Fourth Actuarial Pricing Game - 2018

Game rules

Arthur Charpentier

Ali Farzanehfar

Yves-Alexandre de Montjoye

Welcome to the fourth actuarial pricing game!

This document outlines the basics of actuarial pricing and the game rules.

What is the game?

The fourth actuarial pricing game is a Kaggle-style competition in insurance pricing. In this iteration of the game, **we** (the game organisers) will provide **you** (the players) with real historical home insurance data[1] and ask you to give us your best[2] model for offering insurance premium prices according to the data. We will then use previously unseen data and your pricing models to simulate a **competitive** insurance market where players will compete for customers that choose the cheapest premium offered to them. In each market, the player that makes the most money will win.

The game will occur in two stages (see section 1) during each you will submit a pricing model. In the first stage you can submit only *simple* models, while in the second stage you can submit any model you wish. There will be some criteria that models should satisfy (see section 6). The rest of this document will provide details on the game stages, deadlines, the customer model, model requirements, the data dictionary and other useful details.

1 Structure of the game

1.1 Stage 1 (Deadline April 9th at 11pm GMT)

The first stage of the game, involves some limitations on model complexity. The aim for this stage is for the models to be simple.

The requirements are:

1. **DO** feel free to convert `INSEE` codes to population density, latitude and longitude etc.
2. **DO** feel free to use **a single Decision Tree** or a single GLM for **feature selection**.
3. **DO** use only GLMs for offering premium prices to contracts.
4. **DON'T** use ensemble models. This means, no bagging, bootstrapping, or, boosting. This point removes many models including Random Forests.
5. **DON'T** use Artificial Neural Networks in any shape or form.
6. **DON'T** bootstrap the data for optimizing feature selection in your model.

[1] Data covers 2 years worth of contracts ($\approx 70,000$ per player) gathered from France in this decade. Each player will get a disjoint and separate part of the dataset.

[2] i.e. Most profitable

1.2 Stage 2 (Deadline May 14th at 11pm GMT)

The second stage of the game starts shortly after the first[3]. There are no requirements on the models in this case. In summary:

- **DO** feel free to use **any model** that you think is best![4]

2 The Dataset

The dataset contains approximately **2 million real historical contracts** across **3 consecutive years**. These are contracts concerning **home insurance policies from this decade in France**.

For the purposes of the game, you will each receive a **separate and disjoint** set of contracts from the first two years of the dataset. This will be your training data during the game.

In this iteration of the game, there are more than 200 registered players. However, each market we simulate will contain a random selection of $N \approx 20$ players. This is so that each player obtains a sizeable part of the dataset.

The remaining contracts from the third year ($\approx 716,000$) will be used in the market simulation and will not be provided to you in advance (like a real market). However, you are provided with a **fake**, randomly generated, dataset (100 contracts) for which you should offer premiums. The purpose of this fake dataset is to ensure your model is reproducible.

3 Actuarial Pricing in a Nutshell

Let's clarify some terminology. The **insurance premium** is the amount of money the insured is charged by the insurer for coverage set forth in the insurance policy. If the insured experiences a loss covered by the policy, they can submit a claim to the insurer. For the more insurance savvy of you, note that in this game, there is no investment income nor any underwriting expenses. The profit of an insurer is the difference between their earned premiums and their incurred losses.

[3] You will receive an email reminding of this them shortly after stage one is complete.
[4] Your models still need to respect requirements outlined in section 6.

3.1 Actuarial Pure Premium

Consider a person facing a total possible loss of S during a period of one year. The actuarial pure premium, or fair premium, is the expected value of S, $\mathbb{E}[S]$.

3.2 Actuarial Pure Premium in a Segmented Environment

This is best described through an example. Let X be a binary variable specifying the location of the home of a customer such that, $X \in \{\text{village, city}\}$. If the loss S is dependent on the value of X, then it is reasonable to suppose that the people living in cities and villages should pay different premiums. Therefore the premium π would be:

- People living in a village: $\pi(\text{village}) = \mathbb{E}[S|X = \text{village}]$
- People living in a city: $\pi(\text{city}) = \mathbb{E}[S|X = \text{city}]$

3.3 Annual Result

For a given year, let π_i denote the premium paid by the insured $i \in \{1, \cdots, n\}$. Further, let s_i denote the sum of all losses related to the insured i during the period covered:

- if $\sum_{i=1}^{n} \pi_i > \sum_{i=1}^{n} s_i$ then the insurance company has a positive result.
- if $\sum_{i=1}^{n} \pi_i < \sum_{i=1}^{n} s_i$ then the insurance company has a negative result and is losing money.

4 Mimicking an Insurance Market

After the deadline for stage 2, the market simulation can begin!

4.1 Generating all prices

Before any analysis, after everyone has submitted the necessary files (see section 5), your submitted models will be used to generate premium prices for all the $\approx 700,000$

contracts available in the third year. These will be used to represent **your model** in the final market competition alongside **other player models**.

Therefore, each player j will offer each customer i a premium π_i^j.

4.2 Behaviour of customers

Consider customer i. In a given market, this customer will have a to choose a value among premiums $\Pi_i = \{\pi_i^1, \cdots, \pi_i^N\}$ as offered by all the player models. Let π_i^j denote the ordered premium values:

$$\pi_i^1 \leq \pi_i^2 \leq \cdots \leq \pi_i^N.$$

Given that we are using data from home insurance, we will use a purely price-drive customer model. Specifically, we will use the **lowest affectation rule** where the customer i will always choose the company offering the cheapest price, π_i^1.

4.3 Metrics for insurers

Let $\mathcal{C}_j$ denote the set of customers $i \in \mathcal{C}_j$ that chose insurer j.

Definition 4.1. The annual **earned premium** of insurer j is:

$$P_j = \sum_{i \in \mathcal{C}_j} \pi_i^j$$

Definition 4.2. The annual **total loss** of insurer j, calculated at the end of the year and not known in advance, is:

$$L_j = \sum_{i \in \mathcal{C}_j} s_i$$

Here, s_i is the amount that customer i has cost their insurer. For example, if in one year, Ali's house burns down[5] causing damage worth $\$150,000$, then $s_{\text{Ali}} = 150,000$[6].

Definition 4.3. The annual **earned profit** of insurer j is:

$$EP_j = P_j - L_j.$$

Definition 4.4. The **final score** of insurer j in the market is the sum of their profits from stage 1, $EP_j^{\text{stage 1}}$ and stage 2, $EP_j^{\text{stage 2}}$. Therefore:

$$FS_j = EP_j^{\text{stage 1}} + EP_j^{\text{stage 2}}$$

Note that in each market of N players, **the winner** is the player with the maximum total profit over **both stages** of the game.

[5] Hopefully it won't.

[6] Assuming his insurer covers all costs.

5 Accessing the data, submitting models

Accessing the data and submitting models will all occur via the website:

https://pricing-game.dsi.ic.ac.uk

To allow you to login, your password will be sent to your emails separately on **March 9th**. After a simple and short consent form you will be directed to the a webpage where you will see two sections, **Download the data** and **Upload your model**.

5.1 Download the data

Once you click on this section you will be able to download a `pricing_game.zip` file. You should uncompress this file. Inside of it you will find the following:

1. `data_year1.csv` and `data_year2.csv`: these are your training data files.
2. `fake_year3.csv`: this is a CSV file containing 100 rows of randomly generated fake data. Your code (see section 5.2) will be used to generate prices for the fake contracts in this file. This is to ensure that your model is reproducible.
3. `code.py` and `code.R`: you should edit one of these files. **Please read the code files carefully and follow the instructions therein.** These files already contains a toy example for demonstration purposes only.
4. `model_checker.py` and `model_checker.R`: this file will be used by us to check that your models are satisfying our requirements (see section 6). Do **not** alter this. **Run it as is, after you have edited the code file.** If it does not allow you to submit please refer to your code files.
5. `python_packages.txt` and `R_packages.txt`: these text files list the python and R packages that we have available on our side. If you find that a package that you need is not available in here, please add it to you `model_description.txt` (see section 5.2) file when you upload your model. Please follow the format already present in the example description file `model_description.txt`.
6. `model_description.txt`: this is an example description file which demonstrates the format you should use for package installation requests (see next section), as well as, a description of the toy model provide.

5.2 Upload your model (Available from March 23rd)

In this section of the website you will find two links: one for stage 1 and the other for stage 2. For each stage you are required to upload 4 files including your code (**either** in Python **OR** R):

1. `code.py` **OR** `code.R`: this is your filled in code file. The information here will be used to generate prices for the market. **It is crucial that you follow the instructions in the code files.** Please read this carefully.

2. `trained_model`: this is essentially the result of training your model (e.g. parameter values, saved model etc.). Its purpose is to allow us to use your model for pricing in the game, without having to retrain on the training data. **For players using R** this will be taken care of internally and the file will be called `trained_model.RData`. **For players using Python** they will have to name this object in their `code.py` file as per the instructions in that file.

3. `validation_year3.csv`: This is a CSV file containing the premium values offered to each contract in the `fake_year3.csv` file by your model. We will use this to test that we can reproduce your model.

4. `model_description.txt`: In this text file you should describe how you constructed your model. See the example file provided for guidance. You should also **include any packages** that you are using that are **not** already listed in `python_packages.txt` or `R_packages.txt` file.

6 A note on model qualification

As we have over 200 players in this iteration of the game, model qualification will be checked automatically. Therefore, **all submitted models** are required to satisfy the criteria described in this section.

6.1 Single row input, Single price output

All submitted models must take **one row as input** and should return **one value** as the output. The toy model provided in `code.R` and `code.py` demonstrates this.

The reason for this requirement is that no insurance company, knows in advance the exact distribution of customers they will end up with in a particular year. They *have to* provide prices on-demand.

6.2 Models must be reproducible

The validation_year3.csv that you upload is to contain premium prices for all of the contracts present in fake_year3.csv. In order to validate that this is indeed true, we will use the uploaded code (code.py **OR** code.R) and the uploaded saved object (trained_model) and use it to recreate the validation_year3.csv file on our side. This should match the one you have uploaded.

The reason for this constraint is to make sure the model that on your side, the model that you have uploaded is indeed the same model you used to price the validation_year3.csv. This way, we can be sure that we are using the exact same model when simulating the market.

6.3 Models must be profitable on training data

All submitted models are required to have a net positive profit on **both** data_year1.csv **and** data_year2.csv. This means that the **earned profit** EP of the model needs to be **larger** than zero for **both** data_year1.csv **and** data_year2.csv. This will be checked automatically using the code in the files model_checker.py and model_checker.R.

6.4 Stage 1 models cannot be too complex

The model that you upload during the **first stage** will be checked for complexity, using various different metrics such as the total number of parameters, the number and degree of interaction terms etc. Models that, with consideration to their description and uploaded code, greatly exceed a certain pre-set complexity threshold can be excluded to make sure the rules of the game are followed. This is **not a strict cut-off** and will not immediately disqualify the models on its own.

If we find that your model is too complex we will contact you.

6.5 Versions, packages, and, formatting

Our system to validate your models, and, offer premiums for the third year, is fully automated. Therefore, the syntactic rules and package requirements specified here are **very important** for the smooth running of the game.

6.5.1 Versions

For R, use R 3.0.1 and above. For Python, use Python 3.5 and above.

6.5.2 Packages for for R and Python

In the first stage of the game, all the packages that you use in your files **need to be feely available**. For example, available on CRAN for R, or available on PyPI for Python.

Therefore, in your code files `code.py` or `code.R`, you should ensure that all packages that are used **can be installed by us** if not already installed and present in the `python_packages.txt` or `R_packages.txt` file. If you are using some specially, rare package, please specify it in the `model_description.txt` that you upload.

6.5.3 File names and function names

Please ensure that you **do not alter** the file names or functions names specified in the files in section 5. If you do then not only does this **potentially break** `model_checker.py` or `model_checker.R` which are used to validate your models but more seriously **it will break our automated system** that uses your `code.py` or `code.R` files to create prices for the actual market!

If you wish to add more details to your files please use comments inside the code file if necessary and do not alter the file names or function names.

In all the cases outlined above, if there is an issue and there is still time, the player will be contacted and given a chance to rectify the issue, however, **this should not be relied upon**.

If there is sufficient interest by the end of the game and after the publication of the results, we will endeavour to create a forum for discussion among the players. Here players can engage in discussion and exchange ideas, plots, code, analysis and more!

On the forum we will also publish ancillary results using other metrics such as earned market share, as well as, global rankings across markets and the effect of smaller markets to motivate discussion.

If you have any questions please email Ali at ali.farzanehfar@imperial.ac.uk.

Good luck to all!

Appendix: Variable dictionary

The two training datasets (data_year1.csv and data_year2.csv) contain the following 25 variables,

1. IDCNT Unique contract ID
2. RISK_NATURE_1 An Indicator for whether this is a primary house PH or a secondary house SH.
3. RISK_NATURE_2 Nature of the ownership and house type. This indicates, both if this is a house H or a flat F, and, if the client is the owner O or a tenant T. Hence there are 4 possible values FO, FT, HO, HT.)
4. RISK_NATURE_3 The type of occupancy. This can be one of '001', '002', '003', '004', '005'. For example, '002' indicates a tenant while the others are different types of occupancy.
5. RISK_FAMILY_STRUCT The family structure. This can be one of '000', '001', '002', '003', '004' representing different combinations of adults and children in the client family.
6. RISK_ROOMS_NB Number of declared rooms. Note that '011' means 11 or more (can be up to 20 in real life).
7. RISK_PROT_1 Protection indicator 1. This can be one of '001', '002', '003', '004' and represents different house protection schemes.
8. RISK_PROT_2 Protection indicator 2. This can be one of '000', '001', '002'. This represents another class of house protection schemes.
9. SURF_HOUSE Total house surface area. This ranges from '001' to '028', representing the real house surface area in ascending order.
10. SURF_VER Veranda surface area. This ranges from '000' to '021', in ascending order of the declared veranda surface area. For example, '000' means no veranda present.
11. SURF_OUTB Extra buildings surface area. This ranges from '000' to '011', in ascending order of area. This captures the total area of external buildings other than the main house. For example, '000' means no other building than the main house.
12. SURF_GR Ground surface. Ranges from '000' to '014', in ascending order of the declared terrain surface. Again '000' means no free terrain surface.
13. OPTION_1 Different options (true or false)

14. `OPTION_2` Different options (true or false)
15. `OPTION_3` Different options (true or false)
16. `OPTION_4` Different options (true or false)
17. `OPTION_5` Different options (true or false)
18. `OPTION_6` Different options (true or false)
19. `OPTION_7` Different options (true or false)
20. `OPTION_8` Different options (true or false)
21. `INSEE_CODE` The INSEE code is a numerical index used by the French National Institute for Statistics and Economic Studies[7] (INSEE) to identify various entities, including communes, departments[8].
22. `ZONE_1` Some geographical information. The zone scales, alphabetically ordered. Each zone is a combination of communes (see below).
23. `ZONE_2` Other geographical zones.
24. `ZONE_3` Other geographical zones.
25. `AMOUNT` Total loss for that policy.

Please note: The `INSEE_CODE` is a 5-digits alphanumeric code used by the French National Institute for Statistics and Economic Studies identify communes and departments in France. There are about 36,000 *communes* in France, but not every one of them is present in the dataset. The first 2 digits of `INSEE_CODE` identifies the *department*. The `INSEE_CODE` (or department code) can be used to merge external data to the datasets: population density, OpenStreetMap data, etc. If needed, two *shapefiles* are available online:

- For French regional Departments

  https://pricing-game.dsi.ic.ac.uk/static/DEPARTEMENTS.zip

- For communes

  https://pricing-game.dsi.ic.ac.uk/static/COMMUNES.zip

Be aware that, if you need to graph geographical information, the French reference system is RGF93 / Lambert-93 (EPSG: 2154) and not the common WGS84.

[7] Institut National de la Statistique et des Études Économiques (INSEE).
[8] These are one of the 96 French departments and specifies a region in the country.