Lab 01 - Techniques for Handling Large Datasets in Classification

In this scenario we will explore several techniques for handling large datasets in classification tasks. We will compare the performance of different approaches, including:

- processing the full dataset in parts
- sampling techniques
- summarization techniques
- quantization techniques

1. The Dataset

During this lab, we will use the NYC Yellow Taxi Trip Data from Kaggle. In fact, this is a subset of the much larger and continuously updated dataset available from the NYC Taxi & Limousine Commission (TLC). The original dataset is evolving over time, with respect to its schema, therefore to avoid issues, we will use a preprocessed version from a Kaggle user.

Get familiar with the dataset page, read the description, check available columns, as well as their types and meaning. Download all available CSV files - use any means you prefer (Kaggle API, kagglehub package, direct download using browser, etc.).

In [1]:
```
# write your code here
```

2. Investigate the Dataset

Get familiar with the dataset. Check the sizeof the files, the number of rows and columns, the actual data types, etc. You can use a tool of your choice (pandas, dask, pyspark, pyarrow, etc.) - the one you can install and run locally or an online tool it supports large files. Is it possible to load the entire dataset in memory? Even if it is possible in your case, try to think about scenarios where there is not enough memory available. Consequently, you should design your solutions so that they do not require loading the entire dataset at once.

In [2]:
```
# write your code here
```

3. Preprocess the Dataset

Clean and preprocess the dataset, e.g., handle missing values, engineer new features, perform discretization, etc. You may approach this task in an iterative manner, i.e., you may

implement a first version and then back to it later, if needed.

Classification task

There is no predefined classification task for this dataset. You should define your own. Prepare your solution introducing some problem parameters so that they can be easily changed in the future. For example:

- You may want to predict whether the tip amount will be above a certain threshold `LARGE_TIP_THRESHOLD` (e.g., above 20% of the total amount or total amount with tip excluded) - investigate the distribution of the target variable with respect to different values of `LARGE_TIP_THRESHOLD`, plot it, and choose a reasonable value. A hint: the tip amount is reported only for cashless payments, so you may want to first filter the dataset to include only such records.
- You may want to predict whether the total amount will be above a certain threshold `HIGH_TOTAL_AMOUNT_THRESHOLD` - what is a reasonable value for this threshold?
- etc.

What evaluation approach will you use? How will you split the dataset into training and test sets?

4. Implement Different Techniques for Handling Large Datasets

Classification Model

Implement a simple Naive Bayes classifier on your own that can be applied to large datasets - it should be able to process the dataset in parts, e.g., by processing one column or batch of rows at a time. Alternatively, you can use different model from available libraries, as long as it can be applied to large datasets. Be creative!

Techniques for Handling Large Datasets

You can incorporate the following techniques for handling large datasets directly in your solution, or you can split the process into two steps - first, preprocess the data and save it (use `parquet` format), then use the saved file to train and evaluate the model. Prepare your solution so that you can easily change the parameters of the techniques you use and compare the results, computation time and the size of the processed data used for training the model.

- Process the full dataset in parts - process the dataset in parts, e.g., by processing one column at a time for Naive Bayes (compute the necessary distributions for each column) or by processing batches of rows and creating an ensemble of models.

- Sampling techniques - sample a subset of the dataset, e.g., using random sampling, stratified sampling, - introduce a parameter `FRAC` that defines the fraction of the dataset to be sampled.
- Summarization techniques - apply clustering/gridding/binning to group similar instances and aggregate the values within each cluster/bin. E.g., you can use location-related features (pickup and dropoff coordinates) to create bins and aggregate (you can take the mean for numerical features and the majority value "*mode*" for categorical features) the rides which have pickup and dropoff locations within the same bin.
- Quantization techniques - reduce the number of distinct values for numerical features, e.g., by applying rounding, flooring, ceiling, or other techniques. Store values with reduced precision - it can reduce I/O time and memory usage and consequently speed up the computations.

In [3]:
```
# write your code here
```