Machine Learning Project
Stage I
CS-C3240

06.02.2022

Contents

Important note for the stage I submission:

I will most likely move some of the stuff from problem formulation -section to the introduction -section and then make them work together. I just thought that my problem might be quite difficult to understand without some proper context, so I decided to include a "mini-introduction" already with this submission. Enjoy!

1 Problem formulation

My goal with this project is to tie it in with a project from my work. I've been working as a software consultant for almost a year now, and for the last two years my project has been the leading key control system in Finland.

The product's specific inner workings and customer base is confidential under my non-disclosure agreement, but I managed to get a permission from my customer to use the actual production data from the system. All the data is must made anonymous and must not shared with outside parties, including course staff. I understood that this kind of arrangement is okay in this course.

1.1 Background of the Problem

I will begin by describing the most basic workflow in our product on a high abstraction level.

Our system can be thought as having two main events: Issuance, which means giving out a key, and return, which, surprise surprise, means the return of a key. The interesting part is that the party giving the key out can specify a certain date-time that the key should be returned, called an expiration-date of the issuance. This is highly useful when the key to be issued is "powerful" and shouldn't be lost, or there would be serious financial and even juridical ramifications. For example a master key of an apartment building or Teekkarikylä would be a good example of such key. There's also the possibility that the key has been "booked" for a issuance immediately after expiration of the previous issuance, so the delay in return can cause also process issues with the party giving out the keys.

As there's multiple different parties which have a need to receive these powerful keys, and the party giving out the key might now know the person receiving it, it can be hard to define how likely it is that a powerful key will be returned in time, which will cause uncertainty and problems described earlier. This is why I think there's a possibility to solve or at least ease the problem by developing a machine learning algorithm.

1.2 The Problem

This machine learning algorithm would take in the historical data from the system and determine from previous issuances and other parameters how likely it it that a certain key with a certain expiration-date would be returned on

time. This prediction could be shown in the user interface as some kind of "trustfactor" score which would guide the user to either detain from issuing the key, or at least be wary of the fact that the expiration-date set might not be met and the parameters should be tweaked.

This means that the problem will be a categorization problem, as either the key was returned before the expiration-date or not. Then the certainty of the algorithm could be used as the "trustfactor" score.

1.3 The Data

I've already created an anonymized dump of the production database containing all the actual data in the system, with all sensitive fields removed. The data therefore is a SQL dump with millions of lines. I will write a query that will extract the data needed from this raw dump, and export this data to a CSV file which can be easily used from python.

My datapoints will be issuance with expiration set before the dumping of the database. This way we can say if the expiration-date was met or not, because if we would also include issuances where the expiration is set to year 2050 we wouldn't be able to determine if the key was returned on time and the data would therefore be skewed. Preliminary testing shows that there's about 16 000 issuances meeting this criteria, which means that I'll probably have around 16 000 datapoints to play around with.

1.4 Labels and Values to be used

As mentioned above, my datapoint will be a single issuance with an expiration-date. **The label** will be a boolean value (which can be of course mapped to an integer if needed) which describes whether or not the key was returned before the expiration-date.

The features are quite a lot of harder to decide. As I have access to a almost a gigabyte of raw data, the possibilities are quite endless. That being said I also need to scope them wisely to optimize the amount of SQL needed and also make sure that the model benefits from them.

I will use at least the following features, as I think they'll be essential indicators for the problem, but I might add or remove them if it turns out to be a better approach:

Features:

- prev_loans `int`
  - The amount of previous loans the recipient of the issuance has.
  - The system saves the recipients, and they're often re-used.
- loan_length `int`
  - The suspected length of the loan from the issuance date to the expiration-date

  - The length of the loan differs from hours to multiple years

- has_company `bool`
  - Describes whether or not the recipient is associated with a company in our system
  - This can add trustworthiness as being affiliated with a company can mean that you're more trustworthy

- recipient_age `int`
  - Tells how long the recipients information has been in our system
  - Usually a long "age" of larger than zero indicates that the recipient is "pre-created" which can be thought as a sign of trust as opposed to a recipient being created "on-demand".

- business_id `int`
  - Unique identifier of a party giving out the keys
  - This is provided as different parties have different use cases, so this could help with the redaction of noise.
  - I might also look into grouping and labeling businesses in some kind of computational manner, instead of using this id.