Web Log Analytics

The data is large enough for processing on the laptop, so a sample was taken for processing the analytics and training models.

Due to time constraints, not all the options are explored in training the model, but the other possible options are discussed in this document and also there is opportunity to improving the model further.

The code at few places is repetitive and can be implemented as callable function, but not implemented to avoid retesting of the function.

User Session

The user session is defined as any activity from a given client's IP, for a maximum of 15 minutes. If the activity from same IP exceeds 15 minutes, then it is considered as another session incrementally for every 15 minutes. Here the session is considered closed after the response is received by the client, so the response times are also added to the session time.

The session tine is calculated as below.

(LRT – IRT) + LBRT + LSRT + LCRT

LRT = The last request time from the client within the 15 minutes window

IRT = The initial request time from the client or the request time for new session after 15 minutes

LBRT = The load balancer response time for the last request of that session

LSRT = The backend server response time for the last request of that session

LCRT = The time for response received by client for the last request of that session.

The other parameters like status codes for load balancer and backend server are not considered for this exercise, while computing session time, but it is debatable.

IP Addresses

It is understood that one user is not associated with one IP alone, as the same user may access the website from several devices (computer, mobile or tablet) and accessing through several IP addresses, but here every device will have its own session, even though operated by the same user.

The IP addresses do not remain same all the time, as they are dynamic. The data trained on a set of IP addresses may not be available tomorrow, so the trained model may not generalize well to the new IP addresses coming in the future.

IP addresses are reserved for various organizations and government and some are allocated for internet service providers. The initial assumption is that IP addresses are associated with geographical location. For example some IP address patterns may be originating from certain country or a certain location within a country. In that case there will be certain internet usage patterns that impact the session length (time spent) and number of page visits on the public web

servers. As an example, urban population may spend more time than rural population, because of the accessibility and speed of internet connection. I tried to flatten the IP addresses to a uniform fixed length with the below formula, based on the above assumption.

IP are in the form of 1.2.3.4 and each of the integers is separated by dot and can vary from 0 to 255. To standardise the length of each integers to 3 fixed digits, a number 100 is added to each integer in the IP and concatenated them to form 12 (4 X 3) fixed digit number. For example, 1.2.3.4 will be transformed as (1+100)||(2+100)||(3+100)||(4+100), which translates into 101102103104. The IP address can be reconstructed back, when needed, by subtracting 100 from 3 fixed digits and separating them with dot. This type of transformation requires good processing power.

There are more options to normalize IP addresses, as discussed below.

1. Transform of IP address using one-hot-encoding, increases the feature space and results in multi dimensionality, so not recommended.
2. If each of the four integers in the IP address relates to some geographical location, then we can generalize the model to some extent by stripping off first or last 3 digits from the IP address to identify it with similar locations.
3. The other forms of data conversion for IP address include 32bit binary, but the binary form as number yields the same value for all IP addresses, so this is not suitable for classification.
4. Separate all four integers from the IP address and consider them as four individual features.

Feature Selection

Apart from the features discussed above, there are other important features that may impact the user behaviour, but we really need to process large amounts of data to observe such behaviour. The below features are not tried in my model, as I am running the model on a small sample.

1. **Session start time**: the start time may influence the user on how long they are spending on the site and how many pages they are visiting. A session started by a user in the morning may spend less time and visit few pages. The same user who started the session in the night may spend more time and visit more pages.
2. **Response time**: Users tend to spend more time and visit more pages, when the response time from the servers is good, otherwise they may abruptly end the session.

Feature Extraction

1. If we are considering the four integers in the IP address as four separate features, then it is a good candidate for feature extraction and dimensionality reduction.

Data Cleaning and Transformation

1. The millisecond from the request time is stripped for predicting the next time period load, to keep the time intervals at minimum.
2. Port number from the IP address is stripped, as this is redundant for the given requirement.
3. The data is a web log generated by the system, so it is not expected to have outliers, but the outliers occurred in a computed column for number of page visits and the extreme outliers are removed where needed.

4. The GET and POST keywords from the URL are removed, to make the unique URLs look cleaner.
5. The log transformation is applied to make data normally distributed.
6. The numeric data is normalized with MinMaxScaler, which applies the formula (given value – min value) / (max value – min value), to bring the data range between 0 and 1.

Observations

1. Even though the data contains web activity for one day, but there is no continuity in the data, as the time series is not continuous.
2. Here the data is in time series, so the data shouldn't be shuffled or randomized from the dataset, instead the sampling is done as a continuous block from the original set, whenever needed.
3. It seems there are some clients, which have opened the website once and then closed, without accessing any other pages. In that case session time will become zero, so the response times are also added in the session time, as end of the session.
4. Client request size is zero for most of the Get requests.

Other Potential Stats

1. Identify any bottlenecks in load balancer to backend server, backend server to load balancer and load balancer to client. Also pinpoint specific pages causing those bottlenecks.
2. Identify the number of failures at load balancer and the backend server.
3. Find any correlation between the size of request and processing times. Similarly the size of response and response processing times.
4. Stats can also be generated using the product info in the URL.

Software Challenges

1. Python has no SQL type of language, so querying is not very efficient. Though Pandas provides data frames for similar functionality, but they are not comparable to SQL.

Personal Challenges

1. New to the time series data and handling IP addresses and also the related python libraries like Keras.
2. The data is large enough for my old laptop for mathematical computing, so data sampling is done for quick processing.