Modeling Approaches

In [1]:
```
# pip install xgboost
```

In [2]:
```
import pandas as pd
import numpy as np
# import dask.dataframe as dd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from datetime import datetime
from sklearn.metrics import roc_auc_score, f1_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
from collections import defaultdict
# Week 7
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
import xgboost as xgb
```

Reading into the data

In [3]:
```
pd.set_option('display.max_columns', None)

train = pd.read_csv('train_final.csv', low_memory=False)
validation = pd.read_csv('val_set_final.csv')
```

In [4]:
```
train.head()
```

Out[4]:

	Unnamed: 0	date	customer_code	employee_index	country	female	age	new_cust	seniority_
0	0	2015-07-28	664160	N	1	0	0.632653	0	
1	1	2016-01-28	1076784	N	1	0	0.214286	0	
2	2	2015-12-28	672465	N	1	0	0.387755	0	
3	3	2015-10-28	774528	N	1	0	0.397959	0	
4	4	2016-05-28	569598	N	1	0	0.459184	0	

In [5]:
```
validation.head()
```

Out[5]:

	Unnamed: 0.1	Unnamed: 0	date	customer_code	employee_index	country_spain	female	age	fi
0	0	0	2015-11-28	161428	N	1	1	0.744898	
1	1	1	2015-12-28	367478	N	1	1	0.418367	
2	2	2	2015-11-28	643150	N	1	0	0.520408	
3	3	3	2016-04-28	1385854	N	1	0	0.367347	
4	4	4	2015-08-28	495733	N	1	0	0.346939	

Pre-processing

Changing columns name and dropping columns so both datasets are the same

In [6]:

```
train = train.rename(columns={'country': 'country_spain'})
```

In [7]:

```
train = train.drop(columns=['Unnamed: 0'])
validation = validation.drop(columns=['Unnamed: 0', 'Unnamed: 0.1'])
drop = ['join_channel', 'province_name', 'employee_index', 'segment', 'total_products'
train = train.drop(columns=drop + ['customer_code_encoded'])
validation = validation.drop(columns=drop + ['payroll_acct.1', 'first_contract_date',
```

Setting products we want to predict

In [8]:

```
products = ['savings_acct', 'guarantees', 'current_acct',
       'derivada_acct', 'payroll_acct', 'junior_acct', 'mas_particular_acct',
       'particular_acct', 'particular_plus_acct', 'short_term_depo',
       'medium_term_depo', 'long_term_depo', 'e_acct', 'funds', 'mortgage',
       'pension', 'loans', 'taxes', 'credit_card', 'securities', 'home_acct',
       'pensions_2', 'direct_debt']
```

Dropping duplicates on customer code column since the last instance will show all the products a client has

In [9]:

```
train = train.drop_duplicates(subset=['customer_code'], keep='last')
validation = validation.drop_duplicates(subset=['customer_code'], keep='last')

# Removing customers from validation set that appear in training set
validation = validation[~validation['customer_code'].isin(train['customer_code'])]
```

In [10]:

```
print(train.info())
print(validation.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 706866 entries, 39288 to 6579716
Data columns (total 42 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   date                    706866 non-null  object 
 1   customer_code           706866 non-null  int64  
 2   country_spain           706866 non-null  int64  
 3   female                  706866 non-null  int64  
 4   age                     706866 non-null  float64
 5   new_cust                706866 non-null  int64  
 6   seniority_in_months     706866 non-null  float64
 7   cust_type               706866 non-null  int64  
 8   residency_spain         706866 non-null  int64  
 9   birth_spain             706866 non-null  int64  
 10  active_cust             706866 non-null  int64  
 11  income                  706866 non-null  float64
 12  savings_acct            706866 non-null  int64  
 13  guarantees              706866 non-null  int64  
 14  current_acct            706866 non-null  int64  
 15  derivada_acct           706866 non-null  int64  
 16  payroll_acct            706866 non-null  int64  
 17  junior_acct             706866 non-null  int64  
 18  mas_particular_acct     706866 non-null  int64  
 19  particular_acct         706866 non-null  int64  
 20  particular_plus_acct    706866 non-null  int64  
 21  short_term_depo         706866 non-null  int64  
 22  medium_term_depo        706866 non-null  int64  
 23  long_term_depo          706866 non-null  int64  
 24  e_acct                  706866 non-null  int64  
 25  funds                   706866 non-null  int64  
 26  mortgage                706866 non-null  int64  
 27  pension                 706866 non-null  int64  
 28  loans                   706866 non-null  int64  
 29  taxes                   706866 non-null  int64  
 30  credit_card             706866 non-null  int64  
 31  securities              706866 non-null  int64  
 32  home_acct               706866 non-null  int64  
 33  pensions_2              706866 non-null  int64  
 34  direct_debt             706866 non-null  int64  
 35  01 - TOP                706866 non-null  int64  
 36  02 - PARTICULARES       706866 non-null  int64  
 37  03 - UNIVERSITARIO      706866 non-null  int64  
 38  join_channel_encoded    706866 non-null  float64
 39  province_name_encoded   706866 non-null  float64
 40  employee_index_encoded  706866 non-null  float64
 41  income_to_age           706866 non-null  float64
dtypes: float64(7), int64(34), object(1)
memory usage: 231.9+ MB
None
<class 'pandas.core.frame.DataFrame'>
Index: 200333 entries, 51 to 2100200
Data columns (total 42 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   date                    200333 non-null  object 
 1   customer_code           200333 non-null  int64  
 2   country_spain           200333 non-null  int64  
 3   female                  200333 non-null  int64  
 4   age                     200333 non-null  float64
```

```
 5   new_cust                 200333 non-null  int64
 6   seniority_in_months      200333 non-null  float64
 7   cust_type                200333 non-null  int64
 8   residency_spain          200333 non-null  int64
 9   birth_spain              200333 non-null  int64
 10  active_cust              200333 non-null  int64
 11  income                   200333 non-null  float64
 12  savings_acct             200333 non-null  int64
 13  guarantees               200333 non-null  int64
 14  current_acct             200333 non-null  int64
 15  derivada_acct            200333 non-null  int64
 16  payroll_acct             200333 non-null  int64
 17  junior_acct              200333 non-null  int64
 18  mas_particular_acct      200333 non-null  int64
 19  particular_acct          200333 non-null  int64
 20  particular_plus_acct     200333 non-null  int64
 21  short_term_depo          200333 non-null  int64
 22  medium_term_depo         200333 non-null  int64
 23  long_term_depo           200333 non-null  int64
 24  e_acct                   200333 non-null  int64
 25  funds                    200333 non-null  int64
 26  mortgage                 200333 non-null  int64
 27  pension                  200333 non-null  int64
 28  loans                    200333 non-null  int64
 29  taxes                    200333 non-null  int64
 30  credit_card              200333 non-null  int64
 31  securities               200333 non-null  int64
 32  home_acct                200333 non-null  int64
 33  pensions_2               200333 non-null  int64
 34  direct_debt              200333 non-null  int64
 35  01 - TOP                 200333 non-null  int64
 36  02 - PARTICULARES        200333 non-null  int64
 37  03 - UNIVERSITARIO       200333 non-null  int64
 38  join_channel_encoded     200333 non-null  float64
 39  province_name_encoded    200333 non-null  float64
 40  employee_index_encoded   200333 non-null  float64
 41  income_to_age            200333 non-null  float64
dtypes: float64(7), int64(34), object(1)
memory usage: 65.7+ MB
None
```

Defining our Xs and Ys

In [11]:
```
X_train = train.drop(['customer_code', 'date'] + products, axis=1)
y_train = train[products]

X_val = validation.drop(['customer_code', 'date'] + products, axis=1)
y_val = validation[products]
```

In [12]:
```
print("Shape of X_train:", X_train.shape)
print("Shape of y_train:", y_train.shape)

print("Shape of X_val:", X_val.shape)
print("Shape of y_val:", y_val.shape)
```

```
Shape of X_train: (706866, 17)
Shape of y_train: (706866, 23)
Shape of X_val: (200333, 17)
Shape of y_val: (200333, 23)
```

Week 7 - Develop Second Modeling Approach

Setting hyperparameters for XGBoost model

In [13]:
```
# Define hyperparameter variations
hyperparameters = [
    {
        'max_depth': 3,
        'learning_rate': 0.1,
        'subsample': 0.8,
        'colsample_bytree': 0.8,
    },
    {
        'max_depth': 5,
        'learning_rate': 0.05,
        'subsample': 0.7,
        'colsample_bytree': 0.7,
    },
    {
        'max_depth': 7,
        'learning_rate': 0.01,
        'subsample': 0.6,
        'colsample_bytree': 0.6,
    }
]
```

Creating funtion to train and evaluate the XGBoost model

This week we will keep the same metrics used last week -ROC AUC, F1 Score and Confusion Matrix, so we can compare in the end which model is better

We are looping through ech one of the products, and creating a DMatrix to store the data. To avoid overfitting, we are using early stopping, which means that if the model does not improve for 20 rounds, it will stop early

In [14]:
```
# Model function
def xgb_model(X_train, y_train, X_val, y_val, params):
    metrics = {}
    for product in y_train.columns:
        # Creating DMatrix for XGBoost
        dtrain = xgb.DMatrix(X_train, label=y_train[product])
        dval = xgb.DMatrix(X_val, label=y_val[product])

        # Setting up early stopping to prevent overfitting
        early_stopping_rounds = 20
        eval_set = [(dtrain, 'train'), (dval, 'val')]

        # Train
        model = xgb.train(
            params,
            dtrain,
            num_boost_round=1000,
```

```
            evals=[(dval, 'val')],
            early_stopping_rounds=early_stopping_rounds,
            verbose_eval=False
        )

        # Predictions
        y_train_pred = model.predict(dtrain)
        y_val_pred = model.predict(dval)

        # Metrics
        metrics[product] = {
            'train': calc_metrics(y_train[product], y_train_pred),
            'val': calc_metrics(y_val[product], y_val_pred)
        }

    return metrics

# Metrics Function
def calc_metrics(y_true, y_pred):
    return {
        'ROC AUC': roc_auc_score(y_true, y_pred),
        'F1 Score': f1_score(y_true, y_pred > 0.5),
        'Confusion Matrix': confusion_matrix(y_true, y_pred > 0.5)
    }
```

In [15]:

```
# Calling the funtions to run the model
results = []
for i, params in enumerate(hyperparameters):
    metrics = xgb_model(X_train, y_train, X_val, y_val, params)

    avg_train_roc_auc = np.mean([metrics[product]['train']['ROC AUC'] for product in y
    avg_train_f1 = np.mean([metrics[product]['train']['F1 Score'] for product in y_tra
    avg_val_roc_auc = np.mean([metrics[product]['val']['ROC AUC'] for product in y_tra
    avg_val_f1 = np.mean([metrics[product]['val']['F1 Score'] for product in y_train.c

    results.append({
        'Variation': f'Variation {i+1}',
        'Train ROC AUC': avg_train_roc_auc,
        'Train F1 Score': avg_train_f1,
        'Val ROC AUC': avg_val_roc_auc,
        'Val F1 Score': avg_val_f1
    })
```

Creating a table to summarize results and define which variation is the best

In [28]:

```
# Summary table
results = pd.DataFrame(results)
print(results.to_string(index=False))

best_model = results.loc[results['Val ROC AUC'].idxmax()]['Variation']
print(f"\nThe best model for this week is {best_model}")
```

```
  Variation  Train ROC AUC  Train F1 Score  Val ROC AUC  Val F1 Score
Variation 1       0.918933        0.144641     0.904372      0.132815
Variation 2       0.926318        0.150042     0.923959      0.139032
Variation 3       0.933340        0.135798     0.884652      0.117693

The best model for this week is Variation 2
```

Creating a table to see the results in a easier to interpret way

In [19]:
```
train_metrics_list = []
val_metrics_list = []

for variation_idx, params in enumerate(hyperparameters, 1):
    variation_name = f"Variation {variation_idx}"

    for product in y_train.columns:

        train_roc_auc = metrics[product]['train']['ROC AUC']
        train_f1_score = metrics[product]['train']['F1 Score']
        train_conf_matrix = metrics[product]['train']['Confusion Matrix']

        val_roc_auc = metrics[product]['val']['ROC AUC']
        val_f1_score = metrics[product]['val']['F1 Score']
        val_conf_matrix = metrics[product]['val']['Confusion Matrix']

        # Appending lists
        train_metrics_list.append({
            'Variation': variation_name,
            'Dataset': 'Train',
            'Product': product,
            'ROC AUC': train_roc_auc,
            'F1 Score': train_f1_score,
            'Confusion Matrix': train_conf_matrix
        })


        val_metrics_list.append({
            'Variation': variation_name,
            'Dataset': 'Validation',
            'Product': product,
            'ROC AUC': val_roc_auc,
            'F1 Score': val_f1_score,
            'Confusion Matrix': val_conf_matrix
        })


metrics_df = pd.DataFrame(train_metrics_list + val_metrics_list)
print(metrics_df.to_string(index=False))
```

```
  Variation    Dataset               Product  ROC AUC  F1 Score                  Con
fusion Matrix
Variation 1      Train          savings_acct 0.993083  0.000000              [[706792,
0], [74, 0]]
Variation 1      Train            guarantees 0.999966  0.000000              [[706850,
0], [16, 0]]
Variation 1      Train          current_acct 0.808806  0.804876 [[171753, 105207], [69
525, 360381]]
Variation 1      Train         derivada_acct 0.967359  0.000000             [[706603,
0], [263, 0]]
Variation 1      Train          payroll_acct 0.896745  0.013931         [[667803, 58],
[38731, 274]]
Variation 1      Train           junior_acct 0.999975  0.966054          [[699891, 39
4], [64, 6517]]
Variation 1      Train  mas_particular_acct 0.981853  0.015199            [[701120,
1], [5701, 44]]
Variation 1      Train       particular_acct 0.925143  0.438586   [[602473, 17660], [5
7410, 29323]]
Variation 1      Train particular_plus_acct 0.921532  0.001422           [[677355, 3],
[29487, 21]]
Variation 1      Train       short_term_depo 0.969419  0.000000            [[705857,
0], [1009, 0]]
Variation 1      Train      medium_term_depo 0.945110  0.000000            [[705817,
0], [1049, 0]]
Variation 1      Train        long_term_depo 0.936072  0.363833      [[672909, 4435],
[21971, 7551]]
Variation 1      Train                e_acct 0.899611  0.274052    [[642067, 5918], [4
8592, 10289]]
Variation 1      Train                 funds 0.930750  0.000000           [[694099,
0], [12767, 0]]
Variation 1      Train              mortgage 0.957530  0.000000            [[702807,
0], [4059, 0]]
Variation 1      Train               pension 0.936456  0.001558            [[700454,
0], [6407, 5]]
Variation 1      Train                 loans 0.962701  0.065306           [[705207,
3], [1600, 56]]
Variation 1      Train                 taxes 0.882262  0.000105           [[668644,
0], [38220, 2]]
Variation 1      Train           credit_card 0.907053  0.004812           [[675770, 8],
[31013, 75]]
Variation 1      Train            securities 0.921937  0.002945           [[689233, 2],
[17605, 26]]
Variation 1      Train             home_acct 0.935767  0.000000            [[704156,
0], [2710, 0]]
Variation 1      Train            pensions_2 0.887281  0.012208         [[664206, 54],
[42344, 262]]
Variation 1      Train           direct_debt 0.900405  0.158468      [[612621, 4243],
[81892, 8110]]
Variation 2      Train          savings_acct 0.993083  0.000000              [[706792,
0], [74, 0]]
Variation 2      Train            guarantees 0.999966  0.000000              [[706850,
0], [16, 0]]
Variation 2      Train          current_acct 0.808806  0.804876 [[171753, 105207], [69
525, 360381]]
Variation 2      Train         derivada_acct 0.967359  0.000000             [[706603,
0], [263, 0]]
Variation 2      Train          payroll_acct 0.896745  0.013931         [[667803, 58],
[38731, 274]]
Variation 2      Train           junior_acct 0.999975  0.966054          [[699891, 39
4], [64, 6517]]
```

```
Variation 2      Train  mas_particular_acct 0.981853  0.015199          [[701120,
1], [5701, 44]]
Variation 2      Train      particular_acct 0.925143  0.438586   [[602473, 17660], [5
7410, 29323]]
Variation 2      Train particular_plus_acct 0.921532  0.001422         [[677355, 3],
[29487, 21]]
Variation 2      Train      short_term_depo 0.969419  0.000000           [[705857,
0], [1009, 0]]
Variation 2      Train     medium_term_depo 0.945110  0.000000           [[705817,
0], [1049, 0]]
Variation 2      Train       long_term_depo 0.936072  0.363833     [[672909, 4435],
[21971, 7551]]
Variation 2      Train               e_acct 0.899611  0.274052    [[642067, 5918], [4
8592, 10289]]
Variation 2      Train                funds 0.930750  0.000000          [[694099,
0], [12767, 0]]
Variation 2      Train             mortgage 0.957530  0.000000           [[702807,
0], [4059, 0]]
Variation 2      Train              pension 0.936456  0.001558           [[700454,
0], [6407, 5]]
Variation 2      Train                loans 0.962701  0.065306          [[705207,
3], [1600, 56]]
Variation 2      Train                taxes 0.882262  0.000105          [[668644,
0], [38220, 2]]
Variation 2      Train          credit_card 0.907053  0.004812         [[675770, 8],
[31013, 75]]
Variation 2      Train           securities 0.921937  0.002945         [[689233, 2],
[17605, 26]]
Variation 2      Train            home_acct 0.935767  0.000000           [[704156,
0], [2710, 0]]
Variation 2      Train           pensions_2 0.887281  0.012208       [[664206, 54],
[42344, 262]]
Variation 2      Train          direct_debt 0.900405  0.158468     [[612621, 4243],
[81892, 8110]]
Variation 3      Train         savings_acct 0.993083  0.000000             [[706792,
0], [74, 0]]
Variation 3      Train           guarantees 0.999966  0.000000             [[706850,
0], [16, 0]]
Variation 3      Train         current_acct 0.808806  0.804876 [[171753, 105207], [69
525, 360381]]
Variation 3      Train        derivada_acct 0.967359  0.000000            [[706603,
0], [263, 0]]
Variation 3      Train         payroll_acct 0.896745  0.013931       [[667803, 58],
[38731, 274]]
Variation 3      Train          junior_acct 0.999975  0.966054        [[699891, 39
4], [64, 6517]]
Variation 3      Train  mas_particular_acct 0.981853  0.015199          [[701120,
1], [5701, 44]]
Variation 3      Train      particular_acct 0.925143  0.438586   [[602473, 17660], [5
7410, 29323]]
Variation 3      Train particular_plus_acct 0.921532  0.001422         [[677355, 3],
[29487, 21]]
Variation 3      Train      short_term_depo 0.969419  0.000000           [[705857,
0], [1009, 0]]
Variation 3      Train     medium_term_depo 0.945110  0.000000           [[705817,
0], [1049, 0]]
Variation 3      Train       long_term_depo 0.936072  0.363833     [[672909, 4435],
[21971, 7551]]
Variation 3      Train               e_acct 0.899611  0.274052    [[642067, 5918], [4
8592, 10289]]
```

```
Variation 3      Train                funds 0.930750  0.000000           [[694099,
0], [12767, 0]]
Variation 3      Train             mortgage 0.957530  0.000000            [[702807,
0], [4059, 0]]
Variation 3      Train              pension 0.936456  0.001558            [[700454,
0], [6407, 5]]
Variation 3      Train                loans 0.962701  0.065306           [[705207,
3], [1600, 56]]
Variation 3      Train                taxes 0.882262  0.000105           [[668644,
0], [38220, 2]]
Variation 3      Train          credit_card 0.907053  0.004812          [[675770, 8],
[31013, 75]]
Variation 3      Train           securities 0.921937  0.002945          [[689233, 2],
[17605, 26]]
Variation 3      Train            home_acct 0.935767  0.000000            [[704156,
0], [2710, 0]]
Variation 3      Train           pensions_2 0.887281  0.012208        [[664206, 54],
[42344, 262]]
Variation 3      Train          direct_debt 0.900405  0.158468     [[612621, 4243],
[81892, 8110]]
Variation 1 Validation         savings_acct 0.904347  0.000000               [[20032
5, 0], [8, 0]]
Variation 1 Validation           guarantees 0.029764  0.000000               [[20033
1, 0], [2, 0]]
Variation 1 Validation         current_acct 0.813402  0.818449   [[51050, 27102], [18
774, 103407]]
Variation 1 Validation        derivada_acct 0.900264  0.000000             [[200289,
0], [44, 0]]
Variation 1 Validation         payroll_acct 0.897229  0.009591          [[194107, 1
2], [6184, 30]]
Variation 1 Validation          junior_acct 0.999902  0.933043         [[199106, 10
8], [46, 1073]]
Variation 1 Validation  mas_particular_acct 0.955139  0.000712            [[197526,
0], [2806, 1]]
Variation 1 Validation      particular_acct 0.952627  0.404784     [[182926, 2344],
[10646, 4417]]
Variation 1 Validation particular_plus_acct 0.944120  0.000000            [[195857,
0], [4476, 0]]
Variation 1 Validation      short_term_depo 0.947906  0.000000             [[200009,
0], [324, 0]]
Variation 1 Validation     medium_term_depo 0.901511  0.000000             [[200152,
0], [181, 0]]
Variation 1 Validation       long_term_depo 0.947198  0.266093        [[195094, 578],
[3857, 804]]
Variation 1 Validation               e_acct 0.914806  0.146672        [[190199, 574],
[8758, 802]]
Variation 1 Validation                funds 0.928412  0.000000            [[198246,
0], [2087, 0]]
Variation 1 Validation             mortgage 0.969950  0.000000             [[199812,
0], [521, 0]]
Variation 1 Validation              pension 0.941016  0.000000             [[199454,
0], [879, 0]]
Variation 1 Validation                loans 0.931207  0.023121             [[199991,
4], [334, 4]]
Variation 1 Validation                taxes 0.886356  0.000000            [[195396,
0], [4937, 0]]
Variation 1 Validation          credit_card 0.932572  0.001829            [[195963,
6], [4360, 4]]
Variation 1 Validation           securities 0.924884  0.000000            [[197281,
0], [3052, 0]]
```

```
Variation 1 Validation            home_acct 0.926991  0.000000             [[200015,
0], [318, 0]]
Variation 1 Validation           pensions_2 0.891483  0.007692          [[193599, 1
0], [6698, 26]]
Variation 1 Validation          direct_debt 0.905907  0.094950       [[183640, 592],
[15269, 832]]
Variation 2 Validation         savings_acct 0.904347  0.000000               [[20032
5, 0], [8, 0]]
Variation 2 Validation           guarantees 0.029764  0.000000               [[20033
1, 0], [2, 0]]
Variation 2 Validation         current_acct 0.813402  0.818449   [[51050, 27102], [18
774, 103407]]
Variation 2 Validation        derivada_acct 0.900264  0.000000              [[200289,
0], [44, 0]]
Variation 2 Validation         payroll_acct 0.897229  0.009591          [[194107, 1
2], [6184, 30]]
Variation 2 Validation          junior_acct 0.999902  0.933043         [[199106, 10
8], [46, 1073]]
Variation 2 Validation  mas_particular_acct 0.955139  0.000712            [[197526,
0], [2806, 1]]
Variation 2 Validation      particular_acct 0.952627  0.404784     [[182926, 2344],
[10646, 4417]]
Variation 2 Validation particular_plus_acct 0.944120  0.000000            [[195857,
0], [4476, 0]]
Variation 2 Validation      short_term_depo 0.947906  0.000000             [[200009,
0], [324, 0]]
Variation 2 Validation     medium_term_depo 0.901511  0.000000             [[200152,
0], [181, 0]]
Variation 2 Validation       long_term_depo 0.947198  0.266093        [[195094, 578],
[3857, 804]]
Variation 2 Validation               e_acct 0.914806  0.146672        [[190199, 574],
[8758, 802]]
Variation 2 Validation                funds 0.928412  0.000000            [[198246,
0], [2087, 0]]
Variation 2 Validation             mortgage 0.969950  0.000000             [[199812,
0], [521, 0]]
Variation 2 Validation              pension 0.941016  0.000000             [[199454,
0], [879, 0]]
Variation 2 Validation                loans 0.931207  0.023121             [[199991,
4], [334, 4]]
Variation 2 Validation                taxes 0.886356  0.000000            [[195396,
0], [4937, 0]]
Variation 2 Validation          credit_card 0.932572  0.001829            [[195963,
6], [4360, 4]]
Variation 2 Validation           securities 0.924884  0.000000            [[197281,
0], [3052, 0]]
Variation 2 Validation            home_acct 0.926991  0.000000             [[200015,
0], [318, 0]]
Variation 2 Validation           pensions_2 0.891483  0.007692          [[193599, 1
0], [6698, 26]]
Variation 2 Validation          direct_debt 0.905907  0.094950       [[183640, 592],
[15269, 832]]
Variation 3 Validation         savings_acct 0.904347  0.000000               [[20032
5, 0], [8, 0]]
Variation 3 Validation           guarantees 0.029764  0.000000               [[20033
1, 0], [2, 0]]
Variation 3 Validation         current_acct 0.813402  0.818449   [[51050, 27102], [18
774, 103407]]
Variation 3 Validation        derivada_acct 0.900264  0.000000              [[200289,
0], [44, 0]]
```

```
Variation 3 Validation         payroll_acct 0.897229  0.009591          [[194107, 1
2], [6184, 30]]
Variation 3 Validation          junior_acct 0.999902  0.933043          [[199106, 10
8], [46, 1073]]
Variation 3 Validation  mas_particular_acct 0.955139  0.000712             [[197526,
0], [2806, 1]]
Variation 3 Validation      particular_acct 0.952627  0.404784     [[182926, 2344],
[10646, 4417]]
Variation 3 Validation particular_plus_acct 0.944120  0.000000             [[195857,
0], [4476, 0]]
Variation 3 Validation      short_term_depo 0.947906  0.000000              [[200009,
0], [324, 0]]
Variation 3 Validation     medium_term_depo 0.901511  0.000000              [[200152,
0], [181, 0]]
Variation 3 Validation       long_term_depo 0.947198  0.266093        [[195094, 578],
[3857, 804]]
Variation 3 Validation               e_acct 0.914806  0.146672        [[190199, 574],
[8758, 802]]
Variation 3 Validation                funds 0.928412  0.000000             [[198246,
0], [2087, 0]]
Variation 3 Validation             mortgage 0.969950  0.000000              [[199812,
0], [521, 0]]
Variation 3 Validation              pension 0.941016  0.000000              [[199454,
0], [879, 0]]
Variation 3 Validation                loans 0.931207  0.023121              [[199991,
4], [334, 4]]
Variation 3 Validation                taxes 0.886356  0.000000             [[195396,
0], [4937, 0]]
Variation 3 Validation          credit_card 0.932572  0.001829             [[195963,
6], [4360, 4]]
Variation 3 Validation           securities 0.924884  0.000000             [[197281,
0], [3052, 0]]
Variation 3 Validation            home_acct 0.926991  0.000000              [[200015,
0], [318, 0]]
Variation 3 Validation           pensions_2 0.891483  0.007692          [[193599, 1
0], [6698, 26]]
Variation 3 Validation          direct_debt 0.905907  0.094950       [[183640, 592],
[15269, 832]]
```

Generate product recommendations

Here we want to visualize the product recommendations for each customer
We will generate the product recommendations using Variation 2 - that had the best performance between all variations

In [31]:

```python
best_params = hyperparameters[int(best_model.split()[-1]) - 1]

# Train model using Variation 2
product_models = {}
for product in y_train.columns:
    dtrain = xgb.DMatrix(X_train, label=y_train[product])
    model = xgb.train(
        best_params,
        dtrain,
        num_boost_round=1000,
        verbose_eval=False
    )
```

```python
    product_models[product] = model

# Generate predictions
train_preds = {}
for product in y_train.columns:
    dtrain = xgb.DMatrix(X_train)
    proba = product_models[product].predict(dtrain)

    for customer_id, prob in zip(train['customer_code'], proba):
        if customer_id not in train_preds:
            train_preds[customer_id] = []
        train_preds[customer_id].append((product, prob))
```

We want to make sure that we are not recommending a product that the customer already own, so we will store the products that customers already have

In [32]:
```python
def get_active_products(customer_data):
    return set(product for product in y_train.columns if customer_data[product] > 0)
```

We will sort the recommended products by the probability of a client getting it and we will get the top 7 recommendations

In [33]:
```python
# Generate top 7 products, excluding owned products
for customer_id in train_preds:
    sorted_prods = sorted(train_preds[customer_id], key=lambda x: x[1], reverse=True)
    customer_data = train[train['customer_code'] == customer_id].iloc[0]
    active_products = get_active_products(customer_data)
    recommended_products = [prod for prod, _ in sorted_prods if prod not in active_pro

    # Top 7
    train_preds[customer_id] = recommended_products[:7]

# Recommendations for the first 5 customers
print("Recommendations:")
for customer_id in list(train_preds.keys())[:5]:
    print(f"Customer {customer_id}: {train_preds[customer_id]}")
```

```
Recommendations:
Customer 1225385: ['current_acct', 'taxes', 'pension', 'particular_acct', 'funds', 'c
redit_card', 'securities']
Customer 1358829: ['direct_debt', 'pensions_2', 'e_acct', 'payroll_acct', 'taxes', 's
ecurities', 'pension']
Customer 1436539: ['pensions_2', 'payroll_acct', 'direct_debt', 'mas_particular_acc
t', 'e_acct', 'particular_acct', 'credit_card']
Customer 1448049: ['current_acct', 'payroll_acct', 'particular_plus_acct', 'mas_parti
cular_acct', 'mortgage', 'pension', 'loans']
Customer 1396837: ['direct_debt', 'pensions_2', 'payroll_acct', 'e_acct', 'mas_partic
ular_acct', 'taxes', 'long_term_depo']
```

Lastly, we want to identify which product has been recommended the most and least in the model

In [34]:
```python
product_rec_counts = {product: sum(1 for recs in train_preds.values() if product in re
most_rec = max(product_rec_counts, key=product_rec_counts.get)
least_rec = min(product_rec_counts, key=product_rec_counts.get)
```

```
print(f"Most frequently recommended product: {most_rec} ({product_rec_counts[most_rec]
print(f"Least frequently recommended product: {least_rec} ({product_rec_counts[least_r
```

```
Most frequently recommended product: taxes (533949 times)
Least frequently recommended product: guarantees (18541 times)
```