Week 6 - Develop First modeling approach

In [1]:
```
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
```

In [2]:
```
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 [3]:
```
train.head()
```

Out[3]:

	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 [4]:
```
validation.head()
```

Out[4]:

	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	

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

In [5]:
```
train = train.rename(columns={'country': 'country_spain'})
```

In [6]:
```
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',
```

Reading into the data

Setting products we want to predict

In [7]:
```
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 [8]:
```
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 [9]:
```
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
<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
```

Pre-processing

Defining our Xs and Ys

In [10]:
```
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 [11]:
```
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)
```

Training

In [12]:
```
# Hyperparameters
hyperparameter_variations = [
```

```
    {'C': 0.01, 'solver': 'liblinear', 'max_iter': 100},
    {'C': 1, 'solver': 'lbfgs', 'max_iter': 500},
    {'C': 10, 'solver': 'liblinear', 'max_iter': 300},
]
```

In [13]:
```
# Storing trained models and predictions
models = {}
metrics = defaultdict(lambda: defaultdict(dict))
```

We will create a model to train on the training data using all 3 hyperparameters we set. We will use this trained model to predict the product recommendations on the validation set and compare the results between the different hyperparameters and different metrics we chose to use, which are ROC AUC, F1 Score and Confusion Matrix.

We will calculate ROC AUC using probabilities (predict_proba() method), which is more appropriate for this metric since ROC AUC works with predicted probabilities for the positive class and not binary predictions.

F1 Score and confusion matrix were calculated using the binary predictions (predict() method), which is the correct approach for these metrics.

In [14]:
```
# Train and evaluate each hyperparameter variation
for i, params in enumerate(hyperparameter_variations):
    print(f"\nTraining variation {i + 1} with parameters: {params}")

    for product in products:
        print(f"Training model for product: {product}")
        clf = LogisticRegression(**params)

        # Target column for current product
        y_train_product = y_train[product].values
        y_val_product = y_val[product].values

        # Train the model on current product
        clf.fit(X_train, y_train_product)

        # Make predictions
        y_train_pred = clf.predict(X_train)
        y_val_pred = clf.predict(X_val)
        y_train_pred_proba = clf.predict_proba(X_train)[:, 1]
        y_val_pred_proba = clf.predict_proba(X_val)[:, 1]

        # Calculate metrics for training set and validation sets
        metrics[f'Variation {i + 1}']['train'][product] = {
            'ROC AUC': roc_auc_score(y_train_product, y_train_pred_proba),
            'F1 Score': f1_score(y_train_product, y_train_pred),
            'Confusion Matrix': confusion_matrix(y_train_product, y_train_pred)
        }

        metrics[f'Variation {i + 1}']['val'][product] = {
            'ROC AUC': roc_auc_score(y_val_product, y_val_pred_proba),
            'F1 Score': f1_score(y_val_product, y_val_pred),
            'Confusion Matrix': confusion_matrix(y_val_product, y_val_pred)
        }

        print(f"\nResults for'{product}' in variation {i + 1}:")
        print(f"Training - ROC AUC: {metrics[f'Variation {i + 1}']['train'][product]['
              f"F1 Score: {metrics[f'Variation {i + 1}']['train'][product]['F1 Score']
```

```
        print(f"Validation - ROC AUC: {metrics[f'Variation {i + 1}']['val'][product]['
              f"F1 Score: {metrics[f'Variation {i + 1}']['val'][product]['F1 Score']:.
```

```
Training variation 1 with parameters: {'C': 0.01, 'solver': 'liblinear', 'max_iter':
100}
Training model for product: savings_acct

Results for'savings_acct' in variation 1:
Training - ROC AUC: 0.4029, F1 Score: 0.0000
Validation - ROC AUC: 0.4489, F1 Score: 0.0000
Training model for product: guarantees

Results for'guarantees' in variation 1:
Training - ROC AUC: 0.2619, F1 Score: 0.0000
Validation - ROC AUC: 0.1172, F1 Score: 0.0000
Training model for product: current_acct

Results for'current_acct' in variation 1:
Training - ROC AUC: 0.7485, F1 Score: 0.7865
Validation - ROC AUC: 0.7564, F1 Score: 0.7899
Training model for product: derivada_acct

Results for'derivada_acct' in variation 1:
Training - ROC AUC: 0.7772, F1 Score: 0.0000
Validation - ROC AUC: 0.7503, F1 Score: 0.0000
Training model for product: payroll_acct

Results for'payroll_acct' in variation 1:
Training - ROC AUC: 0.8646, F1 Score: 0.0001
Validation - ROC AUC: 0.8595, F1 Score: 0.0075
Training model for product: junior_acct

Results for'junior_acct' in variation 1:
Training - ROC AUC: 0.9990, F1 Score: 0.1412
Validation - ROC AUC: 0.9972, F1 Score: 0.1272
Training model for product: mas_particular_acct

Results for'mas_particular_acct' in variation 1:
Training - ROC AUC: 0.8256, F1 Score: 0.0000
Validation - ROC AUC: 0.8864, F1 Score: 0.0000
Training model for product: particular_acct

Results for'particular_acct' in variation 1:
Training - ROC AUC: 0.8847, F1 Score: 0.2141
Validation - ROC AUC: 0.9270, F1 Score: 0.2457
Training model for product: particular_plus_acct

Results for'particular_plus_acct' in variation 1:
Training - ROC AUC: 0.8132, F1 Score: 0.0000
Validation - ROC AUC: 0.8590, F1 Score: 0.0288
Training model for product: short_term_depo

Results for'short_term_depo' in variation 1:
Training - ROC AUC: 0.9249, F1 Score: 0.0000
Validation - ROC AUC: 0.9126, F1 Score: 0.0000
Training model for product: medium_term_depo

Results for'medium_term_depo' in variation 1:
Training - ROC AUC: 0.8772, F1 Score: 0.0000
Validation - ROC AUC: 0.8977, F1 Score: 0.0000
Training model for product: long_term_depo

Results for'long_term_depo' in variation 1:
```

```
Training - ROC AUC: 0.9263, F1 Score: 0.3412
Validation - ROC AUC: 0.9359, F1 Score: 0.2931
Training model for product: e_acct

Results for'e_acct' in variation 1:
Training - ROC AUC: 0.8606, F1 Score: 0.2165
Validation - ROC AUC: 0.8769, F1 Score: 0.1683
Training model for product: funds

Results for'funds' in variation 1:
Training - ROC AUC: 0.9216, F1 Score: 0.0000
Validation - ROC AUC: 0.9357, F1 Score: 0.0584
Training model for product: mortgage

Results for'mortgage' in variation 1:
Training - ROC AUC: 0.9208, F1 Score: 0.0000
Validation - ROC AUC: 0.9466, F1 Score: 0.0000
Training model for product: pension

Results for'pension' in variation 1:
Training - ROC AUC: 0.9208, F1 Score: 0.0000
Validation - ROC AUC: 0.9321, F1 Score: 0.0378
Training model for product: loans

Results for'loans' in variation 1:
Training - ROC AUC: 0.8397, F1 Score: 0.0000
Validation - ROC AUC: 0.8581, F1 Score: 0.0000
Training model for product: taxes

Results for'taxes' in variation 1:
Training - ROC AUC: 0.8592, F1 Score: 0.0008
Validation - ROC AUC: 0.8657, F1 Score: 0.0307
Training model for product: credit_card

Results for'credit_card' in variation 1:
Training - ROC AUC: 0.8904, F1 Score: 0.0001
Validation - ROC AUC: 0.9184, F1 Score: 0.0405
Training model for product: securities

Results for'securities' in variation 1:
Training - ROC AUC: 0.9137, F1 Score: 0.0002
Validation - ROC AUC: 0.9285, F1 Score: 0.0860
Training model for product: home_acct

Results for'home_acct' in variation 1:
Training - ROC AUC: 0.8714, F1 Score: 0.0000
Validation - ROC AUC: 0.8997, F1 Score: 0.0000
Training model for product: pensions_2

Results for'pensions_2' in variation 1:
Training - ROC AUC: 0.8614, F1 Score: 0.0001
Validation - ROC AUC: 0.8587, F1 Score: 0.0091
Training model for product: direct_debt

Results for'direct_debt' in variation 1:
Training - ROC AUC: 0.8688, F1 Score: 0.0356
Validation - ROC AUC: 0.8709, F1 Score: 0.0514

Training variation 2 with parameters: {'C': 1, 'solver': 'lbfgs', 'max_iter': 500}
Training model for product: savings_acct
```

```
Results for'savings_acct' in variation 2:
Training - ROC AUC: 0.8689, F1 Score: 0.0000
Validation - ROC AUC: 0.9334, F1 Score: 0.0000
Training model for product: guarantees

Results for'guarantees' in variation 2:
Training - ROC AUC: 0.9642, F1 Score: 0.0000
Validation - ROC AUC: 0.9887, F1 Score: 0.0000
Training model for product: current_acct
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Results for'current_acct' in variation 2:
Training - ROC AUC: 0.7480, F1 Score: 0.7863
Validation - ROC AUC: 0.7569, F1 Score: 0.7895
Training model for product: derivada_acct
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Results for'derivada_acct' in variation 2:
Training - ROC AUC: 0.8804, F1 Score: 0.0000
Validation - ROC AUC: 0.9080, F1 Score: 0.0000
Training model for product: payroll_acct
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Results for'payroll_acct' in variation 2:
Training - ROC AUC: 0.8663, F1 Score: 0.0003
Validation - ROC AUC: 0.8602, F1 Score: 0.0090
Training model for product: junior_acct
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Results for'junior_acct' in variation 2:
Training - ROC AUC: 0.9994, F1 Score: 0.8811
Validation - ROC AUC: 0.9986, F1 Score: 0.8270
Training model for product: mas_particular_acct

Results for'mas_particular_acct' in variation 2:
Training - ROC AUC: 0.8412, F1 Score: 0.0000
Validation - ROC AUC: 0.8851, F1 Score: 0.0000
Training model for product: particular_acct

Results for'particular_acct' in variation 2:
Training - ROC AUC: 0.8848, F1 Score: 0.2291
Validation - ROC AUC: 0.9254, F1 Score: 0.2534
Training model for product: particular_plus_acct
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Results for'particular_plus_acct' in variation 2:
Training - ROC AUC: 0.8138, F1 Score: 0.0000
Validation - ROC AUC: 0.8616, F1 Score: 0.0298
Training model for product: short_term_depo

Results for'short_term_depo' in variation 2:
Training - ROC AUC: 0.9455, F1 Score: 0.0000
Validation - ROC AUC: 0.9457, F1 Score: 0.0058
Training model for product: medium_term_depo

Results for'medium_term_depo' in variation 2:
Training - ROC AUC: 0.8952, F1 Score: 0.0000
Validation - ROC AUC: 0.9271, F1 Score: 0.0377
Training model for product: long_term_depo
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Results for'long_term_depo' in variation 2:
Training - ROC AUC: 0.9270, F1 Score: 0.3508
Validation - ROC AUC: 0.9384, F1 Score: 0.3035
Training model for product: e_acct

Results for'e_acct' in variation 2:
Training - ROC AUC: 0.8608, F1 Score: 0.2218
Validation - ROC AUC: 0.8738, F1 Score: 0.1717
Training model for product: funds
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Results for'funds' in variation 2:
Training - ROC AUC: 0.9232, F1 Score: 0.0044
Validation - ROC AUC: 0.9381, F1 Score: 0.0611
Training model for product: mortgage

Results for'mortgage' in variation 2:
Training - ROC AUC: 0.9267, F1 Score: 0.0000
Validation - ROC AUC: 0.9486, F1 Score: 0.0059
Training model for product: pension
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Results for'pension' in variation 2:
Training - ROC AUC: 0.9223, F1 Score: 0.0056
Validation - ROC AUC: 0.9338, F1 Score: 0.0317
Training model for product: loans
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Results for'loans' in variation 2:
Training - ROC AUC: 0.8544, F1 Score: 0.0000
Validation - ROC AUC: 0.8878, F1 Score: 0.0000
Training model for product: taxes

Results for'taxes' in variation 2:
Training - ROC AUC: 0.8597, F1 Score: 0.0012
Validation - ROC AUC: 0.8660, F1 Score: 0.0311
Training model for product: credit_card
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Results for'credit_card' in variation 2:
Training - ROC AUC: 0.8906, F1 Score: 0.0061
Validation - ROC AUC: 0.9202, F1 Score: 0.0440
Training model for product: securities
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Results for'securities' in variation 2:
Training - ROC AUC: 0.9142, F1 Score: 0.0074
Validation - ROC AUC: 0.9301, F1 Score: 0.0965
Training model for product: home_acct
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Results for'home_acct' in variation 2:
Training - ROC AUC: 0.8889, F1 Score: 0.0000
Validation - ROC AUC: 0.9137, F1 Score: 0.0091
Training model for product: pensions_2
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Results for'pensions_2' in variation 2:
Training - ROC AUC: 0.8623, F1 Score: 0.0013
Validation - ROC AUC: 0.8596, F1 Score: 0.0105
Training model for product: direct_debt
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Results for'direct_debt' in variation 2:
Training - ROC AUC: 0.8691, F1 Score: 0.0500
Validation - ROC AUC: 0.8709, F1 Score: 0.0606

Training variation 3 with parameters: {'C': 10, 'solver': 'liblinear', 'max_iter': 30
0}
Training model for product: savings_acct

Results for'savings_acct' in variation 3:
Training - ROC AUC: 0.8729, F1 Score: 0.0000
Validation - ROC AUC: 0.9315, F1 Score: 0.0000
Training model for product: guarantees

Results for'guarantees' in variation 3:
Training - ROC AUC: 0.9734, F1 Score: 0.0000
Validation - ROC AUC: 0.9106, F1 Score: 0.0000
Training model for product: current_acct

Results for'current_acct' in variation 3:
Training - ROC AUC: 0.7490, F1 Score: 0.7866
Validation - ROC AUC: 0.7587, F1 Score: 0.7901
Training model for product: derivada_acct

Results for'derivada_acct' in variation 3:
Training - ROC AUC: 0.8808, F1 Score: 0.0000
Validation - ROC AUC: 0.8979, F1 Score: 0.0000
Training model for product: payroll_acct

Results for'payroll_acct' in variation 3:
Training - ROC AUC: 0.8663, F1 Score: 0.0010
Validation - ROC AUC: 0.8642, F1 Score: 0.0090
Training model for product: junior_acct

Results for'junior_acct' in variation 3:
Training - ROC AUC: 0.9996, F1 Score: 0.8854
Validation - ROC AUC: 0.9996, F1 Score: 0.8401
Training model for product: mas_particular_acct

Results for'mas_particular_acct' in variation 3:
Training - ROC AUC: 0.8412, F1 Score: 0.0000
Validation - ROC AUC: 0.8902, F1 Score: 0.0000
Training model for product: particular_acct

Results for'particular_acct' in variation 3:
Training - ROC AUC: 0.8849, F1 Score: 0.2309
Validation - ROC AUC: 0.9303, F1 Score: 0.2574
Training model for product: particular_plus_acct

Results for'particular_plus_acct' in variation 3:
Training - ROC AUC: 0.8137, F1 Score: 0.0000
Validation - ROC AUC: 0.8670, F1 Score: 0.0000
Training model for product: short_term_depo

Results for'short_term_depo' in variation 3:
Training - ROC AUC: 0.9473, F1 Score: 0.0000
Validation - ROC AUC: 0.9438, F1 Score: 0.0126
Training model for product: medium_term_depo

Results for'medium_term_depo' in variation 3:
Training - ROC AUC: 0.8952, F1 Score: 0.0000
```

```
Validation - ROC AUC: 0.9285, F1 Score: 0.0358
Training model for product: long_term_depo

Results for'long_term_depo' in variation 3:
Training - ROC AUC: 0.9270, F1 Score: 0.3501
Validation - ROC AUC: 0.9401, F1 Score: 0.3052
Training model for product: e_acct

Results for'e_acct' in variation 3:
Training - ROC AUC: 0.8612, F1 Score: 0.2203
Validation - ROC AUC: 0.8791, F1 Score: 0.1709
Training model for product: funds

Results for'funds' in variation 3:
Training - ROC AUC: 0.9231, F1 Score: 0.0042
Validation - ROC AUC: 0.9420, F1 Score: 0.0597
Training model for product: mortgage

Results for'mortgage' in variation 3:
Training - ROC AUC: 0.9270, F1 Score: 0.0000
Validation - ROC AUC: 0.9518, F1 Score: 0.0079
Training model for product: pension

Results for'pension' in variation 3:
Training - ROC AUC: 0.9224, F1 Score: 0.0056
Validation - ROC AUC: 0.9397, F1 Score: 0.0367
Training model for product: loans

Results for'loans' in variation 3:
Training - ROC AUC: 0.8544, F1 Score: 0.0000
Validation - ROC AUC: 0.8876, F1 Score: 0.0000
Training model for product: taxes

Results for'taxes' in variation 3:
Training - ROC AUC: 0.8598, F1 Score: 0.0012
Validation - ROC AUC: 0.8734, F1 Score: 0.0234
Training model for product: credit_card

Results for'credit_card' in variation 3:
Training - ROC AUC: 0.8907, F1 Score: 0.0063
Validation - ROC AUC: 0.9235, F1 Score: 0.0438
Training model for product: securities

Results for'securities' in variation 3:
Training - ROC AUC: 0.9142, F1 Score: 0.0074
Validation - ROC AUC: 0.9320, F1 Score: 0.0970
Training model for product: home_acct

Results for'home_acct' in variation 3:
Training - ROC AUC: 0.8892, F1 Score: 0.0000
Validation - ROC AUC: 0.9174, F1 Score: 0.0000
Training model for product: pensions_2

Results for'pensions_2' in variation 3:
Training - ROC AUC: 0.8625, F1 Score: 0.0010
Validation - ROC AUC: 0.8615, F1 Score: 0.0107
Training model for product: direct_debt

Results for'direct_debt' in variation 3:
```

```
Training - ROC AUC: 0.8693, F1 Score: 0.0468
Validation - ROC AUC: 0.8705, F1 Score: 0.0586
```

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

In [15]:
```
train_metrics_df = pd.DataFrame.from_dict({(i,j): metrics[i]['train'][j]
                     for i in metrics.keys()
                     for j in products},
                    orient='index')

val_metrics_df = pd.DataFrame.from_dict({(i,j): metrics[i]['val'][j]
                     for i in metrics.keys()
                     for j in products},
                    orient='index')
```

In [16]:
```
pd.set_option('display.max_rows', None)
print("Training Metrics Table:")
train_metrics_df
```

```
Training Metrics Table:
```

Out[16]:

		ROC AUC	F1 Score	Confusion Matrix
Variation 1	**savings_acct**	0.402916	0.000000	[[706792, 0], [74, 0]]
	guarantees	0.261854	0.000000	[[706850, 0], [16, 0]]
	current_acct	0.748522	0.786467	[[150022, 126938], [69026, 360880]]
	derivada_acct	0.777204	0.000000	[[706603, 0], [263, 0]]
	payroll_acct	0.864570	0.000103	[[667859, 2], [39003, 2]]
	junior_acct	0.998986	0.141183	[[700283, 2], [6081, 500]]
	mas_particular_acct	0.825569	0.000000	[[701121, 0], [5745, 0]]
	particular_acct	0.884666	0.214141	[[600002, 20131], [73919, 12814]]
	particular_plus_acct	0.813231	0.000000	[[677357, 1], [29508, 0]]
	short_term_depo	0.924857	0.000000	[[705857, 0], [1009, 0]]
	medium_term_depo	0.877228	0.000000	[[705817, 0], [1049, 0]]
	long_term_depo	0.926317	0.341200	[[672212, 5132], [22394, 7128]]
	e_acct	0.860581	0.216472	[[640624, 7361], [50841, 8040]]
	funds	0.921595	0.000000	[[694096, 3], [12767, 0]]
	mortgage	0.920838	0.000000	[[702807, 0], [4059, 0]]
	pension	0.920788	0.000000	[[700453, 1], [6412, 0]]
	loans	0.839712	0.000000	[[705210, 0], [1656, 0]]
	taxes	0.859193	0.000784	[[668627, 17], [38207, 15]]
	credit_card	0.890356	0.000129	[[675775, 3], [31086, 2]]
	securities	0.913740	0.000227	[[689223, 12], [17629, 2]]
	home_acct	0.871426	0.000000	[[704156, 0], [2710, 0]]
	pensions_2	0.861401	0.000094	[[664258, 2], [42604, 2]]
	direct_debt	0.868809	0.035647	[[615556, 1308], [88345, 1657]]
Variation 2	**savings_acct**	0.868903	0.000000	[[706792, 0], [74, 0]]
	guarantees	0.964197	0.000000	[[706850, 0], [16, 0]]
	current_acct	0.748034	0.786279	[[150207, 126753], [69288, 360618]]
	derivada_acct	0.880394	0.000000	[[706603, 0], [263, 0]]
	payroll_acct	0.866318	0.000256	[[667856, 5], [39000, 5]]
	junior_acct	0.999425	0.881061	[[699481, 804], [766, 5815]]
	mas_particular_acct	0.841209	0.000000	[[701121, 0], [5745, 0]]
	particular_acct	0.884820	0.229081	[[598646, 21487], [72734, 13999]]
	particular_plus_acct	0.813776	0.000000	[[677356, 2], [29508, 0]]
	short_term_depo	0.945518	0.000000	[[705857, 0], [1009, 0]]
	medium_term_depo	0.895236	0.000000	[[705817, 0], [1049, 0]]

		ROC AUC	F1 Score	Confusion Matrix
	long_term_depo	0.927018	0.350823	[[671929, 5415], [22090, 7432]]
	e_acct	0.860760	0.221820	[[640475, 7510], [50599, 8282]]
	funds	0.923179	0.004361	[[694052, 47], [12739, 28]]
	mortgage	0.926707	0.000000	[[702806, 1], [4059, 0]]
	pension	0.922319	0.005588	[[700442, 12], [6394, 18]]
	loans	0.854446	0.000000	[[705210, 0], [1656, 0]]
	taxes	0.859697	0.001202	[[668620, 24], [38199, 23]]
	credit_card	0.890593	0.006080	[[675713, 65], [30993, 95]]
	securities	0.914192	0.007435	[[689178, 57], [17565, 66]]
	home_acct	0.888943	0.000000	[[704156, 0], [2710, 0]]
	pensions_2	0.862302	0.001266	[[664248, 12], [42579, 27]]
	direct_debt	0.869138	0.049997	[[614470, 2394], [87633, 2369]]
Variation 3	savings_acct	0.872940	0.000000	[[706792, 0], [74, 0]]
	guarantees	0.973446	0.000000	[[706850, 0], [16, 0]]
	current_acct	0.748993	0.786626	[[149726, 127234], [68714, 361192]]
	derivada_acct	0.880811	0.000000	[[706603, 0], [263, 0]]
	payroll_acct	0.866259	0.001025	[[667849, 12], [38985, 20]]
	junior_acct	0.999570	0.885389	[[699470, 815], [706, 5875]]
	mas_particular_acct	0.841194	0.000000	[[701121, 0], [5745, 0]]
	particular_acct	0.884887	0.230861	[[598531, 21602], [72596, 14137]]
	particular_plus_acct	0.813682	0.000000	[[677358, 0], [29508, 0]]
	short_term_depo	0.947280	0.000000	[[705857, 0], [1009, 0]]
	medium_term_depo	0.895238	0.000000	[[705817, 0], [1049, 0]]
	long_term_depo	0.927018	0.350078	[[671952, 5392], [22114, 7408]]
	e_acct	0.861151	0.220272	[[640548, 7437], [50673, 8208]]
	funds	0.923120	0.004206	[[694055, 44], [12740, 27]]
	mortgage	0.927031	0.000000	[[702807, 0], [4059, 0]]
	pension	0.922354	0.005587	[[700441, 13], [6394, 18]]
	loans	0.854412	0.000000	[[705210, 0], [1656, 0]]
	taxes	0.859777	0.001202	[[668623, 21], [38199, 23]]
	credit_card	0.890683	0.006335	[[675711, 67], [30989, 99]]
	securities	0.914217	0.007435	[[689178, 57], [17565, 66]]
	home_acct	0.889242	0.000000	[[704156, 0], [2710, 0]]
	pensions_2	0.862506	0.001032	[[664248, 12], [42584, 22]]

	ROC AUC	F1 Score	Confusion Matrix
direct_debt	0.869315	0.046836	[[614537, 2327], [87788, 2214]]

In [17]:
```
print("Validation Metrics Table:")
val_metrics_df
```

```
Validation Metrics Table:
```

Out[17]:

		ROC AUC	F1 Score	Confusion Matrix
Variation 1	**savings_acct**	0.448868	0.000000	[[200325, 0], [8, 0]]
	guarantees	0.117191	0.000000	[[200328, 3], [2, 0]]
	current_acct	0.756396	0.789899	[[39830, 38322], [17412, 104769]]
	derivada_acct	0.750282	0.000000	[[200289, 0], [44, 0]]
	payroll_acct	0.859517	0.007521	[[192120, 1999], [6183, 31]]
	junior_acct	0.997220	0.127197	[[199214, 0], [1043, 76]]
	mas_particular_acct	0.886404	0.000000	[[197523, 3], [2807, 0]]
	particular_acct	0.926977	0.245744	[[180445, 4825], [12277, 2786]]
	particular_plus_acct	0.858969	0.028845	[[194046, 1811], [4384, 92]]
	short_term_depo	0.912585	0.000000	[[200006, 3], [324, 0]]
	medium_term_depo	0.897696	0.000000	[[200152, 0], [181, 0]]
	long_term_depo	0.935919	0.293059	[[193029, 2643], [3407, 1254]]
	e_acct	0.876874	0.168256	[[187944, 2829], [8422, 1138]]
	funds	0.935718	0.058437	[[197675, 571], [2007, 80]]
	mortgage	0.946640	0.000000	[[199771, 41], [521, 0]]
	pension	0.932103	0.037847	[[198674, 780], [847, 32]]
	loans	0.858123	0.000000	[[199995, 0], [338, 0]]
	taxes	0.865688	0.030680	[[193529, 1867], [4831, 106]]
	credit_card	0.918365	0.040483	[[194089, 1880], [4235, 129]]
	securities	0.928463	0.085966	[[195880, 1401], [2852, 200]]
	home_acct	0.899709	0.000000	[[200015, 0], [318, 0]]
	pensions_2	0.858688	0.009136	[[191616, 1993], [6684, 40]]
	direct_debt	0.870896	0.051411	[[182140, 2092], [15621, 480]]
Variation 2	**savings_acct**	0.933433	0.000000	[[200325, 0], [8, 0]]
	guarantees	0.988746	0.000000	[[200331, 0], [2, 0]]
	current_acct	0.756892	0.789503	[[39858, 38294], [17517, 104664]]
	derivada_acct	0.907970	0.000000	[[200289, 0], [44, 0]]
	payroll_acct	0.860161	0.008951	[[192103, 2016], [6177, 37]]
	junior_acct	0.998574	0.827032	[[199092, 122], [244, 875]]
	mas_particular_acct	0.885058	0.000000	[[197474, 52], [2807, 0]]
	particular_acct	0.925391	0.253438	[[179507, 5763], [12041, 3022]]
	particular_plus_acct	0.861638	0.029767	[[193979, 1878], [4380, 96]]
	short_term_depo	0.945722	0.005831	[[199991, 18], [323, 1]]
	medium_term_depo	0.927063	0.037736	[[199969, 183], [174, 7]]

		ROC AUC	F1 Score	Confusion Matrix
	long_term_depo	0.938429	0.303536	[[192704, 2968], [3296, 1365]]
	e_acct	0.873791	0.171671	[[187851, 2922], [8388, 1172]]
	funds	0.938061	0.061080	[[196365, 1881], [1962, 125]]
	mortgage	0.948643	0.005950	[[197987, 1825], [514, 7]]
	pension	0.933830	0.031701	[[197539, 1915], [834, 45]]
	loans	0.887794	0.000000	[[199576, 419], [338, 0]]
	taxes	0.865983	0.031130	[[193439, 1957], [4828, 109]]
	credit_card	0.920199	0.044042	[[194071, 1898], [4223, 141]]
	securities	0.930085	0.096486	[[195539, 1742], [2809, 243]]
	home_acct	0.913726	0.009112	[[199896, 119], [316, 2]]
	pensions_2	0.859630	0.010488	[[191607, 2002], [6678, 46]]
	direct_debt	0.870889	0.060638	[[181975, 2257], [15527, 574]]
Variation 3	savings_acct	0.931511	0.000000	[[200325, 0], [8, 0]]
	guarantees	0.910590	0.000000	[[200315, 16], [2, 0]]
	current_acct	0.758703	0.790143	[[39445, 38707], [17107, 105074]]
	derivada_acct	0.897898	0.000000	[[200284, 5], [44, 0]]
	payroll_acct	0.864173	0.008957	[[192108, 2011], [6177, 37]]
	junior_acct	0.999552	0.840149	[[199085, 129], [215, 904]]
	mas_particular_acct	0.890196	0.000000	[[197523, 3], [2807, 0]]
	particular_acct	0.930324	0.257369	[[181266, 4004], [12247, 2816]]
	particular_plus_acct	0.866979	0.000000	[[195857, 0], [4476, 0]]
	short_term_depo	0.943757	0.012579	[[199701, 308], [320, 4]]
	medium_term_depo	0.928459	0.035794	[[199894, 258], [173, 8]]
	long_term_depo	0.940121	0.305180	[[192758, 2914], [3297, 1364]]
	e_acct	0.879096	0.170943	[[187857, 2916], [8394, 1166]]
	funds	0.942005	0.059712	[[197506, 740], [2000, 87]]
	mortgage	0.951792	0.007890	[[198818, 994], [515, 6]]
	pension	0.939714	0.036748	[[198570, 884], [846, 33]]
	loans	0.887644	0.000000	[[199658, 337], [338, 0]]
	taxes	0.873381	0.023401	[[194167, 1229], [4864, 73]]
	credit_card	0.923466	0.043750	[[194073, 1896], [4224, 140]]
	securities	0.931958	0.097025	[[195567, 1714], [2809, 243]]
	home_acct	0.917353	0.000000	[[200015, 0], [318, 0]]
	pensions_2	0.861479	0.010707	[[191601, 2008], [6677, 47]]

	ROC AUC	F1 Score	Confusion Matrix
direct_debt	0.870459	0.058622	[[181953, 2279], [15546, 555]]

Creating a summary table for all the variations and different datasets

In [18]:
```
summary_data = []
for variation in metrics:
    for dataset in ['train', 'val']:
        avg_roc_auc = np.mean([metrics[variation][dataset][p]['ROC AUC'] for p in proc
        avg_f1 = np.mean([metrics[variation][dataset][p]['F1 Score'] for p in products
        summary_data.append([variation, dataset, avg_roc_auc, avg_f1])

summary_df = pd.DataFrame(summary_data, columns=['Variation', 'Dataset', 'Avg ROC AUC'
print("Summary Table:")
print(summary_df.to_string(index=False))

best_variation = summary_df[summary_df['Dataset'] == 'val'].sort_values('Avg ROC AUC',
print(f"\nBest Model For This Week: {best_variation}")
```

```
Summary Table:
  Variation Dataset  Avg ROC AUC  Avg F1 Score
Variation 1   train     0.827581      0.075498
Variation 1     val     0.836491      0.085847
Variation 2   train     0.887266      0.110663
Variation 2     val     0.907466      0.120787
Variation 3   train     0.888049      0.110734
Variation 3     val     0.906114      0.119955

Best Model For This Week: Variation 2
```

Generate product recommendations

Here we want to visualize the product recommendations for each customer

In [19]:
```
best_params = hyperparameter_variations[int(best_variation.split()[-1]) - 1]

# Train models and predict for each product for all customers using parameters from Va
product_models = {}
for product in products:
    clf = LogisticRegression(**best_params)
    clf.fit(X_train, y_train[product])
    product_models[product] = clf

train_preds = {}
for product in products:
    proba = product_models[product].predict_proba(X_train)[:, 1]
    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))
```

```
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
c:\Users\MARIA\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\line
ar_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

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 [20]:
```python
def get_active_products(customer_data):
    return set(product for product in products 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 [21]:
```python
for customer_id in train_preds:
    # Sorting by probability
    sorted_prods = sorted(train_preds[customer_id], key=lambda x: x[1], reverse=True)
    customer_data = train[train['customer_code'] == customer_id].iloc[0]
    # Filter out already active products
    active_products = get_active_products(customer_data)
    recommended_products = [prod for prod, _ in sorted_prods if prod not in active_prc

    # Get top 7
    train_preds[customer_id] = recommended_products[:7]

# Example recommendations
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', 'e_acct', 'particular_acct', 'taxes', 'particular_
plus_acct', 'mas_particular_acct', 'direct_debt']
Customer 1358829: ['direct_debt', 'payroll_acct', 'pensions_2', 'e_acct', 'taxes', 'c
redit_card', 'long_term_depo']
Customer 1436539: ['direct_debt', 'payroll_acct', 'pensions_2', 'e_acct', 'short_term
_depo', 'taxes', 'credit_card']
Customer 1448049: ['current_acct', 'e_acct', 'taxes', 'mas_particular_acct', 'direct_
debt', 'home_acct', 'payroll_acct']
Customer 1396837: ['direct_debt', 'payroll_acct', 'pensions_2', 'mas_particular_acc
t', 'e_acct', 'junior_acct', 'long_term_depo']
```

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

In [22]:
```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 (652073 times)
Least frequently recommended product: savings_acct (2 times)
```