AI vs Churn Rate

Implementing a predictive model in robo-advisors to optimise client retention and reduce churn rate: Micappital S.L.

MBA 24D
BHASKAR Raghu

CONTENTS

USER GUIDE

1. Overview

1.1. AI vs churn

One of the key methods AI can help reduce churn is through predictive analytics. By using machine learning algorithms, companies can gain insights into customer behaviour patterns and identify customers at risk of leaving. These models analyse customer data, such as subscription duration and website activity, to determine which customers are likely to churn.

Understanding the trigger events that lead to customers leaving enables businesses to take proactive measures to prevent churn. This approach helps companies make better decisions that benefit both the customers and the business.

1.2. Objective

Micappital currently suffers from a 10% customer churn rate. Based on 2023 annual revenue of €150k, and taking into account that the median gross dollar churn (i.e. revenue lost YoY due to churn rate) was in line with the churn rate, there is a potential value of €15k if the rate decreased to 0%. In addition, we should factor in that, according to Bain, acquiring new customers is around 5x more expensive than retaining existing ones, so this could lead to substantial cost savings too.

This user guide is aimed to provide Micappital with a reference tool to implement an AI model to proactively identify and address factors leading to their customer attrition in order to prevent their ARR to plummet and grow more efficiently.

2. Model development/Methodology

2.1. Model explanation

2.1.1. Logistic regression

Logistic regression is a statistical model used for binary classification tasks, where the outcome is either 0 or 1, such as predicting whether a customer will churn or not. It estimates the probability of a given outcome based on one or more independent variables. The model uses a logistic function to transform the output into a probability value between 0 and 1. It's simple, interpretable, and often serves as a baseline model in predictive analytics.

Think of Logistic Regression like a yes-or-no decision maker. It looks at data about your customers (like how often they use or invest using Micappital) and tries to figure out the chance that they will stop being customers (churn). It gives a percentage score that tells you how likely each customer is to leave.

2.1.2. Random CART

Random CART (classification and regression trees) is a type of decision-making tool that helps you sort data into different groups or categories. It is used to make predictions, such as determining whether a customer is likely to stay with a company or leave (churn). The "random" part means that the model adds an element of randomness to improve accuracy and prevent mistakes.

Imagine you have a big tree with branches that help you make decisions. Each branch asks a simple question like, "Does this customer use the Micappital app every day?" Depending on the answer, you move to a different branch until you reach a final decision, like "This customer is likely to leave" or "This customer will stay."

2.1.3. Random forest

Similar to random CART, random forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode (for classification) or mean prediction (for regression) of the individual trees. Each tree is built on a random subset of features and data points, which helps in reducing overfitting and improving generalisation. In the context of churn prediction, random forest can capture complex patterns in customer behaviour that single models might miss. Whereas CART emphasises simplicity, random forest emphasises accuracy.

Imagine you're trying to make a decision, and you ask a group of experts for advice. Random forest works like that group of experts. It creates many small decision-makers (trees) that each give an opinion on whether a customer will leave or stay. Then, it combines all those opinions to make a final prediction. This helps make sure the prediction is accurate and not based on just one opinion.

2.1.4. Random forest with hyperparameter tuning

This model is the same as the random forest but with hyperparameter tuning. Hyperparameters are settings that control the training process (like the number of trees or the depth of each tree). Tuning involves systematically adjusting these parameters to improve the model's performance. Techniques like grid search or random search are commonly used for this. The goal is to find the best combination of hyperparameters that lead to the most accurate churn predictions.

This is just like the random forest we talked about, but with a twist. Here, we spend some time adjusting the settings for how the trees are made to make them even better at predicting who might leave. It's like fine-tuning a recipe to get the best possible dish.

2.1.5. Gradient boosting classifier

Gradient boosting classifier is another ensemble learning method that builds models sequentially. Each new model attempts to correct the errors made by the previous models. This approach focuses on "boosting" the performance by combining weak learners (typically shallow decision trees) to form a strong predictor. Gradient boosting is powerful for complex datasets and can provide high predictive accuracy for churn analysis, but it can be prone to overfitting if not properly tuned.

Imagine you're building a team of decision-makers, but instead of creating them all at once, you create them one by one. Each new team member is trained to fix the mistakes made by the previous ones. Over time, this makes the team really good at predicting which customers will leave. That is what gradient boosting does –it builds smarter and smarter predictors to get the best results.

2.1.6. XGBoost with fine-tuning of hyperparameters

XGBoost (extreme gradient boosting) is an advanced implementation of gradient boosting that is highly efficient and scalable. It includes regularisation terms to prevent overfitting and is known for its speed and performance. Fine-tuning the hyperparameters of XGBoost involves carefully adjusting parameters such as learning rate, tree depth, and the number of trees, to optimise the model's performance. When finely tuned, XGBoost is often one of the top-performing models for tasks like churn prediction.

XGBoost is like a supercharged version of gradient boosting. It's faster and often gives better results. We also take extra care to tweak its settings to make it work as well as possible. This is like getting the best equipment for your team to make sure they perform at their peak.

2.1.7. Artificial neural network with TensorFlow

An artificial neural network (ANN) is a computational model inspired by the way biological neural networks in the human brain work. It consists of layers of interconnected nodes (neurons) that process and learn from input data. TensorFlow is a popular open-source library for building and training ANNs. In the context of churn prediction, an ANN can learn complex, non-linear relationships between customer data features and the likelihood of churn, potentially outperforming more traditional models like logistic regression or random forests.

An artificial neural network is inspired by how the human brain works. It's like a virtual brain that can learn patterns from your data. With TensorFlow, a tool to help build these networks, you can create a model that learns to recognize the patterns that indicate a customer might leave. It's especially good at understanding complex and subtle clues in the data.

2.1.8. Artificial neural network with fine-tuned hyperparameters

As in random forest and XGBoost, hyperparameter tuning in ANNs is crucial as it can significantly impact the model's accuracy and ability to generalise to new data. Fine-tuning often involves techniques like grid search, random search, or more advanced methods like bayesian optimization. This tuned ANN can provide highly accurate churn predictions if done correctly.

This is the same as the artificial neural network, but with some adjustments to make it even better. By tweaking its settings, like how many "neurons" it has or how fast it learns, you can make it more accurate in predicting which customers might leave. It is like training an athlete with a personalised plan to get the best performance.

2.2. Data preparation

Before preparing the model, we followed all the steps to clean our raw data for the modelling. The data cleaning and preparation process involved several key steps to ensure the data was ready for feeding into machine learning models. Here's an overview:

2.2.1. Data loading and initial inspection

The dataset was loaded into a Pandas DataFrame, and basic information about its structure was inspected, such as column names, data types, and the presence of missing values. The data had a mix of numeric, categorical, and object data types. Some columns were missing values and required cleaning to remove the spaces and make the data types uniform.

2.2.2. Handling missing values

Missing data was imputed for key columns. For numeric columns (e.g., `CAC`, `revenue`, etc.), missing values were filled using the column mean. For categorical columns (`experience`, `invAssets`, `invRevenue`, etc.), the most frequent category (mode) was used to fill in missing values. Columns like `dateOUT` and `No. of days` were dropped because they were either irrelevant or had too many missing values.

2.2.3. Categorical variables

Firstly, many columns containing categorical data (e.g., `bank`, `recommended`, `invHorizon`, etc.) were converted to the appropriate data types (e.g., category or boolean). Secondly, one-hot encoding (dummy variables) was applied to categorical columns, converting each categorical level into a binary feature. This ensures the data is in a format that can be used in machine learning models.

2.2.4. Combining rare categories

The custom function `CombineRareCategories` was used to handle rare categories in categorical columns. Categories that appeared less frequently than a specified threshold were combined into an "Other" category to reduce noise and sparsity in the data.

2.2.5. Data splitting

After cleaning and processing the data, it was split into training and testing sets using an 80/20 split. The target variable was `Leave in less than 15 months`, a binary classification indicating whether the customer would leave in less than 15 months or not. The splitting was stratified to ensure that the proportion of positive and negative classes in both the training and testing datasets was maintained.

2.2.6. Feature selection

During the preparation phase, irrelevant features like `idClient`, `dateIN`, `dateOUT_value`, etc., were dropped from the feature set as they did not contribute to the prediction task. Only the most relevant features were retained in the final dataset to feed the models.

2.2.7. Final preprocessed dataset

The cleaned and processed dataset consisted of 69 features (after encoding) and was ready to be used in the model training process.

The data cleaning and preparation involved converting data types, handling missing values, transforming categorical variables, and scaling numerical features. These steps ensured that the dataset was ready for robust machine learning model training and testing. The goal was to provide clean, well-structured data to optimise model performance.

2.3. Data performance summary

After preparing the data, the next step was to apply it to our suite of data science models. We evaluated six different models, rigorously testing each one against key classification performance metrics, including AUC (Area Under the Curve), sensitivity, and specificity. Additionally, for each model, we identified an optimal classification threshold by analysing its respective profit curve, ensuring that our final recommendations balanced both predictive accuracy and profitability. We didn't run random forest because of its similarity with random CART.

2.3.1. Logistic regression

A linear model trained to predict customer retention. The model shows an accuracy of approximately 60%, sensitivity of 64%, and specificity of 57%. The optimal threshold for maximising profit was found at 0.21, yielding a maximum profit of 127.66.

2.3.2. Random CART

This model divides the data into decision-based categories, and hypertuning was applied for better performance. The model performed well with an accuracy of 89.36%, sensitivity of 89.28%, and specificity of 89.41%. The optimal threshold was 0.11, and it produced a maximum profit of 744.68.

2.3.3. Gradient boosting classifier

A boosting algorithm designed to improve prediction accuracy. It resulted in an accuracy of 92.2%, with a maximum profit of 744.68 and an optimal threshold of 0.27. The model exhibited strong classification abilities.

2.3.4. XGBoost with fine-tuning of hyperparameters

After hyperparameter tuning, XGBoost achieved an accuracy of 92.9%, with a maximum profit of 776.60 and an optimal threshold of 0.29. This model provided the best profit among the models tested.

2.3.5. Artificial neural network with TensorFlow

A neural network model built using TensorFlow. It achieved an accuracy of 62.41%, sensitivity of 55.36%, and specificity of 67%. The maximum profit was 53.19, with an optimal threshold of 0.15. While the performance was modest, the neural network has potential for further tuning.

2.3.6. Artificial neural network with fine-tuned hyperparameters

GridSearchCV was used to optimise the hyperparameters (batch size and epochs). While the results of this tuning are not fully detailed in the snippet, neural networks tend to improve with careful tuning of hyperparameters.

2.4. Summary of findings

XGBoost produced the highest profit and accuracy, making it the most effective model in this scenario. However, Random CART and Gradient Boosting also performed well, particularly in terms of profit maximisation. Neural Networks showed potential but underperformed compared to tree-based models in this task, indicating that further tuning and experimentation are required for competitive results.

Overall, tree-based models like Random CART and XGBoost outperformed linear and neural network models in terms of profit and accuracy, with XGBoost being the top performer after hyperparameter tuning.

3. Implementation plan

3.1. Model choice

XGBoost provided the highest accuracy (92.9%) and maximum profit (776.60). Its strong performance in both areas makes it ideal for predicting customer retention and ensuring profitability, so it is the best choice for deployment. Deploy the XGBoost model into the client's system, making it a key part of their customer retention strategy. Integrate it with the data pipeline to make real-time predictions on whether customers will stay or leave.

Although XGBoost outperformed Random CART, the latter also achieved high accuracy (89.36%) and a comparable profit (744.68). It could serve as a fallback option if XGBoost does not perform well in specific scenarios or requires adjustments. Keep Random CART as a backup in production, using it for comparison or specific cases where a simpler, interpretable model is preferred.

3.2. Monitoring and maintenance

3.2.1. Fine-tuning and continuous improvement

Further fine-tune XGBoost: Hyperparameter tuning was already conducted, but further adjustments could yield better results. Continue experimenting with parameters such as `learning_rate`, `max_depth`, and `min_child_weight`. Set up a process for continuous improvement of the model, monitoring its performance in production and refining it based on real-time data and feedback.

3.2.2. Neural network tuning:

The Artificial Neural Network showed potential but underperformed compared to tree-based models. This could be due to inadequate hyperparameter tuning or architecture choices. Experiment with different architectures, more epochs, or varied batch sizes. Continue experimenting with ANNs, keeping them as part of an ongoing R&D process. They could eventually outperform tree-based models with the right configuration.

3.2.3. Monitoring and evaluation

Establish Key Metrics: The performance metrics from the models, such as accuracy, sensitivity, specificity, and profit, should be tracked over time. Build a dashboard to monitor these metrics and flag any potential performance degradation. Use this dashboard to evaluate the impact of the model on customer retention and profitability regularly.

3.2.4. Threshold adjustments

The threshold for classification (i.e., deciding whether a customer will leave or not) plays a crucial role in balancing precision and recall, as well as profit. Regularly assess and adjust this threshold based on business objectives, such as maximising retention or focusing on high-value customers. Implement a process for dynamically adjusting the classification threshold based on current business priorities. For example, during high churn seasons, lowering the threshold might help retain more customers.

3.2.5. Customer retention strategy

Segment High-Risk and Low-Risk Customers: Classify customers based on the predicted risk levels from the models. Focus on saving high-value customers while optimising efforts for low-value customers. For example, classify predicted customer attrition into High (above 70%), Medium (40-70%), and Low (below 40%) risk categories, with periodic adjustments to reflect changes in customer behaviour. Develop targeted retention campaigns for each group such as personalised offers, targeted communication, or loyalty programs.

3.2.6. Data pipeline and automation

Automate Data Preparation: Ensure that the data pipeline is fully automated so that new data is cleaned, processed, and fed into the model without manual intervention. Develop an end-to-end data pipeline using tools like Airflow, AWS Lambda, or similar automation tools to prepare and clean data in real time.

3.2.7. Regular data refresh:

Customer data needs to be refreshed periodically to ensure that the model stays relevant and continues to perform well. Set up daily or weekly data ingestion and model retraining processes to keep the model updated with the latest information.

3.2.8. Evaluate profit and ROI

Assess the Business Impact: Once the model is deployed, track its impact on business performance, especially in terms of profit, ROI, and customer retention rates. Set clear business KPIs related to customer retention, such as the number of customers saved, the total cost saved from retaining customers, and the ROI of retention efforts. Compare these metrics to the model's predictions to ensure it is delivering business value.

3.3. Client training and adoption

Model interpretability: Ensure that key stakeholders understand the models, especially if they are complex (like XGBoost). The client should make a strong business judgement to decide on the probability thresholds to classify an individual at High, medium or Low risk of attrition and keep modifying it with time. Create training sessions for business users and decision-makers to familiarise them with model outputs, thresholds, and key metrics, making it easier for them to trust and act on the predictions.

4. Moving forward and future considerations

Firstly, it is advisable to combine multiple models, such as XGBoost and Random CART, through an ensemble approach that could yield better performance and robustness. Experiment with ensemble techniques like stacking or bagging to combine the predictions of the best-performing models.

Secondly, explore data enrichment by adding new features or external data sources to enhance model performance. For instance, data related to customer behaviour such as how frequently they reach out to customer service, market trends such as market share or what are other robo advisors doing, or economic conditions could improve prediction accuracy. Partnering with data providers or leveraging internal analytics to enrich the dataset with more features and insights could also be valuable.

MEMO

This memo outlines the actions, meetings, calls, and touchpoints taken to implement the AI predictive model aimed at reducing the churn rate for Micappital. The collaboration includes interactions with various stakeholders and detailed steps taken to achieve our objectives.

1. *Initial assessment:* Conducted an initial meeting with Micappital to understand the company's churn-related challenges, agree on project scope and objectives, and gathered requirements for the AI model.

2. *Data collection and preparation:* Collected historical customer data, including demographics, customer portfolio, and market evolution. Cleaned and preprocessed the data for model training.

3. *Exploratory data analysis:* Performed EDA to identify patterns and correlations related to customer churn.

4. *Model selection and development:* Evaluated six machine learning algorithms and selected the most suitable one for predicting churn. Developed the initial version of the predictive model.

5. *Model testing and validation:* Tested the model using a validation dataset and refined it based on performance metrics such as accuracy, precision, and recall.

6. *Presentation meeting:* Introduced the model to Micappital, refined objectives, and outlined the project timeline.

7. *Weekly progress meetings over July:* Provided updates on progress, addressed any issues, and ensured alignment with project goals.

8. *Technical deep dive sessions:* Discussed technical details of the model, including data handling, algorithm selection, and model evaluation metrics.

9. *Final review meeting:* Reviewed the final model, discussed results and empowered Micappital team with knowledge to utilise the model effectively. We presented the user guide for correct deployment, monitoring and potential enhancement.

The collaborative efforts have led to the successful development and validation of an AI predictive model to predict and address churn rate reduction. The next steps involve deploying the model into production, monitoring its performance, and continuously refining it to adapt to changing customer behaviours.

EXHIBITS

Exhibit 1. Data dictionary

VariableName	Definition
idClient	Client Identification
dateIN	Registration date
dateOUT	Dropping date
bank	Bank the client uses for investing
CAC	Customer acquisition cost
risk	Level of risk assigned by Micappital based on onboarding survey
now_value	Current value of client's portfolio [EUR]
contribution	Sum of all the contributions the client has made to the portfolio [EUR]
result	Current profit or loss of the client portfolio [EUR]
max_value	Maximum historic value of portfolio [EUR]
revenue	Revenue Micappital has earned from that client [EUR]
dateOUT_value	Value of client's portfolio when dropping out [EUR]
recommended	Has the client used a recommendation code when registering?
age	Age of the client
invHorizon	Client's investment time horizon
investor_type	Client's level of risk stated by client
experience	Client's investing experience stated by client
invAssets	Client's total assets
invRevenue	Client's total annual revenues
invRevenueXpent	Client's portion of revenues that expends
invRevStability	Client's revenue stability

SOURCES

Key Banc Capital Markets (2022), *"KBCM TECHNOLOGY GROUP 2022 PRIVATE SAAS 13th ANNUAL COMPANY SURVEY."* https://www.key.com/content/dam/kco/documents/businesses___institutions/2022_kbcm_saas_survey_10-20-22_vF.pdf

Brownlee, Jason (2020), *"A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning."* Machine Learning Mastery. August 15. https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

Micappital team. Co-Founders: Miguel Camiña (CEO), Borja Nieto (CFO), Carlos Bernabeu (CTO); Tech lead: Javier Sandoval

Stewart, Matthew (2019), *"Simple Guide to Hyperparameter Tuning in Neural Networks".* Towards Data Science. July 9. https://towardsdatascience.com/simple-guide-to-hyperparameter-tuning-in-neural-networks-3fe03dad8594

Tariq, Farina (2023), *"Building Neural Network with TensorFlow: A Beginner's Guide."* Medium. May 12. https://medium.com/@beingfarina/building-neural-network-with-tensorflow-a-beginners-guide-6e11ca98cc38