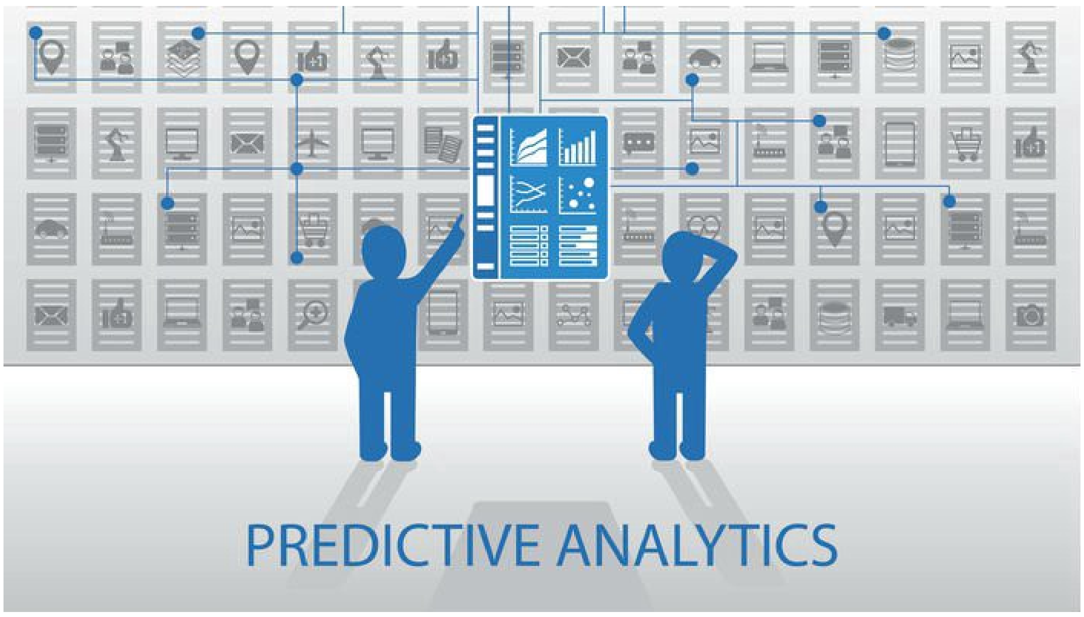
Clustering and Predictive Analysis to group problems

Ryan Aminollahi
Feb 17, 2018 · 11 min read

Predictive Customer Analytics — Part IV



If you read my previous articles you know, in the story, customer X has purchased a laptop. Now, imagine that he's trying to set it up in his home office. Upon doing so, he finds that his laptop keeps losing connection with his wireless keyboard. He thinks he messed up the setup and he tries to find ways to fix it. He goes to the customer website to see if there are any self-help videos that will help him identify the cause. He cannot find any. He then calls the company's number. It goes to a self-service prompt that asks him for all his information.

He's then put on a queue for the next 15 minutes. Eventually, he gets to an agent who asks him for all the information all over again. As you can probably imagine, X is getting irritated. After listening to his problem the agent provides him a couple of instructions over the phone, but X is unable to understand and follow them. He requests for in-house help but the agent declines it, saying, he does not have that covered under his warranty. Frustrated, He is thinking of never buying from this vendor again.

He is an example of countless customers who are frustrated with bad-quality service and support. In today's Internet world, an alternate vendor is only a click away. Businesses need to work harder and smarter to keep their existing customers happy and get referrals for new customers. The customer service process starts with providing customers with interaction options, from website self-help to e-mails to phone support, today's support centres run on omni channels.

Once a customer initiates a contact, the business needs to provide them with quality service:

> *ask minimum questions and resolve problems quickly.*

This requires that the business not only knows about the customer but also keeps context on his current problems and related interactions. It needs to let the customer feel that the business truly knows about his problems and cares for him.

Today's customer service world has moved towards self-service and automation using interactive voice response and chatbot.

These automation pieces need intelligence to predict what the customer is going through and ask relevant questions. Customers service is definitely an area where predictive analytics can help in understanding customer problems and providing fast resolutions.

Predict intent of contact

Let's think about what went wrong. Given that X has already given his phone number during the sales process, shouldn't customer support automatically identify him with this caller ID?

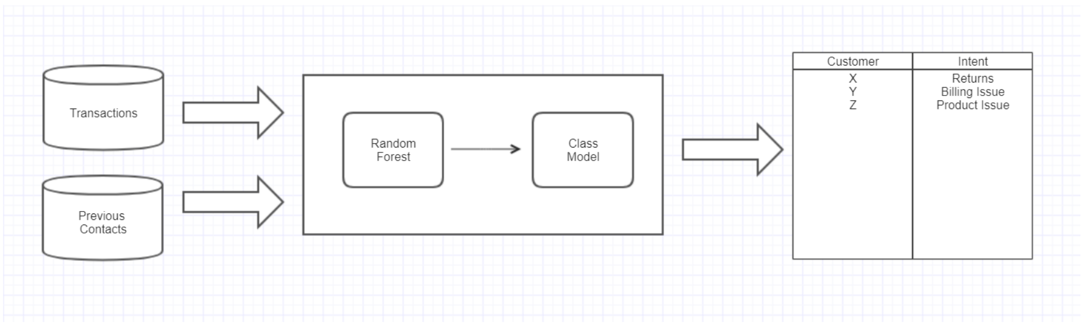
He is calling again within two days after the delivery of the laptop. Wouldn't that mean that his call is most likely related to that purchase?

During the first call, customer support created a ticket for his laptop keyboard troubles and gave him some advice.

However, he is calling again after two hours. Can't customer support guess that his second call is most likely related to this first ticket? The goal of this use case is to build a prediction model that can predict the possible reason the customer is contacting the business through customer support. Knowing the reason will help the business to quickly direct the contact to the right skilled person and get it resolved on the first interaction. The data for this use case would be all the previous transactions that happened between the customer and the business.

It contains sales transactions that include details like products, status, and delivery. We will also use data related to previous contacts between the customer and the business. For each contact, we will use reason, the agent who handled the contact, duration, root cause, resolution, status, etcetera. This is a classification problem, so previous algorithms will be a good fit. It is important to keep the number of target dot classes to less than 10.

The algorithm will be used to generate the model for predicting intent. we will use sale transactions and previous contacts data to generate the model that can then be used to predict the intent of contact. The call for action is as follows. The model for this classification will be built offline. When a customer contacts the business through a phone call or chat, using caller ID, or user ID, identify the customer, then use the algorithm to predict the reason for calling. If the prediction says that the reason will be last purchase, then answer through the IVR with “Hello, are you calling regarding your “recently purchased laptop?” He would definitely be happy and relieved that the business can guess his problem, even if it might not be the actual reason.



It aptly shows that we know him and what he’s going through.

Find unsatisfied customers

How does a business find out if their customers are happy or unhappy with their products or service?

Typically they do so with a survey and compute an overall customer satisfaction score. Surveys work great when customers answer them but survey results are skewed in two primary ways.

> *First, unsatisfied customers typically take the surveys.*

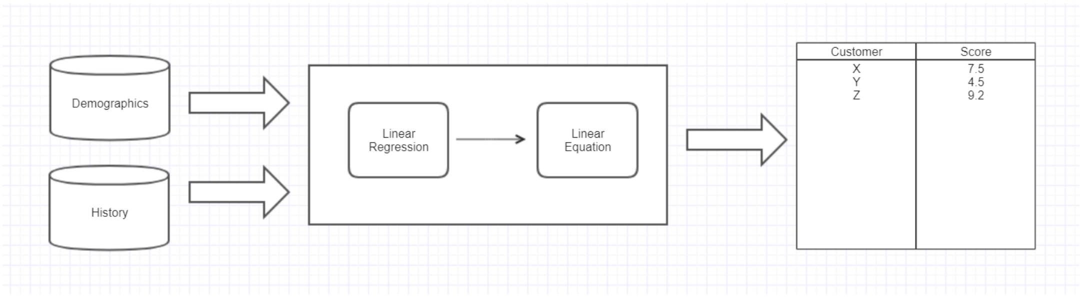
Think about it, since Mr.X is upset about the service and product he’s probably more willing to spend time on a survey and vent, while Mrs. Y might not be that forthcoming.

> *Second, only 10% of the customers respond to a survey.*

So how does the business identify those silent, unsatisfied customers who might be thinking of or have already moved on to other options? The goal of this use case is to build a model that would predict a customer satisfaction score for all its customers, regardless of whether they filled out a survey. The data used for building this model is from customers who actually answered the survey themselves.

The data included customer demographics and customer history. This includes events and transactions with the company and their results. Specifically, it contains information about defect, returns, open tickets, time to solve issues, number of service calls, etc. They form the feature variables. The target variable is going to be the satisfaction score that comes out of the survey. Typically, that score could in the range of zero to five or zero to 10 with one or two decimal points.

If the range of scores is discrete like one, two, three, four, five we can go with the classification of Gordon. In this case, assuming it to be continuous we will go with **Linear Regression**. Using demographics and history data you build a Linear Regression equation that would predict a customer satisfaction score. Call to Action is this, using data from customers who actually answered surveys we build a model to predict the customer satisfaction score. We want to normalize the survey data to eliminate outliers that would impact the overall model, then we apply this model on all of the customers to find out what their satisfaction score would be.



Then our customer support team can take the list of top unsatisfied customers and go to work with them like contacting them proactively and asking them how they are doing with their products or offering them discounts, at least.

Group problem types

Technical support teams handle various types of problems every day. Some problems are frequent and straightforward that the technician on call can spell out the solution even before the customer has completed the explanation. Others are infrequent and complex that would require multiple phone calls and onsite visit to fix. An issue like internet connectivity is easier to troubleshoot, but a blue screen on a laptop is much harder. Real people resources are expensive, so businesses need to find ways to optimize their time.

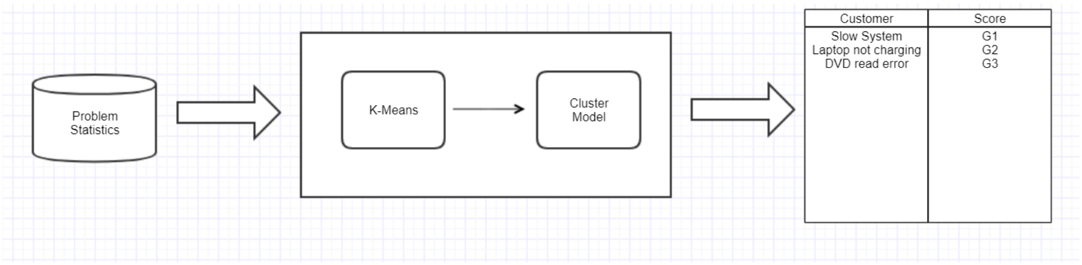
This includes providing online help for some problems. It includes more rigorous training for the tech support team or creating new specialist positions for complex issues. To help businesses make these decisions they want to group problems based on similar attributes. Here is the goal for this use case. From a long list of problem types, the business should identify logical groups that can then be used to develop an optimal resolution plan with least human effort.

The data used for this use case would be problem statistics from case data captured from technical support. The case data is summarized by problem type. The featured data would be items like average time to resolve the problem, average number of calls, replacement rate, etc. Given that we are going to create logical groups we would go with **K-Means clustering** or variants of it. It is important to measure the optimal number of groups by repeating this exercise for multiple values of K and take the need test.

Using problem statistic data, we would group problems into similar types.

The call for action is as follows:

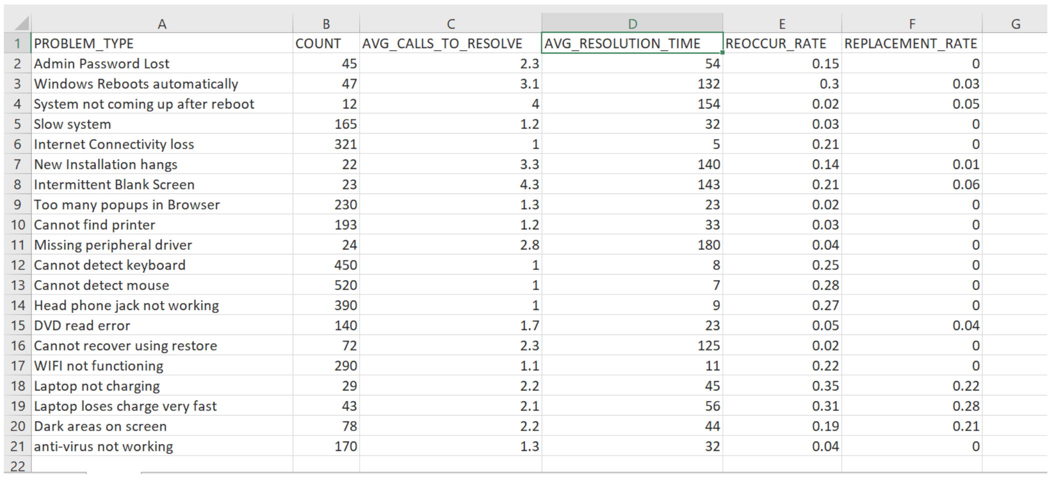
Once the problem types are grouped we would analyse each group to find similarities between items in the group, like very high times to solve the problem or very low occurrence. Then we would come up with plans to optimize their resolution, like providing self-help on our website, providing videos on YouTube, or better training to technicians in these problem areas.



As a business, we want to achieve maximum efficiency and effectiveness when it comes to customer support. This use case helps us in that goal.

Use case Group problem types

How we can group different problem types into similar items and then analyse them as a group to see if see any patterns and then make some actions to solve them efficiently and effectively.



	A	B	C	D	E	F	G
1	PROBLEM_TYPE	COUNT	AVG_CALLS_TO_RESOLVE	AVG_RESOLUTION_TIME	REOCCUR_RATE	REPLACEMENT_RATE	
2	Admin Password Lost	45	2.3	54	0.15	0	
3	Windows Reboots automatically	47	3.1	132	0.3	0.03	
4	System not coming up after reboot	12	4	154	0.02	0.05	
5	Slow system	165	1.2	32	0.03	0	
6	Internet Connectivity loss	321	1	5	0.21	0	
7	New Installation hangs	22	3.3	140	0.14	0.01	
8	Intermittent Blank Screen	23	4.3	143	0.21	0.06	
9	Too many popups in Browser	230	1.3	23	0.02	0	
10	Cannot find printer	193	1.2	33	0.03	0	
11	Missing peripheral driver	24	2.8	180	0.04	0	
12	Cannot detect keyboard	450	1	8	0.25	0	
13	Cannot detect mouse	520	1	7	0.28	0	
14	Head phone jack not working	390	1	9	0.27	0	
15	DVD read error	140	1.7	23	0.05	0.04	
16	Cannot recover using restore	72	2.3	125	0.02	0	
17	WIFI not functioning	290	1.1	11	0.22	0	
18	Laptop not charging	29	2.2	45	0.35	0.22	
19	Laptop loses charge very fast	43	2.1	56	0.31	0.28	
20	Dark areas on screen	78	2.2	44	0.19	0.21	
21	anti-virus not working	170	1.3	32	0.04	0	
22							

Loading the Dataset

In [1]:

```
%matplotlib inline

from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import os
import matplotlib.pylab as plt
from sklearn.model_selection  import train_test_split
from sklearn.cluster import KMeans
import sklearn.metrics

raw_data = pd.read_csv("issues.csv")
raw_data.dtypes
```

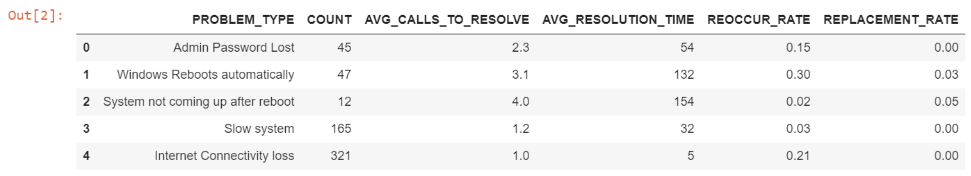
Out[1]:

```
PROBLEM_TYPE             object
COUNT                     int64
AVG_CALLS_TO_RESOLVE    float64
```

```
AVG_RESOLUTION_TIME       int64
REOCCUR_RATE            float64
REPLACEMENT_RATE        float64
dtype: object
```

The dataset contains one record for each unique problem type. It has metrics for each type like count, average calls to resolve, average resolution time etc.

```
raw_data.head()
```



Out[2]:

	PROBLEM_TYPE	COUNT	AVG_CALLS_TO_RESOLVE	AVG_RESOLUTION_TIME	REOCCUR_RATE	REPLACEMENT_RATE
0	Admin Password Lost	45	2.3	54	0.15	0.00
1	Windows Reboots automatically	47	3.1	132	0.30	0.03
2	System not coming up after reboot	12	4.0	154	0.02	0.05
3	Slow system	165	1.2	32	0.03	0.00
4	Internet Connectivity loss	321	1.0	5	0.21	0.00

Group Data into similar clusters

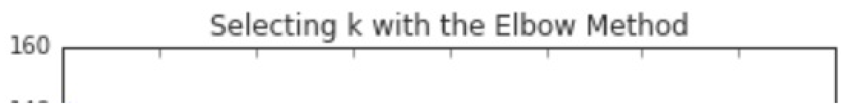
Now, we will use K-Means clustering to group data based on their attribute. First, we need to determine the optimal number of groups. For that, we conduct the knee test to see where the knee happens.

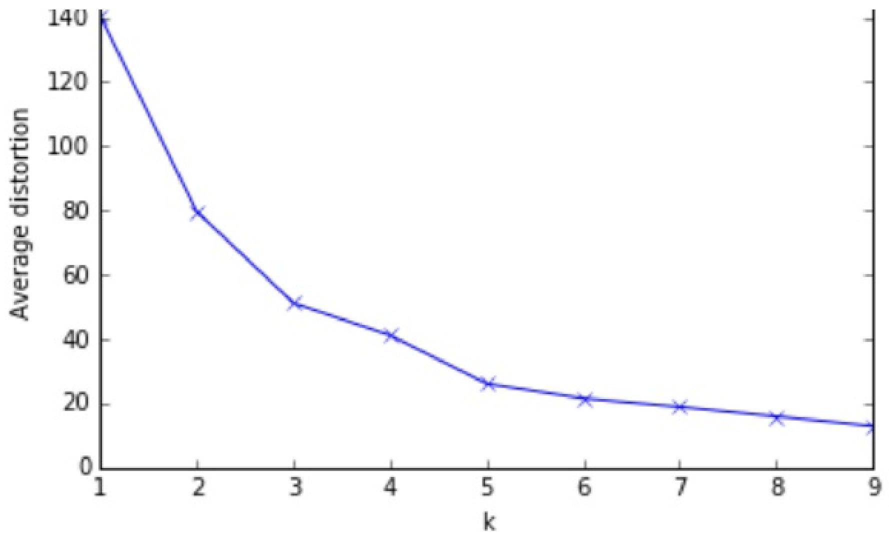
In [3]:

```
clust_data = raw_data.drop("PROBLEM_TYPE",axis=1)

#Finding optimal no. of clusters
from scipy.spatial.distance import cdist
clusters=range(1,10)
meanDistortions=[]

for k in clusters:
    model=KMeans(n_clusters=k)
    model.fit(clust_data)
    prediction=model.predict(clust_data)
    meanDistortions.append(sum(np.min(cdist(clust_data, model.cluster_centers_, 'euclidean'), axis=1)) /
clust_data.shape[0])

#plt.cla()
plt.plot(clusters, meanDistortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Average distortion')
plt.title('Selecting k with the Elbow Method')
```

Out[3]:

```
<matplotlib.text.Text at 0x27298f49eb8>
```





Looking at the plot, we see that the knee happens at cluster=3.

That is the ideal number of clusters. We now perform the actual clustering for 3.

Then we add the cluster ID to the original dataset.

In [4]:

```
#Optimal clusters is 3
final_model=KMeans(3)
final_model.fit(clust_data)
prediction=final_model.predict(clust_data)

#Join predicted clusters back to raw data
raw_data["GROUP"] = prediction
print("Groups Assigned : \n")
raw_data[["GROUP","PROBLEM_TYPE"]]
Groups Assigned :
```

Out[4]:

	GROUP	PROBLEM_TYPE
0	0	Admin Password Lost
1	0	Windows Reboots automatically
2	0	System not coming up after reboot
3	2	Slow system
4	1	Internet Connectivity loss
5	0	New Installation hangs
6	0	Intermittent Blank Screen
7	2	Too many popups in Browser
8	2	Cannot find printer
9	0	Missing peripheral driver
10	1	Cannot detect keyboard
11	1	Cannot detect mouse
12	1	Head phone jack not working
13	2	DVD read error
14	0	Cannot recover using restore

15	2	WIFI not functioning
16	0	Laptop not charging
17	0	Laptop loses charge very fast
18	0	Dark areas on screen
19	2	anti-virus not working

Analyse the groups

We now do a set of *boxplots* to see how the groups differ for various feature attributes.

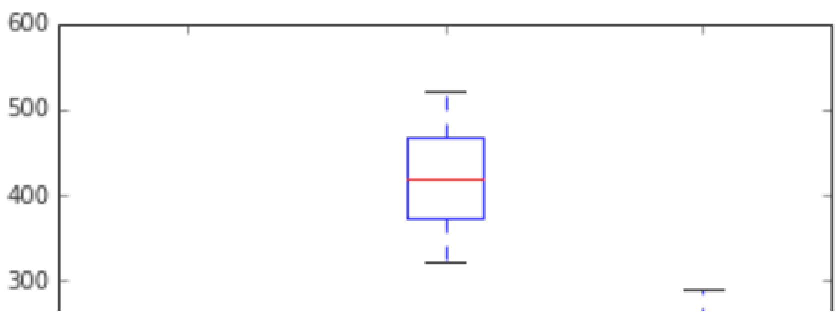
We start off with Count.

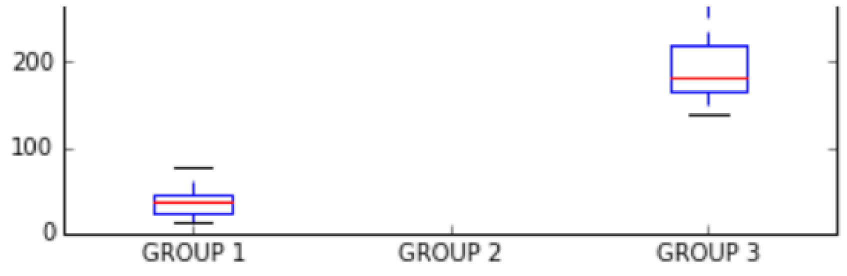
In [5]:

```
plt.cla()
plt.boxplot([[raw_data["COUNT"][raw_data.GROUP==0]],
              [raw_data["COUNT"][raw_data.GROUP==1]] ,
                [raw_data["COUNT"][raw_data.GROUP==2]] ],
            labels=('GROUP 1','GROUP 2','GROUP 3'))
```

Out[5]:

```
{'boxes': [<matplotlib.lines.Line2D at 0x27299349ef0>,
  <matplotlib.lines.Line2D at 0x27299362390>,
  <matplotlib.lines.Line2D at 0x27299375cc0>],
 'caps': [<matplotlib.lines.Line2D at 0x272993589b0>,
  <matplotlib.lines.Line2D at 0x27299358ac8>,
  <matplotlib.lines.Line2D at 0x2729936abe0>,
  <matplotlib.lines.Line2D at 0x2729936eb38>,
  <matplotlib.lines.Line2D at 0x2729937ec50>,
  <matplotlib.lines.Line2D at 0x2729937ed68>],
 'fliers': [<matplotlib.lines.Line2D at 0x2729935db38>,
  <matplotlib.lines.Line2D at 0x27299375ba8>,
  <matplotlib.lines.Line2D at 0x27299385dd8>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x2729935d320>,
  <matplotlib.lines.Line2D at 0x2729936ec50>,
  <matplotlib.lines.Line2D at 0x272993855c0>],
 'whiskers': [<matplotlib.lines.Line2D at 0x27299350940>,
  <matplotlib.lines.Line2D at 0x27299350a58>,
  <matplotlib.lines.Line2D at 0x27299362b00>,
  <matplotlib.lines.Line2D at 0x2729936aac8>,
  <matplotlib.lines.Line2D at 0x27299379be0>,
  <matplotlib.lines.Line2D at 0x27299379cf8>]}
```





In [ ]:

We can see that the count of incidents range differently for different groups.

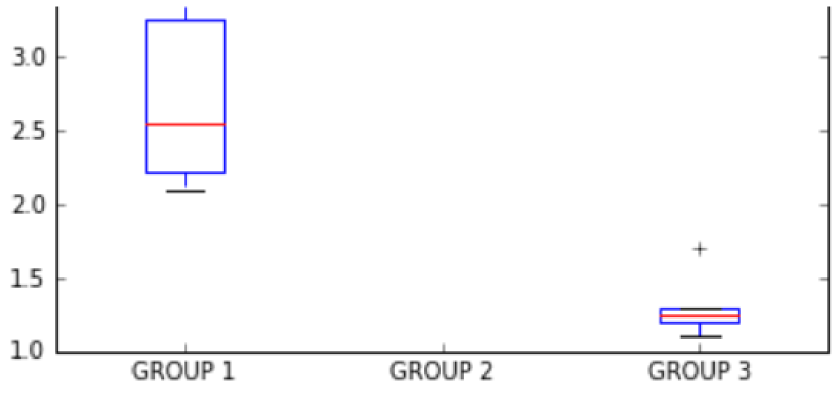
Next we see avg. calls to resolve.

In [6]:

```
#Now for Avg. Calls to resolve
plt.cla()
plt.boxplot([[raw_data["AVG_CALLS_TO_RESOLVE"][raw_data.GROUP==0]],
              [raw_data["AVG_CALLS_TO_RESOLVE"][raw_data.GROUP==1]] ,
                [raw_data["AVG_CALLS_TO_RESOLVE"][raw_data.GROUP==2]] ],
            labels=('GROUP 1','GROUP 2','GROUP 3'))
```

Out[6]:

```
{'boxes': [<matplotlib.lines.Line2D at 0x272993f2ba8>,
  <matplotlib.lines.Line2D at 0x27299405da0>,
  <matplotlib.lines.Line2D at 0x2729941a710>],
 'caps': [<matplotlib.lines.Line2D at 0x272993f8cc0>,
  <matplotlib.lines.Line2D at 0x272993fdc18>,
  <matplotlib.lines.Line2D at 0x2729940fd30>,
  <matplotlib.lines.Line2D at 0x2729940fe48>,
  <matplotlib.lines.Line2D at 0x27299421f60>,
  <matplotlib.lines.Line2D at 0x27299428eb8>],
 'fliers': [<matplotlib.lines.Line2D at 0x27299405c88>,
  <matplotlib.lines.Line2D at 0x27299415eb8>,
  <matplotlib.lines.Line2D at 0x2729942ef28>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x272993fdd30>,
  <matplotlib.lines.Line2D at 0x272994156a0>,
  <matplotlib.lines.Line2D at 0x27299428fd0>],
 'whiskers': [<matplotlib.lines.Line2D at 0x272993f2a90>,
  <matplotlib.lines.Line2D at 0x272993f8ba8>,
  <matplotlib.lines.Line2D at 0x2729940acc0>,
  <matplotlib.lines.Line2D at 0x2729940add8>,
  <matplotlib.lines.Line2D at 0x2729941ae80>,
  <matplotlib.lines.Line2D at 0x27299421e48>]}
```



Group 2 has hardly any time needed to resolve. This points to problems that are simple and straightforward. The business needs to look at these incidents and provide a self-service path (product help, online help) for the customer instead of wasting agent’s time

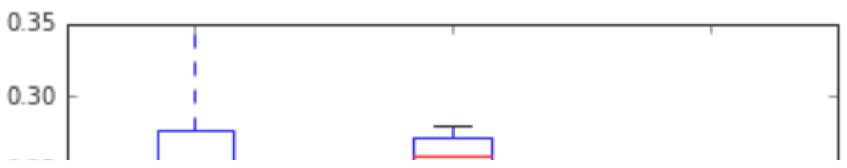
Next, we see Reoccurance Rate.

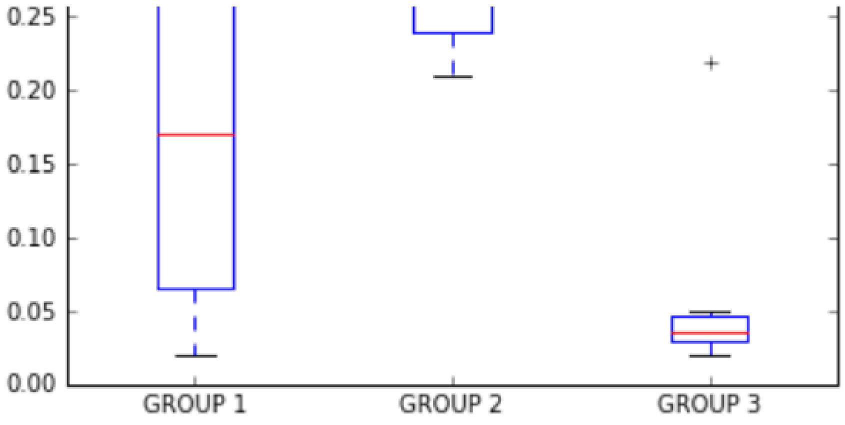
In [7]:

```
plt.cla()
plt.boxplot([[raw_data["REOCCUR_RATE"][raw_data.GROUP==0]],
              [raw_data["REOCCUR_RATE"][raw_data.GROUP==1]] ,
                [raw_data["REOCCUR_RATE"][raw_data.GROUP==2]] ],
            labels=('GROUP 1','GROUP 2','GROUP 3'))
```

Out[7]:

```
{'boxes': [<matplotlib.lines.Line2D at 0x27299485c88>,
  <matplotlib.lines.Line2D at 0x27299499e80>,
  <matplotlib.lines.Line2D at 0x272994b07f0>],
 'caps': [<matplotlib.lines.Line2D at 0x2729948eda0>,
  <matplotlib.lines.Line2D at 0x27299494cf8>,
  <matplotlib.lines.Line2D at 0x272994a4e10>,
  <matplotlib.lines.Line2D at 0x272994a4f28>,
  <matplotlib.lines.Line2D at 0x272994bc780>,
  <matplotlib.lines.Line2D at 0x272994bcf98>],
 'fliers': [<matplotlib.lines.Line2D at 0x27299499d68>,
  <matplotlib.lines.Line2D at 0x272994a8f98>,
  <matplotlib.lines.Line2D at 0x272994c2ef0>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x27299494e10>,
  <matplotlib.lines.Line2D at 0x272994a8780>,
  <matplotlib.lines.Line2D at 0x272994c20f0>],
 'whiskers': [<matplotlib.lines.Line2D at 0x27299485b70>,
  <matplotlib.lines.Line2D at 0x2729948ec88>,
  <matplotlib.lines.Line2D at 0x2729949eda0>,
  <matplotlib.lines.Line2D at 0x2729949eeb8>,
  <matplotlib.lines.Line2D at 0x272994b0f60>,
  <matplotlib.lines.Line2D at 0x272994b6f28>]}
```





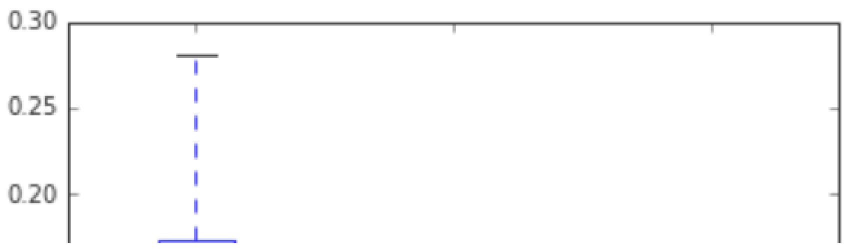
Group 2 has really high reoccurrence rate. This set of incidents need to be analysed to see how the product quality can be improved to prevent these from happening.

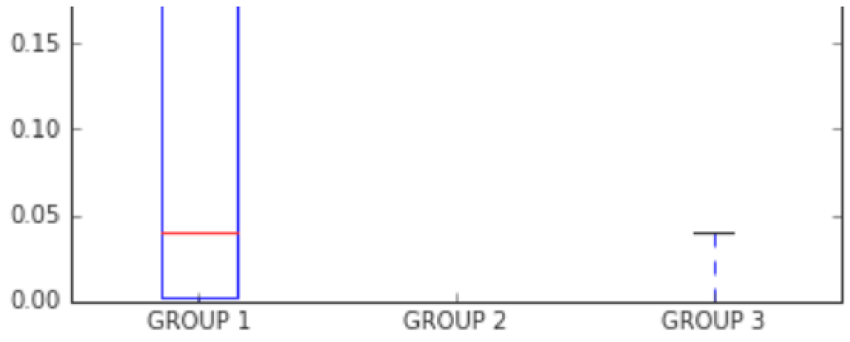
In [8]:

```
plt.cla()
plt.boxplot([[raw_data["REPLACEMENT_RATE"][raw_data.GROUP==0]],
             [raw_data["REPLACEMENT_RATE"][raw_data.GROUP==1]] ,
              [raw_data["REPLACEMENT_RATE"][raw_data.GROUP==2]] ],
            labels=('GROUP 1','GROUP 2','GROUP 3'))
```

Out[8]:

```
{'boxes': [<matplotlib.lines.Line2D at 0x27299526e10>,
  <matplotlib.lines.Line2D at 0x27299538f98>,
  <matplotlib.lines.Line2D at 0x2729954e978>],
 'caps': [<matplotlib.lines.Line2D at 0x2729952ef28>,
  <matplotlib.lines.Line2D at 0x27299532e80>,
  <matplotlib.lines.Line2D at 0x27299544f98>,
  <matplotlib.lines.Line2D at 0x272995497f0>,
  <matplotlib.lines.Line2D at 0x2729955a908>,
  <matplotlib.lines.Line2D at 0x2729955aa20>],
 'fliers': [<matplotlib.lines.Line2D at 0x27299538ef0>,
  <matplotlib.lines.Line2D at 0x2729954e860>,
  <matplotlib.lines.Line2D at 0x2729955fa90>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x27299532f98>,
  <matplotlib.lines.Line2D at 0x27299549908>,
  <matplotlib.lines.Line2D at 0x2729955f278>],
 'whiskers': [<matplotlib.lines.Line2D at 0x27299526cf8>,
  <matplotlib.lines.Line2D at 0x2729952ee10>,
  <matplotlib.lines.Line2D at 0x2729953ff28>,
  <matplotlib.lines.Line2D at 0x27299544780>,
  <matplotlib.lines.Line2D at 0x27299555898>,
  <matplotlib.lines.Line2D at 0x27299555fd0>]}
```





Replacement rates vary widely for Group 1. It does not provide any significant pattern to act upon.

Group 2 has no replacement at all, which is good. As we can see there group level tendencies that are showing now and based on this then we can make some decisions at group level like for example, group 2 calls a lot of times, hardly take any time to resolve, so maybe we can use some self-help. Group 1, on the other hand, occurs a few times, takes a lot of time resolve, higher replacement, and reoccurrence rate. Maybe we need to look at the product to make sure there's nothing wrong with the product or fix something in the product to see how this cannot happen again.

So this is how we can use clustering and predictive analysis to group our problems and analyse them as groups.