PaytmLabs Challenge

Basic Processing

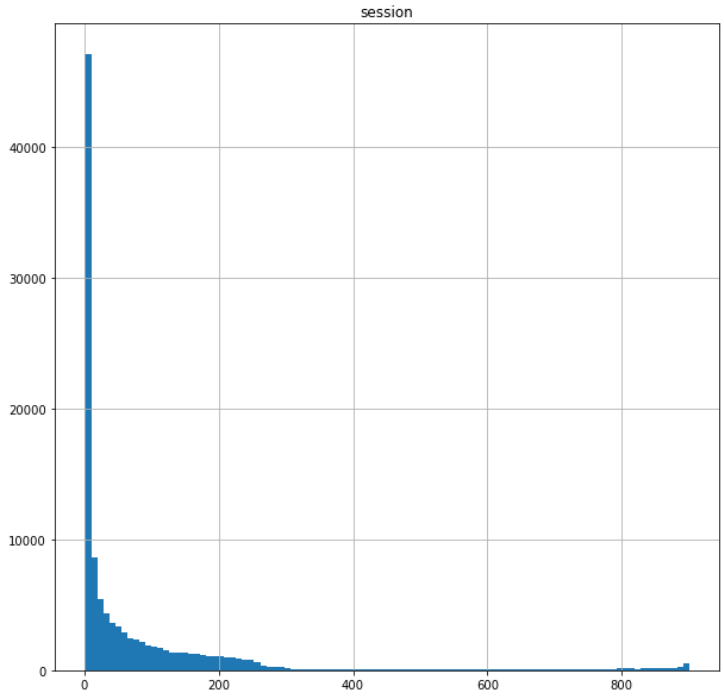
1. Average session time and unique URL visits per session

Take 15 minutes as a fixed session time window, here is the statistics: (session_analysis.py)

	session(second)	url_count
count	113370.000	113370.000
mean	89.792273	8.193861
std	168.489399	54.157797
min	1.000000	1.000000
25%	1.000000	2.000000
50%	20.000000	3.000000
75%	103.000000	7.000000
max	900.000000	8016.000000

Average session length = 90 seconds,
Average unique URL visits per session = 8.2, check the visits count for each session by "ip_session.data"

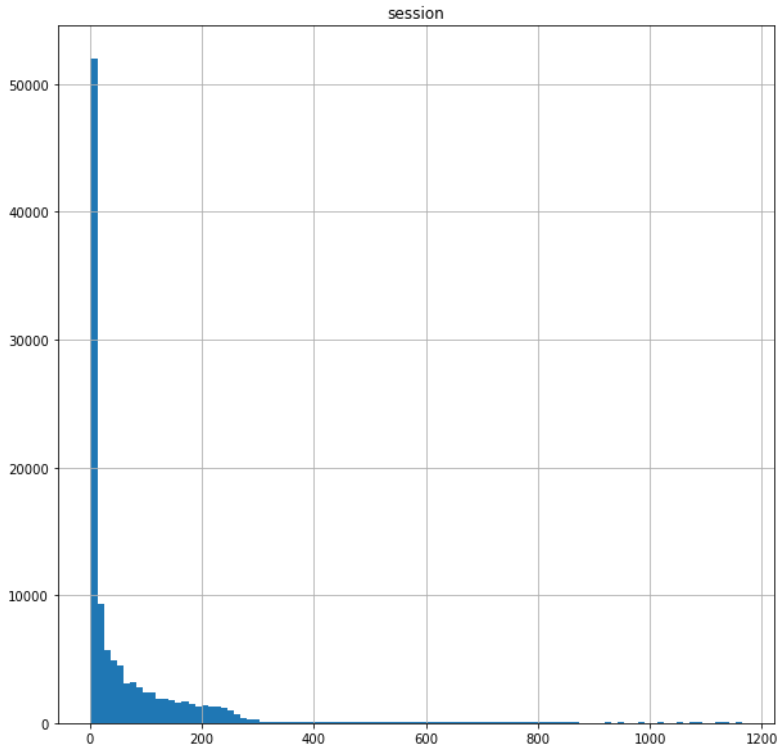
Here is the histogram graph for session length, it roughly follows power law distribution:



Another choice: If the interval between the two adjacent is greater than 10 minutes, a new session starts. Here is the statistic:

	session	url_count
count	114229.000	114229.000
mean	76.063346	8.120180
std	142.018872	56.225612
min	1.000000	1.000000
25%	1.000000	2.000000
50%	18.000000	3.000000
75%	95.000000	6.000000
max	1164.000	8016.000

Here is the histogram graph for session length, it roughly follows power law distribution:



They are quite similar, result of the second method has smaller std.

Additional questions

1. Predict the expected load in the next minute

1.1 Density estimation

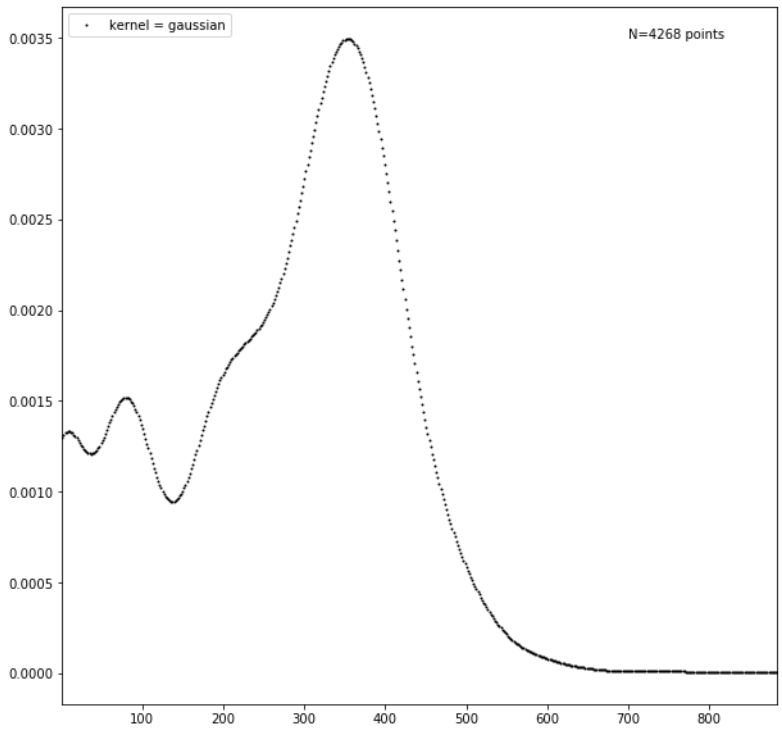
If we can get the density estimation of the data, we can sample from the density as a effective prediction for future.
There are two kinds of methods for density estimation, parametric approach and non-parametric appoach. Generally, non-parametric approach named ***kernel density estimation*** has better performance, because it does not make any assumption about the data. I choose this method to fit the data.

If we choose minute as the unit time, the dataset will be very small and sparse, so I use second as time unit. Here is the histogram of the request frequency per second.

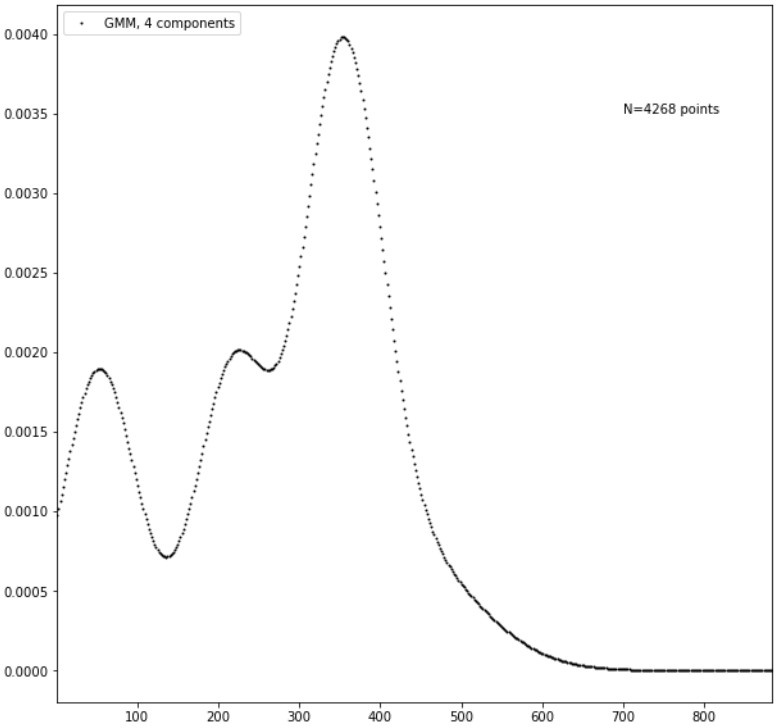
Here is the statistics of the request counts per second:

```
count     4268.000000
mean       271.414480
std        142.966698
min          1.000000
25%        175.000000
50%        304.000000
75%        374.000000
max        884.000000
```

Here is the result of kde using Gaussian kernel, (predict_load\kde.py)



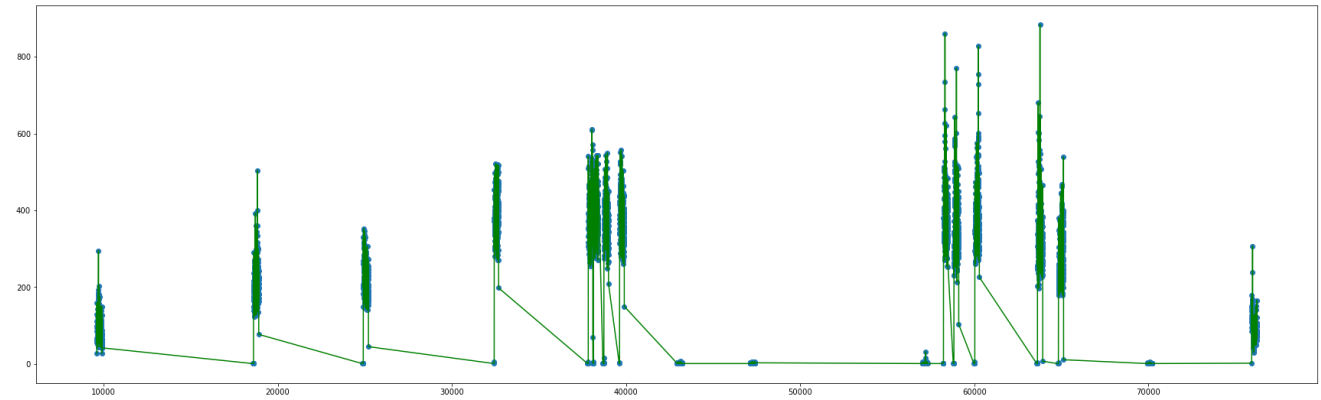
Here is the result for the GMM

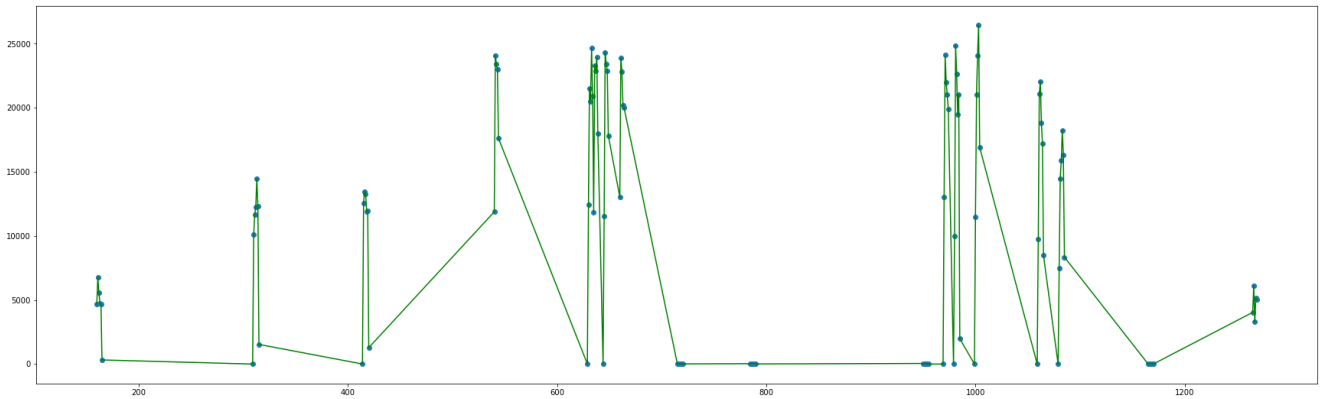


They are rather similar. We can do sampling from them to make predictions. See kde.py

1.2 Time series curve

Second as time unit and Minute as time unit





It is clear that it is not good choice to make sample based on these time series curve, there are too many gaps.

1.3 Regression and classfication

The source code is predict_load\cl_reg.py

1.3.1 Regresson

After analysing the all fields in a single request, it seems that only the time field has correlations with the users' behaviors. For example, it is obviously that there should be more requests in the afternoon than requests in early morning.

But due to the limited amounts of data, the correlation is not clear. Here is the correlation matrix (respect to count):

```
hour      0.070500
minute   -0.055574
second    0.004524
count     1.000000
```

It is clear tha only the original features are not enough, we need more transformations.

We can use kernel models to expand the feature space, such as SVR.
mean squared error of test dataset: 68.4

We also can employ models that can directly learn features tranformations from data, such as GBM, Randforst, ANN
mean squared error of test dataset using GBM: 67.7

1.3.2 Classification

Can we discretize the request counts into some intervals, and we assign a label for each interval.

1.4 Sequential Learning

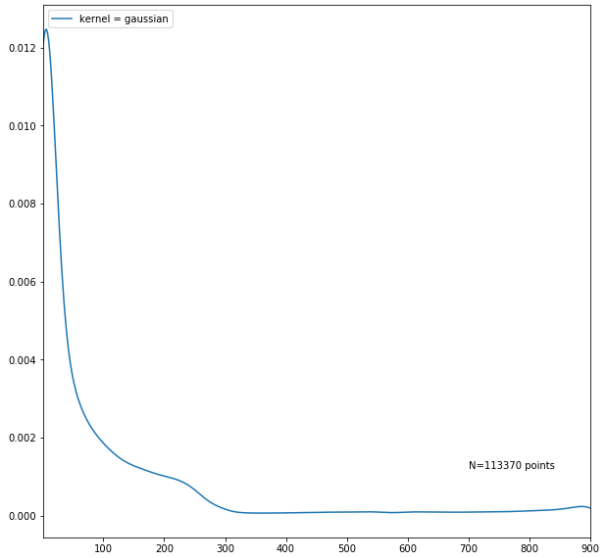
Maybe better if there are more data, such RNN.

2. Predict the session length for a given IP

Here is the statistics of the session legnth for IPs (predict_ip_info\ip_session_kde.py):

```
count   113370.000
mean        89.792273
std        168.489399
min          1.000000
25%          1.000000
50%         20.000000
75%        103.000000
max        900.000000
```

Use kernel density estimation estimate the probablity distribution of the session length, here is a showing graph.



We can sample values from this distribution as predictions.

3. Predict the number of unique URL visits by a given IP

Here is the statistics of the numer of unique URL visits by IPs (predict_ip_info\ip_url_kde.py):

```
count     90544.000000
mean         10.259520
std         159.893034
min           1.000000
25%           2.000000
50%           3.000000
75%           8.000000
max       32174.000000
```

Use kernel density estimation estimate probablity distribution of the unqiue URL visits by a given IP: