# Predicting future calls to the Customer Service

Application for CRM & VIP Data Scientist

**Cainã Max Couto da Silva**, PhD

April 2024

# Agenda

1

## Introduction

- Business context
- Delivered files

2

## Exploratory Data Analysis

- Statistical analysis
- Feature selection

3

## Predictive modeling

- Model selection
- Model validation
- Model deployment

**How does it help the business?**

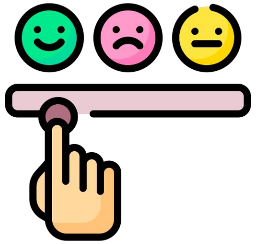
# Business context

For efficiency purposes, Betsson is trying to predict which customers are going to call the Customer Service, based on their past behaviour.

For a given day, predict whether a customer will call the Customer Service in the following 14 days.

**Why?** → **Optimize** customer relationship → 

therefore reducing churn



# File delivery structure

**project_folder**

- **notebooks**
  - 0-eda-ipynb
  - 1-feature-selection.ipynb
  - 2-predictive-modeling.ipynb
  - pycaret.ipynb
- requirements.txt

**GitHub repositories:**

- Project
- Streamlit App
- FastAPI

**Docker hub (images):**

- Streamlit App
- FastAPI

**APP/API endpoints:**

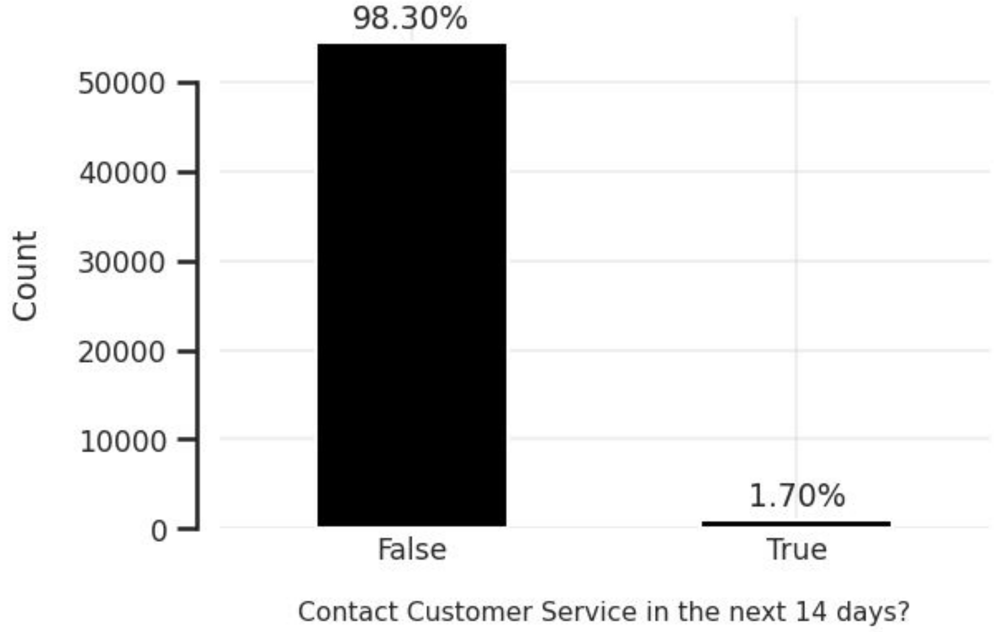
- App: https://crm-app.datargs.com
- FastAPI: https://crm-app.datargs.com

**Note:** these web services were not shared with anyone.
I can delete them after our interview if necessary :)

# EDA

# Data quality checks

- No missing values (probably they've been filled with -1)
- No duplicated rows
- Constant & quasi-constant features
- Presence of outliers
- Target imbalance



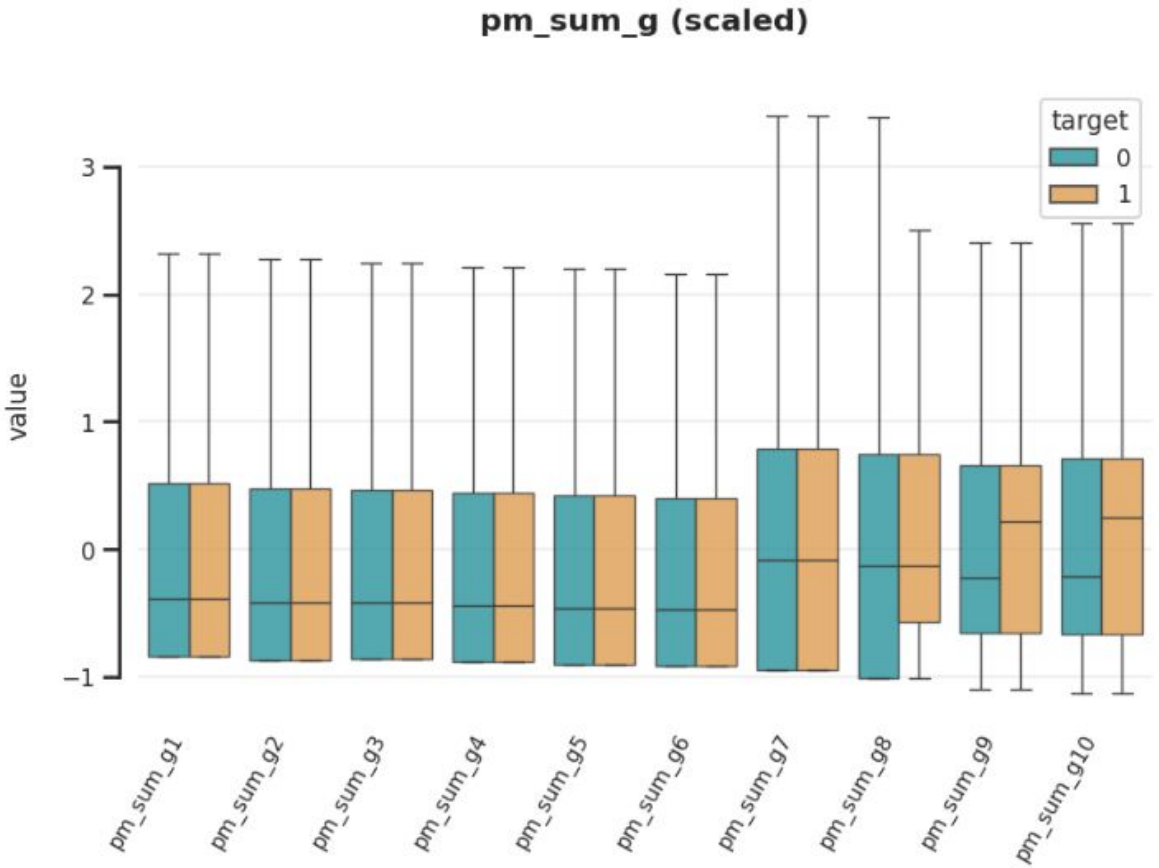
# Statistical Analysis

Example of outliers:

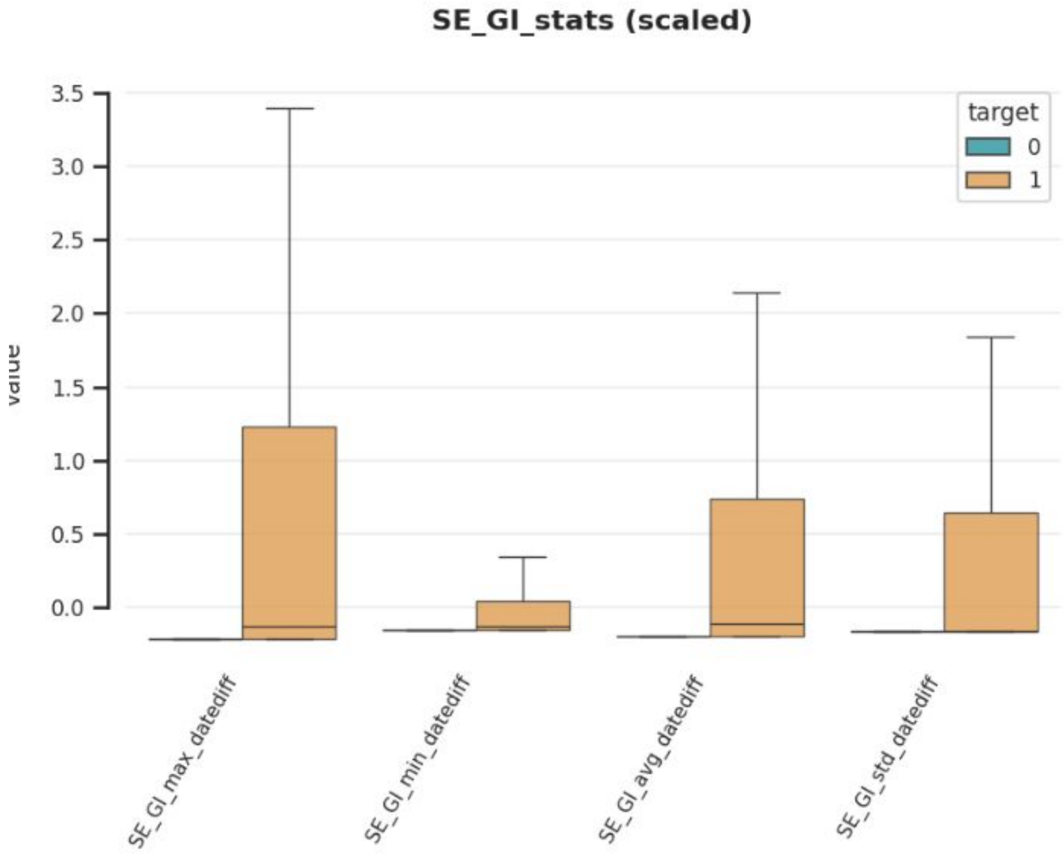
| | gw_g8 | gw_g7 | SE_GI_wrt_days_70days | mar_g3 | bon_wrt_succdep_g8 | unsucc_dep_g8 | SB_to_g8 | gw_g2 | SB_to_g7 | with_sum_g8 |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 55415.000000 | 55415.000000 | 55415.000000 | 55415.000000 | 55415.000000 | 55415.000000 | 5.541500e+04 | 55415.000000 | 5.541500e+04 | 55415.000000 |
| mean | -86.992825 | -85.059200 | 0.000018 | -0.092597 | 0.971746 | 173.537111 | 4.237183e+02 | -91.566777 | 3.586638e+02 | 222.618962 |
| std | 2582.774584 | 2532.532325 | 0.004248 | 2.330595 | 32.793466 | 4083.854676 | 8.789732e+03 | 1517.749186 | 6.627518e+03 | 2465.312225 |
| min | -163365.400000 | -85766.660000 | 0.000000 | -0.999800 | 0.000000 | 0.000000 | 0.000000e+00 | -176602.780000 | 0.000000e+00 | 0.000000 |
| 25% | -103.864700 | -100.050000 | 0.000000 | -0.202600 | 0.000000 | 0.000000 | 0.000000e+00 | -85.464550 | 0.000000e+00 | 0.000000 |
| 50% | -14.950000 | -11.600000 | 0.000000 | -0.024500 | 0.000000 | 0.000000 | 0.000000e+00 | -3.100000 | 0.000000e+00 | 0.000000 |
| 75% | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.053100 | 30.000000 | 4.826810e+01 | 0.000000 | 2.266515e+01 | 65.000000 |
| 99% | 1378.881476 | 1304.765296 | 0.000000 | 0.814158 | 16.044000 | 2758.542340 | 6.758736e+03 | 1199.805400 | 6.206709e+03 | 3362.277996 |
| 99.5% | 2372.264353 | 2463.224288 | 0.000000 | 1.289123 | 32.545768 | 4639.725315 | 1.188462e+04 | 2235.062937 | 1.066179e+04 | 5379.028963 |
| max | 478379.799800 | 513582.087500 | 1.000000 | 541.421200 | 7152.580000 | 887001.000000 | 1.853940e+06 | 142312.570000 | 1.350032e+06 | 479756.089000 |

# Statistical Analysis

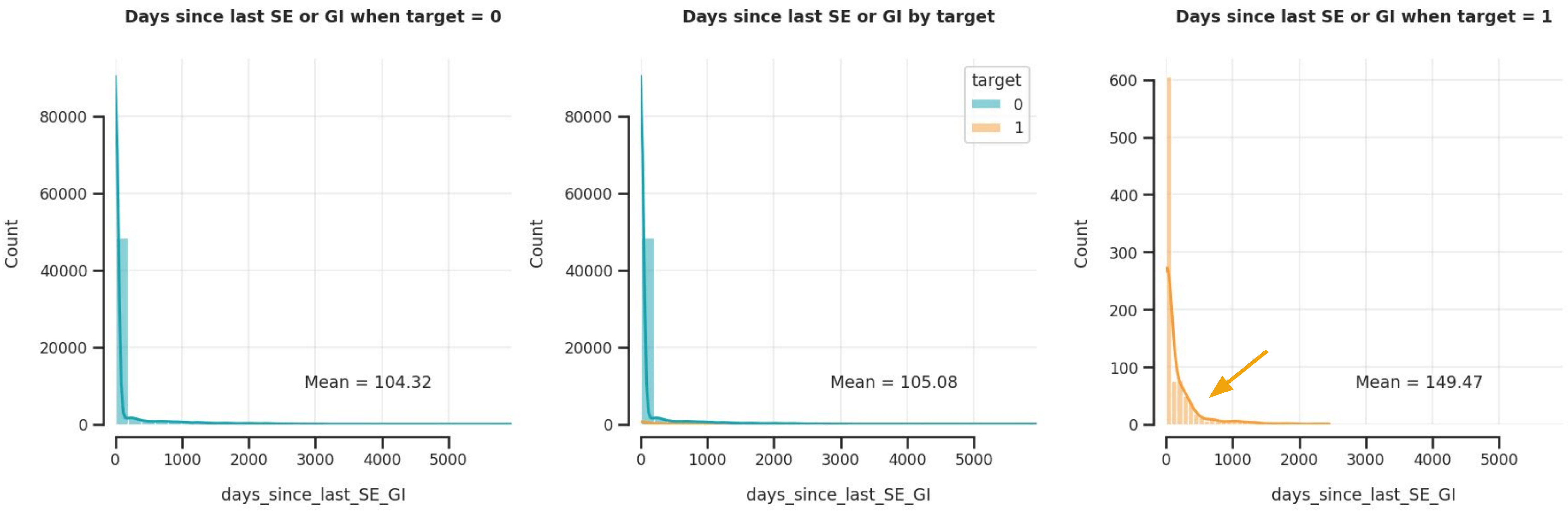
Example of non-informative features




Example of informative features



# Statistical Analysis




Example of feature with potential impact to the model

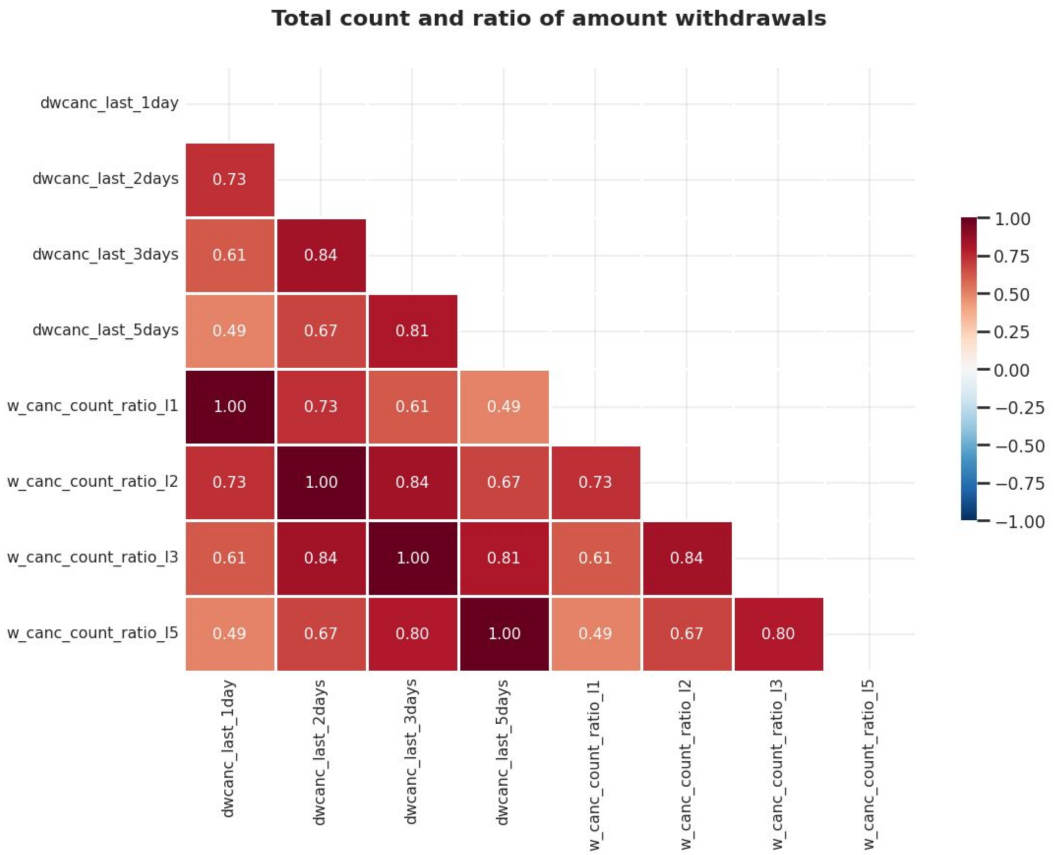
# Statistical Analysis

**Mann–Whitney U test:**

- 198 statistically significant features
- 69 non-statistically significant features

**Spearman correlation:**

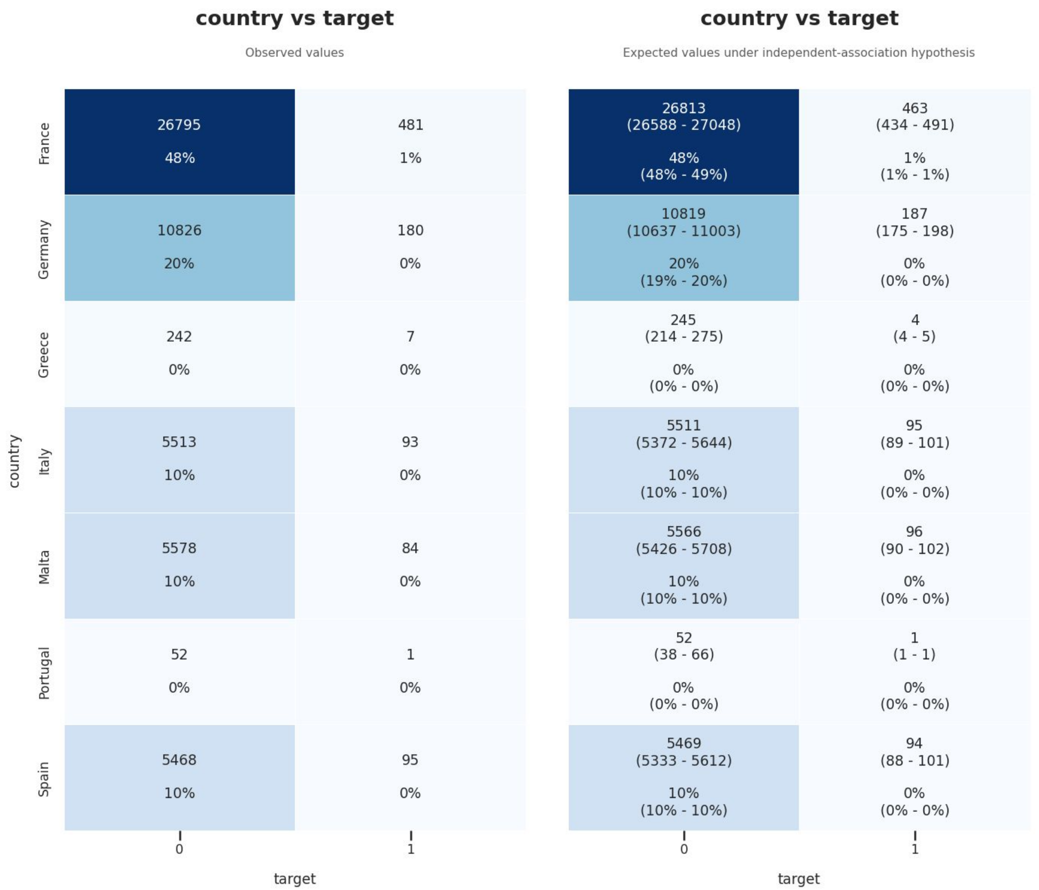
- Many features are highly correlated (multicollinearity)

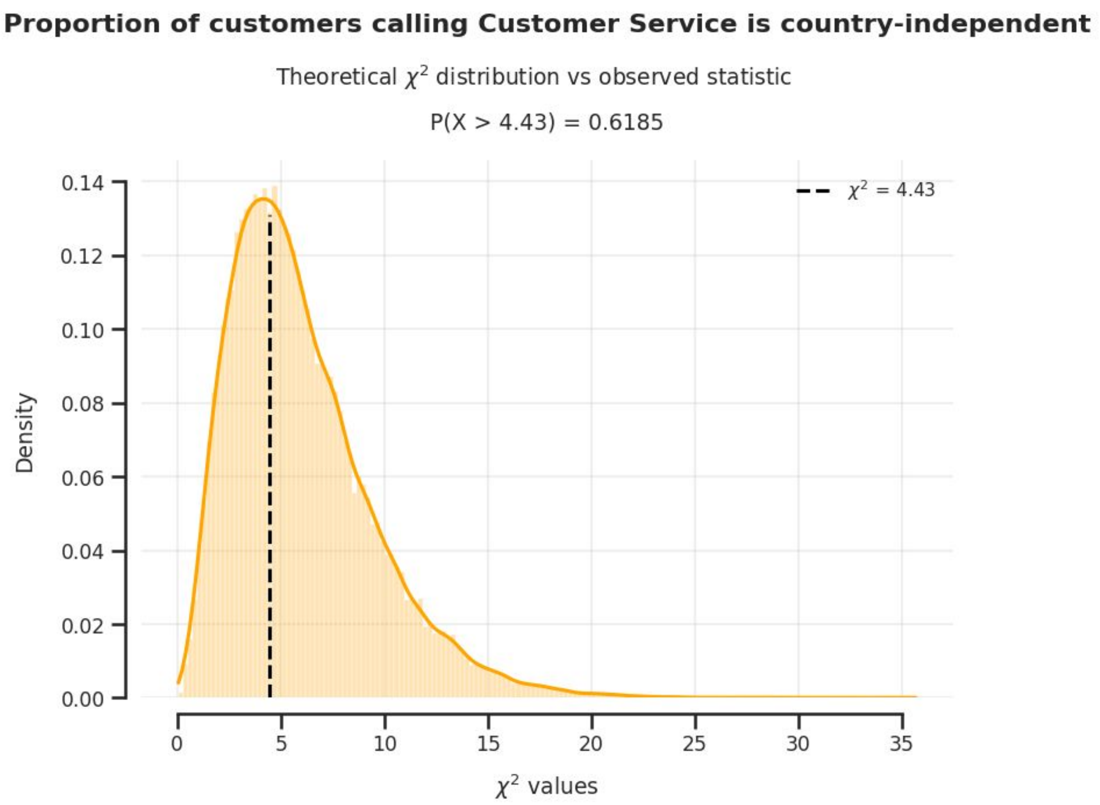


# Statistical Analysis

**Chi2 test:**

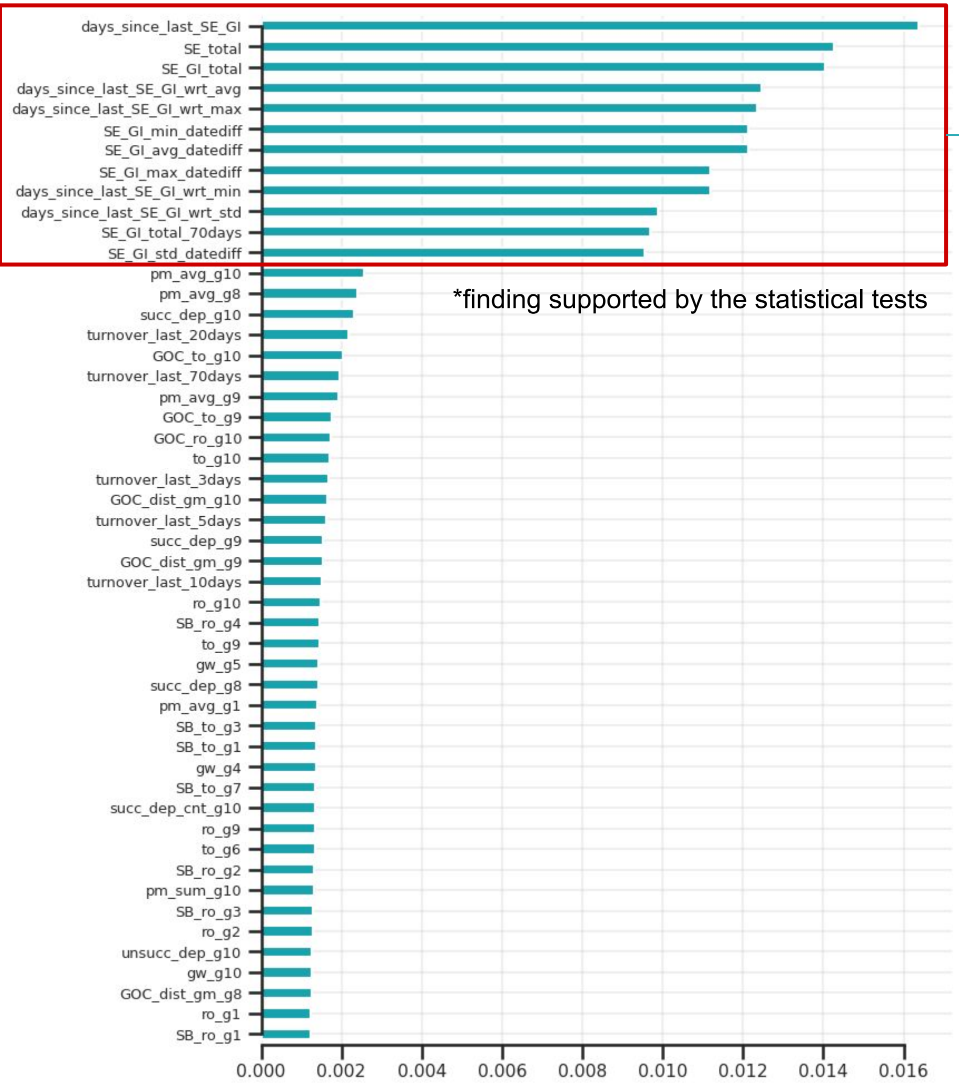
- The country feature **is not** associated with our target



# Statistical Analysis

**Mutual Information (MI) scores:**




According to the mutual information (MI) scores and statistical tests, features related to Gambling issues and self-exclusion might be important to predict if the the customer will or will not call the customer service.

# Feature Selection

- Removal of low-variance (quasi-constant) features (>0.01) → 3+1 features
- Winsorizing extreme outliers within the ML pipeline (99.99th percentile) → 0 features
- Removal of correlated features (keep the ones with the highest variance) → 69 features
- Select top 12 features performing similarly to a the models with all features

# Predictive Modeling

# Model Selection

Test models with default parameters and all variables results in poor recall or accuracy:

| | Model | Accuracy | AUC | Recall | Prec. | F1 | FPR | FNR | AUC-PR | TT (Sec) |
|---|---|---|---|---|---|---|---|---|---|---|
| **nb** | Naive Bayes | 0.1629 | 0.6843 | 0.9078 | 0.0181 | 0.0355 | 0.8500 | 0.0922 | 0.4637 | 1.0200 |
| **qda** | Quadratic Discriminant Analysis | 0.2420 | 0.5458 | 0.8511 | 0.0188 | 0.0367 | 0.7685 | 0.1489 | 0.4362 | 1.7300 |
| **lda** | Linear Discriminant Analysis | 0.9739 | 0.7765 | 0.2624 | 0.2467 | 0.2543 | 0.0138 | 0.7376 | 0.2608 | 1.8600 |
| **dt** | Decision Tree Classifier | 0.9670 | 0.5755 | 0.1702 | 0.1326 | 0.1491 | 0.0192 | 0.8298 | 0.1584 | 16.1600 |
| **ada** | Ada Boost Classifier | 0.9829 | 0.8056 | 0.0851 | 0.4800 | 0.1446 | 0.0016 | 0.9149 | 0.2903 | 30.3500 |
| **lightgbm** | Light Gradient Boosting Machine | 0.9827 | 0.8263 | 0.0851 | 0.4444 | 0.1429 | 0.0018 | 0.9149 | 0.2725 | 2.3200 |
| **gbc** | Gradient Boosting Classifier | 0.9820 | 0.8449 | 0.0780 | 0.3548 | 0.1279 | 0.0024 | 0.9220 | 0.2242 | 162.8400 |
| **xgboost** | Extreme Gradient Boosting | 0.9832 | 0.8065 | 0.0674 | 0.5429 | 0.1199 | 0.0010 | 0.9326 | 0.3130 | 1.6800 |
| **catboost** | CatBoost Classifier | 0.9831 | 0.8479 | 0.0603 | 0.5152 | 0.1079 | 0.0010 | 0.9397 | 0.2957 | 33.1800 |
| **ridge** | Ridge Classifier | 0.9835 | 0.5281 | 0.0567 | 0.6667 | 0.1046 | 0.0005 | 0.9433 | 0.3697 | 1.0200 |
| **rf** | Random Forest Classifier | 0.9833 | 0.7896 | 0.0142 | 1.0000 | 0.0280 | 0.0000 | 0.9858 | 0.5155 | 7.7300 |
| **et** | Extra Trees Classifier | 0.9830 | 0.7849 | 0.0035 | 0.5000 | 0.0070 | 0.0001 | 0.9965 | 0.2602 | 3.4800 |
| **lr** | Logistic Regression | 0.9806 | 0.5092 | 0.0000 | 0.0000 | 0.0000 | 0.0025 | 1.0000 | 0.0085 | 17.4100 |
| **knn** | K Neighbors Classifier | 0.9829 | 0.5203 | 0.0000 | 0.0000 | 0.0000 | 0.0002 | 1.0000 | 0.0085 | 0.8800 |
| **svm** | SVM - Linear Kernel | 0.9806 | 0.4988 | 0.0000 | 0.0000 | 0.0000 | 0.0024 | 1.0000 | 0.0085 | 1.6100 |
| **dummy** | Dummy Classifier | 0.9830 | 0.5000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 1.0000 | 0.5085 | 0.8200 |

# Model Selection

I wanted to optimize:

- Recall >70%
- Accuracy >70%
- ROC-AUC > 80%

Trade-off = lower precision

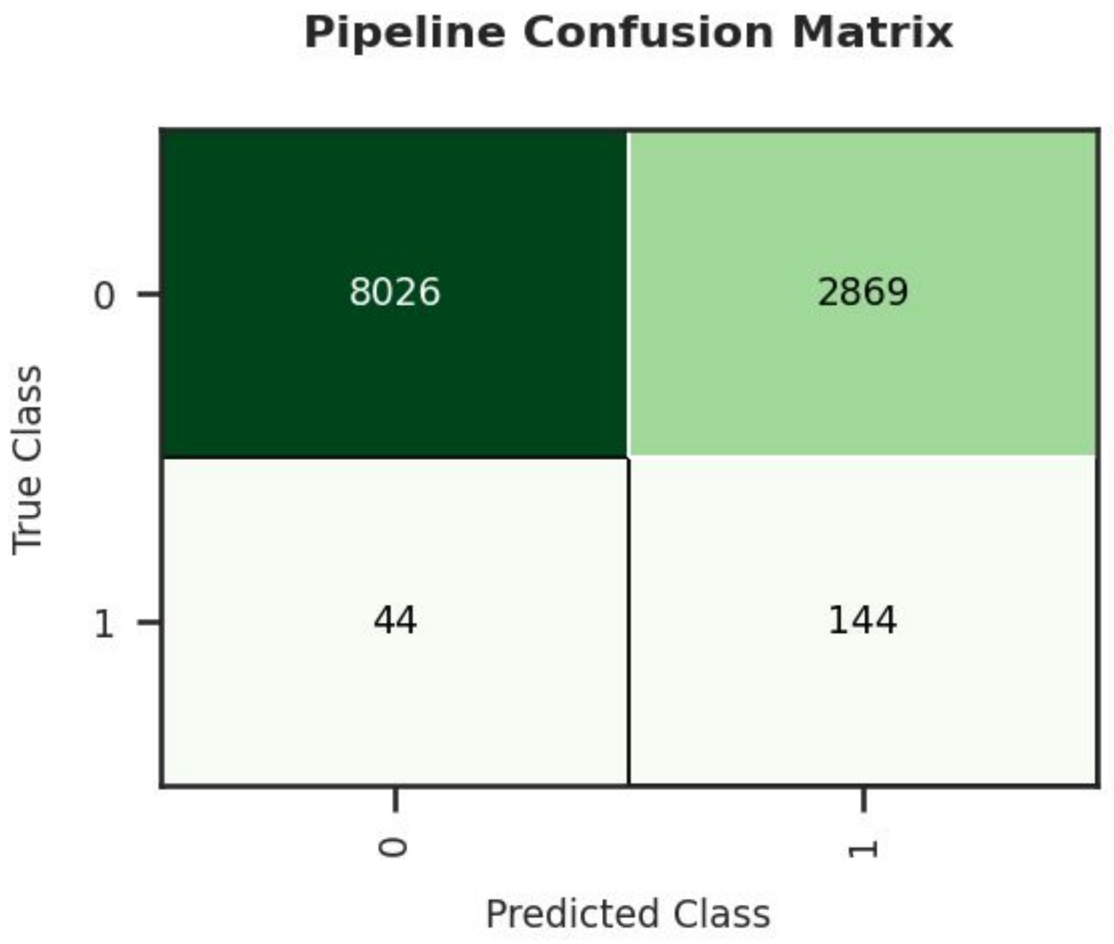
Experiment setup

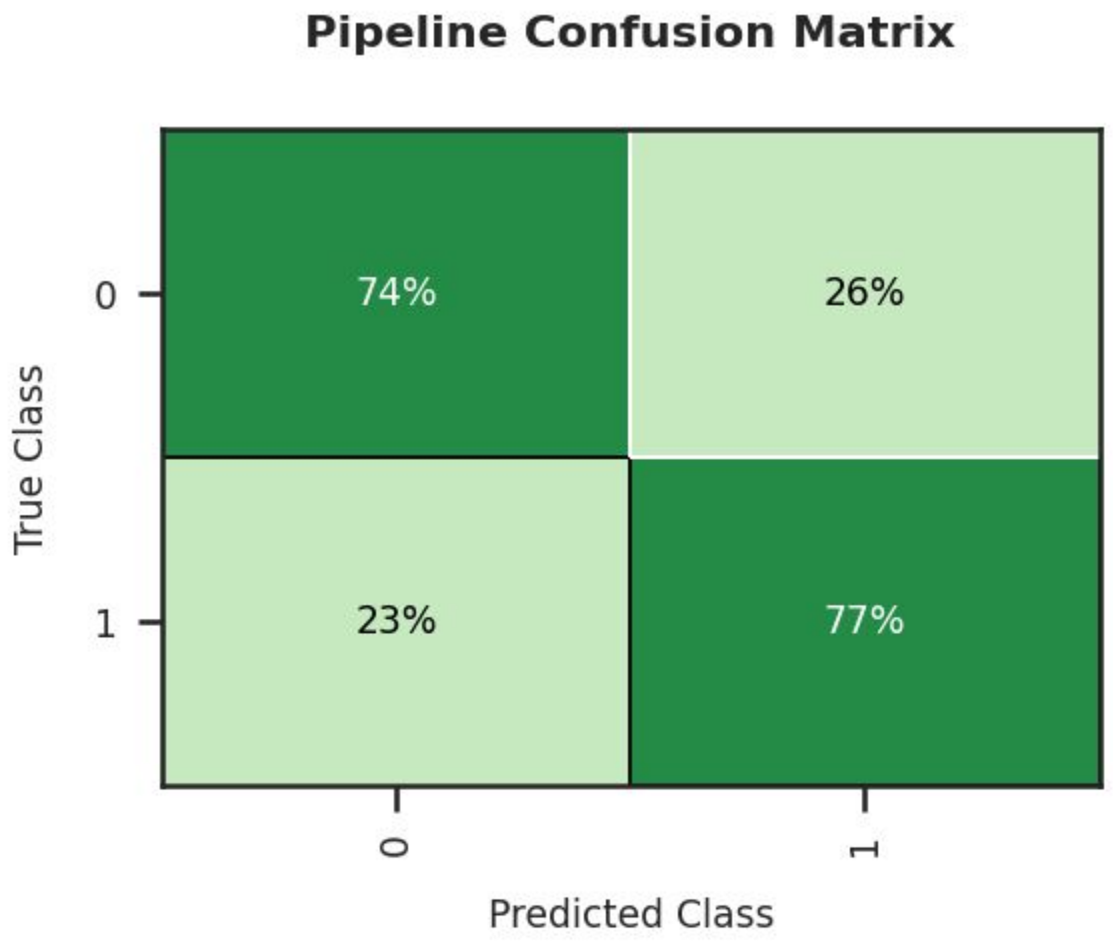
```
# Instantiate experiment
exp = ClassificationExperiment()

# Default experiment setup
exp.setup(
    df, target='target',
    fix_imbalance=True,
    fix_imbalance_method='RandomUnderSampler',
    normalize=True,
    normalize_method='robust',
    remove_multicollinearity=True,
    multicollinearity_threshold=0.9,
    low_variance_threshold=0.01,
    remove_outliers=False,
    feature_selection=False,
    use_gpu=True,
    session_id=42
)

# Set custom metrics
set_custom_metrics(exp)

# Compare available models
best = exp.compare_models(sort='recall', cross_validation=False)
```

Models with basic preprocessing, training set resample, and all features, provided better results:

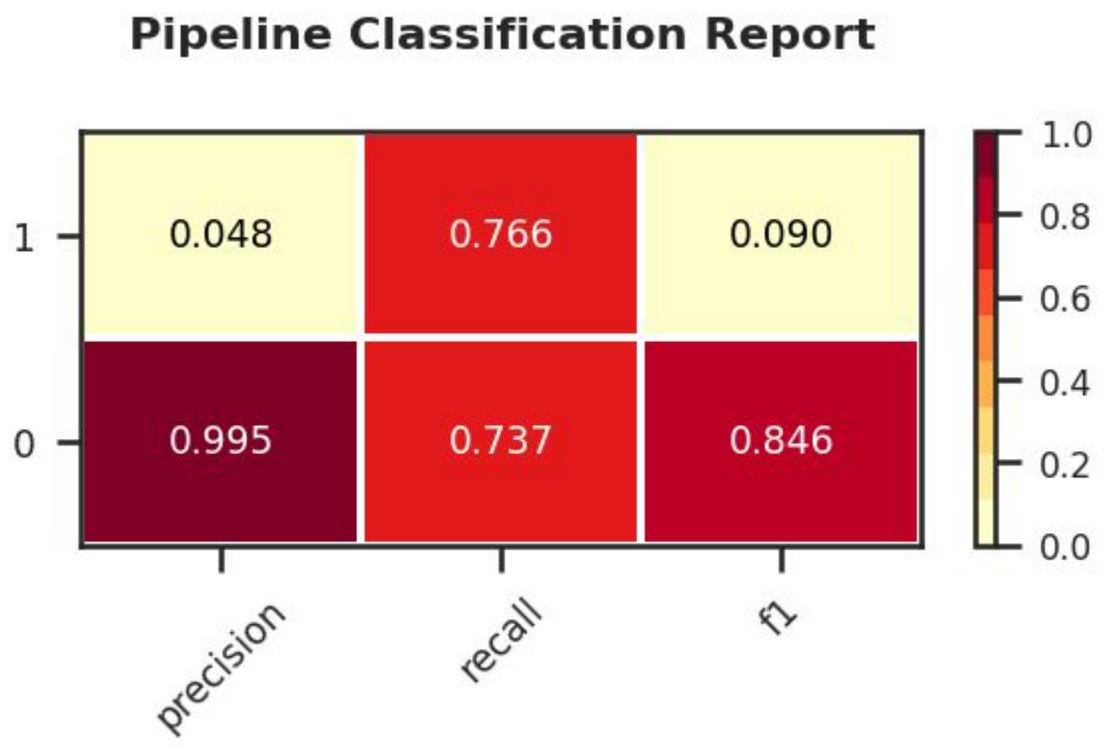
| | Model | Accuracy | AUC | Recall | Prec. | F1 | FPR | FNR | AUC-PR | TT (Sec) |
|---|---|---|---|---|---|---|---|---|---|---|
| **svm** | SVM - Linear Kernel | 0.3881 | 0.5912 | 0.8014 | 0.0219 | 0.0425 | 0.6190 | 0.1986 | 0.4133 | 8.9200 |
| **et** | Extra Trees Classifier | 0.7711 | 0.8135 | 0.7128 | 0.0512 | 0.0955 | 0.2279 | 0.2872 | 0.3844 | 9.0900 |
| **catboost** | CatBoost Classifier | 0.8075 | 0.8310 | 0.7128 | 0.0605 | 0.1116 | 0.1908 | 0.2872 | 0.3891 | 31.2400 |
| **xgboost** | Extreme Gradient Boosting | 0.7815 | 0.8206 | 0.7092 | 0.0533 | 0.0992 | 0.2173 | 0.2908 | 0.3837 | 9.1900 |
| **dt** | Decision Tree Classifier | 0.6811 | 0.6915 | 0.7021 | 0.0366 | 0.0695 | 0.3192 | 0.2979 | 0.3719 | 9.1500 |
| **lightgbm** | Light Gradient Boosting Machine | 0.8002 | 0.8301 | 0.6950 | 0.0571 | 0.1055 | 0.1980 | 0.3050 | 0.3787 | 9.6500 |
| **gbc** | Gradient Boosting Classifier | 0.8218 | 0.8331 | 0.6915 | 0.0635 | 0.1163 | 0.1760 | 0.3085 | 0.3801 | 13.3700 |
| **rf** | Random Forest Classifier | 0.8225 | 0.8293 | 0.6809 | 0.0629 | 0.1151 | 0.1751 | 0.3191 | 0.3746 | 9.2100 |
| **ada** | Ada Boost Classifier | 0.7719 | 0.7895 | 0.6702 | 0.0486 | 0.0906 | 0.2263 | 0.3298 | 0.3622 | 9.8300 |
| **lr** | Logistic Regression | 0.7402 | 0.7391 | 0.6489 | 0.0416 | 0.0781 | 0.2582 | 0.3511 | 0.3482 | 9.5600 |
| **lda** | Linear Discriminant Analysis | 0.7428 | 0.7029 | 0.5816 | 0.0379 | 0.0712 | 0.2544 | 0.4184 | 0.3133 | 8.9400 |
| **ridge** | Ridge Classifier | 0.7444 | 0.6626 | 0.5780 | 0.0380 | 0.0712 | 0.2528 | 0.4220 | 0.3116 | 9.0100 |
| **qda** | Quadratic Discriminant Analysis | 0.7183 | 0.6693 | 0.5390 | 0.0323 | 0.0609 | 0.2787 | 0.4610 | 0.2896 | 8.9600 |
| **knn** | K Neighbors Classifier | 0.6592 | 0.6197 | 0.5106 | 0.0254 | 0.0484 | 0.3382 | 0.4894 | 0.2722 | 9.0700 |
| **nb** | Naive Bayes | 0.8346 | 0.6823 | 0.3794 | 0.0399 | 0.0722 | 0.1576 | 0.6206 | 0.2149 | 8.9300 |
| **dummy** | Dummy Classifier | 0.9830 | 0.5000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 1.0000 | 0.5085 | 8.9000 |

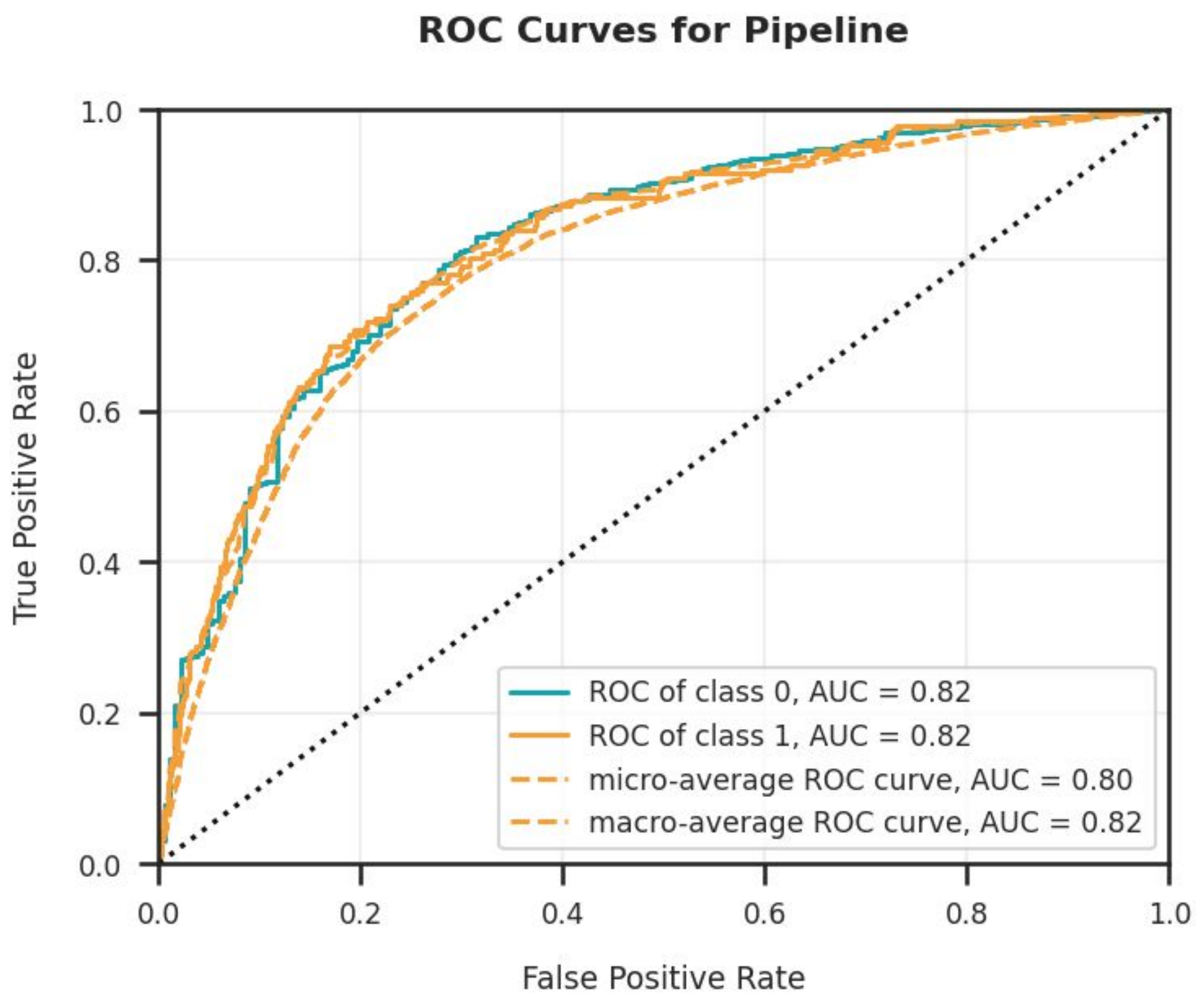
# Model evaluation



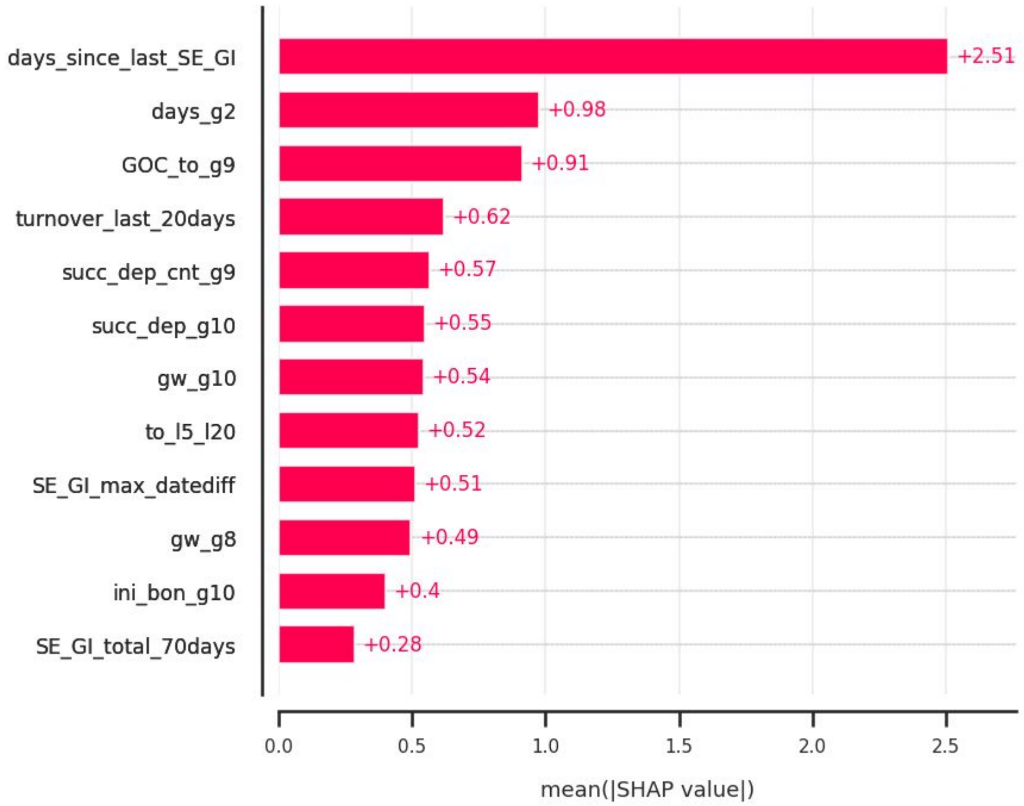



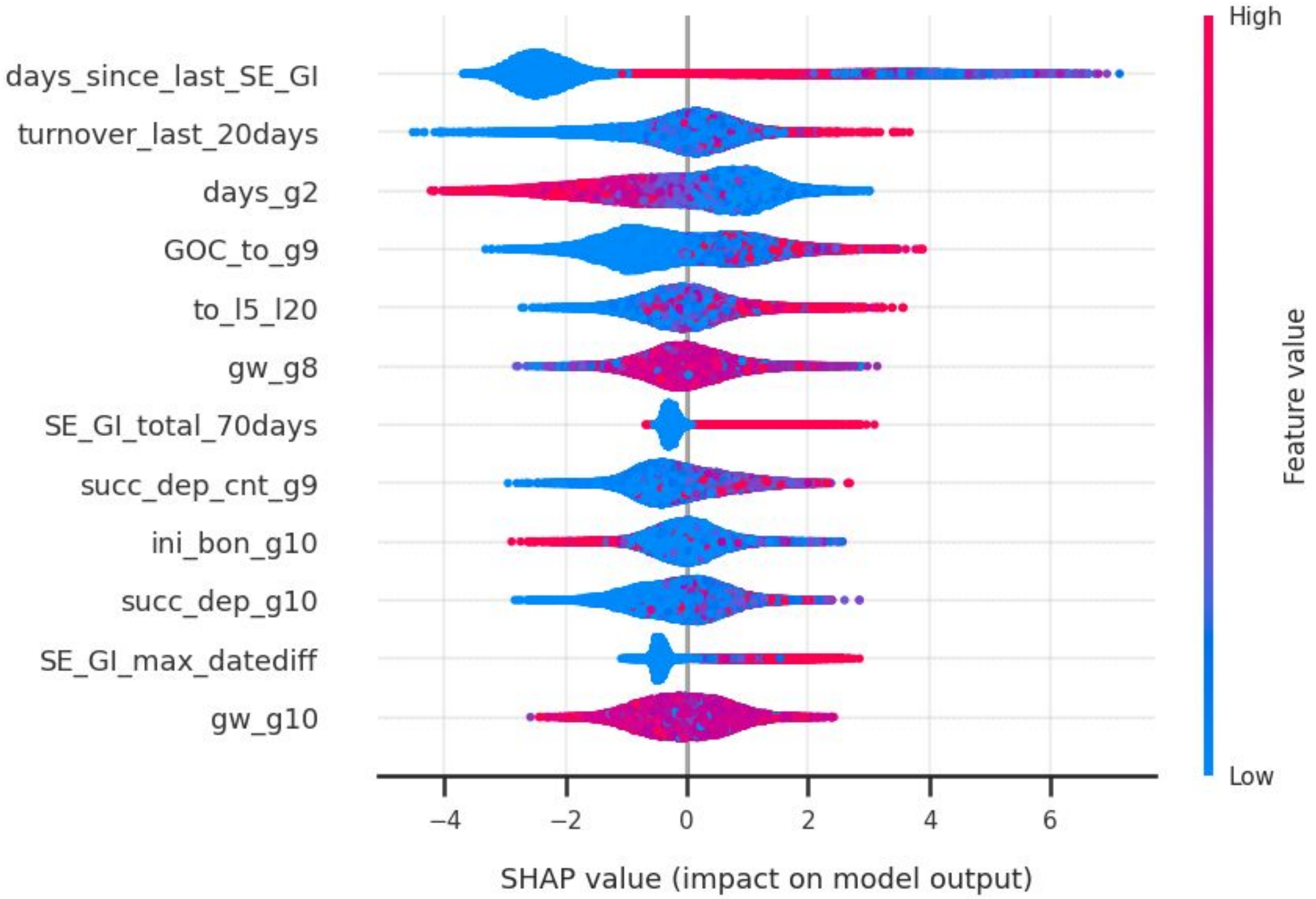
# Model evaluation



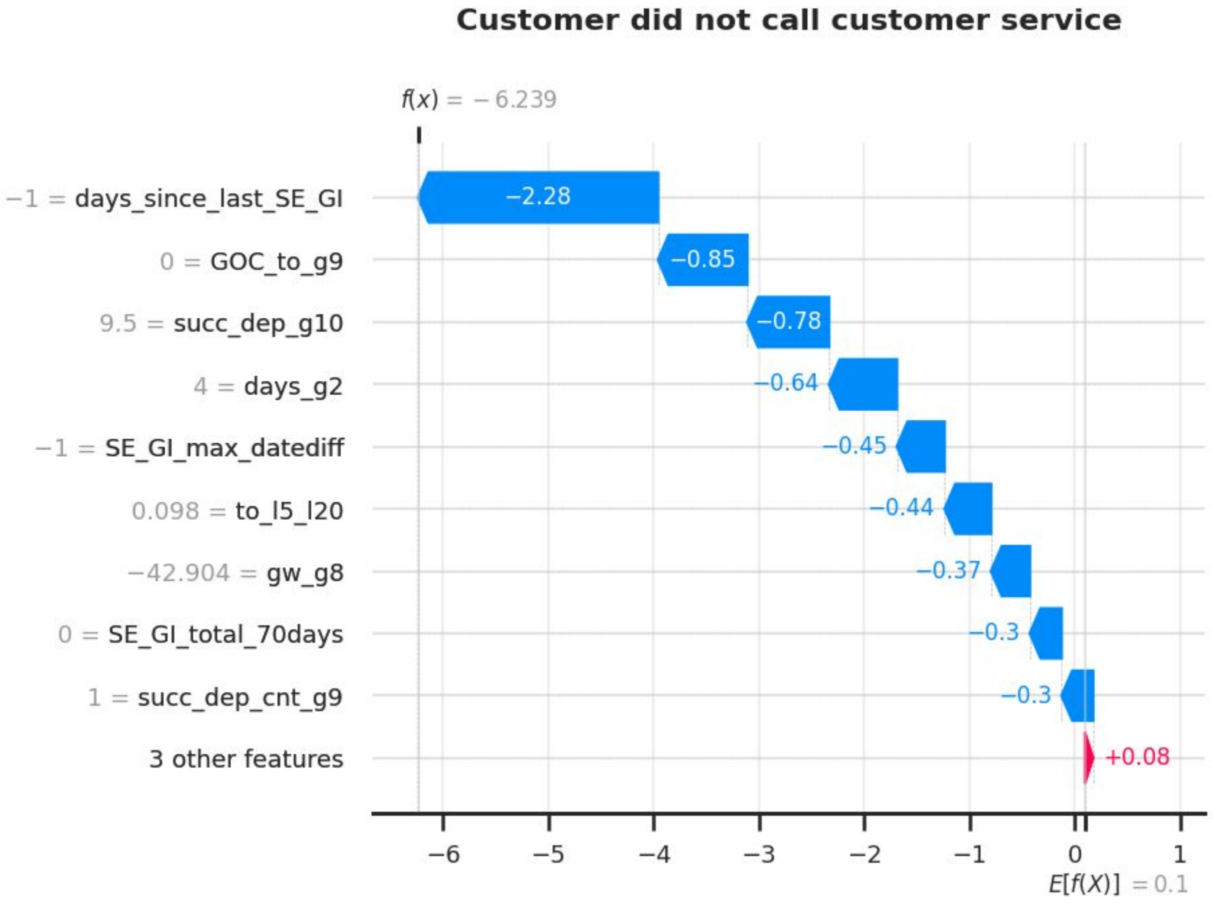



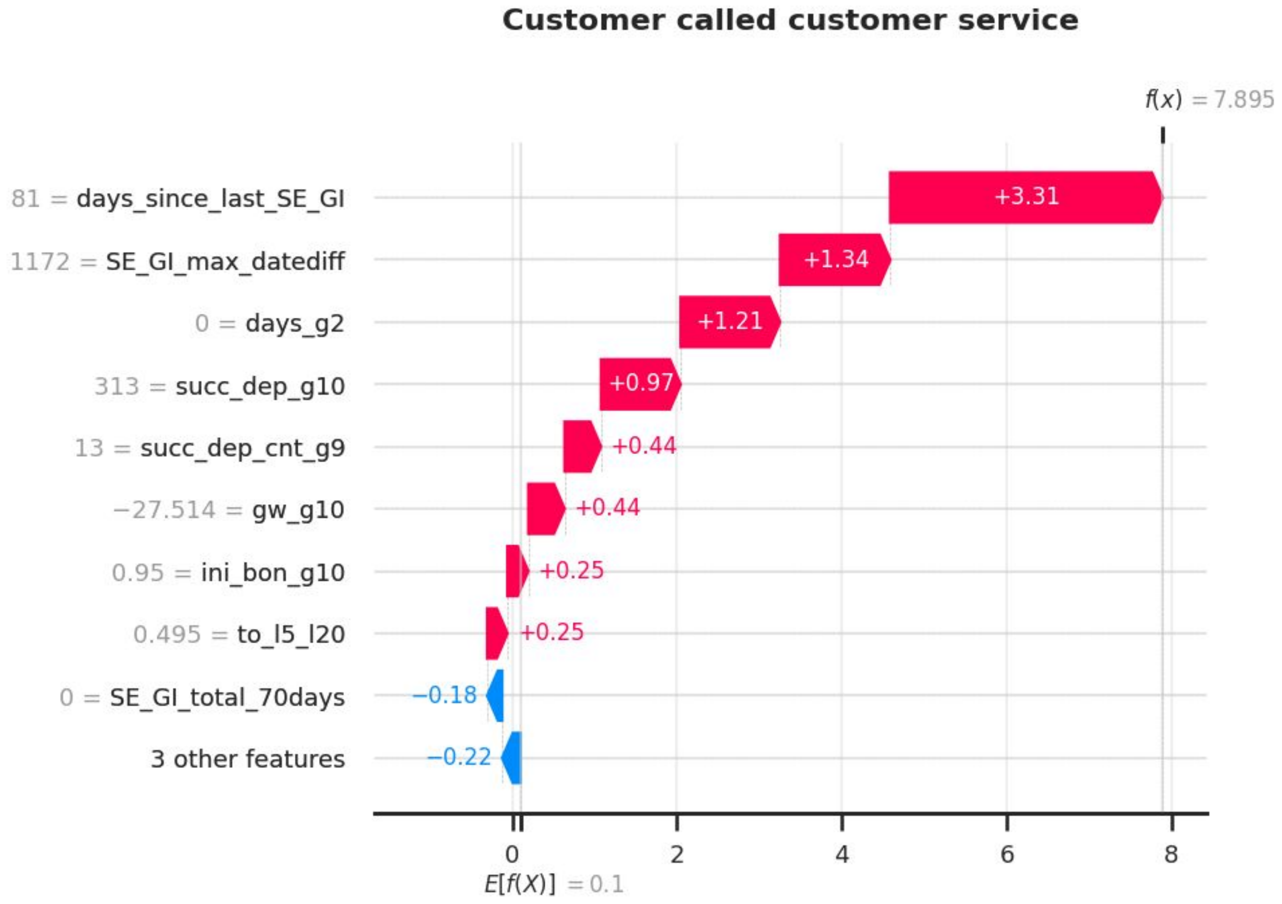
# Model Interpretation



# Model Interpretation



# Business alignment

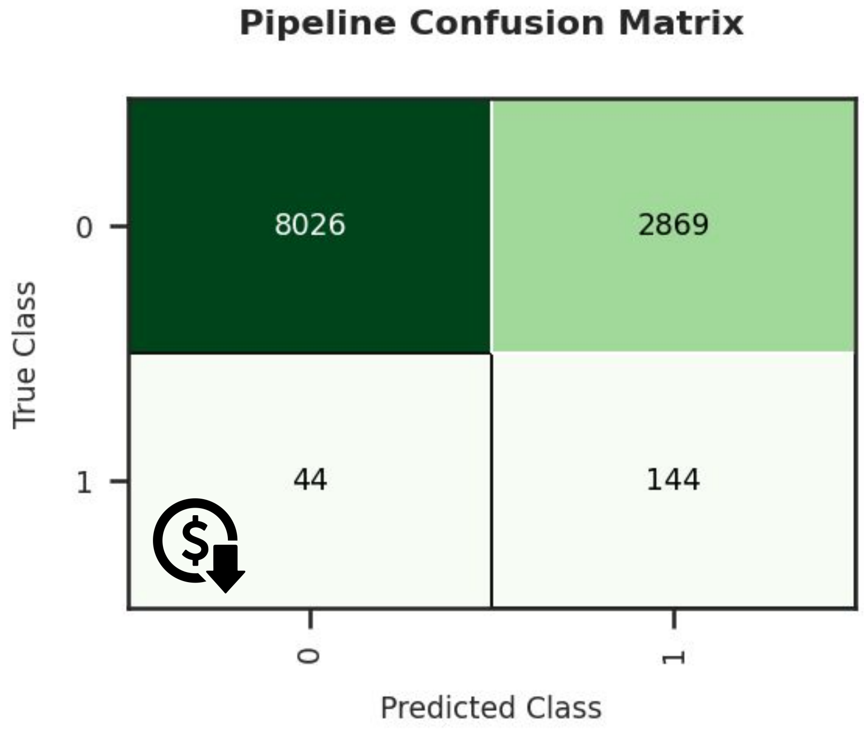
# Will this model be used in production?

## It depends!

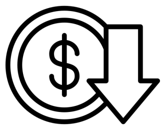
→ How much do we earn by contacting a customer?
→ What's the intervention action cost?
→ What about resources allocation?

## Possible improvements:

→ Brainstorming and collection of meaningful features
→ Extensive hyperparameter tuning on new models
→ Smarter feature selection using forward/backward selection




**Cost?**



**Earnings?**

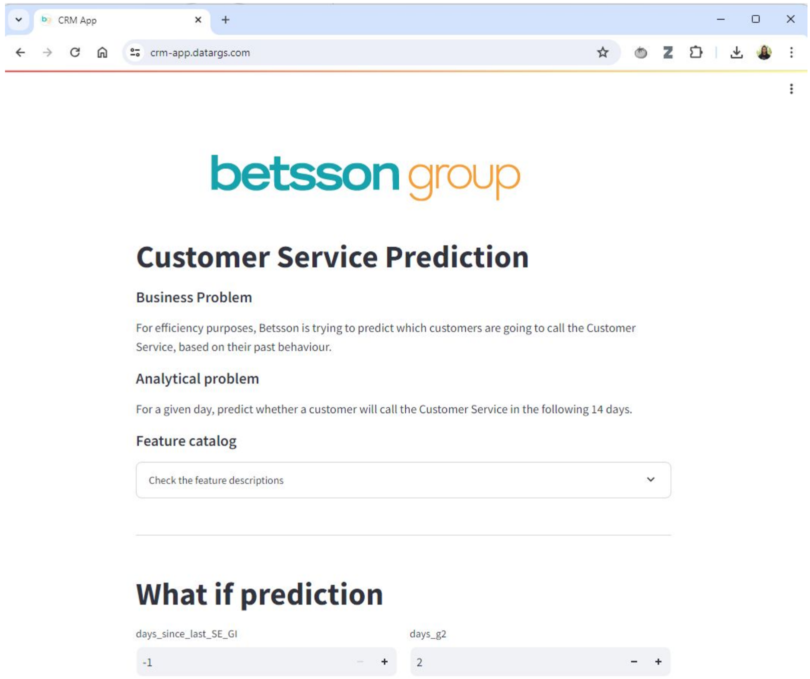
↑ **CLV**
↓ **Churn**

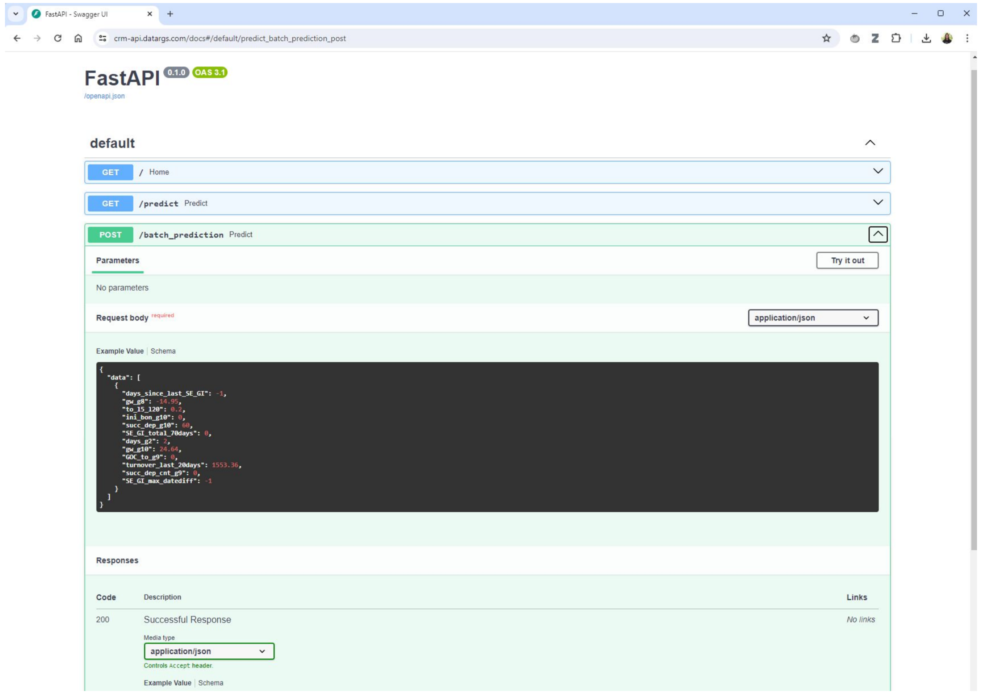
**Note:** Perhaps the customers who didn't contact Customer Service,
who have features similar to those who contacted them, would benefit from a contact as well.

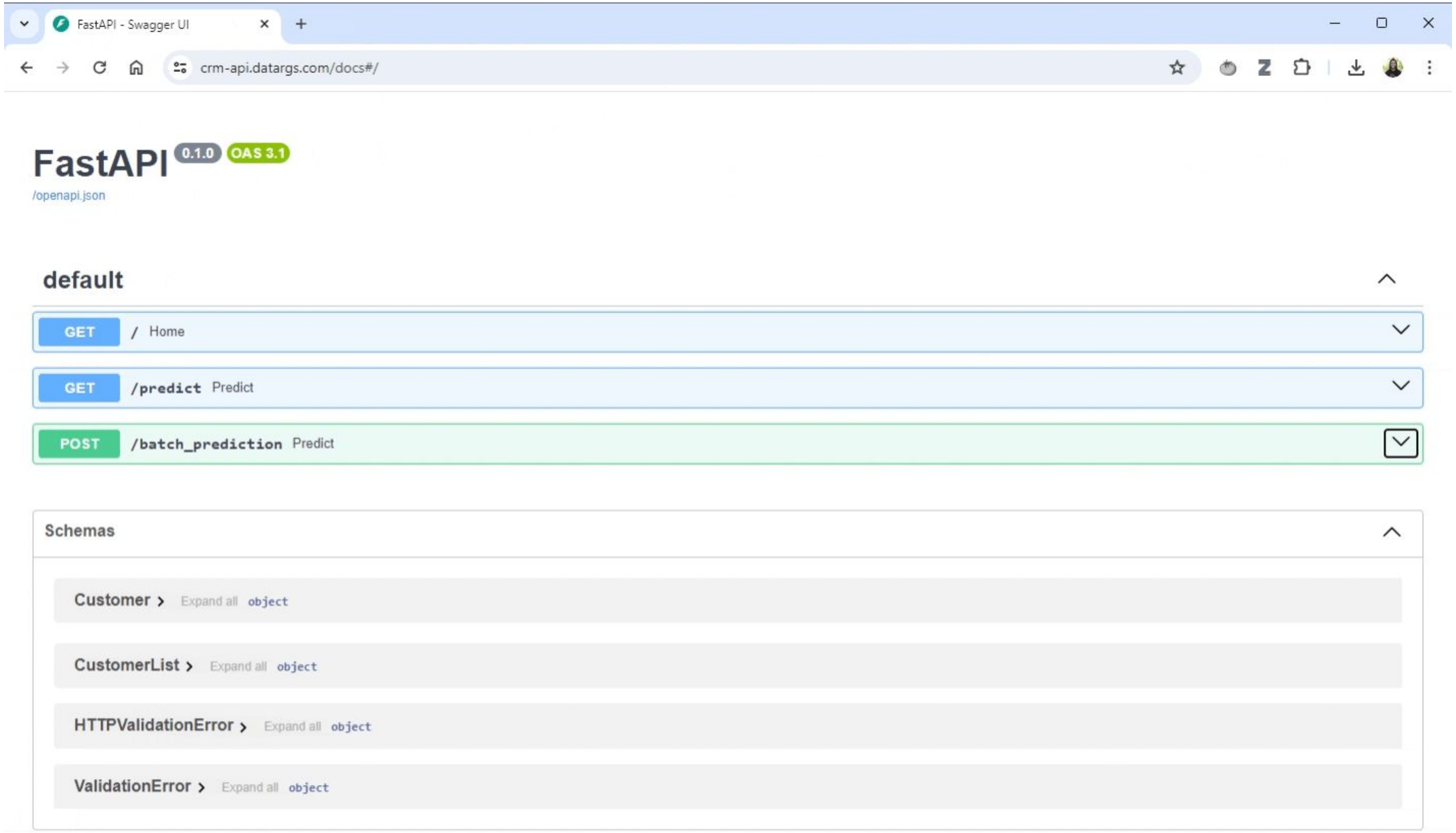
# Model Deployment

# Model Deployment

- API on Oracle Cloud with a custom domain: crm-api.datargs.com
- Streamlit App on Oracle Cloud with a custom domain: crm-api.datargs.com




FastAPI - Swagger UI
crm-api.datargs.com/docs#/
FastAPI 0.1.0 OAS 3.1
/openapi.json
default
GET / Home
GET /predict Predict
POST /batch_prediction Predict
Schemas
Customer Expand all object
CustomerList Expand all object
HTTPValidationError Expand all object
ValidationError Expand all object


# Thank you!