Advanced Topics Notes

Data Discovery

Combining data from different sources. In this context: we start from a base dataset, and we want to add more information to it. There are some repositories of data known as data lakes that we can look for information in. However, they are massive and heterogeneous, meaning formats might be different (.csv, .json, etc) and the nomenclatures might be different (ESP/Spain, FRA/France).

Our goal is to go in the data lake and find data related to our project. Being related means being able to perform a join on our data, or perform unions (append data), or fill in missing values.

Data Augmentation

I want to extend my data from a data lake, but the difference is that in my base data I have picked a particular column and defined a task. I want to find some data to improve the performance. In data discovery, we augment the dataset in an abstract way, but in this way we want to improve our data. The matter is: how do we define a criteria to determine if taking data from the data lake improves the data in our set?

A modern approach does not require the model. Just by measuring some characteristics, we can assess if they are relevant or not.

Another approach is to use generative approach, where we generate synthetic data.

Feature Selection

The process by which I want to filter the features that are truly useful.

- Filter-based methods: compute some statistics between categories to see if they are relevant. Very fast, very simple but not very good.
- Wrapper methods: re-train and re-evaluate the model by iteratively removing features and seeing how it affects the accuracy. Slow but better than filter-based methods, and it able to capture interactions between features.

- Embedded methods: do feature selection during training. They are restricted to certain models (random forest, etc), and they allow to see which features are most relevant during the training phase. They work well, capture interactions between features and are faster than wrapper methods, but quite restrictive.

Data Quality

For example, $(!Order[total_amount] < 0)$ means that I prohibit records that have $total_amount$ less than zero from being in my database.

$!(Client[email]=Client[email] && Client[id] != Client[id]) means that two clients cannot have the same email.

This is essentially defining things logically to check in the databse.

Entity Resolution

Imagine in a databse I have these rows in different datasets:

- JohnSmith@gmail.com
- JonSmith@gmail.compute

Entity resolution is determining if they represent the same thing or not. There are these methods:

- Exact Matching: If there are identical parameters, they are the same. For example, if the names are the same, the entities are the same.
- Fuzzy Matching: Takes parameters and uses a similarity measure. For example, takes the names, measures a distance and if it exceeds a threshold, assume they are the same.
- Blocking: Uses a hash function, and entities that are similar are moved to the same bucket. In this bucket, apply the similarity measure.
- Supervised learning: Trains a model
- Rules