# Customer Churn Example (R)

ValidMind

2023-09-29

## Analysis

## Quickstart - Customer Churn Full Suite Model Documentation

This interactive notebook will guide you through documenting a model using the ValidMind Developer framework. We will use sample datasets provided by the library and train a simple classification model.

For this simple demonstration, we will use the following bank customer churn dataset from Kaggle: https://www.kaggle.com/code/kmalit/bank-customer-churn-prediction/data.

We will train a sample model and demonstrate the following documentation functionalities:

- Initializing the ValidMind Developer Framework

- Using a sample datasets provided by the library to train a simple classification model

- Running a test suite to quickly generate document about the data and model

### Initializing the Python environment

### Initializing the ValidMind Client Library

Log in to the ValidMind platform with your registered email address, and navigate to the Documentation Projects page.

#### Creating a new Documentation Project

*(Note: if a documentation project has already been created, you can skip this section and head directly "Finding Project API key and secret")*

Clicking on "Create a new project" allows to you to register a new documentation project for our demo model.

Select "Customer Churn model" from the Model drop-down, and "Initial Validation" as Type. Finally, click on "Create Project".

#### Finding the project API key and secret

In the "Client Integration" page of the newly created project, you will find the initialization code that allows the client library to associate documentation and tests with the appropriate project. The initialization code configures the following arguments:

- api_host: Location of the ValidMind API.
- api_key: Account API key.
- api_secret: Account Secret key.

Table 1: Numerical Variables

| Name | Count | Mean | Std | Min | 25% | 50% | 75% | 90% | |
|---|---|---|---|---|---|---|---|---|---|
| RowNumber | 8000 | 5.020520e+03 | 2885.7185 | 1 | 2519 | 5036 | 7512 | 9015 | |
| CustomerId | 8000 | 1.569047e+07 | 71902.4733 | 15565701 | 15628164 | 15690144 | 15752378 | 15790809 | 1580 |
| CreditScore | 8000 | 6.501596e+02 | 96.8462 | 350 | 583 | 652 | 717 | 778 | |
| Age | 8000 | 3.894890e+01 | 10.4590 | 18 | 32 | 37 | 44 | 53 | |
| Tenure | 8000 | 5.033900e+00 | 2.8853 | 0 | 3 | 5 | 8 | 9 | |
| Balance | 8000 | 7.643410e+04 | 62612.2513 | 0 | 0 | 97264 | 128045 | 149545 | 16 |
| NumOfProducts | 8000 | 1.532500e+00 | 0.5805 | 1 | 1 | 1 | 2 | 2 | |
| HasCrCard | 8000 | 7.026000e-01 | 0.4571 | 0 | 0 | 1 | 1 | 1 | |
| IsActiveMember | 8000 | 5.199000e-01 | 0.4996 | 0 | 0 | 1 | 1 | 1 | |
| EstimatedSalary | 8000 | 9.979019e+04 | 57520.5089 | 12 | 50857 | 99505 | 149216 | 179486 | 18 |

- project: The project identifier. The 'project' argument is mandatory since it allows the library to associate all data collected with a specific account project.

The code snippet can be copied and pasted directly in the cell below to initialize the ValidMind Developer Framework when run:

```
vm_r <- vm(
  api_key="b34dfe4dcb5491212be3eefe77c85cd6",
  api_secret="40f8d2d583baa9e730a7f8872dd57e2f4657c7918c13fa259ba7ccda8a60e858",
  project="clmp6k8e800ds19mot0zu8o34",
  python_version=python_version,
  api_host="https://api.dev.vm.validmind.ai/api/v1/tracking"
)
```

## Load the Demo Dataset

For the purpose of this demonstration, we will use a sample dataset provided by the ValidMind library.

```
# Read the dataset
data <- read.csv('../datasets/bank_customer_churn.csv')
```

**Initialize a dataset object for ValidMind** Before running the test plan, we must first initialize a ValidMind dataset object using the `init_dataset` function from the `vm` module. This function takes in arguments: `dataset` which is the dataset that we want to analyze; `target_column` which is used to identify the target variable; `class_labels` which is used to identify the labels used for classification model training.

```
vm_dataset = vm_r$init_dataset(
    dataset=data,
    target_column="Exited",
    class_labels=list("0" = "Did not exit", "1" = "Exited")
)
```

## Run the Data Validation Test Plan

```
tabular_suite_results <- vm_r$run_test_suite("tabular_dataset", dataset=vm_dataset)
```

```
## HBox(children=(Label(value='Running test suite...'), IntProgress(value=0, max=24)))
```

```
raw_summary_results <- summarize_results(tabular_suite_results)
```

Table 2: Categorical Variables

| Name | Count | Number of Unique Values | Top Value | Top Value Frequency | Top Value Frequency % |
|---|---|---|---|---|---|
| Surname | 8000 | 2616 | Smith | 29 | 0.3625 |
| Geography | 8000 | 3 | France | 4010 | 50.1250 |
| Gender | 8000 | 2 | Male | 4396 | 54.9500 |

Table 3: Class Imbalance Results for Column Exited

| Exited | Percentage of Rows (%) | Pass/Fail |
|---|---|---|
| 0 | 79.80% | Pass |
| 1 | 20.20% | Pass |

Table 4: High Cardinality Results for Dataset

| Column | Number of Distinct Values | Percentage of Distinct Values (%) | Pass/Fail |
|---|---|---|---|
| Geography | 3 | 0.0375 | Pass |
| Gender | 2 | 0.0250 | Pass |
| NumOfProducts | 4 | 0.0500 | Pass |
| HasCrCard | 2 | 0.0250 | Pass |
| IsActiveMember | 2 | 0.0250 | Pass |
| Exited | 2 | 0.0250 | Pass |

Table 5: High Pearson Correlation Results for Dataset

| Columns | Coefficient | Pass/Fail |
|---|---|---|
| (Balance, NumOfProducts) | -0.3044646 | Fail |
| (Age, Exited) | 0.2810129 | Pass |
| (Exited, IsActiveMember) | -0.1515037 | Pass |
| (Balance, Exited) | 0.1174399 | Pass |
| (Age, IsActiveMember) | 0.0873114 | Pass |
| (Exited, NumOfProducts) | -0.0523056 | Pass |
| (Age, NumOfProducts) | -0.0306226 | Pass |
| (CreditScore, IsActiveMember) | 0.0305541 | Pass |
| (IsActiveMember, Tenure) | -0.0293018 | Pass |
| (Age, Balance) | 0.0289801 | Pass |

Table 6: Missing Values Results for Dataset

| Column | Number of Missing Values | Percentage of Missing Values (%) | Pass/Fail |
|---|---|---|---|
| RowNumber | 0 | 0 | Pass |
| CustomerId | 0 | 0 | Pass |
| Surname | 0 | 0 | Pass |
| CreditScore | 0 | 0 | Pass |
| Geography | 0 | 0 | Pass |
| Gender | 0 | 0 | Pass |
| Age | 0 | 0 | Pass |
| Tenure | 0 | 0 | Pass |
| Balance | 0 | 0 | Pass |
| NumOfProducts | 0 | 0 | Pass |
| HasCrCard | 0 | 0 | Pass |
| IsActiveMember | 0 | 0 | Pass |
| EstimatedSalary | 0 | 0 | Pass |
| Exited | 0 | 0 | Pass |

Table 7: Skewness Results for Dataset

| Column | Skewness | Pass/Fail |
|---|---|---|
| RowNumber | -0.0059207 | Pass |
| CustomerId | 0.0100323 | Pass |
| CreditScore | -0.0619516 | Pass |
| Age | 1.0245221 | Fail |
| Tenure | 0.0076920 | Pass |
| Balance | -0.1352769 | Pass |
| EstimatedSalary | 0.0095104 | Pass |

Table 8: Unique Rows Results for Dataset

| Column | Number of Unique Values | Percentage of Unique Values (%) | Pass/Fail |
|---|---|---|---|
| RowNumber | 8000 | 100.0000 | Fail |
| CustomerId | 8000 | 100.0000 | Fail |
| Surname | 2616 | 32.7000 | Pass |
| CreditScore | 452 | 5.6500 | Pass |
| Geography | 3 | 0.0375 | Pass |
| Gender | 2 | 0.0250 | Pass |
| Age | 69 | 0.8625 | Pass |
| Tenure | 11 | 0.1375 | Pass |
| Balance | 5088 | 63.6000 | Pass |
| NumOfProducts | 4 | 0.0500 | Pass |
| HasCrCard | 2 | 0.0250 | Pass |
| IsActiveMember | 2 | 0.0250 | Pass |
| EstimatedSalary | 8000 | 100.0000 | Fail |
| Exited | 2 | 0.0250 | Pass |

Table 9: Zeros Results for Dataset

| Column | Number of Zero Values | Percentage of Zero Values (%) | Pass/Fail |
|---|---|---|---|
| Tenure | 323 | 4.0375 | Fail |
| Balance | 2912 | 36.4000 | Fail |

Table 10: Model Summary Results

| term | estimate | std.error | statistic | p.value |
|---|---|---|---|---|
| (Intercept) | -3.6279796 | 0.3351288 | -10.8256285 | 0.0000000 |
| CreditScore | -0.0008556 | 0.0003737 | -2.2895400 | 0.0220480 |
| Age | 0.0717281 | 0.0034578 | 20.7438009 | 0.0000000 |
| Tenure | -0.0167515 | 0.0124169 | -1.3490929 | 0.1773071 |
| Balance | 0.0000017 | 0.0000007 | 2.4866260 | 0.0128961 |
| NumOfProducts | -0.1089455 | 0.0637106 | -1.7100051 | 0.0872649 |
| HasCrCard | -0.0409089 | 0.0790667 | -0.5173971 | 0.6048790 |
| IsActiveMember | -1.0521795 | 0.0769549 | -13.6726793 | 0.0000000 |
| EstimatedSalary | 0.0000005 | 0.0000006 | 0.7195061 | 0.4718291 |
| GeographyFrance | -0.0541293 | 0.0944621 | -0.5730267 | 0.5666266 |
| GeographyGermany | 0.7316662 | 0.1054571 | 6.9380484 | 0.0000000 |
| GeographySpain | NA | NA | NA | NA |
| GenderFemale | 0.4824388 | 0.0726963 | 6.6363583 | 0.0000000 |
| GenderMale | NA | NA | NA | NA |

## Run the Model Validation Test Plan

We will need to preprocess the dataset and produce the training, test and validation splits first.

### Preprocess the Raw Dataset

```
# Handle categorical variables using one-hot encoding and remove unnecessary columns
data <- data %>% select(-RowNumber, -CustomerId, -Surname)
geo_dummies <- model.matrix(~Geography - 1, data=data)
gender_dummies <- model.matrix(~Gender - 1, data=data)
data <- data %>% select(-Geography, -Gender)
data <- cbind(data, geo_dummies, gender_dummies)
```

```
# Split the dataset into training and testing sets
set.seed(123) # Setting seed for reproducibility
split <- sample.split(data$Exited, SplitRatio = 0.7)
train_data <- subset(data, split == TRUE)
test_data <- subset(data, split == FALSE)
```

```
# Run a logistic regression model
model <- glm(Exited ~ ., family = binomial(link = 'logit'), data = train_data)
```

```
model_path = save_model(model)
```

```
vm_train_ds = vm_r$init_dataset(
  dataset=train_data,
  target_column="Exited"
)

vm_test_ds = vm_r$init_dataset(
  dataset=test_data,
  target_column="Exited"
)

vm_model = vm_r$init_r_model(
  model_path=model_path,
```

Table 11: NULL

| Attribute | Value |
|---|---|
| Modeling Technique | glm.fit (Family: binomial, Link function: logit) |
| Modeling Framework | R |
| Framework Version | N/A |
| Programming Language | R version 4.3.1 (2023-06-16) |

Table 12: NULL

| Dataset | Size | Proportion |
|---|---|---|
| Training | 5600 | 70.00% |
| Test | 2400 | 30.00% |
| Total | 8000 | 100% |

```
  train_ds=vm_train_ds,
  test_ds=vm_test_ds,
)
```

**Run the Binary Classification Test Plan**

```
model_validation_results = vm_r$run_test_suite("classifier_model_validation", model=vm_model, fail_fast
```

```
## HBox(children=(Label(value='Running test suite...'), IntProgress(value=0, max=36)))
```

```
raw_model_results <- summarize_results(model_validation_results)
```

Table 13: Classification Report

| Metric | Value |
|---|---|
| Precision | 0.6214099 |
| Recall | 0.2104332 |
| F1 | 0.3143989 |
| Accuracy | 0.8146429 |
| ROC AUC | 0.5889938 |

Table 14: Classification Report

| Metric | Value |
|---|---|
| Precision | 0.6153846 |
| Recall | 0.1814433 |
| F1 | 0.2802548 |
| Accuracy | 0.8116667 |
| ROC AUC | 0.5763613 |

Table 15: Population Stability Index for Training and Test Datasets

| Bin | Count Initial | Percent Initial (%) | Count New | Percent New (%) | PSI |
|---|---|---|---|---|---|
| 0 | 1845 | 32.9464286 | 825 | 34.3750000 | 0.0006064 |
| 1 | 1570 | 28.0357143 | 664 | 27.6666667 | 0.0000489 |
| 2 | 900 | 16.0714286 | 403 | 16.7916667 | 0.0003157 |
| 3 | 501 | 8.9464286 | 213 | 8.8750000 | 0.0000057 |
| 4 | 305 | 5.4464286 | 124 | 5.1666667 | 0.0001475 |
| 5 | 230 | 4.1071429 | 76 | 3.1666667 | 0.0024457 |
| 6 | 131 | 2.3392857 | 51 | 2.1250000 | 0.0002059 |
| 7 | 73 | 1.3035714 | 28 | 1.1666667 | 0.0001519 |
| 8 | 37 | 0.6607143 | 12 | 0.5000000 | 0.0004479 |
| 9 | 8 | 0.1428571 | 4 | 0.1666667 | 0.0000367 |

Table 16: Minimum Accuracy Test on Test Data

| Score | Threshold | Pass/Fail |
|---|---|---|
| 0.8116667 | 0.7 | Pass |

Table 17: Minimum F1 Score Test

| Score | Threshold | Pass/Fail |
|---|---|---|
| 0.2802548 | 0.5 | Fail |

Table 18: Minimum ROC AUC Score Test

| Score | Threshold | Pass/Fail |
|---|---|---|
| 0.5763613 | 0.5 | Pass |

Table 19: Training-Test Degradation Test

| Metric | Train Score | Test Score | Degradation (%) | Pass/Fail |
|---|---|---|---|---|
| Accuracy | 0.8146429 | 0.8116667 | 0.3653368 | Pass |
| Precision | 0.8146429 | 0.8116667 | 0.3653368 | Pass |
| Recall | 0.8146429 | 0.8116667 | 0.3653368 | Pass |
| F1 | 0.8146429 | 0.8116667 | 0.3653368 | Pass |

Table 20: Overfit Regions Data

| slice | training records | feature | training accuracy | test accuracy | gap |
|---|---|---|---|---|---|
| (349.5, 400.0] | 10 | CreditScore | 20.000000 | 0.00000 | 20.000000 |
| (17.926, 25.4] | 321 | Age | 94.080997 | 89.03226 | 5.048739 |
| (47.6, 55.0] | 504 | Age | 60.317460 | 54.76190 | 5.555556 |
| (77.2, 84.6] | 12 | Age | 8.333333 | 0.00000 | 8.333333 |
| (6.0, 7.0] | 558 | Tenure | 86.559140 | 80.82707 | 5.732072 |
| (25089.809, 50179.618] | 35 | Balance | 85.714286 | 63.15789 | 22.556391 |
| (125449.045, 150538.854] | 1015 | Balance | 80.886700 | 76.55860 | 4.328096 |
| (175628.663, 200718.472] | 113 | Balance | 78.761062 | 63.15789 | 15.603167 |
| (200718.472, 225808.281] | 13 | Balance | 53.846154 | 30.76923 | 23.076923 |
| (-188.338, 20003.339] | 566 | EstimatedSalary | 82.862191 | 78.57143 | 4.290762 |

Table 21: Weakspots Test

| slice | shape | feature | accuracy | precision | recall | f1 | Dataset Type |
|---|---|---|---|---|---|---|---|
| (349.5, 400.0] | 10 | CreditScore | 0.2000000 | 1.0000000 | 0.2000000 | 0.3333333 | Training |
| (400.0, 450.0] | 89 | CreditScore | 0.8202247 | 0.7142857 | 0.4545455 | 0.5555556 | Training |
| (450.0, 500.0] | 249 | CreditScore | 0.8152610 | 0.7200000 | 0.3157895 | 0.4390244 | Training |
| (500.0, 550.0] | 595 | CreditScore | 0.7949580 | 0.6666667 | 0.2446043 | 0.3578947 | Training |
| (550.0, 600.0] | 827 | CreditScore | 0.8210399 | 0.7166667 | 0.2471264 | 0.3675214 | Training |
| (600.0, 650.0] | 1020 | CreditScore | 0.8107843 | 0.5468750 | 0.1758794 | 0.2661597 | Training |
| (650.0, 700.0] | 1084 | CreditScore | 0.8256458 | 0.5409836 | 0.1701031 | 0.2588235 | Training |
| (700.0, 750.0] | 828 | CreditScore | 0.8200483 | 0.5909091 | 0.1656051 | 0.2587065 | Training |
| (750.0, 800.0] | 516 | CreditScore | 0.8217054 | 0.6750000 | 0.2547170 | 0.3698630 | Training |
| (800.0, 850.0] | 382 | CreditScore | 0.8036649 | 0.4545455 | 0.1369863 | 0.2105263 | Training |
| (349.5, 400.0] | 4 | CreditScore | 0.0000000 | 0.0000000 | 0.0000000 | 0.0000000 | Test |
| (400.0, 450.0] | 46 | CreditScore | 0.8043478 | 0.8000000 | 0.3333333 | 0.4705882 | Test |
| (450.0, 500.0] | 118 | CreditScore | 0.8474576 | 0.4285714 | 0.1764706 | 0.2500000 | Test |
| (500.0, 550.0] | 214 | CreditScore | 0.8224299 | 0.4666667 | 0.1891892 | 0.2692308 | Test |
| (550.0, 600.0] | 317 | CreditScore | 0.8233438 | 0.6875000 | 0.1774194 | 0.2820513 | Test |
| (600.0, 650.0] | 467 | CreditScore | 0.8094218 | 0.6538462 | 0.1752577 | 0.2764228 | Test |
| (650.0, 700.0] | 480 | CreditScore | 0.7895833 | 0.5312500 | 0.1650485 | 0.2518519 | Test |
| (700.0, 750.0] | 385 | CreditScore | 0.8000000 | 0.7647059 | 0.1511628 | 0.2524272 | Test |
| (750.0, 800.0] | 230 | CreditScore | 0.8347826 | 0.5789474 | 0.2682927 | 0.3666667 | Test |
| (800.0, 850.0] | 139 | CreditScore | 0.8417266 | 0.8333333 | 0.1923077 | 0.3125000 | Test |
| (17.926, 25.4] | 321 | Age | 0.9408100 | 0.0000000 | 0.0000000 | 0.0000000 | Training |
| (25.4, 32.8] | 1213 | Age | 0.9216818 | 0.0000000 | 0.0000000 | 0.0000000 | Training |
| (32.8, 40.2] | 2005 | Age | 0.8708229 | 0.6666667 | 0.0076923 | 0.0152091 | Training |
| (40.2, 47.6] | 1101 | Age | 0.7275204 | 0.6376812 | 0.1379310 | 0.2268041 | Training |
| (47.6, 55.0] | 504 | Age | 0.6031746 | 0.8301887 | 0.3259259 | 0.4680851 | Training |
| (55.0, 62.4] | 256 | Age | 0.7226562 | 0.8488372 | 0.5572519 | 0.6728111 | Training |
| (62.4, 69.8] | 111 | Age | 0.7567568 | 0.5531915 | 0.8125000 | 0.6582278 | Training |
| (69.8, 77.2] | 75 | Age | 0.2800000 | 0.0689655 | 1.0000000 | 0.1290323 | Training |
| (77.2, 84.6] | 12 | Age | 0.0833333 | 0.0833333 | 1.0000000 | 0.1538462 | Training |
| (84.6, 92.0] | 2 | Age | 0.0000000 | 0.0000000 | 0.0000000 | 0.0000000 | Training |
| (17.926, 25.4] | 155 | Age | 0.8903226 | 0.0000000 | 0.0000000 | 0.0000000 | Test |
| (25.4, 32.8] | 519 | Age | 0.9248555 | 0.0000000 | 0.0000000 | 0.0000000 | Test |
| (32.8, 40.2] | 913 | Age | 0.8707558 | 0.0000000 | 0.0000000 | 0.0000000 | Test |
| (40.2, 47.6] | 419 | Age | 0.6992840 | 0.5833333 | 0.1076923 | 0.1818182 | Test |
| (47.6, 55.0] | 210 | Age | 0.5476190 | 0.7435897 | 0.2543860 | 0.3790850 | Test |
| (55.0, 62.4] | 101 | Age | 0.7425743 | 0.7894737 | 0.6250000 | 0.6976744 | Test |
| (62.4, 69.8] | 49 | Age | 0.8367347 | 0.6250000 | 0.8333333 | 0.7142857 | Test |
| (69.8, 77.2] | 29 | Age | 0.3793103 | 0.2380952 | 0.7142857 | 0.3571429 | Test |
| (77.2, 84.6] | 3 | Age | 0.0000000 | 0.0000000 | 0.0000000 | 0.0000000 | Test |
| (84.6, 92.0] | 2 | Age | 0.0000000 | 0.0000000 | 0.0000000 | 0.0000000 | Test |
| (-0.01, 1.0] | 793 | Tenure | 0.8108449 | 0.7285714 | 0.2802198 | 0.4047619 | Training |
| (1.0, 2.0] | 606 | Tenure | 0.8168317 | 0.6000000 | 0.2479339 | 0.3508772 | Training |
| (2.0, 3.0] | 586 | Tenure | 0.8071672 | 0.5000000 | 0.1592920 | 0.2416107 | Training |
| (3.0, 4.0] | 545 | Tenure | 0.8036697 | 0.6969697 | 0.1916667 | 0.3006536 | Training |
| (4.0, 5.0] | 545 | Tenure | 0.8128440 | 0.4166667 | 0.1562500 | 0.2272727 | Training |
| (5.0, 6.0] | 547 | Tenure | 0.8025594 | 0.6451613 | 0.1709402 | 0.2702703 | Training |
| (6.0, 7.0] | 558 | Tenure | 0.8655914 | 0.7058824 | 0.2696629 | 0.3902439 | Training |
| (7.0, 8.0] | 576 | Tenure | 0.8246528 | 0.7000000 | 0.1858407 | 0.2937063 | Training |
| (8.0, 9.0] | 557 | Tenure | 0.8007181 | 0.5833333 | 0.1794872 | 0.2745098 | Training |
| (9.0, 10.0] | 287 | Tenure | 0.7909408 | 0.5555556 | 0.2380952 | 0.3333333 | Training |
| (-0.01, 1.0] | 329 | Tenure | 0.7933131 | 0.5357143 | 0.2142857 | 0.3061224 | Test |
| (1.0, 2.0] | 235 | Tenure | 0.8382979 | 0.4285714 | 0.0810811 | 0.1363636 | Test |
| (2.0, 3.0] | 243 | Tenure | 0.7860082 | 0.7142857 | 0.2459016 | 0.3658537 | Test |
| (3.0, 4.0] | 239 | Tenure | 0.8075314 | 0.6666667 | 0.1960784 | 0.3030303 | Test |
| (4.0, 5.0] | 264 | Tenure | 0.7878788 | 0.6818182 | 0.2343750 | 0.3488372 | Test |

Table 22: Robustness test

| Perturbation Size | Dataset Type | Records | accuracy | Passed |
|---|---|---|---|---|
| 0.0 | Training | 5600 | 81.46429 | TRUE |
| 0.0 | Test | 2400 | 81.16667 | TRUE |
| 0.1 | Training | 5600 | 81.55357 | TRUE |
| 0.1 | Test | 2400 | 81.08333 | TRUE |
| 0.2 | Training | 5600 | 81.55357 | TRUE |
| 0.2 | Test | 2400 | 80.70833 | TRUE |
| 0.3 | Training | 5600 | 81.16071 | TRUE |
| 0.3 | Test | 2400 | 80.37500 | TRUE |
| 0.4 | Training | 5600 | 80.80357 | TRUE |
| 0.4 | Test | 2400 | 80.29167 | TRUE |
| 0.5 | Training | 5600 | 80.58929 | TRUE |
| 0.5 | Test | 2400 | 80.70833 | TRUE |