Make-a-thon Challenge 2024, PostFinance

**Background**

Explainable AI (XAI) is a hot topic. Consumers and regulators of AI products have embraced the power of machine learning in various business contexts but are becoming increasingly wary of reliance on a system that is by its very nature, too complex to fully understand. Explainability is often discussed in terms of global explainability, which is understanding the model itself, and local explainability, which is understanding why the model arrived at a particular decision in a particular case. Global explainability can, for example, be provided by the relative feature importances in a tree-based model, while local explainability is often provided by Shapley values.

Technical metrics, like Shapley values, can provide valuable information to the analyst programming the model itself. But they can be difficult to properly interpret by an end-user or regulator who is far-removed from the model development. Since the end-user and regulator are main targets of XAI, this means there is a gap between the technical output of the model metrics and the people who need to know how the model works. Your task will be to address this gap.

**Deliverables / Ideas**

We want a product, the exact shape of which is up to you, but it should address the translation of technical model explainability parameters to usable output in a business or regulatory context. We anticipate that Large Language Models (LLMs) will be useful in this regard, but are skeptical that an LLM will be able to solve the problem on its own with nothing more than a clever prompt. Some features we might look for:

* A clean API with defined inputs that can provide explainability for an arbitrary context
* Adjustable output options based on the target audience
* Callable within a technical system, such that we can provide explainability blurbs automatically within existing programs

Example: if you choose to work in python, a call might look like

get\_explainability(

shapley\_values: list,

feature\_documentation: dict,

tone=’regulatory’,

language=’de’,

target\_length=’medium’

)

for a regulatory application, for example in compliance, or

get\_explainability(

shapley\_values: list,

feature\_documentation: dict,

tone=conversational,

language=’fr’,

target\_length=’short’

)

for a non-regulatory application, for example for the marketing use case described below. Note that you are neither bound to this format nor to python, but this is just a rough example. Also note that you are not limited to the marketing case but can create your own examples in a variety of contexts to demonstrate the applicability of your product to other business areas.

**Case Study**

To make this entire thing less abstract, we provide an example use-case to start. This use-case involves marketing of residential mortgages to private customers. Postfinance offers a residential mortgage product and is obviously interested in enticing existing customers in other product areas to come to us for any mortgage needs. One tool we use to do this are marketing models, which try to identify possible candidates for a new home loan based on historical data from customers who have taken out mortgages with us in the past.

These model scores are used in a variety of contexts. One context is in marketing campaigns like emails and flyers, but the context we want to highlight here is for counter-personnel and customer service representatives. If a customer goes to a Post or Postfinance location to do some other business, a little window might pop up to indicate that they could be a good candidate for a home loan. But this feature is not so terribly well fleshed out. Right now there is a fixed score threshold, and if the score from the model is above this threshold the popup just say ‘candidate for home loan’. We think we can do this better in a couple ways:

* First, and this should be relatively trivial, is this a **strong** candidate with a very high score, or rather a score just above threshold. Due to their expense and long running times, there are simply less people interested in a mortgage exactly now compared to, for example, a credit card or another more common product. So perhaps it makes sense to differentiate here.
* To give the service representative a bit of an *in* with the customer, we want a brief sentence saying why this person might be a good candidate. For example:
  + ‘Large buildup of cash balance might indicate customer is saving for a mortgage’
  + ‘Large salary and recent asset grown shows optimal conditions for a mortgage’
  + ‘A large investment balance in low-risk assets may indicate this person is interested in real-estate’
  + And so on…

We provide here a (simulated) dataset containing the features described below. This dataset contains a TARGET parameter, where:

* **Target 1** means the customer took out a home loan with us (the features in this case come from the month before they closed the loan)
* **Target 0** means the customer has *not yet* taken out a home loan, though we of course hope that some customers will be interested in the future. The features come from the present month.

Because this dataset is (probably poorly) simulated, it should be possible to create a relatively good machine learning model to separate these classes. Don’t get too hung up on the model performance, as this data is only provided to get you going into the actual meat of the problem, which is the explainability.

Features:

| Age | Integer | Customer age in years |
| --- | --- | --- |
| balance\_cash | Float | Balance in liquid cash assets across all product categories (payment account, savings account, non-invested cash in non-regulated funds accounts) |
| balance\_investment | Float | Total invested cash in stocks, bonds, mutual funds, or other financial instruments excluding pillar 3 accounts |
| balance\_3a | Float | Balance of all 3a accounts for customer including cash and invested balance |
| app\_logins | Int | Number of app or webapp logins in the past month |
| income | Float | Monthly income from salary or pension (CHF) |
| has\_twint | Int | Flag indicating customer has twint connected to their payment account |
| has\_credit\_card | Int | Flag indicating customer has a prepay or postpay credit card of any type associated with their account |
| has\_partner\_account | Int | Flag indicating whether or not a customer has a joint account with a partner (balances in this case reflect assets on the personal account plus half of shared assets) |
| account\_trx\_ls\_amount | Float | Monthly amount of debit transactions of all types excluding debit- or credit-card transactions. |
| card\_trx\_ls\_amount | Float | Monthly amount of card debit transactions via either the debit- or credit-card channel |
| investment risk score | Int | Value from 1 to 5 indicating the total risk-profile of all investment products held by the customer (1 is all-cash or cash-equivalents and 5 is everything in etherium) |
| vermoegenszuwachs\_3mo | Float | Total increase or decrease in wealth over the past 3 months excluding fluctuations in securities prices |
| vermoegenszuwachs\_6mo | Float | Total increase or decrease in wealth over the past 6 months excluding fluctuations in securities prices |
| residence\_status | Char | Residence status in Switzerland. One of ‘ch’ (Swiss-citizen) or ‘b’, ‘c’, ‘l’, ‘s’ for (b-, c-, l-, s-permit holders) or ‘aus’ for nonresidents. |