Week 9 - Select the Winning Model

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')
test = pd.read_csv('test_4_11.csv')
```

In [3]:

```
train.head()
```

Out[3]:

	Unnamed: 0	date	customer_code	employee_index	country_spain	female	age	new_cust	sen
0	0	2016-04-28	1334092	N	1	0	0.234694	0	
1	1	2015-07-28	1024586	N	1	0	0.234694	0	
2	2	2016-04-28	856204	N	1	0	0.306122	0	
3	3	2015-08-28	295807	N	1	0	0.489796	0	
4	4	2016-03-28	942624	N	1	1	0.224490	0	

In [4]:

```
validation.head()
```

Out[4]:

	Unnamed: 0	date	customer_code	employee_index	country_spain	female	age	first_contract_
0	0	2016-05-28	1212130	N	1	0	0.204082	2013-1
1	1	2015-07-28	84306	N	1	0	0.500000	1998-0
2	2	2015-07-28	883630	N	1	0	0.418367	2010-0
3	3	2016-05-28	1464700	N	1	1	0.183673	2015-0
4	4	2015-12-28	487783	N	1	1	0.418367	2004-1

In [5]:

```
test.head()
```

Out[5]:

	Unnamed: 0	date	customer_code	employee_index	country_spain	female	age	new_cust	sen
0	0	2015-06-28	49335	N	1	0	0.734694	0	
1	1	2016-02-28	1174349	N	1	0	0.214286	0	
2	2	2015-07-28	1393286	N	1	0	0.244898	1	
3	3	2016-03-28	1454346	N	1	0	0.183673	0	
4	4	2016-02-28	1074431	N	1	0	0.234694	0	

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'])
drop = ['join_channel', 'province_name', 'employee_index', 'segment', 'total_products'
train = train.drop(columns=drop)
validation = validation.drop(columns=drop + ['payroll_acct.1', 'first_contract_date',

test = test.drop(columns=['Unnamed: 0'])
test = test.drop(columns=drop + ['payroll_acct.1'])
```

Reading into the data

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'])]
```

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]

X_test = test.drop(['customer_code', 'date'] + products, axis=1)
y_test = test[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)

print("Shape of X_test:", X_test.shape)
print("Shape of y_test:", y_test.shape)
```

```
Shape of X_train: (706816, 17)
Shape of y_train: (706816, 23)
Shape of X_val: (179432, 17)
Shape of y_val: (179432, 23)
Shape of X_test: (1236744, 17)
Shape of y_test: (1236744, 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):

    for product in products:
        clf = LogisticRegression(**params)

        y_train_product = y_train[product].values
        y_val_product = y_val[product].values

        # Train the model on each 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)
        }
```

```
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(
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(
```

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')
```

Creating a summary table for all the variations and different datasets

In [16]:

```
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     0.827547      0.075518
Variation 1     val     0.833304      0.085716
Variation 2   train     0.887300      0.110169
Variation 2     val     0.907706      0.119564
Variation 3   train     0.888038      0.110746
Variation 3     val     0.909773      0.118087

Best Model For This Week: Variation 3
```

In [17]:

```
# Getting the best variation
best_index = int(best_variation.split()[-1]) - 1
best_params = hyperparameter_variations[best_index]
```

In [19]:

```
test_metrics = {}

for product in products:

    clf = LogisticRegression(**best_params)

    y_train_product = y_train[product].values
    y_test_product = y_test[product].values

    clf.fit(X_train, y_train_product)

    y_test_pred = clf.predict(X_test)
    y_test_pred_proba = clf.predict_proba(X_test)[:, 1]

    test_metrics[product] = {
        'ROC AUC': roc_auc_score(y_test_product, y_test_pred_proba),
        'F1 Score': f1_score(y_test_product, y_test_pred),
        'Confusion Matrix': confusion_matrix(y_test_product, y_test_pred)
    }

for product, metric in test_metrics.items():
    print(f"\nResults for '{product}' on the test set:")
    print(f"ROC AUC: {metric['ROC AUC']:.4f}")
    print(f"F1 Score: {metric['F1 Score']:.4f}")
    print(f"Confusion Matrix:\n{metric['Confusion Matrix']}")
```

```
Results for 'savings_acct' on the test set:
ROC AUC: 0.8784
F1 Score: 0.0000
Confusion Matrix:
[[1236601       0]
 [    143       0]]

Results for 'guarantees' on the test set:
ROC AUC: 0.9692
F1 Score: 0.0000
Confusion Matrix:
[[1236709       0]
 [     35       0]]

Results for 'current_acct' on the test set:
ROC AUC: 0.7454
F1 Score: 0.7896
Confusion Matrix:
[[246591 225370]
 [118899 645884]]

Results for 'derivada_acct' on the test set:
ROC AUC: 0.8503
F1 Score: 0.0039
Confusion Matrix:
[[1236230      33]
 [    480       1]]

Results for 'payroll_acct' on the test set:
ROC AUC: 0.8640
F1 Score: 0.2868
Confusion Matrix:
[[937865 229213]
 [ 19627  50039]]

Results for 'junior_acct' on the test set:
ROC AUC: 0.9996
F1 Score: 0.8862
Confusion Matrix:
[[1224003    1146]
 [   1458   10137]]

Results for 'mas_particular_acct' on the test set:
ROC AUC: 0.8396
F1 Score: 0.0000
Confusion Matrix:
[[1226568       9]
 [  10167       0]]

Results for 'particular_acct' on the test set:
ROC AUC: 0.8831
F1 Score: 0.1925
Confusion Matrix:
[[1047910   31921]
 [ 136798   20115]]

Results for 'particular_plus_acct' on the test set:
ROC AUC: 0.8101
F1 Score: 0.0006
Confusion Matrix:
```

```
[[1183613      30]
 [  53085      16]]

Results for 'short_term_depo' on the test set:
ROC AUC: 0.9399
F1 Score: 0.1394
Confusion Matrix:
[[1224649   10498]
 [    691     906]]

Results for 'medium_term_depo' on the test set:
ROC AUC: 0.8942
F1 Score: 0.0000
Confusion Matrix:
[[1234898       0]
 [   1846       0]]

Results for 'long_term_depo' on the test set:
ROC AUC: 0.9239
F1 Score: 0.2238
Confusion Matrix:
[[828437 355603]
 [  1255  51449]]

Results for 'e_acct' on the test set:
ROC AUC: 0.8570
F1 Score: 0.3687
Confusion Matrix:
[[853632 277938]
 [ 18592  86582]]

Results for 'funds' on the test set:
ROC AUC: 0.9190
F1 Score: 0.2889
Confusion Matrix:
[[1181411   32575]
 [  13414    9344]]

Results for 'mortgage' on the test set:
ROC AUC: 0.9238
F1 Score: 0.0419
Confusion Matrix:
[[1225048    4446]
 [   7000     250]]

Results for 'pension' on the test set:
ROC AUC: 0.9201
F1 Score: 0.1304
Confusion Matrix:
[[1220905    4185]
 [  10549    1105]]

Results for 'loans' on the test set:
ROC AUC: 0.8322
F1 Score: 0.0000
Confusion Matrix:
[[1233790       0]
 [   2954       0]]

Results for 'taxes' on the test set:
```

```
ROC AUC: 0.8532
F1 Score: 0.0681
Confusion Matrix:
[[1164200    3919]
 [  66067    2558]]

Results for 'credit_card' on the test set:
ROC AUC: 0.8862
F1 Score: 0.2446
Confusion Matrix:
[[867925 313061]
 [  4359  51399]]

Results for 'securities' on the test set:
ROC AUC: 0.9110
F1 Score: 0.2633
Confusion Matrix:
[[1119610   85764]
 [  13614   17756]]

Results for 'home_acct' on the test set:
ROC AUC: 0.8856
F1 Score: 0.0000
Confusion Matrix:
[[1231825       0]
 [   4919       0]]

Results for 'pensions_2' on the test set:
ROC AUC: 0.8600
F1 Score: 0.2513
Confusion Matrix:
[[720045 440504]
 [  1927  74268]]

Results for 'direct_debt' on the test set:
ROC AUC: 0.8656
F1 Score: 0.4523
Confusion Matrix:
[[688926 387106]
 [   632 160080]]
```

In [20]:

```
test_summary_data = []
for product, metric in test_metrics.items():
    test_summary_data.append([product, metric['ROC AUC'], metric['F1 Score']])

test_summary_df = pd.DataFrame(test_summary_data, columns=['Product', 'ROC AUC', 'F1 S
print("\nTest Summary Table:")
print(test_summary_df.to_string(index=False))
```

```
Test Summary Table:
             Product  ROC AUC  F1 Score
        savings_acct 0.878352  0.000000
          guarantees 0.969186  0.000000
        current_acct 0.745434  0.789571
       derivada_acct 0.850344  0.003883
        payroll_acct 0.863972  0.286824
         junior_acct 0.999608  0.886179
  mas_particular_acct 0.839557  0.000000
     particular_acct 0.883083  0.192535
particular_plus_acct 0.810068  0.000602
     short_term_depo 0.939935  0.139374
    medium_term_depo 0.894157  0.000000
      long_term_depo 0.923875  0.223810
              e_acct 0.857049  0.368674
               funds 0.918984  0.288944
            mortgage 0.923781  0.041855
             pension 0.920110  0.130430
               loans 0.832159  0.000000
               taxes 0.853217  0.068121
         credit_card 0.886230  0.244630
          securities 0.910982  0.263266
           home_acct 0.885591  0.000000
          pensions_2 0.860020  0.251344
         direct_debt 0.865578  0.452269
```

In [22]:

```
average_roc_auc = test_summary_df['ROC AUC'].mean()
average_f1_score = test_summary_df['F1 Score'].mean()

print(f"Average ROC AUC for the whole model: {average_roc_auc:.4f}")
print(f"Average F1 Score for the whole model: {average_f1_score:.4f}")
```

```
Average ROC AUC for the whole model: 0.8831
Average F1 Score for the whole model: 0.2014
```