Problem definition

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

My problem is about prediction HTTP-Request response size before process a request. I want to design a prediction system based on HTTP-Request header's fields. I will explain my problem with an **example**.

What's in an HTTP request?

Whenever your web browser fetches a file (a page, a picture, etc) from a web server, it does so using *HTTP* - that's "Hypertext Transfer Protocol". HTTP is a request/response protocol, which means your computer sends a request for some file (e.g. "Get me the file 'home.html'"), and the web server sends back a response ("Here's the file", followed by the file itself).

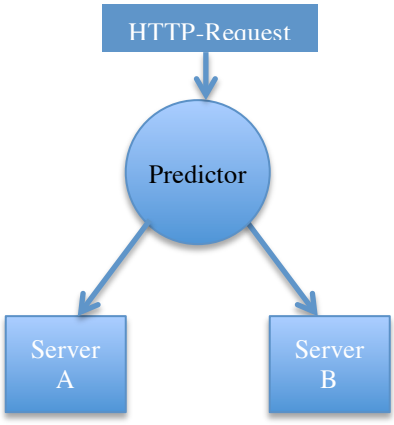
Example:

- It is an HTTP-Request for Udacity logo file:

```
GET /udacity-content/rebrand/svg/logomark.svg HTTP/1.1
Host: s3-us-west-1.amazonaws.com
Connection: keep-alive
Cache-Control: max-age=0
Accept: image/webp,image/*,*/*;q=0.8
If-None-Match: "5160692df65598202eec0f02e3f76e1d"
If-Modified-Since: Fri, 05 Feb 2016 19:36:30 GMT
User-Agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWebKit/537.36 (KHTML, like
Gecko)
Referer: https://www.udacity.com/me
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,fa;q=0.6,es;q=0.4 2.
```

- When you send this request to the Udacity-Server, Udacity-Server starts to process this request and returns a response, this response is a file. If you look at the HTTP-Request, you can realize this file is logomark.svg.

- Now for simplicity Imagine I have 2 serves. When I receive an HTTP-Request (in server side) I want to categorize my HTTP-Requests between these two servers based on the response size (at this example response is a file logomark.svg). This figure illustrates the problem.



- In Predictor we have to make an important decision (predict), at this level we have to predict how much is response size ($_{\text{the response is a file}}$) for this HTTP-Request. If response size is more than 1MB send the HTTP-Request to Server A if it is more than 1MB send it to Server B to for processing.
- Most probably you want to ask why prediction? When you send an HTTP-Request you don't mention about response size, it means there is no field in HTTP-Request header field to mention about response size, and when there is no response size I have to predict how big the response is.

- When we realize how big is response size? After we process the HTTP-Request in the server.

Hopefully you have a good understanding of the problem and you know why I want to predict response size in the Predictor. Because I want to categorize or split my request to two types, HTTP-Request with response size less than 1MB and HTTP-Request with response size more than 1MB.

Now in the real problem I have more than two classes (1MB and more than 1MB). I will explain how many class do I need, future in this report.

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

One of the biggest challenges for smart load balancing in Content Delivery Network (CDN) system is about response size. 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. (For more information about load balancing and CDN check Wikipedia)

All of the content on the CDN are static, it means they only serve static content or files; there is no process in the CDN systems.

Now let's make our example close to the real problem. We had two servers (A and B), They can process files with response size less than 1MB in 1 second and HTTP-Request with response size more than 1MB in 5 seconds.

Now if I receive a list of HTTP-Requests, we can split them between two servers and process all HTTP-Requests with small response faster than HTTP-Request with large response size, Exactly like Express line in Grocery Store. You are serving all HTTP-Request with small response size together and we don't let any HTTP-Request with large response size come to this server because take more time process and make or process speed slow.

Analyze The Problem

What is the size of your dataset?

My data set is almost as big as 10% of all HTTP-Requests on the Internet from 5 years ago. We have access to a large set off HTTP-Requests plus their HTTP-Responses header. I cannot use this data set because it is too big. For this practice I will use only 500,000 unique Http-Requests plus their HTTP-Responses.

How many features are present?

All I have as a data set is a list off HTTP-Request and HTTP-Response headers fields, to check list of HTTP headers files please check this link. (https://en.wikipedia.org/wiki/List_of_HTTP_header_fields). You have to consider most of these HTTP header fields are optional and only few of them always available on each HTTP-Request.

I am using only four of them. This table shows my sample data set.

Features				Target
HOST	File Extension	Accept	Is-Mobile	Response Size
www.udacity.com	jpg	image/webp,image/*,*/*;q=0.8	0	19
www.netflix.com	rt	movie/webp	1	23

I choose these four feature because, they are always available in the HTTP-Request and they are not optional (it means I always have these features and none of them miss). Second, because based on my calculation with "information gain" and "entropy" formula, these four are bringing the most information about the target into the model.

For other features "Information gain" and "entropy" aren't zero, but they are making the model bigger (size) and slower to make a prediction, but just adding some information plus a lot of noise, and they are not always available in the HTTP-Request.

Which features seem most promising?

Host: Extracted from the URL (**www.udacity.com**/Machine_learning/images/demo.jpg

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

Accept: Extracted from the http-Request header (Accept: text/html))

Is-Mobile: I extracted this feature and made a lot of changes on the USER-AGENT field in the http-request header to make this feature as a Boolean value, this feature only says whether this request is coming from a mobile devices (i.e. smartphone, tablets and etc.) or not.

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

All of my features are categorical, but there is no end for these 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.

For more information:
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 and Accept 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?

Based on the Information Gain and entropy for all my four features. Extension and Host perform well.

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.

Describe a solution.

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

Ideally accuracy must be 100%, what does it mean 100% accuracy? It means we classify all Http-Request with response size less than 1MB as A and all Http-Request with response size more than 1MB as B.

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 a tangible improvement or real impact on our performance.

My result needs to be at least 85% accrue.

The speed is one of my biggest concerns on this system, because this application (Predictor) should be capable to predict more than 1,000,000 requests per second, and also my second problem and concern is about online learning, this application has to be online and learn from upcoming requests. It has to be able to keep the model up-to-date during the time.

Why we have to keep the model up-to-date during the time? Because some Http-Request response size (file size) are variable and they change during the time, it means if for the first time a Http-Request response size be less than 1MB there is no guaranty that second time should be less than 1MB, because file owner can change his file, and changing the file, change its size.

Why I think this problem is online? Because after we process an Http-Request in server A or B, in the way back in response header, we can read response size and we can check our prediction with actual value. For example if I predict it as A but when I process it I realize it is B, then I can update my model to not make same mistake again, next time predictor will predict same Http-Request as a B Request.

When you can see the actual value for your prediction, it calls online learning, because you able to compare your prediction with actual value and improve your system based on the feedback.

I will talk more about my implementation and solution under *what is my solution and which Machine-learning Algorithm I want to use? Section.*

What are the labels?

In our simple example we only had two classes (A and B) but in the real problem we have more than two classes. In this system I don't know how big is the biggest file but I know the smallest one can be 1 byte.

We can use regression for this problem because response size has continues value and there is no end, but I want to change my problem to a classification problem. My solution to change this problem from continues value to categorical value is applying ($Log_2^{(Size)}$) on the response size. So my target label become like this.

#	Response Size **From** (Byte)	Response Size **To** (Byte)	Label
1	0	524288	19
2	524289	1048576	20
3	1048577	2097152	21
4	2097153	4194304	22
5	4194305	8388608	23

#	Response Size **From** (Byte)	Response Size **To** (Byte)	Label
6	8388609	16777216	24
7	16777217	33554432	25
8	33554431	67108864	26
9	134217729	134217728	27
10	268435457	268435456 and bigger	28

I merged label 0 to 19 together and also all labels above 28.This table says if http request response size is (equal or grater than 0) and (equal or less than 524288), the best label for this Http-Request is 19. Why 19? Because $Log_2^{(524288)} = 19$.

Now we know this problem is a classifier, with 10 target labels (19 to 28) and each one represents a range of Http-Request response size.

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 fields to realize where this request is coming from (i.e. OS, Browser, OS version and etc.)

Extracting the "Extension" is a difficult task, because most of the URLs don't contain extension, I had to use their "content-type" in response or ACCEPT in the HTTP-Request to realize what is the extension. 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.

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 I clustered my target data with "Log_2".

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**.

I started to work with Perceptron model to take full advantage of its online capability, and simultaneously I was thinking about other Neural Network, Deep learning and reinforcement learning techniques, but after spending a lot of time to implement those Ideas, I realized Neural Network is really slow for my system despite getting really good result. Perceptron guarantees to converge, if data is linearly separable. But my data is not linearly separable.

I continued my research 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.

My model has to be able to **up-to-date its self** with new data point (unseen data point). This metric is highly depending on how I implement my Decision Tree(ID3).

I am keeping my Tree up-to-date by adding new data point (unseen data point) to the Model (TREE). If I get a data point which is not exists in the tree, I will predict it with the biggest class (or default class which is 19). After processing the same request we have access to the real class (actual value) and we can compare the prediction with actual value. If prediction was correct we don't need to do any thing even if data point is new. If prediction was Incorrect it means the model is not working well and we need add this data point to the model (Only when data point is new)

Similar Idea to keep our model Up-to-date with changes and new data point is to 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 will add it to the tree.

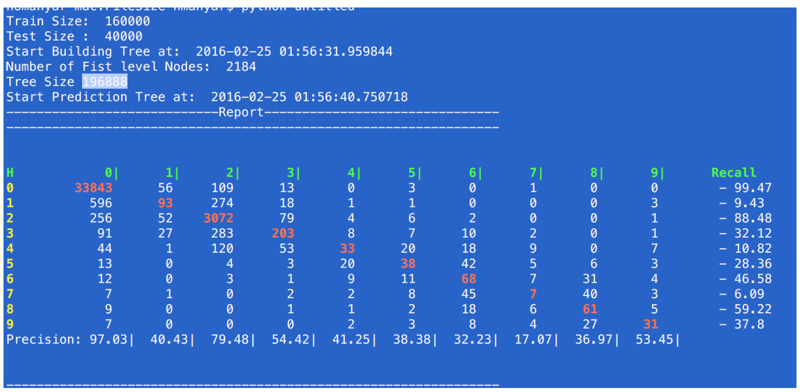
I decided to use Decision Tree (ID3) because it supports most of my criteria.

I am using Decision Tree but I am aware of its weakness, for example over-fitting is one of them, but despite this problem, it is still the best algorithm for Reponses size prediction.

This is my core implementation for creating and updating tree. I am using nested list(dictionary) and recursive technique.

```python
def UpdateTree(self,node,features):
      if features != None:
         if features[0] in node:
            if len(features)>1:
               self.UpdateTree(node[features[0]],features[1:])
            else:
                 node[features[0]]+=1
         else:
            if len(features)==1:
               if features[0] in node:
                  node[features[0]]+=1
               else:
                  node[features[0]]=1
            else:
               node[features[0]] = defaultdict(dict)
              self.UpdateTree(node[features[0]],features[1:])
```

You will see some output like this if you run my code(FS.py). 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. And I can keep this score during running this application, because I check my prediction with actual values after prediction process.



```
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|	

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

This image shows, precision and Recall for each label (19-28)