardrop.   979
0    warezclient.   1020
0    warezmaster.   20
1    portsweep.   1
```

**Analysis:** This is not a good way to model the data. According to the results printed, there is only one data point in cluster 1, and all other data are in cluster 0. Only the cluster 0 works and the cluster 1 does not work. Therefore, this model is not reasonable. I think we should at least 23 clusters and may be more to have a good clustering because there are 23 labels in the data,. It is better to iterate more times at a given *k* or do something to select better initializations.

## Choosing *k*

In this part, Euclidean distance between vectors and their cluster center is used as metric to evaluate the *k* and choose the proper k for the KDD cup data of 1999.

A function named *dist_to_centroid* is defined to calculate the Euclidean distance between a data point and its centroid. The parameters of this function are *datum* (vectors in data)and *model*(clustering model we built before). At first, *model.predict* function is called to get the which cluster this datum belongs to. Later, *model.clusterCenters* function to get a list of cluster centers. According to the cluster number , we can get the corresponding cluster center from the list. Because the *datum* and *centroid* are arrays, and we could get the distance between them by using the formula of Euclidean distance.

```
def dist_to_centroid(datum, model):
   cluster =model.predict(datum)
   centroid =model.clusterCenters[cluster]
   return sqrt(sum([x**2 for x in (centroid -datum)]))
```

We also define a*clustering* function to return the mean distances between vectors and their cluster centroid for each *k*. In this *KMeans.train* function, we set the *maxIterations* as 10, parallel *runs* as 5 and the *initializationMode* as random. The result is a tuple which includes *k*, *model* and mean distance.

```
def clustering(data, k):
   model =KMeans.train(data, k, maxIterations=10, runs=5, initializationMode="random")
   result =(k, model, data.map(lambda datum: dist_to_centroid(datum, model)).mean())
   return result
```

After defining the functions, we use *map* function to evaluate for *k* =5 to 181 in increments of 5.

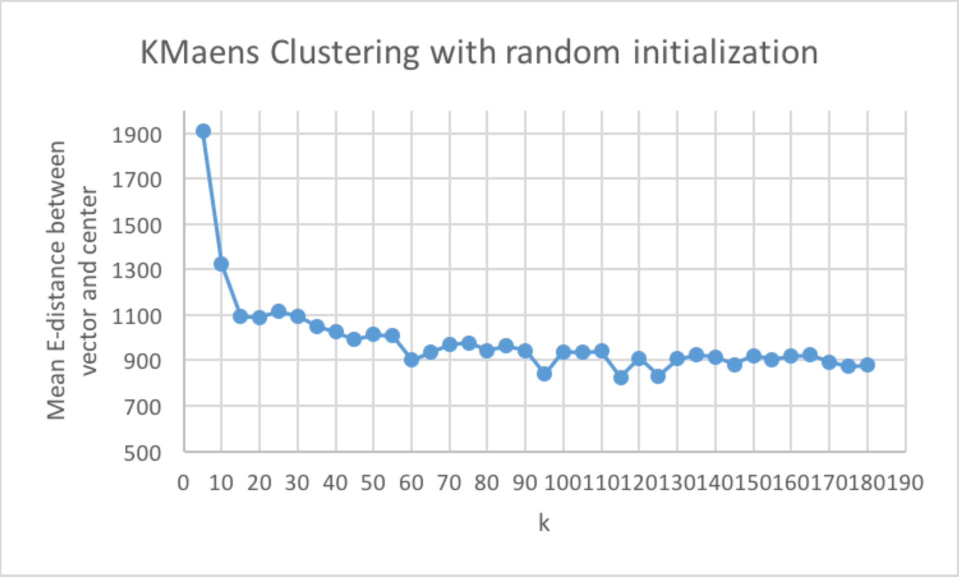
```
def clustering(data, k):
  scores =map(lambda k: clustering(parsed_data_values, k), range(5,181,5 ))
```

The *KMeans* Clustering results with random initialization are shown as follows.

```
05     1908.727735
10     1321.518299
15     1091.714547
20     1090.489979
25     1115.812595
30     1094.958445
35     1050.153143
40     1024.964408
45     990.871061
50     1012.281377
55     1007.698614
60     901.820666
65     935.499316
70     968.730234
75     977.053573
80     942.303398
85     963.364015
90     943.956004
```

```
95     839.057928
100    937.902555
105    934.219432
110    940.57706
115    824.372646
120    908.165968
125    831.134445
130    907.199544
135    924.589962
140    915.311954
145    882.231164
150    921.796610
155    904.014104
160    917.843094
165    923.612971
170    892.661797
175    871.437991
180    877.764924
```

Plot the results as a graph as follows.

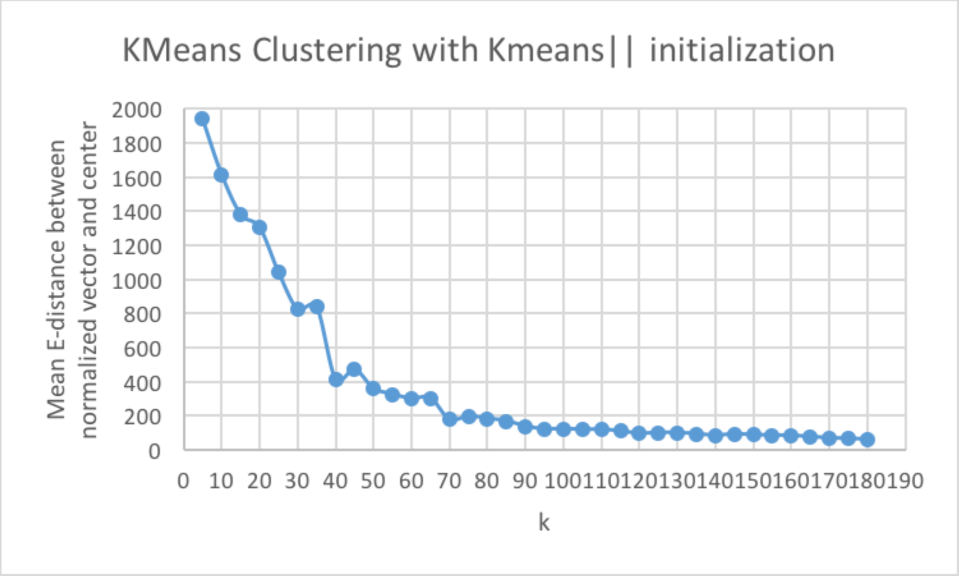


It can be seen from above figure that first three mean distances decrease obviously with the increase of *k*, while other results fluctuate and the downtrend is not obvious. One reason is because the random initialization and another is that sometimes the *kmeans* can only get the local optimum.

So I change the *initializationModel* parameter with *k-means*—— which will have a better initialization, and the results and figure are shown as follows.

```
05    1938.858342
10    1614.751129
15    1381.315621
20    1303.332945
25    1039.438383
30    822.809077
35    838.717617
40    413.258929
45    470.390464
50    358.46826
55    325.269504
60    298.705518
65    301.992488
70    176.440001
75    195.831212
80    182.050028
85    167.843253
90    137.236961
95    120.479014
100   121.813906
105   117.658784
110   117.738596
115   108.564773
120   94.903553
125   99.171808
130   98.253193
135   92.851235
140   82.487367
145   90.954528
150   86.36824
155   83.759404
160   81.694943
165   75.705291
170   67.579603
175   64.632222
180   68.214134
```

The figure is shown as follows.



From above, if we use *kmeas* —— as the initialization, the mean distance decreases with the increase of *k* for most time. It is better than random initialization. The decrease becomes flat start with *k*=100. The next several parts, *kmeas* —— initialization are also used.

## Normalizing our feature space

In this part, we will normalize the data by subtracting the mean of the data, and dividing it by the standard deviation. This normalizing process does not affect the Euclidean distance between data which are in the the original data set, because we do the some operation to each datum. But it truly affect the numeric results since we subtract the mean value.

Because there is normalizing function named *StandardScaler*, in which *fit* function is used to compute the mean and std to be used for later scaling and *transform* function is used to perform standardization by centering and scaling. We use these function to normalizing the data.

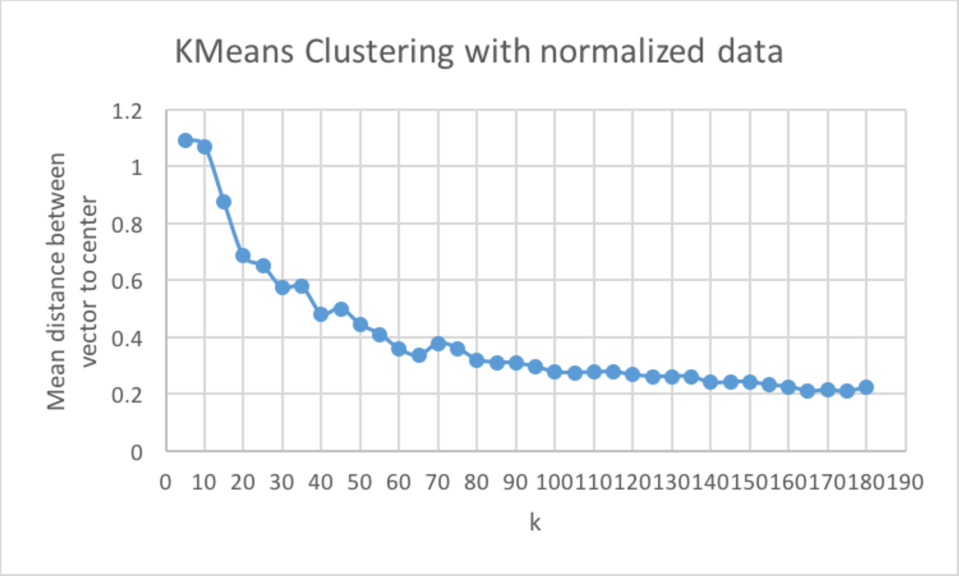
```
standardizer =StandardScaler(True, True)
standardizer_model =standardizer.fit(parsed_data.values())
standardized_data_values =standardizer_model.transform(parsed_data.values())
```

Using these normalized data to run the clustering programming again. Results are shown as follows.

```
05     1.090081
```

```
10    1.071025
15    0.87477
20    0.684846
25    0.6493
30    0.572975
35    0.577037
40    0.481622
45    0.500058
50    0.443797
55    0.409584
60    0.35814
65    0.334645
70    0.377353
75    0.360279
80    0.319062
85    0.311221
90    0.307578
95    0.295063
100   0.277036
105   0.274279
110   0.276436
115   0.27707
120   0.266617
125   0.260922
130   0.261289
135   0.260355
140   0.239378
145   0.241943
150   0.241932
155   0.230417
160   0.224942
165   0.209555
170   0.213833
175   0.208491
180   0.225541
```

The figure is plotted as follows.



## Improving the clustering with labels

In this part, entropy is used to evaluate the clustering. If there is only one label in each cluster, then the entropy index is zero which means this is a perfect clustering. Therefore, we would select the *k* with low entropy. Firstly, we define a function to compute the entropy according to its formula for later use.

```
def entropy(counts):
   n=sum(counts)
   value=[v/n for v in counts]
   result=sum([-x*log(x) for x in value])
   return result
```

Secondly, We define a function named *clustering_new* to include the requirements listed in the Assignment 2. The input parameters are *data* ( labels and arrays) and *k*. As used before, *KMeans.train* function is used to set the model and *model.predict* function is used to predict which cluster the datum belongs to. Each tuple in *cluster_label_count* records the cluster number, label and the frequency of appearance of label in this cluster. After that, *sorted* function is used to sort *cluster_label_count* first by cluster number, and if has the same cluster number, then by the alphabets of label in ascending order. Later, we transform the sorted results into RDD and group them according to the cluster number. Till now, we get all the labels and their frequency in each cluster. It is now to call *emtropy* compute the weighted entropy.

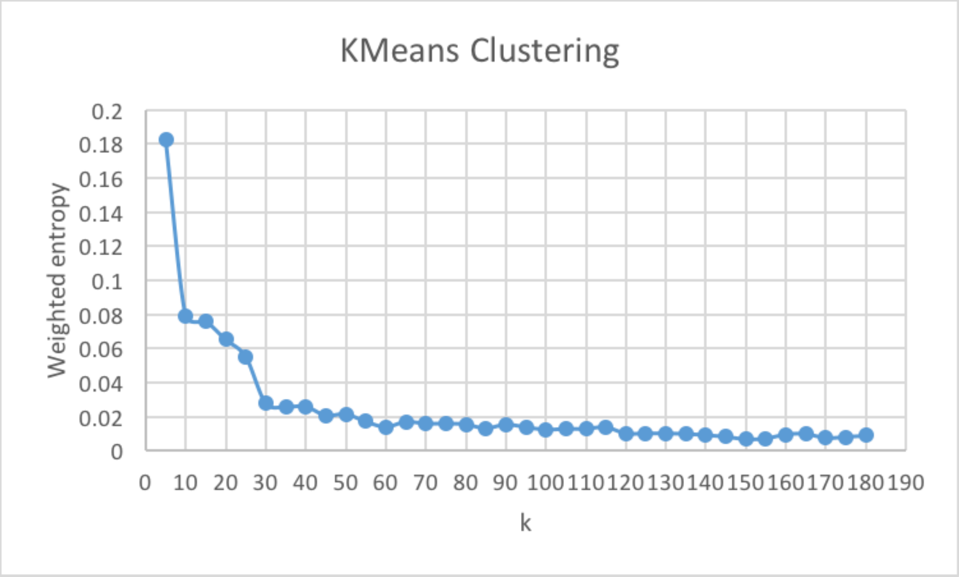
```
def clustering_new(data,k):
    clusters =KMeans.train(data.values(), k,maxIterations=10, runs=5,
                                                initializationMode="k-means||")
    cluster_label=data.map(lambda(label,datum): (clusters.predict(datum),label))
    cluster_label_count=cluster_label.countByValue()
    cluster_label_count_sorted=OrderedDict(sorted(cluster_label_count.items(),
                                                key=lambda t: (t[0][0],t[0][1])))
    res=list((v,k) for v,k in cluster_label_count_sorted.items())
    for (a,cc),ddd in cluster_label_count_sorted.items():
      print((a,str(cc)),ddd)
    clusterlabelcount=sc.parallelize(res)
    labelsInCluster=clusterlabelcount.groupBy(lambda x: x[0][0]).map(lambda
                                                x:(x[0],list((label,count) for
                                                (c,label),count in x[1]))).collect()
    print("Extract labels of each cluster:")
    print([labelsInCluster[x] for x in range(0,len(labelsInCluster)-1)])

     labelsAndCounts=clusterlabelcount.groupBy(lambda x: x[0][0]).map(lambda
                                                 x:(x[0],list(count for (tupe,count) in
                                                 x[1])))
    num=data.count()
    entroy_weight=labelsAndCounts.map(lambda x:sum(x[1])*entropy(x[1])).sum()/num
    print("Clustering score for k=%(k)d is %(score)f" % {"k": k, "score":
                                                entroy_weight})
    return entroy_weight
```

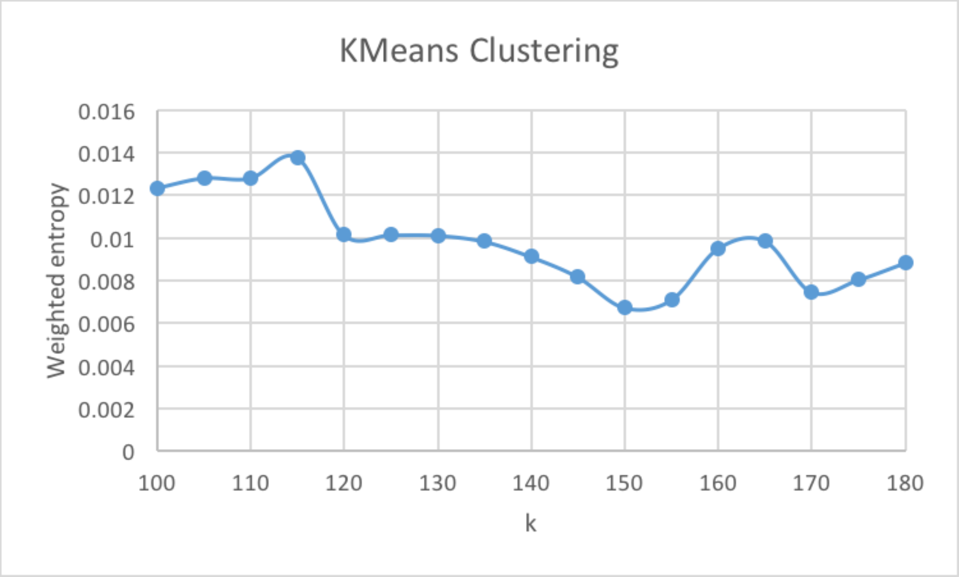
*Clustering_new* function is now used to evaluate *k* ranging from 5 to 180 with step 5. The weighted entropies are shown as follows.

```
05      0.182345
10      0.079161
15      0.075823
20      0.065175
25      0.055127
30      0.027954
35      0.025756
40      0.025903
45      0.020678
50      0.021451
55      0.01713
60      0.013905
65      0.016962
70      0.015912
75      0.015918
```

```
80      0.015473
85      0.013121
90      0.015357
95      0.013827
100     0.012346
105     0.01281
110     0.012808
115     0.013785
120     0.010146
125     0.010141
130     0.010117
135     0.009835
140     0.009117
145     0.008171
150     0.006732
155     0.007095
160     0.009504
165     0.009859
170     0.007463
175     0.008038
180     0.008833
```

The figure is shown as follows.





We can get that entropy does not always decrease with the increase of *k* from both data results and figures. In this case, we would like to choose *k* with the lowest entropy. If we focus on the rang from 100 to 180, we could find that *k*=150 has the lowest entropy. So we choose *k*=150 to cluster KDD cup data 1999.

## Interpreting our clusters

We have chosen *k*=150 in this case and set the model for clustering. The results are shown as follows.

```
((0, 'neptune.'), 103995)
((0, 'normal.'), 1
((0, 'portsweep.'), 65)
((1, 'normal.'), 18)
((1, 'smurf.'), 2624671)
((2, 'ipsweep.'), 7980)
((2, 'nmap.'), 3)
((2, 'normal.'), 48)
((2, 'warezclient.'), 2)
((3, 'normal.'), 700)
((3, 'portsweep.'), 234)
((4, 'neptune.'), 235070)
```

```
((4, ’normal.’), 1)
((5, ’buffer_overflow.’), 2)
((5, ’normal.’), 6159)
((5, ’rootkit.’), 1)
((5, ’warezclient.’), 352)
((6, ’normal.’), 4383)
((7, ’guess_passwd.’), 1)
((8, ’teardrop.’), 970)
((9, ’normal.’), 4)
((10, ’land.’), 21)
((10, ’normal.’), 7)
((11, ’ipsweep.’), 2)
((11, ’normal.’), 3697)
((11, ’portsweep.’), 1)
((12, ’back.’), 10)
((12, ’normal.’), 184208)
((13, ’portsweep.’), 2)
((14, ’back.’), 5)
((14, ’normal.’), 12635)
((15, ’normal.’), 24)
((15, ’spy.’), 1)
((16, ’normal.’), 1156)
((16, ’warezclient.’), 14)
((17, ’normal.’), 23)
((17, ’warezmaster.’), 1)
((18, ’satan.’), 11898)
((19, ’portsweep.’), 1)
((20, ’normal.’), 1)
((21, ’normal.’), 2)
((22, ’normal.’), 2859)
((22, ’portsweep.’), 108)
((23, ’normal.’), 1059)
((23, ’warezclient.’), 14)
((24, ’normal.’), 17)
((25, ’ftp_write.’), 1)
((25, ’normal.’), 3)
((25, ’rootkit.’), 1)
((26, ’nmap.’), 16)
((26, ’normal.’), 9709)
((26, ’portsweep.’), 4)
```

```
((26, ’rootkit.’), 2)
((26, ’satan.’), 391)
((26, ’smurf.’), 459)
((26, ’spy.’), 1)
((27, ’portsweep.’), 2)
((28, ’normal.’), 4798)
((28, ’satan.’), 162)
((29, ’ipsweep.’), 10)
((29, ’normal.’), 13731)
((30, ’normal.’), 348)
((31, ’neptune.’), 254336)
((31, ’normal.’), 1)
((32, ’normal.’), 1)
((33, ’buffer_overflow.’), 18)
((33, ’loadmodule.’), 1)
((33, ’normal.’), 216)
((33, ’rootkit.’), 1)
((34, ’imap.’), 5)
((34, ’neptune.’), 1243)
((34, ’normal.’), 46)
((35, ’normal.’), 1)
((36, ’normal.’), 6)
((36, ’satan.’), 1)
((37, ’normal.’), 12)
((37, ’portsweep.’), 2624)
((37, ’satan.’), 2)
((38, ’nmap.’), 16)
((38, ’normal.’), 6267)
((38, ’satan.’), 49)
((39, ’normal.’), 16241)
((40, ’buffer_overflow.’), 2)
((40, ’loadmodule.’), 1)
((40, ’normal.’), 362)
((41, ’normal.’), 1)
((42, ’normal.’), 6)
((43, ’neptune.’), 5)
((43, ’normal.’), 2444)
((44, ’normal.’), 3312)
((45, ’guess_passwd.’), 51)
((45, ’normal.’), 53)
```

```
((45, 'rootkit.'), 1)
((46, 'normal.'), 544)
((47, 'normal.'), 99805)
((48, 'imap.'), 2)
((48, 'neptune.'), 5)
((48, 'normal.'), 2406)
((48, 'satan.'), 1)
((49, 'normal.'), 290)
((49, 'warezclient.'), 274)
((50, 'normal.'), 1)
((51, 'normal.'), 5)
((52, 'back.'), 38)
((52, 'normal.'), 13025)
((53, 'normal.'), 91448)
((54, 'normal.'), 1)
((55, 'normal.'), 7)
((56, 'back.'), 3)
((56, 'normal.'), 1942)
((56, 'portsweep.'), 203)
((56, 'satan.'), 1)
((56, 'warezclient.'), 1)
((57, 'portsweep.'), 2)
((58, 'imap.'), 1)
((58, 'ipsweep.'), 69)
((58, 'loadmodule.'), 1)
((58, 'multihop.'), 2)
((58, 'nmap.'), 145)
((58, 'normal.'), 10349)
((58, 'pod.'), 5)
((58, 'portsweep.'), 2)
((58, 'rootkit.'), 2)
((58, 'satan.'), 6)
((58, 'smurf.'), 92)
((58, 'warezmaster.'), 18)
((59, 'back.'), 4)
((59, 'normal.'), 3482)
((59, 'portsweep.'), 2)
((59, 'satan.'), 128)
((59, 'warezclient.'), 1)
((60, 'ftp_write.'), 1)
```

```
((60, 'ipsweep.'), 756)
((60, 'neptune.'), 4)
((60, 'nmap.'), 18)
((60, 'normal.'), 493)
((60, 'warezclient.'), 144)
((61, 'ipsweep.'), 2524)
((61, 'neptune.'), 2)
((61, 'normal.'), 266)
((61, 'portsweep.'), 2)
((62, 'portsweep.'), 2)
((63, 'ipsweep.'), 1)
((63, 'nmap.'), 1)
((63, 'normal.'), 4648)
((63, 'satan.'), 41)
((63, 'warezclient.'), 3)
((64, 'normal.'), 4515)
((65, 'ipsweep.'), 1)
((65, 'normal.'), 51)
((66, 'normal.'), 13)
((67, 'normal.'), 5568)
((68, 'imap.'), 1)
((68, 'normal.'), 26455)
((68, 'rootkit.'), 1)
((68, 'satan.'), 2)
((68, 'warezclient.'), 36)
((69, 'normal.'), 199)
((69, 'smurf.'), 179559)
((70, 'loadmodule.'), 2)
((70, 'multihop.'), 1)
((70, 'normal.'), 1)
((71, 'normal.'), 181)
((72, 'back.'), 4)
((72, 'normal.'), 16022)
((73, 'normal.'), 6)
((74, 'normal.'), 4)
((75, 'normal.'), 2613)
((76, 'ipsweep.'), 28)
((76, 'neptune.'), 1)
((76, 'nmap.'), 984)
((76, 'normal.'), 152)
```

```
((76, ’warezclient.’), 4)
((77, ’normal.’), 5)
((78, ’neptune.’), 9284)
((78, ’normal.’), 1)
((78, ’portsweep.’), 214)
((79, ’back.’), 6)
((79, ’buffer_overflow.’), 3)
((79, ’normal.’), 26731)
((79, ’warezclient.’), 77)
((80, ’normal.’), 28)
((80, ’portsweep.’), 396)
((81, ’normal.’), 1)
((82, ’normal.’), 15)
((83, ’ipsweep.’), 679)
((83, ’normal.’), 9)
((83, ’portsweep.’), 1)
((84, ’normal.’), 11)
((85, ’normal.’), 18495)
((86, ’neptune.’), 11)
((86, ’nmap.’), 1018)
((86, ’normal.’), 4)
((86, ’portsweep.’), 9)
((87, ’normal.’), 37)
((88, ’ipsweep.’), 6)
((88, ’loadmodule.’), 2)
((88, ’normal.’), 419)
((88, ’satan.’), 1)
((89, ’nmap.’), 1)
((89, ’normal.’), 4662)
((90, ’normal.’), 4)
((91, ’normal.’), 1339)
((92, ’normal.’), 13)
((93, ’normal.’), 7)
((93, ’portsweep.’), 2)
((94, ’ipsweep.’), 3)
((94, ’neptune.’), 15)
((94, ’normal.’), 3136)
((94, ’satan.’), 4)
((95, ’normal.’), 685)
((95, ’portsweep.’), 1)
```

```
((95, ’satan.’), 1)
((96, ’multihop.’), 1)
((96, ’normal.’), 1)
((96, ’perl.’), 3)
((97, ’phf.’), 4)
((98, ’normal.’), 5828)
((99, ’normal.’), 2199)
((99, ’portsweep.’), 145)
((100, ’portsweep.’), 13)
((100, ’satan.’), 1783)
((101, ’normal.’), 24259)
((102, ’normal.’), 18656)
((102, ’warezclient.’), 7)
((103, ’imap.’), 1)
((103, ’nmap.’), 24)
((103, ’normal.’), 10834)
((103, ’portsweep.’), 3)
((103, ’satan.’), 39)
((103, ’smurf.’), 35)
((104, ’normal.’), 2574)
((104, ’portsweep.’), 5)
((104, ’satan.’), 258)
((105, ’normal.’), 18228)
((106, ’buffer_overflow.’), 4)
((106, ’loadmodule.’), 1)
((106, ’normal.’), 1694)
((106, ’rootkit.’), 1)
((107, ’normal.’), 8778)
((107, ’portsweep.’), 1)
((108, ’back.’), 6)
((108, ’ipsweep.’), 2)
((108, ’neptune.’), 33)
((108, ’normal.’), 4202)
((109, ’normal.’), 2)
((109, ’portsweep.’), 1545)
((109, ’satan.’), 3)
((110, ’normal.’), 2714)
((111, ’ftp_write.’), 2)
((111, ’guess_passwd.’), 1)
((111, ’multihop.’), 2)
```

```
((111, ’normal.’), 872)
((111, ’warezclient.’), 4)
((111, ’warezmaster.’), 1)
((112, ’neptune.’), 1)
((112, ’normal.’), 16)
((112, ’portsweep.’), 992)
((112, ’satan.’), 385)
((113, ’back.’), 1)
((113, ’normal.’), 2850)
((114, ’ftp_write.’), 1)
((114, ’normal.’), 2)
((115, ’ipsweep.’), 12)
((115, ’loadmodule.’), 1)
((115, ’nmap.’), 4)
((115, ’normal.’), 2225)
((115, ’portsweep.’), 1)
((115, ’satan.’), 12)
((115, ’warezclient.’), 4)
((116, ’normal.’), 45424)
((117, ’buffer_overflow.’), 1)
((117, ’ftp_write.’), 1)
((117, ’normal.’), 40328)
((118, ’back.’), 471)
((118, ’imap.’), 2)
((118, ’normal.’), 292)
((118, ’warezclient.’), 43)
((119, ’neptune.’), 1)
((119, ’normal.’), 360)
((119, ’portsweep.’), 1)
((119, ’satan.’), 70)
((119, ’smurf.’), 24)
((120, ’nmap.’), 2)
((120, ’normal.’), 700)
((121, ’back.’), 1)
((121, ’normal.’), 9904)
((121, ’warezclient.’), 1)
((122, ’back.’), 1640)
((122, ’normal.’), 153)
((123, ’normal.’), 100)
((124, ’normal.’), 4506)
```

```
((125, ’ftp_write.’), 1)
((125, ’ipsweep.’), 164)
((125, ’neptune.’), 7)
((125, ’normal.’), 1231)
((125, ’portsweep.’), 1)
((125, ’satan.’), 1)
((125, ’warezclient.’), 35)
((126, ’normal.’), 3)
((127, ’ftp_write.’), 1)
((127, ’normal.’), 9928)
((127, ’warezclient.’), 2)
((128, ’normal.’), 51304)
((129, ’pod.’), 94)
((130, ’normal.’), 1)
((131, ’neptune.’), 1761)
((131, ’normal.’), 2)
((131, ’portsweep.’), 32)
((132, ’normal.’), 1)
((133, ’back.’), 1)
((133, ’normal.’), 36064)
((134, ’pod.’), 165)
((134, ’teardrop.’), 9)
((135, ’ipsweep.’), 196)
((135, ’normal.’), 6)
((135, ’portsweep.’), 3752)
((136, ’neptune.’), 365687)
((136, ’normal.’), 21)
((136, ’portsweep.’), 1)
((137, ’normal.’), 1)
((138, ’ipsweep.’), 17)
((138, ’normal.’), 18)
((138, ’portsweep.’), 42)
((138, ’satan.’), 457)
((139, ’ipsweep.’), 31)
((139, ’multihop.’), 1)
((139, ’nmap.’), 10)
((139, ’normal.’), 1008)
((140, ’neptune.’), 100540)
((140, ’normal.’), 7)
((141, ’normal.’), 8)
```

```
((142, 'normal.'), 8)
((143, 'neptune.'), 1)
((143, 'normal.'), 2921)
((143, 'portsweep.'), 2)
((144, 'normal.'), 300)
((144, 'warezclient.'), 1)
((145, 'neptune.'), 11)
((145, 'normal.'), 3623)
((146, 'neptune.'), 1)
((146, 'normal.'), 1736)
((146, 'satan.'), 2)
((147, 'back.'), 2)
((147, 'neptune.'), 3)
((147, 'nmap.'), 1)
((147, 'normal.'), 1630)
((147, 'satan.'), 3)
((147, 'warezclient.'), 1)
((148, 'nmap.'), 73)
((148, 'normal.'), 1344)
((148, 'satan.'), 191)
((148, 'smurf.'), 3046)
((149, 'back.'), 11))
```

Now it's time to build anomaly detector. we use the *KMeans.train* to get the model with *k*=150, and calculate the distance between each point and its cluster center. Of all the distances we calculated before, we get the top 100 elements from then and set the last one as the threshold to test data. If distance between a point and its cluster center is larger than the threshold, then we identify this point is anomalous and give its key which is data with nonnumeric and numeric feature. Otherwise it is normal.

```
clusters =KMeans.train(standardized_data_values, k=150,maxIterations=10, runs=5,
                                            initializationMode="k-means||")
  distance=standardized_data_values.map(lambda line: dist_to_centroid(line,clusters))
  threshold=distance.top(100)[-1]#if want to get the 10 farthest data:
                                              threshold=distance.top(11)[-1]
  print(threshold)
  Combine_Data=raw_test.zip(standardized_data_values1)
  anomalies=Combine_Data.filter(lambda(keys,data):dist_to_centroid(data,clusters)>threshld).keys()
  nn=anomalies.count()
  print(nn)
  for kk in anomalies.take(nn):
     print(kk)
```

The threshold we get is 29.8447237849.
**Test1:** The last 10 line in *kddcup.data* dataset are used to test. In the raw dataset, they are all labelled ”normal”. In the test, the results are normal because there is no object printed which means their distances are all less than the threshold we set.
**Test2:** In this test, we use all the *kddcup.data* dataset to test which connections are anomalous. We get 99 connection whose distances are longer than threshold we set. This is because we set the 100th farthest distance as the threshold.

```
5506,tcp,telnet,SF,1983,8012,0,0,0,0,0,1,0,0,0,0,7,0,0,0,0,0,1,1,0.00,0.00,
0.00,0.00,1.00, 0.00,0.00,55,13,0.20,0.05,0.02,0.15,0.04,0.15,0.00,
0.00,normal.
9,tcp,telnet,SF,307,2374,0,0,1,0,0,1,0,1,0,1,3,1,0,0,0,0,1,1,0.00,0.00,0.00,
0.00,1.00,0.00,0.00,69,4,0.03,0.04,0.01,0.75,0.00,0.00,0.00,
0.00,normal.
40504,tcp,telnet,RSTO,8155,15402,0,0,0,0,0,1,0,0,0,0,26,0,0,0,0,0,1,1,
0.00,0.00,1.00,1.00,1.00,0.00,0.00,7,2,0.29,0.29,0.14,0.00,0.00,0.00,
0.14,0.50,normal.
11565,tcp,telnet,SF,565,111864,0,0,0,0,0,1,217,1,2,247,0,0,4,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,8,3,0.38,0.25,0.12,0.00,0.00,0.00,
0.12,0.33,normal.
25,tcp,telnet,SF,269,2333,0,0,0,0,0,1,0,1,0,2,2,1,0,0,0,0,1,1,0.00,0.00,
0.00,0.00,1.00,0.00,0.00,69,2,0.03,0.06,0.01,0.00,0.00,0.00,0.00,
0.00,perl.
0,tcp,telnet,S1,1905,9092,0,0,0,0,0,1,0,0,0,0,7,0,0,0,0,0,1,1,1.00,1.00,
0.00,0.00,1.00,0.00,0.00,176,26,0.14,0.02,0.01,0.08,0.11,0.77,0.00,
0.00,normal.
12123,tcp,telnet,SF,8641,16120,0,0,0,0,0,1,0,0,0,0,29,0,1,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,179,28,0.15,0.02,0.01,0.07,0.12,0.75,
0.00,0.00,normal.
12491,tcp,telnet,SF,8536,16492,0,0,0,0,0,1,3,0,0,2,32,0,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,33,4,0.12,0.09,0.03,0.00,0.00,0.00,0.00,
0.00,normal.
13383,tcp,telnet,SF,597,94761,0,0,0,0,0,1,349,1,2,387,0,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,131,11,0.08,0.03,0.01,0.00,0.00,0.00,
0.00,0.00,normal.
10027,tcp,telnet,SF,5851,15387,0,0,0,0,0,1,1,0,0,0,28,0,4,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,135,12,0.08,0.53,0.01,0.17,0.01,0.08,
0.51,0.17,normal.
14468,tcp,telnet,SF,11343,18127,0,0,0,0,0,1,1,0,0,0,29,0,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,27,13,0.07,0.11,0.04,0.15,0.00,0.00,
0.00,0.00,normal.
11103,tcp,telnet,SF,738,111026,0,0,0,0,0,1,452,1,2,505,0,0,4,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,32,15,0.16,0.09,0.03,0.13,0.00,
0.00,0.00,0.00,normal.
14583,tcp,telnet,SF,756,80753,0,0,0,0,0,1,345,1,2,390,0,0,2,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,203,13,0.06,0.02,0.00,0.00,0.00,
0.00,0.00,0.00,normal.
11010,tcp,telnet,SF,7025,13152,0,0,0,0,0,1,0,0,0,0,21,0,1,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,7,3,0.29,0.29,0.14,0.67,0.00,0.00,0.00,
0.00,normal.
15122,tcp,telnet,SF,1154,176690,0,0,0,0,0,1,884,1,2,975,0,0,5,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,15,4,0.20,0.13,0.07,0.50,0.00,0.00,
0.00,0.00,normal.
10514,tcp,telnet,SF,2625,34656,0,0,0,11,0,1,0,0,0,0,8,0,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,23,5,0.17,0.13,0.04,0.40,0.00,0.00,0.00,
0.00,normal.
11877,tcp,telnet,SF,11486,17241,0,0,0,0,0,1,1,0,0,0,22,0,1,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,108,21,0.19,0.05,0.01,0.00,0.01,0.05,
0.00,0.00,normal.
310,tcp,telnet,SF,595,121612,0,0,1,1,0,1,187,1,2,187,0,0,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,255,1,0.00,0.01,0.00,0.00,0.00,0.00,
0.00,0.00,normal.
2514,tcp,telnet,SF,1937,84751,0,0,2,1,0,1,74,1,2,77,9,0,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,255,2,0.01,0.01,0.00,0.00,0.00,0.00,
0.00,0.00,normal.
2,tcp,finger,RSTO,693375640,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,57,3,0.79,
0.67,0.21,0.33,0.05,0.39,0.00,255, 3,0.01,0.09,0.22,0.00,0.18,0.67,
0.05,0.33,portsweep.
1154,tcp,telnet,SF,360,107836,0,0,1,0,0,1,9,0,0,1,0,0,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,73,1,0.01,0.05,0.01,0.00,0.00,0.00,
0.00,0.00,normal.
67,tcp,login,SF,157,2703,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,
0.00,0.00,1.00,0.00,0.00,2,1,0.50,1.00, 0.50,0.00,0.00,0.00,0.00,
0.00,ftp_write.
408,tcp,telnet,SF,385,3410,0,0,0,0,0,1,0,0,0,0,4,0,1,0,0,0,1,1,0.00,0.00,
0.00,0.00,1.00,0.00,0.00,255,9,0.04,0.70,0.00,0.00,0.69,0.11,0.00,
0.00,normal.
0,tcp,telnet,S1,1419,24871,0,0,0,11,0,1,0,0,0,0,6,0,0,0,0,0,1,1,1.00,1.00,
0.00,0.00,1.00,0.00,0.00,255,16,0.06,0.59,0.00,0.00,0.59,0.19,
0.00,0.00,normal.
```

```
18022,tcp,telnet,SF,4734,54623,0,0,0,44,0,1,0,0,0,0,25,0,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,255,19,0.07,0.57,0.00,0.00,0.57,
0.16,0.00,0.05,normal.
156,tcp,telnet,SF,210,291180,0,0,1,0,0,1,27,0,0,26,0,0,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,255,2,0.01,0.02,0.00,0.00,0.00,0.00,
0.00,0.00,normal.
582,tcp,telnet,SF,451,10934,0,0,0,2,0,1,1,0,0,1,1,0,2,0,0,1,1,1,0.00,0.00,
0.00,0.00,1.00,0.00,0.00,255,2,0.01,0.02,0.00,0.00,0.00,0.00,
0.00,0.00,normal.
189,tcp,ftp,SF,536,1664,0,0,0,11,1,1,0,0,0,0,0,0,0,0,0,1,1,1,0.00,0.00,
0.00,0.00,1.00,0.00,0.00,6,2,0.33,0.50,0.17,0.00,0.00,0.00,0.17,
0.50,normal.
521,tcp,telnet,SF,1205,15071,0,0,0,1,0,1,8,1,1,7,17,0,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,255,20,0.08,0.07,0.00,0.00,0.97,0.90,
0.00,0.00,normal.
12039,tcp,telnet,SF,798,276683,0,0,0,0,0,1,462,1,2,512,0,0,3,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,37,0.15,0.06,0.00,0.00,0.57,
0.30,0.04,0.11,normal.
17903,tcp,telnet,SF,5830,122343,0,0,0,77,0,1,6,0,0,1,19,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,38,0.15,0.06,0.00,0.00,0.56,
0.29,0.04,0.11,normal.
13812,tcp,telnet,SF,10991,83763,0,0,0,0,0,1,0,0,0,0,40,0,1,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,39,0.15,0.06,0.00,0.00,0.54,
0.26,0.04,0.13,normal.
438,tcp,telnet,SF,296,15420,0,0,1,0,0,1,5,1,2,7,0,0,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,255,2,0.01,0.02,0.00,0.00,0.00,0.00,0.00,
0.00,normal.
156,tcp,ftp,SF,950,2551,0,0,0,18,0,1,0,0,0,0,21,0,0,0,0,1,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,218,1,0.00,0.03,0.00,0.00,0.01,0.00,0.07,
0.00,warezmaster.
988,tcp,telnet,SF,14384,74509,0,0,0,0,0,1,247,1,1,402,11,0,0,0,0,0,
1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,2,0.01,0.01,0.00,0.00,0.00,
0.00,0.05,0.00,normal.
16800,tcp,telnet,SF,1970,101286,0,0,0,0,1,1,151,0,1,151,1,0,0,0,0,
0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,38,6,0.16,0.11,0.03,0.00,0.03,
0.00,0.05,0.33,normal.
0,tcp,http,SF,143,11993,0,0,0,21,0,1,21,0,0,0,0,0,0,0,0,0,9,11,0.00,
0.00,0.00,0.00,1.00,0.00,0.18,9,255,1.00,0.00,0.11,0.02,0.00,0.00,0.00,
0.00,normal.
16125,tcp,telnet,SF,882,91068,0,0,0,0,0,1,622,1,2,684,0,0,1,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,16,5,0.31,0.19,0.06,0.00,0.00,0.00,
0.00,0.00,normal.
12161,tcp,telnet,SF,8004,14762,0,0,0,0,0,1,1,0,0,0,29,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,29,8,0.28,0.14,0.03,0.00,0.00,0.00,
0.00,0.00,normal.
9502,tcp,telnet,SF,6190,15338,0,0,0,0,0,1,3,0,0,0,20,0,1,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,134,23,0.17,0.03,0.01,0.00,0.00,0.00,
0.01,0.04,normal.
15211,tcp,telnet,SF,454,144329,0,0,0,0,0,1,177,1,2,190,0,0,1,0,0,0,
1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,3,2,0.33,0.67,0.33,1.00,0.00,0.50,
0.00,0.00,normal.
7782,tcp,telnet,SF,677,67023,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,6,3,0.33,0.33,0.17,0.67,0.00,0.33,0.00,
0.00,normal.
13169,tcp,telnet,SF,10236,14839,0,0,0,0,0,1,0,0,0,0,30,0,1,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,57,11,0.18,0.05,0.02,0.18,0.00,0.09,
0.00,0.00,normal.
12632,tcp,telnet,SF,3416,34430,0,0,0,22,0,1,6,0,0,0,16,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,76,16,0.20,0.04,0.01,0.12,0.00,0.06,
0.00,0.00,normal.
11597,tcp,telnet,SF,7844,19418,0,0,0,0,0,1,2,0,0,0,32,0,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,77,17,0.21,0.04,0.01,0.12,0.00,0.06,
0.00,0.00,normal.
0,tcp,telnet,S1,223,12698,0,0,0,0,0,1,94,1,1,104,0,0,0,0,0,0,1,1,1.00,
1.00,0.00,0.00,1.00,0.00,0.00,134,20,0.15,0.04,0.01,0.00,0.01,0.05,
0.01,0.10,normal.
0,tcp,telnet,S1,1752,5177,0,0,0,0,0,1,0,0,0,0,6,0,0,0,0,0,1,1,1.00,1.00,
0.00,0.00,1.00,0.00,0.00,149,23,0.15,0.03,0.01,0.00,0.01,0.09,0.01,
0.09,normal.
2552,tcp,telnet,SF,589,28390,0,0,0,0,0,1,21,1,1,17,0,0,1,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,2,0.01,0.02,0.00,0.00,0.00,0.00,
0.00,0.00,normal.
54,tcp,telnet,SF,260,2635,0,0,0,0,0,1,0,1,0,2,2,1,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,255,1,0.00,0.01,0.00,0.00,0.00,0.00,0.00,
0.00,perl.
14362,tcp,telnet,SF,705,106734,0,0,0,0,0,1,307,1,2,338,0,0,2,0,0,0,
1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,2,0.01,0.07,0.00,0.00,0.99,
0.00,0.00,0.00,normal.
0,tcp,telnet,S1,1536,13968,0,0,0,0,0,1,2,0,0,0,7,0,0,0,0,0,1,1,1.00,
1.00,0.00,0.00,1.00,0.00,0.00,255,7,0.03,0.07,0.00,0.00,0.84,0.14,0.00,
0.00,normal.
13535,tcp,telnet,SF,10606,17797,0,0,0,0,0,1,4,0,0,2,40,0,0,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,12,0.05,0.06,0.00,0.00,0.76,
0.08,0.00,0.00,normal.
15883,tcp,telnet,SF,2109,144715,0,0,0,0,0,1,37,0,0,36,2,0,1,0,0,0,
151,1,0.99,0.00,0.00,0.00,0.01,0.07,0.00,255,27,0.11,0.05,0.00,0.00,0.42,
0.07,0.02,0.15,normal.
13919,tcp,telnet,SF,7766,11508,0,0,0,0,0,1,0,0,0,0,7,0,1,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,52,0.20,0.02,0.00,0.00,0.60,0.75,
0.00,0.00,normal.
18848,tcp,telnet,SF,2327,418769,0,0,0,0,0,1,281,0,1,278,1,0,0,0,0,
0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,50,0.20,0.02,0.00,0.00,0.36,
0.46,0.00,0.00,normal.
13620,tcp,telnet,SF,1024,165715,0,0,0,0,0,1,174,1,1,204,0,0,3,0,0,
0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,45,0.18,0.02,0.00,0.00,0.24,
0.36,0.00,0.00,normal.
2736,tcp,telnet,SF,2225,35841,0,0,0,0,0,1,110,1,1,121,13,0,0,0,0,0,
1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,1,0.00,0.02,0.00,0.00,0.00,
0.00,0.00,0.00,normal.
98,tcp,telnet,SF,621,8356,0,0,1,1,0,1,5,1,0,14,1,0,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,255,4,0.02,0.02,0.00,0.00,0.00,0.00,0.00,
0.00,rootkit.
15382,tcp,telnet,SF,1011,221754,0,0,0,0,0,1,254,1,2,289,0,0,5,0,0,
0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,25,12,0.48,0.08,0.04,0.00,0.00,
0.00,0.00,0.00,normal.
9378,tcp,telnet,SF,592,95596,0,0,0,0,0,1,275,1,2,306,0,0,3,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,67,21,0.31,0.04,0.01,0.00,0.00,0.00,
0.00,0.00,normal.
14517,tcp,telnet,SF,794,152110,0,0,0,0,0,1,520,1,2,572,0,0,8,0,0,0,
1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,8,3,0.38,0.25,0.12,0.00,0.00,0.00,
0.00,0.00,normal.
15377,tcp,telnet,SF,975,165912,0,0,0,0,0,1,676,1,2,754,0,0,6,0,0,0,
1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,130,25,0.19,0.03,0.01,0.00,0.01,
0.04,0.02,0.08,normal.
18677,tcp,telnet,SF,1544,32120,0,0,0,3,1,1,78,0,0,71,0,0,0,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,142,28,0.20,0.03,0.01,0.00,0.01,0.04,
0.01,0.07,normal.
11816,tcp,telnet,SF,5577,68684,0,0,0,0,0,1,0,0,0,0,20,0,1,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,26,10,0.38,0.15,0.04,0.00,0.04,0.10,
0.00,0.00,normal.
```

```
20252,tcp,telnet,RSTO,737,112155,0,0,0,0,0,1,457,1,2,508,0,0,2,0,
0,0,1,1,0.00,0.00,1.00,1.00,1.00,0.00,0.00,54,14,0.26,0.11,0.02,0.00,0.02,
0.07,0.04,0.14,normal.
3138,tcp,telnet,RSTO,329,205950,0,0,0,0,1,1,4,0,1,3,0,0,0,0,0,0,1,
1,0.00,0.00,1.00,1.00,1.00,0.00,0.00,92,22,0.24,0.07,0.01,0.00,0.01,0.05,
0.09,0.36,normal.
14943,tcp,telnet,SF,1357,227537,0,0,0,0,0,1,756,1,2,857,0,0,7,0,0,
0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,2,2,0.50,1.00,0.50,1.00,0.00,0.00,
0.00,0.00,normal.
0,tcp,telnet,S1,5173,7962,0,0,0,0,0,1,0,0,0,0,11,0,1,0,0,0,1,1,1.00,
1.00,0.00,0.00,1.00,0.00,0.00,88,9,0.10,0.06,0.01,0.00,0.02,0.22,0.00,
0.00,normal.
3396,tcp,telnet,SF,1649,241571,0,0,0,10,0,1,16,1,0,10,8,0,0,0,0,0,0,
1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,31,6,0.19,0.13,0.03,0.00,0.00,0.00,
0.00,0.00,normal.
13956,tcp,telnet,SF,7297,13479,0,0,0,0,0,1,1,0,0,0,32,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,42,8,0.19,0.12,0.02,0.00,0.00,0.00,
0.00,0.00,normal.
13014,tcp,telnet,SF,732,122323,0,0,0,0,0,1,373,1,2,421,0,0,5,0,0,0,
1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,44,9,0.20,0.11,0.02,0.00,0.00,0.00,
0.00,0.00,normal.
13417,tcp,telnet,SF,986,144411,0,0,0,0,0,1,789,1,2,867,0,0,2,0,0,0,
1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,224,37,0.17,0.03,0.00,0.00,0.00,
0.03,0.01,0.05,normal.
45,tcp,telnet,SF,268,2364,0,0,0,0,0,1,0,1,0,2,2,1,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,255,2,0.01,0.02,0.00,0.00,0.00,0.00,
0.69,0.00,perl.
85,tcp,telnet,SF,136,383,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,1,9,1.00,0.00,1.00,0.78,0.00,0.00,0.00,
0.00,normal.
5158,tcp,X11,SF,89581520,7028652,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,
0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,11,0.04,0.03,0.01,0.00,0.00,
0.00,0.00,0.00,normal.
16754,tcp,telnet,SF,3555,198172,0,0,0,2,0,1,151,1,2,146,1,0,1,0,0,
0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,9,3,0.22,0.33,0.11,0.67,0.00,
0.00,0.00,0.00,normal.
18682,tcp,telnet,SF,5058,23616,0,0,0,16,0,1,6,0,0,10,25,0,2,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,19,4,0.21,0.21,0.05,0.00,0.00,
0.00,0.00,0.00,normal.
14697,tcp,telnet,SF,1419,320547,0,0,0,0,0,1,884,1,2,993,0,0,8,0,0,0,
1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,42,11,0.26,0.10,0.02,0.00,0.02,
0.09,0.00,0.00,normal.
11839,tcp,telnet,SF,2469,9509,0,0,0,1,0,1,1,0,0,2,11,0,2,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,101,20,0.20,0.05,0.01,0.00,0.01,0.05,
0.00,0.00,normal.
5930,tcp,telnet,RSTR,197,5339,0,0,0,0,1,1,2,1,2,6,0,0,0,0,0,0,1,1,
0.00,0.00,1.00,1.00,1.00,0.00,0.00,80,4,0.05,0.08,0.01,0.00,0.03,0.25,0.65,
0.25,normal.
14743,tcp,telnet,SF,697,94872,0,0,0,0,0,1,456,1,2,502,0,0,1,0,0,0,
1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,7,0.03,0.06,0.00,0.00,0.93,0.00,
0.01,0.29,normal.
10123,tcp,telnet,SF,6664,15661,0,0,0,0,0,1,2,0,0,2,22,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,29,0.11,0.05,0.00,0.00,0.30,0.00,0.02,
0.14,normal.
14538,tcp,telnet,SF,4454,135624,0,0,0,0,1,1,23,1,1,22,11,0,0,0,0,0,
1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,30,0.12,0.05,0.00,0.00,0.29,0.00,
0.02,0.13,normal.
38259,tcp,discard,RSTOS0,621568663,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1,1,0.00,0.00,1.00,1.00,1.00,0.00,0.00,74,1,0.01,0.08,0.03,0.00,0.00,
0.00,0.03,1.00,portsweep.
36071,tcp,ftp,RSTOS0,1379963888,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
1,1,0.00,0.00,1.00,1.00,1.00,0.00,0.00,77,1,0.01,0.12,0.06,0.00,0.00,0.00,
0.06,1.00,portsweep.
34578,tcp,private,RSTOS0,1167519497,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1,1,0.00,0.00,1.00,1.00,1.00,0.00,0.00,80,1,0.01,0.12,0.10,0.00,0.00,
0.00,0.10,1.00,portsweep.
12345,tcp,telnet,SF,8070,14511,0,0,0,0,0,1,1,0,0,0,35,0,5,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,2,4,1.00,0.00,0.50,0.50,0.00,0.00,0.00,
0.00,normal.
18053,tcp,telnet,SF,4571,29709,0,0,0,3,0,1,0,0,0,0,26,0,2,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,63,0.25,0.02,0.00,0.00,0.92,0.95,
0.00,0.00,normal.
14494,tcp,telnet,SF,293,33542,0,0,0,0,0,1,46,1,2,55,0,0,1,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,67,0.26,0.02,0.00,0.00,0.84,0.84,
0.00,0.00,normal.
12476,tcp,telnet,SF,8102,20827,0,0,0,0,0,1,1,0,0,0,35,0,5,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,2,2,1.00,0.00,0.50,0.00,0.00,0.00,0.00,
0.00,normal.
14664,tcp,telnet,SF,8720,15392,0,0,0,0,0,1,1,0,0,0,41,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,33,7,0.21,0.09,0.03,0.00,0.00,0.00,0.00,
0.00,normal.
4245,tcp,telnet,SF,3887,143093,0,0,0,0,1,1,8,0,1,11,1,0,1,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,35,8,0.23,0.09,0.03,0.00,0.00,0.00,0.00,
0.00,normal.
11680,tcp,telnet,SF,738,103904,0,0,0,0,0,1,375,1,2,425,0,0,1,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,43,11,0.26,0.07,0.02,0.00,0.00,0.00,
0.00,0.00,normal.
12988,tcp,telnet,SF,825,100118,0,0,0,0,0,1,568,1,2,626,0,0,4,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,31,8,0.26,0.13,0.03,0.00,0.00,0.00,
0.00,0.00,normal.
16187,tcp,telnet,SF,1119,176336,0,0,0,0,0,1,691,1,2,766,0,0,8,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,2,2,1.00,0.00,0.50,0.00,0.00,0.00,
0.00,0.00,normal.
11233,tcp,telnet,SF,2361,46593,0,0,0,8,0,1,54,1,2,39,9,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,205,104,0.51,0.01,0.00,0.00,0.98,0.96,
0.00,0.00,normal.
15263,tcp,telnet,SF,9933,14772,0,0,0,0,0,1,0,0,0,0,40,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,113,0.44,0.01,0.00,0.00,0.78,0.88,
0.00,0.00,normal.
8954,tcp,telnet,SF,394,54636,0,0,0,0,0,1,102,1,2,119,0,0,3,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,97,0.38,0.01,0.00,0.00,0.64,0.82,
0.00,0.00,normal.
14301,tcp,telnet,SF,703,107135,0,0,0,0,0,1,405,1,2,450,0,0,2,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,77,0.30,0.02,0.00,0.00,0.43,0.64,
0.00,0.01,normal.
```

From the data get, we can clearly see that most of them are labeled ”normal”, but they may be anomalous. Let’s take two points for example.

```
5506,tcp,telnet,SF,1983,8012,0,0,0,0,0,1,0,0,0,0,7,0,0,0,0,0,1,1,0.00,0.00,
0.00,0.00,1.00, 0.00,0.00,55,13,0.20,0.05,0.02,0.15,0.04,0.15,0.00,
0.00,normal.
```

The first column represents the connection duration. The duration of data above is very long and may be anomalous.

```
0,tcp,http,SF,143,11993,0,0,0,21,0,1,21,0,0,0,0,0,0,0,0,0,9,11,0.00,
0.00,0.00,0.00,1.00,0.00,0.18,9,255,1.00,0.00,0.11,0.02,0.00,0.00,0.00,
0.00,normal.
```

Unlikely the first example, this duration is 0 (Maybe very short I think). However the connection to host is 255 in such a short time. So this may be anomalous in this case.
The 10 farthest point in the KDDdata are shown as follows.

```
9,tcp,telnet,SF,307,2374,0,0,1,0,0,1,0,1,0,1,3,1,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,69,4,0.03,0.04,0.01,0.75,0.00,0.00,
0.00,0.00,normal.
11447,tcp,telnet,SF,3131,45415,0,0,0,1,0,1,0,1,0,0,15,0,0,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,100,10,0.09,0.72,0.01,0.20,
0.01,0.10,0.69,0.20,normal.
13368,tcp,telnet,SF,2987,58843,0,0,0,1,0,1,2,1,0,2,15,0,0,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,127,11,0.08,0.57,0.01,0.18,
0.01,0.09,0.54,0.18,normal.
7805,tcp,telnet,SF,1828,148722,0,0,0,1,2,1,19,0,1,10,8,0,0,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,51,7,0.14,0.08,0.02,0.00,0.00,
0.00,0.00,0.00,normal.
310,tcp,telnet,SF,595,121612,0,0,1,1,0,1,187,1,2,187,0,0,0,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,1,0.00,0.01,0.00,0.00,0.00,
0.00,0.00,0.00,normal.
17903,tcp,telnet,SF,5830,122343,0,0,0,77,0,1,6,0,0,1,19,0,0,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,38,0.15,0.06,0.00,0.00,0.56,
0.29,0.04,0.11,normal.
438,tcp,telnet,SF,296,15420,0,0,1,0,0,1,5,1,2,7,0,0,0,0,0,0,1,1,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,255,2,0.01,0.02,0.00,0.00,0.00,0.00,
0.00,0.00,normal.
5930,tcp,telnet,RSTR,197,5339,0,0,0,0,1,1,2,1,2,6,0,0,0,0,0,0,1,1,
0.00,0.00,1.00,1.00,1.00,0.00,0.00,80,4,0.05,0.08,0.01,0.00,0.03,0.25,
0.65,0.25,normal.
14538,tcp,telnet,SF,4454,135624,0,0,0,0,1,1,23,1,1,22,11,0,0,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,30,0.12,0.05,0.00,0.00,0.29,
0.00,0.02,0.13,normal.
36071,tcp,ftp,RSTOS0,1379963888,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
1,0.00,0.00,1.00,1.00,1.00,0.00,0.00,77,1,0.01,0.12,0.06,0.00,0.00,0.00,
0.06,1.00,portsweep.
```

In the 10 farthest point, 9 of which are labeled ”normal”. However, they may also be anomalous as we said before. The duration of some data above is very long, more than 5000, and serve different host rate is more than 200.

```
17903,tcp,telnet,SF,5830,122343,0,0,0,77,0,1,6,0,0,1,19,0,0,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,38,0.15,0.06,0.00,0.00,0.56,
0.29,0.04,0.11,normal.
```

The point above has a duration longer than 17900, and the number of compromised is 77, which seems a little unusual.

```
17903,tcp,telnet,SF,5830,122343,0,0,0,77,0,1,6,0,0,1,19,0,0,0,0,0,1,
1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,38,0.15,0.06,0.00,0.00,0.56,
0.29,0.04,0.11,normal.
```

This point above has 77 host which is far more than others.