Problem definition

What is the problem you're trying to solve, why is it important to solve it?

Before I explain the problem I need to explain a little about CDN.

The goal of a CDN is to serve content to end-users with high availability and high performance. CDNs serve a large fraction of the Internet content today, including web objects (text, graphics and scripts), Downloadable objects (media files, software, documents), applications (e-commerce, portals), live streaming media, n-demand streaming media, and social networks.(Wikipedia)

One of the important processes in CDNs (or generally any load balancing system) is smart load balancing and for smart load balancing it is really important to know how much is the response size (before processing the request). If you know the response size then you can distribute your requests between your servers and return the response in quicker time. If you don't know how much time each request takes, you have to split all of your requests with round robin algorithm, and it is kind of blind balancing.

The Problem is: Predicting file size based on the http Request for making a smarter load balancing system, and keep learning from mistake and new (unseen) data points during prediction.

In this system I want to predict how big is a requested file (imagine each http request is a request for a file) before to send this http request to a server to being process. If we know how much time each http request takes to get processed, we can make a better balance between our servers to serve our requests better. (In a CDN system the time for each request is highly related to the size of the response for that request.)

How does it affect people? How does it affect you?

This problem is e real problem and this is part of my job that I have to solve it, that's why I chose this problem. I will have a big impact on our data centers to serve a request with less time to wait to get processed.

At the end who ever is using the Internet will give a faster response for every type of contents (Movies, Music, Websites). They don't feel any big delay between their request and their response. So it has a really large impact on all the Internet users.

Describe a solution.

How accurate do your results need to be? How quickly does your algorithm need to produce an answer?

The ideal accuracy for this system is 100 %, but as we know this number is not reachable. The accuracy should be at least more than the random guest. Based on my pre test on the data set we have to reach more than 85% accuracy for this system, other wise we cannot see tangible improvement or real impact on the load balancing system.

The speed is one of my biggest concerns on this system, because this application should be capable to predict more than 1,000,000 requests per second, and also my second problem and

concern as I mention before is about online learning, this application has to be online and learn from upcoming requests. It has to be able to keep its self up-to-date during the time.

Analyze The Problem

What is the size of your dataset?

My data set is so big, almost as big as 10% of all http requests on the Internet from 5 years ago. For this practice I will use only 500,000 unique data points but later I will mention about how result and other stuff are on the real data.

How many features are present?

In the real data set there are so many features, almost every thing from an http request, but I only use five of them.

Host: Extracted from the URL (**www.udacity.com**/Machine_learning/images/demo.jpg), I hashed my host data, because this is private data and I could not make it public, but it does not have any impact on the final result.

File-Extension: Extracted from the URL (www.udacity.com/Machine_learning/images/demo.**jpg**)

Content-Type: Extracted from the http header (`content-type: image/jpg`)

Is-Mobile: I extracted this feature from the AGENT property in the http request header. This feature only says whether this request is coming from a mobile devices (i.e. smartphone, tablets and etc.) or not.

I could not use all of my features, because my model has to be so small and fast.

Which features seem most promising?

None of my features are so good lonely, but a combination of them is working really well. If I want to order them based on their amount of information that they are injecting to the model I would say File-Extension, Host, Content-Type and Is-Mobile.

Are there any categorical variables that may need to be converted?

All of my features are categorical but with one different, there is now end for this categories. They are like words in a text. I tried to use text-processing (NLP) techniques to categorize them into a certain number of categories. But I could not because there is a big difference between NLP and this problem.

In most of NLP tools such as, Dependency Grammar, Semantic Role Labeling (SRL), Information Extraction (IE) or Name Entity Recognition (NRE), they don't work with words directly, most of them use POS tagger to label words and then process those words based on their Part of Speech Tag.

In this system there is now way to combine or categorize some Hosts, file extension, content type to one or many meaning full category.

That's why I cannot work with some techniques like dummy variable.

Which ones would you expect to perform well?

As I mentioned before I don't expect one of them lonely performs well or better than the rest because as time goes on we are getting different type of values for our features, but if I have to choose one of them I would say Extension or Content-Type.

How easy is it to convert the available data into a suitable form?

It was really hard; if I am not mistaken I spent most of my time to crate my data set and doing some data cleaning. I spent a lot of time to remove or replace null value with mean or the biggest category. Unfortunately my data is not so clean and categorize, that's why I spent a lot of time on data cleaning.

Implement a solution

What pre-processing operations do you have to carry out on the features? (e.g. scaling, normalization, selection, transformation)

As I mentioned earlier I extracted all of my features from a bag of uncategorized dataset. For example there wasn't any **Is-mobile** feature and I had to process "Agent" property in the http request to realize where this request is coming from (i.e. OS, Browser, OS version and etc.)

Extracting "Extension" also was a really hard task for me, because most of the URLs don't not contain extension, I had to use their "content-type" to realize what is the extension or some of the URLs are using query string to mention about which file they are looking for, I had to process all query string to realize that, sometimes a query string is encrypted and there is no clear way to extract information from that.

Another thing I encountered was Encoding.

One another thing that I have done was changing my Target-label from a continues value to a categorical value, because I could not solve this problem with regression algorithm, so I changed my solution from regression to classifier, but the next question was how should be the categories

and how big each category should be. Eventually after talking to my manager we came up with a solution, we will cluster our target data with "Log_2". It means we don't use the pure file size for prediction and we will apply ($Log_2^{(Size)}$) on file size and use it as a Target-label.

So my labels become like below chart with two exceptions. I joined label 0 to 19 together and aleo all labels above 28 together as 28.

0-19	20	21	22	23	24	25	26	27	28	28>=

This chart shows the range for each class:

0 KB	512KB	1 MB	2 MB	4MB	8 MB	16 MB	32 MB	64 MB	128MB	256MB
512KB	1 MB	2 MB	4 MB	8 MB	16MB	32 MB	64 MB	128MB	256MB	~

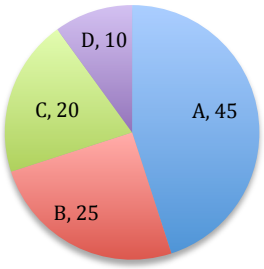
What is my solution and which Machine-learning Algorithm I want to use?

Based on what I have explained until now, my algorithm has to be so **fast**, it has to be able to **up-to-date its self** with new data and simultaneously **forget its past information** and using small amount of **memory** (When my manager came to me and explained what he wants and he talked about the metrics, I was thinking this is impossible ☹).

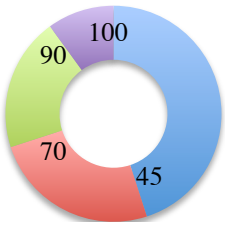
I started to work with Perceptron model to take full advantage of its online capability, and simultaneously I was thinking about NN (Neural Network), Deep learning and reinforcement learning, but after spending a lot of time on implementing those Ideas I realized NN is really slow for my system despite getting really good result.

I started to work with decision tree (ID3). It was really fast (which was my most important criteria) and it would have let me update the Model (Tree Structure) whiteout creating a new model from scratch and it was compatible with text data and I would not have to change text data to numerical equivalent.

I also should mention that, my biggest problem with decision tree was over-fitting. I found some solutions for this problem but I am not still sure this solution can work under all circumstances. For example at then final node I have 45% from class A, 25% class B, 20% class C and 10 & class D.



1) I could choose the biggest class, (this example A with 45%), I am doing this now.



2) Or creating a random number between 0 to 100 and choosing that class which contains the random number.

Using the metric(s) you defined earlier, measure your current performance. Is it close to what you expected?

I had for metrics; fist it has to be fast enough to predicts about 1,000,000 requests per second, Decision Tree is relatively fast and its time complexity (if we consider h is height) is around O(h) .

My second and third metrics are: model should be able to **up-to-date its self** with new data and simultaneously **forget the past information.** This two metrics are highly depending on how you implement your tree. I am keeping my Tree up-to-date by adding new nodes to three when data points are new and we have not seen it before. If I get a data point which is not exists in the tree I will predict it with biggest class in the data set. I'm my data set the biggest class is 19 and then I will process the request, after processing the request I have access to the real class, at this time I can check my prediction with actual value. The gap between prediction and getting actual value is less than 100 milliseconds. After getting Actual value now I can insert new data-point to tree and for next time I don't need to predict this type of data-point with default value. Also in real implementation, which is working on staging environment right now, we wait to see one request N times and after N times we will decide what should be the label for this type of new data-points and then we add it to tree.

One another solution for **forgetting the past information and keeps the model up-to-date** is whenever we predict a new data-point incorrectly we can go to the model and change the wait of each class on that specific node. (I took this idea from Perceptron). And if we don't go to one node for a long time we can remove that node to save memory.

My last metric was Memory, fortunately decision tree use less memory in comparison with other algorithm like NN. At the end I think I could reach most of my goal in this problem

You will see some output like this if you run my code. I calculated F1 separately for each class. And Overall Precision was around 89% percent. But if I run this application with huge amount of data I will get 93% F1 Score. Which is really good.

```
Train Size:  160000
Test Size :  40000
Start Building Tree at:  2016-02-25 01:56:31.959844
Number of Fist level Nodes:  2184
Tree Size 196888
Start Prediction Tree at:  2016-02-25 01:56:40.750718
----------------------------Report--------------------------------
------------------------------------------------------------------


H            0|      1|      2|      3|      4|      5|      6|      7|      8|      9|      Recall
0        33843      56     109      13       0       3       0       1       0       0       - 99.47
1          596      93     274      18       1       1       0       0       0       3       - 9.43
2          256      52    3072      79       4       6       2       0       0       1       - 88.48
3           91      27     283     203       8       7      10       2       0       1       - 32.12
4           44       1     120      53      33      20      18       9       0       7       - 10.82
5           13       0       4       3      20      38      42       5       6       3       - 28.36
6           12       0       3       1       9      11      68       7      31       4       - 46.58
7            7       1       0       2       2       8      45       7      40       3       - 6.09
8            9       0       0       1       1       2      18       6      61       5       - 59.22
9            7       0       0       0       2       3       8       4      27      31       - 37.8
Precision: 97.03|  40.43|  79.48|  54.42|  41.25|  38.38|  32.23|  17.07|  36.97|  53.45|


------------------------------------------------------------------
```