Question 1

1) To access customer behavior from a large amount of data it would be convenient to divide the browsing events into subgroups. Meaning we need to perform a clustering task.
    1. Data analysis following by decision which features we should use
    2. Work with text features: cleaning the text from irrelevant symbols and getting a list of words. The most frequent words are going to be used for training a word embedding model.
    3. Combine the words numerical vectors into vectors that represent events' text data.
    4. Clustering of the browsing events.

2)
    1. For data analysis step I would visualize the data into histograms to access the distribution of this feature. It will give me a direction on feature selection and method of scaling for this feature. But some features I may discard later due to clustering algorithm performance.
    2. I would use a word embedding model such as FastText or Word2Vec. To emphasize connection in a specific event I would combine the information on 'title' and 'url' into one sentence for word embedding purposes.
    3. To create a vector out of a sentence I would use weighted mean to give frequent words less influence. I would try to combine all 'url' features in one or try to separate them into subclasses such as 'domain' and 'path'. The final decision will be done by performance of the clustering models.
    4. For clustering I would train couple of different models of different types of clustering st. centroid based - k-means and MeanShit, probabilistic – Gaussian mixture and density based - DBSCAN. And chose the model according to the scores and clusters balance. For k-means I am using elbow rule of the inertia together with clustering scores.
3) Both models are fast and efficient way to embed words.
   The k-means algorithm is frequently used for customer segmentation.
   DBSCAN works well for large number of samples and medium amount of clusters, also it finds more complex shapes of clusters – using density of data points.
4) For each browsing event I am going to produce a number from 0 to k-1 (k-number of clusters). Different number for k categories I decided to distinguish between.

Question 2

I decided to use two clustering scores: Silhouette score and Davies-Bouldin index. In addition to it, I accessed balance of the clusters, meaning how many samples there are in one cluster. On top of it I looked at the samples of each predicted cluster and made sure that the clustering makes sense.

For the solution I have found it is 0.33 and 0.98 respectively. I was able to get better numbers, but the clusters were heavily unbalanced where one cluster contained most of the samples. To eliminate it I used logarithmic function for weighed average in calculation of sentence embedding in step 3.

I was able to get better numbers for using time features – time and day of the week. But looking at the events in the same cluster I didn't find any resemblance of events except that they were

done at the same time. So, I decided to eliminate these features and use only text. Also, I decided to use all the words in an event together since it gave better clustering performance.

If 10 000 000 more events were available, I think that the embedding algorithm would give much better embedding and hence the cluster model would have better performance. In this case I might want to change the number of vector size for embedding. Now I used vector size 10 since I didn't see considerable improvement when using more.

Advantages:

1) Easy to understand
2) Easy to implement with sklearn
3) Gives reasonable clustering

Disadvantages:

1) Doesn't pay attention to the language (I wasn't able to find a quick solution that gave reasonable language detection)
2) Outlier events were not deleted, this noise may confuse the clustering model that doesn't detect anomalies (such as k-means that was chosen)
3) doesnt use any regularization to avoid noise events.

To improve accuracy of the model better cleaning of text features would help. More events and more complex models (not implemented in sklearn) such as regularized algorithms or manifold learning with ability to predict may improve the clustering.

To improve run time on training stage I was using k-means++ that speeds up the algorithm by finding starting position smartly. For very large dataset we can use Mini batch k-means version to speed up the training. The word2vec algorithm is a neural network, so to speed it up parallel computing can be used (GPU).

To improve cloud cost we can train the word2vec model using some events, saving the model these events can be deleted completely. And for the future events we will just add new words and retrain.