Introduction

In the financial services industry, matching customers with the right financial advisor is critical for improving customer engagement, satisfaction, and policy conversions.

This report explores a data-driven approach to optimising financial advisor matching using machine learning and recommendation techniques. By leveraging historical matches between clients, products and agents, we developed a model that assigns the most suitable agent to each customer. This model assumes that historical matches between clients and agents were good matches to begin with, which is reasonable since all the products were bought by the client, indicating a successful client-agent interaction.

By applying a neural network to model these complex interactions and relationships, we were able to predict certain qualities of the ideal agent for every customer with some degree of success.

Dataset Overview

Client Information

The client dataset consists of 20,000 rows and 10 features. Of the 20,000, 353 rows contain NA values.

Of the 9 variables, **`household_size_grp`**, **`family_size_grp`** and **`cltpcode`** have effects that are summarised by **`household_size`** and **`economic_status`**. Hence, these features were not considered for the final model evaluation.

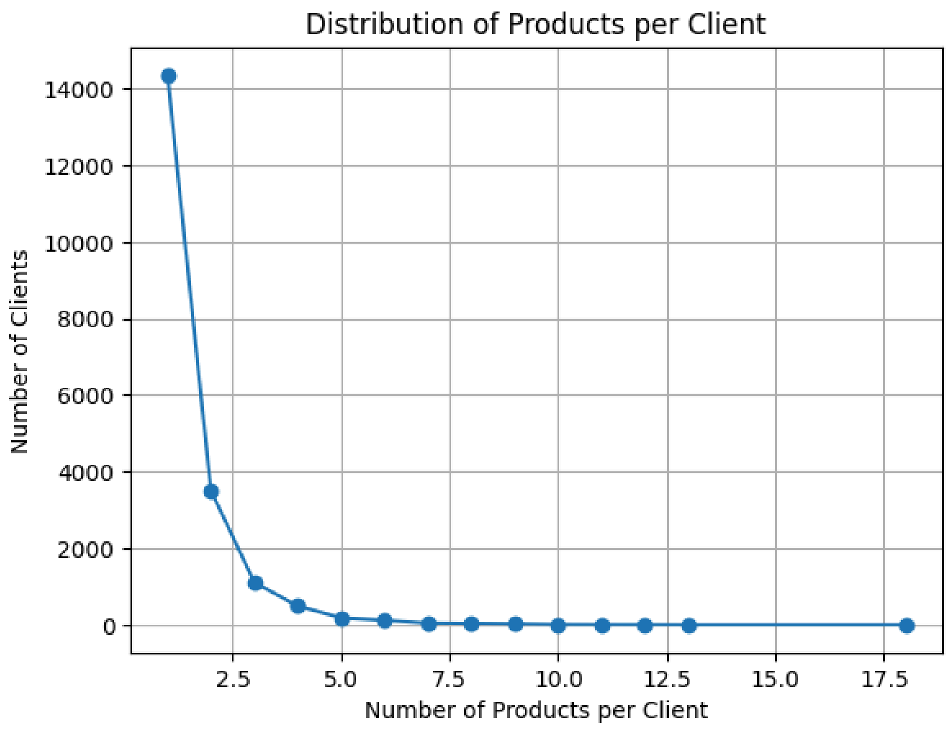
Of the remaining 6 variables, three **`race_desc_map`**, **`marryd`** and **`cltsex`** are categorical.

Policy Information

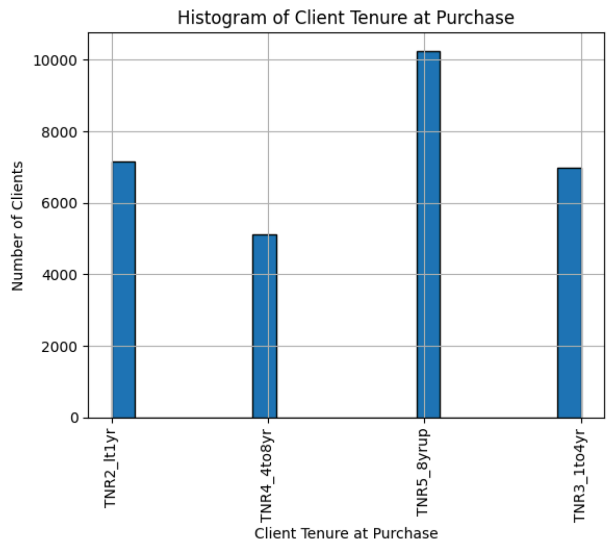
The policy dataset consists of 29,503 rows and 15 variables with no NA values.

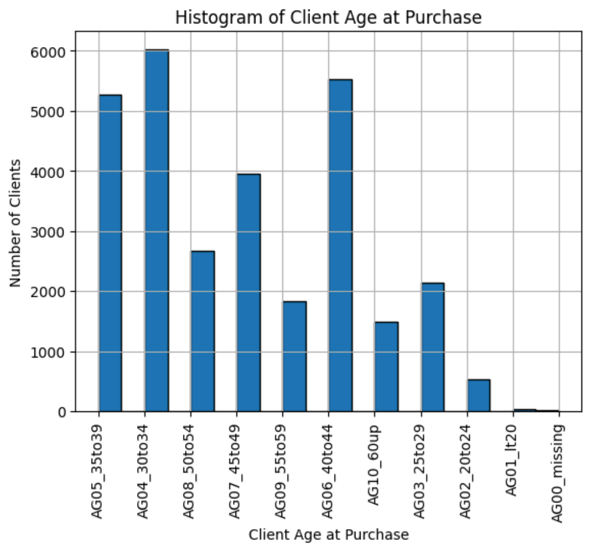
Of the 15 variables, **`flg_converted`**, **`flg_main`**, **`flg_rider`** and **`flg_cancel`** were not meaningful because they contained the same value for all policies (1, 1, 0 and 0 respectively).

Most clients only purchased one product, while a small minority purchased more than 10. The mean product per client is about 1.10.



The 2 categorical variables tenure group **`cust_tenure_at_purchase_grp`** and age group **`cust_age_purchase_grp`** of the clients when buying the products vary widely.





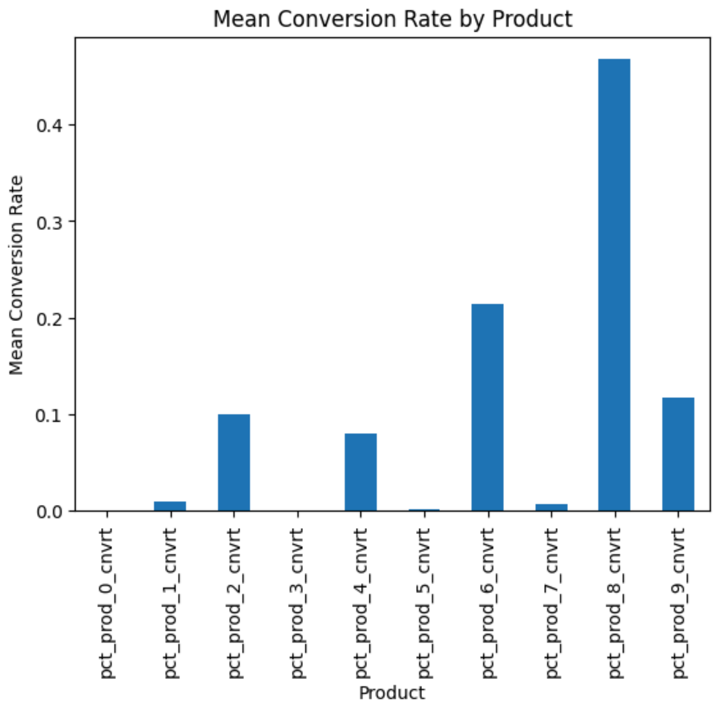
There were also 19 policies with no information on how old the client was when purchased, i.e., **`cust_tenure_at_purchase_grp = AG00_missing`**. These rows were dropped.

Agent Information

The agent dataset consists of 10,129 rows and 34 features. Of the 10,129, 12 rows contain NA values.

Only 3 variables were categorical – the agent's gender, marital status and their product expertise. These were one-hot encoded during the data processing step.

From the variables describing the percentage conversion of each product **`pct_prod_0_cnvrt`**, **`pct_prod_1_cnvrt`**, **…**, **`pct_prod_9_cnvrt`**, we can gain insights on the percentage converted for each agent on each policy, which gives us an insight on which products are the easiest and hardest to convert. From our analysis, product 3 is the hardest to sell, followed by 0, 1, 5 and 7. On the other hand, product 8 had the highest selling success rate.



Methodology

General Approach

Every agent is unique, and the dataset has about 10,000 unique agents. Instead of trying to predict who exactly the ideal agent is for a specific client, we decided to run a model to predict which are the ideal **qualities** this agent will have, and then match the most ideal agent from there.

Neural Network

Neural networks offer several advantages for this task due to their ability to model complex relationships. Choosing the best advisor is something that is incredibly complex that relies on factors like the chemistry between agent and client, which may be a result of a multitude of different factors, such as the backgrounds of both the agent and client involved. These relationships are most likely non-linear, so we decided to use neural networks to try and identify some of these hidden patterns.

We chose to use the PyTorch package to build our own neural network with features we felt were most suitable to handle this problem. Firstly, we created two different neural networks, one for regression and one for classification. This is because the ideal qualities of the agent we are trying to predict involves both continuous and categorical data and using the same neural network for both types of data would not lead to ideal results. All categorical features were one hot encoded.

These were the continuous variables selected:

```
agent_age                agent_tenure             cnt_converted

annual_premium_cnvrt     pct_lapsed               pct_cancel

pct_inforce              pct_prod_0_cnvrt         pct_prod_1_cnvrt

pct_prod_2_cnvrt         pct_prod_3_cnvrt         pct_prod_4_cnvrt

pct_prod_5_cnvrt         pct_prod_6_cnvrt         pct_SX1_male

pct_SX2_female           pct_AG01_lt20            pct_AG02_20to24

pct_AG03_25to29          pct_AG04_30to34          pct_AG05_35to39

pct_AG06_40to44          pct_AG07_45to49          pct_AG08_50to54

pct_AG09_55to59          pct_AG10_60up
```

These were the categorical variables selected:

`agent_gender` `agent_marital`

`agent_marital` was split into 4 dummies while <the last categorical> was split into 12 dummies for fitting in the neural network.

The specific neural network we are using is a multi-layered perceptron model. For this model,we did batch normalisation so that all the data can be easily compared to each other and reduce training time. In tandem with this, we chose to use a LeakyReLU activation function to improve convergence speeds, since batch normalisation might lead to negative values which LeakyReLU is better equipped to handle.

Results

Performance Metrics

The model was trained to predict a total of 32 features of the ideal agent, including 29 continuous features and 3 categorical features. The performance metric we used is the accuracy of the prediction to the actual agent(s) who converted the client according to the policy information.

For continuous variables, the predicted value had to come within 0.025 of the true value to be considered "accurate". We thought this left a narrow enough window for For categorical variables, the predicted value had to be equal to the true value to be considered "accurate".

Model Performance

Out of 5794 * 32 predictions, our model made 115191 "accurate" predictions. This means our model, on average, successfully predicted 19.88 suitable agent features per client. For continuous variables, this means that the difference between the true and predicted values is less than 0.025.

Insights

There is a chance that our model did not perform as well as hoped due to the intricacies that go into human relationships that might not be derivable from data such as a person's background. In the financial advising industry, trust is often a key quality that clients look out for. The ability to build that trust and rapport with another human might not be easy to model for with a simple neural network.

Additionally, it is also possible that not all the client-agent interactions in the dataset were necessarily positive. In the real world, there are many reasons that would drive a client to buy a product from an agent that is unrelated to how much of a "fit" an agent is. For example, a person might be engaging that specific agent only because he is a friend, a relative or someone tangentially related to that person, and is under certain social obligations to buy a product. These confounding factors could be a source of noise in the data and prevent the model from making accurate predictions.

Conclusion

Despite having to account for numerous continuous and categorical data, the neural network model predicted with a decent degree of success the ideal agent for each client. This shows the promise of neural networks in future applications of complex datasets with many potential sources of noise.

However, we do concede that a drawback on neural network models is that they tend to be a black box, and the models do not make it very easy to glean insights on what is the most impactful factor of a client's or agent's background that will draw them towards each other. Future work could involve trying on more models that have more interpretable results, such as decision trees, to isolate important variables and refine the inputs to the neural network model to reduce noise and improve outcomes.