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.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel
```

In [2]:
```
pd.set_option('display.max_columns', None)

train = pd.read_csv('train_data.csv', low_memory=False)
```

In [3]:
```
# # Reading test and validation dataset to transform categorical values on those as we
# test = pd.read_csv('test_data.csv')
# val_set = pd.read_csv('val_set_data.csv')
```

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

Out[4]:

	date	customer_code	employee_index	country	female	age	new_cust	seniority_in_months
0	2015-07-28	664160	N	1	0	0.632653	0	0.402344
1	2016-01-28	1076784	N	1	0	0.214286	0	0.152344
2	2015-12-28	672465	N	1	0	0.387755	0	0.417969
3	2015-10-28	774528	N	1	0	0.397959	0	0.343750
4	2016-05-28	569598	N	1	0	0.459184	0	0.496094

In [5]:
```
train['date'] = pd.to_datetime(train['date'])
```

In [7]:
```
# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Apply LabelEncoder on the customer_code column
train['customer_code_encoded'] = label_encoder.fit_transform(train['customer_code'])

# Apply the same encoding to the test and validation set
# test['customer_code_encoded'] = label_encoder.transform(test['customer_code'])
# val_set['customer_code_encoded'] = label_encoder.transform(val_set['customer_code'])

# Display the first few rows in test and val_set to verify the encoding
```

```
# print(test[['customer_code', 'customer_code_encoded']].head())
# print(val_set[['customer_code', 'customer_code_encoded']].head())
print(train[['customer_code', 'customer_code_encoded']].head())
```

```
   customer_code  customer_code_encoded
0         664160                 263662
1        1076784                 459750
2         672465                 266890
3         774528                 300528
4         569598                 227731
```

In [8]:
```
# Convert all boolean columns (True/False) to integers (1/0)
train = train.map(lambda x: int(x) if isinstance(x, bool) else x)
# test = test.map(lambda x: int(x) if isinstance(x, bool) else x)
# val_set = val_set.map(lambda x: int(x) if isinstance(x, bool) else x)

# Display the first few rows to check the changes
print(train.head())
# print(test.head())
# print(val_set.head())
```

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

   new_cust  seniority_in_months  cust_type  residency_spain  birth_spain  \
0         0             0.402344          1                1            0   
1         0             0.152344          1                1            0   
2         0             0.417969          1                1            0   
3         0             0.343750          1                1            0   
4         0             0.496094          1                1            0   

  join_channel province_name  active_cust    income             segment  \
0          KAR        MADRID            0  1.989686   02 - PARTICULARES   
1          KHE        LERIDA            0 -0.306603  03 - UNIVERSITARIO   
2          KFC       SEVILLA            1 -0.148205   02 - PARTICULARES   
3          KFA        MURCIA            1 -0.228531   02 - PARTICULARES   
4          KAT        MADRID            1  0.588748   02 - PARTICULARES   

   savings_acct  guarantees  current_acct  derivada_acct  payroll_acct  \
0             0           0             1              0             0   
1             0           0             1              0             0   
2             0           0             0              0             1   
3             0           0             1              0             0   
4             0           0             1              0             0   

   junior_acct  mas_particular_acct  particular_acct  particular_plus_acct  \
0            0                    0                0                     0   
1            0                    0                0                     0   
2            0                    0                0                     0   
3            0                    0                0                     0   
4            0                    0                0                     0   

   short_term_depo  medium_term_depo  long_term_depo  e_acct  funds  mortgage  \
0                0                 0               0       0      0         0   
1                0                 0               0       0      0         0   
2                0                 0               0       0      0         0   
3                0                 0               0       0      0         0   
4                0                 0               0       0      0         0   

   pension  loans  taxes  credit_card  securities  home_acct  pensions_2  \
0        0      0      0            0           0          0           0   
1        0      0      0            0           0          0           0   
2        0      0      0            0           0          0           1   
3        0      0      0            0           0          0           0   
4        0      0      0            0           0          0           0   

   direct_debt  total_products  01 - TOP  02 - PARTICULARES  \
0            0               1         0                  1   
1            0               1         0                  0   
2            1               4         0                  1   
3            1               2         0                  1   
4            0               1         0                  1   

   03 - UNIVERSITARIO  join_channel_encoded  province_name_encoded  \
0                   0              1.424185               1.749698   
1                   1              0.886876               1.006139   
2                   0              1.559984               1.382030   
```

```
3                  0            1.850124            1.075147
4                  0            1.942077            1.749698

   employee_index_encoded  customer_code_encoded
0                1.407278                 263662
1                1.407278                 459750
2                1.407278                 266890
3                1.407278                 300528
4                1.407278                 227731
```

In [9]:
```
train = train.rename(columns={'country': 'country_spain'})
```

In [10]:
```
df_encoded = train
```

In [11]:
```
# List of columns you want to drop
columns_to_drop = ['customer_code', 'employee_index', 'join_channel', 'province_name',

# Drop the columns from the DataFrame
df_encoded = df_encoded.drop(columns=columns_to_drop)

# Display the first few rows to confirm the columns were dropped
df_encoded.head()
```

Out[11]:

	date	country_spain	female	age	new_cust	seniority_in_months	cust_type	residency_spain
0	2015-07-28	1	0	0.632653	0	0.402344	1	1
1	2016-01-28	1	0	0.214286	0	0.152344	1	1
2	2015-12-28	1	0	0.387755	0	0.417969	1	1
3	2015-10-28	1	0	0.397959	0	0.343750	1	1
4	2016-05-28	1	0	0.459184	0	0.496094	1	1

New variables

Customers with higher income relative to the number of products they hold may indicate a propensity for wealth management.

In [12]:
```
# 1. Income to Product Ratio
df_encoded['income_to_product_ratio'] = df_encoded['income'] / df_encoded['total_produ
df_encoded['income_to_product_ratio']
```

Out[12]:
```
0          1.989686
1         -0.306603
2         -0.037051
3         -0.114266
4          0.588748
             ...
6579712    1.700197
6579713   -0.402519
6579714    0.954580
6579715   -0.010249
6579716   -0.214432
Name: income_to_product_ratio, Length: 6579717, dtype: float64
```

Income to Age Ratio: This metric helps identify customers who might have high disposable income.

In [13]:
```
# 2. Income to Age Ratio
df_encoded['income_to_age'] = train['income'] / (df_encoded['age'] + 1e-5)  # Avoid di
df_encoded['income_to_age']
```

Out[13]:
```
0          3.144938
1         -1.430748
2         -0.382203
3         -0.574243
4          1.282133
             ...
6579712    2.031918
6579713   -0.730484
6579714    1.670486
6579715   -0.118158
6579716   -0.525345
Name: income_to_age, Length: 6579717, dtype: float64
```

In [14]:
```
df_encoded['total_savings'] = (df_encoded['savings_acct'] + df_encoded['short_term_dep
                               df_encoded['medium_term_depo'] + df_encoded['long_term_depo
```

In [15]:
```
#Create function to calculate the probabiity

def calculate_product_probabilities(df, product_columns):
    product_counts = df_encoded[product_columns].sum()

    # Calculate total number of observations
    total_observations = len(df_encoded)

    # Calculate probabilities
    product_probabilities = product_counts / total_observations

    # Create a DataFrame to return
    probabilities_df = product_probabilities.reset_index()
    probabilities_df.columns = ['Product', 'Probability']

    probabilities_df = probabilities_df.sort_values(by='Probability', ascending=False)

    return probabilities_df
```

In [16]:
```
product_columns = ['savings_acct', 'guarantees', 'current_acct', 'derivada_acct', 'pay
             'junior_acct', 'mas_particular_acct', 'particular_acct', 'particular_plus_
             'short_term_depo', 'medium_term_depo', 'long_term_depo', 'e_acct', 'funds'
             'mortgage', 'pension', 'loans', 'taxes', 'credit_card', 'securities',
```

```
            'home_acct', 'payroll_acct', 'pensions_2', 'direct_debt']

# Call the function with your DataFrame
probabilities = calculate_product_probabilities(df_encoded, product_columns)

# Display the resulting DataFrame
print(probabilities)
```

```
                 Product  Probability
2           current_acct     0.618343
23           direct_debt     0.130519
7        particular_acct     0.126079
12                e_acct     0.085384
22            pensions_2     0.061948
4           payroll_acct     0.056727
21          payroll_acct     0.056727
17                 taxes     0.055590
18           credit_card     0.045349
8   particular_plus_acct     0.043026
11        long_term_depo     0.042750
19            securities     0.025600
13                 funds     0.018571
5            junior_acct     0.009495
15               pension     0.009374
6    mas_particular_acct     0.008207
14              mortgage     0.005955
20             home_acct     0.003935
16                 loans     0.002400
10      medium_term_depo     0.001523
9        short_term_depo     0.001260
3          derivada_acct     0.000398
0           savings_acct     0.000102
1             guarantees     0.000023
```

We will use this as a guidance to recommend the product.

In [17]:
```
df_encoded.shape
```

Out[17]:
```
(6579717, 45)
```

Feature Engineering

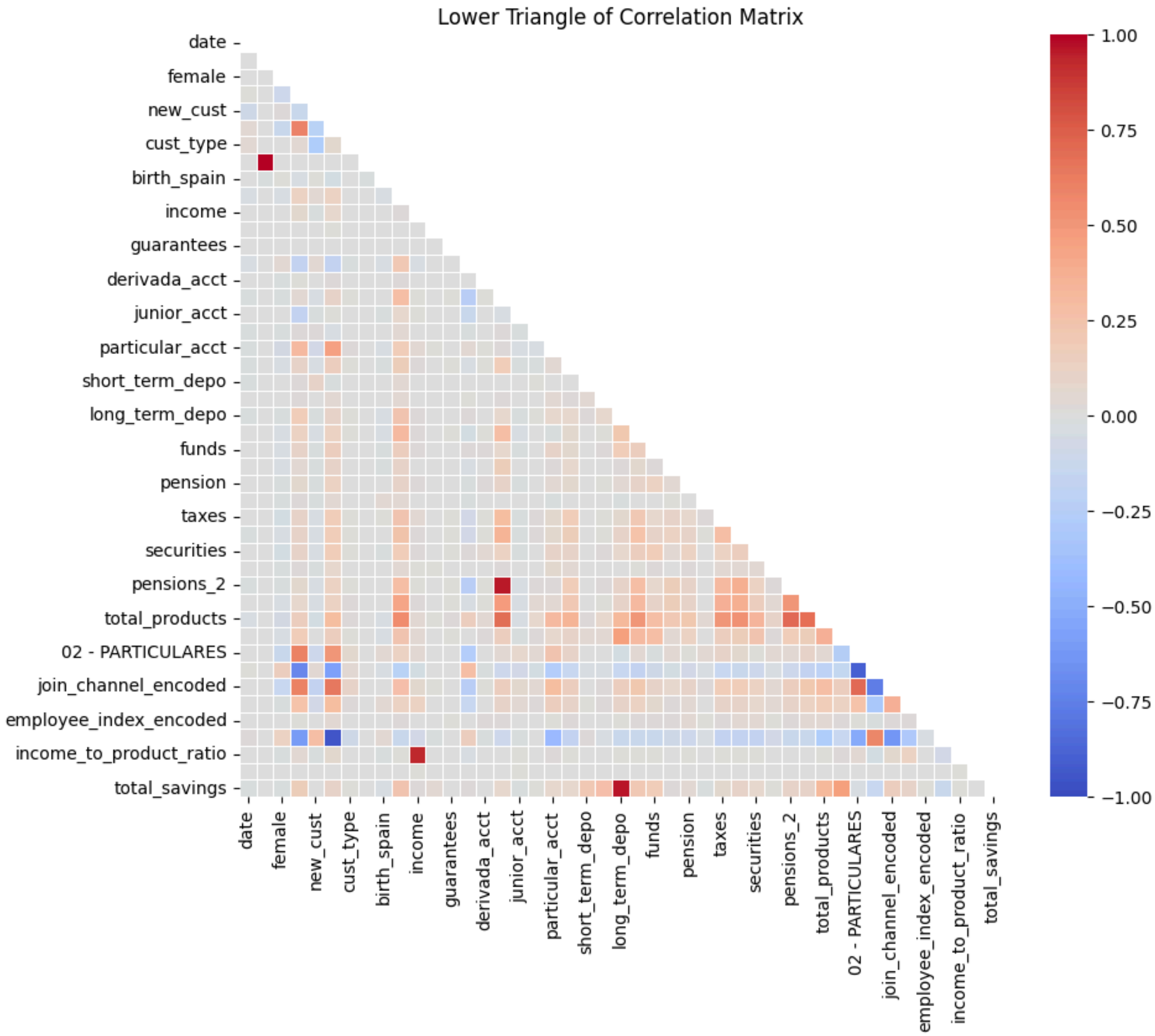
In [18]:
```
# Compute the correlation matrix
corr = df_encoded.corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
plt.figure(figsize=(10, 8))

# Create a seaborn heatmap with the mask for the upper triangle
sns.heatmap(corr, mask=mask, annot=False, cmap='coolwarm', vmin=-1, vmax=1, square=Tru

# Display the plot
plt.title('Lower Triangle of Correlation Matrix')
plt.show()
```



Since these are vaguely correlated, it might not help to use PCA for dimension reduction

PCA

In [17]:
```
# # Select numerical features
# numerical_features = train.select_dtypes(include=['float64', 'int64'])

# # Standardizing the features
# scaler = StandardScaler()
# numerical_features_scaled = scaler.fit_transform(numerical_features)

# # PCA Implementation
# pca = PCA(n_components=0.95)  # Retain 95% of variance
# principal_components = pca.fit_transform(numerical_features_scaled)
```

In [18]:
```
# # Create a DataFrame for the PCA components
# pca_columns = [f'pca_{i+1}' for i in range(principal_components.shape[1])]
# train_pca = pd.DataFrame(data=principal_components, columns=pca_columns)
```

In [19]:
```
# Combine PCA components back with original DataFrame
# train_pca1 = pd.concat([train.reset_index(drop=True), train_pca.reset_index(drop=Tru
```

```
# train.head()
```

In [20]:
```
# #print the eigen values
# print("Explained variance ratio of each component:", pca.explained_variance_ratio_)
```

In [21]:
```
# plt.figure(figsize=(8, 6))
# plt.scatter(train_pca1['pca_1'], train_pca1['pca_2'], alpha=0.5)
# plt.title('PCA Component 1 vs Component 2')
# plt.xlabel('PCA Component 1')
# plt.ylabel('PCA Component 2')
# plt.grid()
# plt.show()
```

Feature selection

In [19]:
```
product_features = df_encoded[product_columns]
```

In [20]:
```
user_features = df_encoded.drop(columns=product_columns)
user_features.drop('date', axis=1, inplace=True)
```

In [21]:
```
print("User Features Shape: ", user_features.shape)
print("Product Features Shape: ", product_features.shape)
```

```
User Features Shape:  (6579717, 21)
Product Features Shape:  (6579717, 24)
```

In [22]:
```
X = user_features
y = product_features
```

In [23]:
```
# Store feature importances for all products
feature_importances = pd.DataFrame(index=X.columns)

for product in y.columns:
    X_train, X_test, y_train, y_test = train_test_split(X, y[product], test_size=0.2,

    # RF with parallel processing
    rf = RandomForestClassifier(n_estimators=50, random_state=42, n_jobs=-1)
    rf.fit(X_train, y_train)

    # Store feature importances for the current product
    feature_importances[product] = rf.feature_importances_

# Print top features for each product
for product in feature_importances.columns:
    print(f"Top features for {product}:")
    top_features = feature_importances[product].sort_values(ascending=False).head(10)
    print(top_features)
    print("\n")
```

```
Top features for savings_acct:
income                     0.150026
customer_code_encoded      0.145615
income_to_product_ratio    0.134967
active_cust                0.131859
income_to_age              0.122386
age                        0.059167
seniority_in_months        0.056159
total_products             0.051564
province_name_encoded      0.043833
total_savings              0.038495
Name: savings_acct, dtype: float64


Top features for guarantees:
income                     0.219881
customer_code_encoded      0.204270
income_to_product_ratio    0.183386
income_to_age              0.177039
seniority_in_months        0.056700
age                        0.055959
total_products             0.041000
female                     0.015331
join_channel_encoded       0.015092
total_savings              0.013095
Name: guarantees, dtype: float64


Top features for current_acct:
total_products             0.481619
customer_code_encoded      0.074443
age                        0.064783
income_to_product_ratio    0.063346
income                     0.057294
income_to_age              0.055301
active_cust                0.043672
join_channel_encoded       0.043264
seniority_in_months        0.040608
province_name_encoded      0.028459
Name: current_acct, dtype: float64


Top features for derivada_acct:
income                     0.207479
income_to_product_ratio    0.184919
customer_code_encoded      0.184267
income_to_age              0.164446
age                        0.072552
seniority_in_months        0.055015
province_name_encoded      0.041464
total_products             0.040045
join_channel_encoded       0.029849
total_savings              0.005193
Name: derivada_acct, dtype: float64


Top features for payroll_acct:
total_products             0.536286
income_to_product_ratio    0.085550
income_to_age              0.058419
```

```
customer_code_encoded      0.056871
income                     0.056316
age                        0.051787
seniority_in_months        0.039896
active_cust                0.038681
join_channel_encoded       0.020124
total_savings              0.016661
Name: payroll_acct, dtype: float64


Top features for junior_acct:
age                        0.782805
total_products             0.083060
income_to_age              0.058719
active_cust                0.016471
join_channel_encoded       0.012058
customer_code_encoded      0.011977
income_to_product_ratio    0.007391
seniority_in_months        0.006929
income                     0.006484
new_cust                   0.005340
Name: junior_acct, dtype: float64


Top features for mas_particular_acct:
customer_code_encoded      0.265559
seniority_in_months        0.153728
income_to_product_ratio    0.095696
income                     0.092929
income_to_age              0.088706
total_products             0.088699
age                        0.072574
join_channel_encoded       0.046050
province_name_encoded      0.039559
active_cust                0.015595
Name: mas_particular_acct, dtype: float64


Top features for particular_acct:
seniority_in_months        0.178946
total_products             0.178607
customer_code_encoded      0.177448
income_to_product_ratio    0.091419
income                     0.083074
age                        0.082105
income_to_age              0.080537
join_channel_encoded       0.035153
province_name_encoded      0.033288
active_cust                0.027269
Name: particular_acct, dtype: float64


Top features for particular_plus_acct:
customer_code_encoded      0.201030
total_products             0.129670
seniority_in_months        0.125185
income_to_product_ratio    0.119368
income                     0.111244
income_to_age              0.104906
age                        0.077333
```

```
province_name_encoded      0.040592
join_channel_encoded       0.039423
active_cust                0.018318
Name: particular_plus_acct, dtype: float64


Top features for short_term_depo:
total_savings              0.334159
seniority_in_months        0.102058
customer_code_encoded      0.094495
income                     0.081934
income_to_product_ratio    0.080532
income_to_age              0.077346
age                        0.067829
join_channel_encoded       0.035363
total_products             0.032061
province_name_encoded      0.031593
Name: short_term_depo, dtype: float64


Top features for medium_term_depo:
total_savings              0.388705
income                     0.105810
customer_code_encoded      0.104587
income_to_product_ratio    0.098489
income_to_age              0.096999
age                        0.058526
seniority_in_months        0.046620
join_channel_encoded       0.028844
province_name_encoded      0.027295
total_products             0.023775
Name: medium_term_depo, dtype: float64


Top features for long_term_depo:
total_savings              0.801103
01 - TOP                   0.072204
total_products             0.024375
02 - PARTICULARES          0.016546
active_cust                0.016341
age                        0.011355
customer_code_encoded      0.010558
seniority_in_months        0.010087
income_to_product_ratio    0.009313
income_to_age              0.007631
Name: long_term_depo, dtype: float64


Top features for e_acct:
total_products             0.152858
customer_code_encoded      0.132103
income_to_product_ratio    0.119507
income                     0.107342
income_to_age              0.101691
age                        0.091362
seniority_in_months        0.076319
join_channel_encoded       0.047845
active_cust                0.044392
province_name_encoded      0.041876
Name: e_acct, dtype: float64
```

```
Top features for funds:
customer_code_encoded      0.148714
income                     0.141107
income_to_product_ratio    0.139097
income_to_age              0.131653
age                        0.092453
total_products             0.085708
seniority_in_months        0.075338
province_name_encoded      0.048089
01 - TOP                   0.041367
join_channel_encoded       0.038511
Name: funds, dtype: float64


Top features for mortgage:
customer_code_encoded      0.160195
income                     0.155460
income_to_product_ratio    0.153717
income_to_age              0.141241
age                        0.087063
total_products             0.081423
seniority_in_months        0.075101
province_name_encoded      0.055274
join_channel_encoded       0.046578
female                     0.012386
Name: mortgage, dtype: float64


Top features for pension:
customer_code_encoded      0.166026
income                     0.159026
income_to_product_ratio    0.154290
income_to_age              0.145950
age                        0.092795
seniority_in_months        0.072714
total_products             0.069665
province_name_encoded      0.047206
join_channel_encoded       0.034225
female                     0.013580
Name: pension, dtype: float64


Top features for loans:
join_channel_encoded       0.171391
customer_code_encoded      0.151088
income                     0.145957
income_to_product_ratio    0.143202
income_to_age              0.122515
age                        0.072709
province_name_encoded      0.056720
seniority_in_months        0.050123
total_products             0.039852
birth_spain                0.014895
Name: loans, dtype: float64


Top features for taxes:
total_products             0.187044
```

```
income_to_product_ratio    0.136643
customer_code_encoded      0.130083
income                     0.122971
income_to_age              0.115140
age                        0.080697
seniority_in_months        0.071173
province_name_encoded      0.044298
join_channel_encoded       0.041436
active_cust                0.032692
Name: taxes, dtype: float64


Top features for credit_card:
total_products             0.251439
income_to_product_ratio    0.128461
customer_code_encoded      0.116539
income                     0.109145
income_to_age              0.103685
seniority_in_months        0.075451
age                        0.074473
province_name_encoded      0.040497
join_channel_encoded       0.036119
active_cust                0.031333
Name: credit_card, dtype: float64


Top features for securities:
customer_code_encoded      0.150649
income_to_product_ratio    0.139164
income                     0.136940
income_to_age              0.128634
total_products             0.104252
age                        0.092613
seniority_in_months        0.080837
province_name_encoded      0.049161
join_channel_encoded       0.038594
01 - TOP                   0.023625
Name: securities, dtype: float64


Top features for home_acct:
income                     0.183951
income_to_product_ratio    0.176212
customer_code_encoded      0.173810
income_to_age              0.160687
age                        0.071120
seniority_in_months        0.065040
total_products             0.046384
province_name_encoded      0.044355
join_channel_encoded       0.036430
female                     0.015797
Name: home_acct, dtype: float64


Top features for pensions_2:
total_products             0.509767
income_to_product_ratio    0.094730
income                     0.069637
customer_code_encoded      0.060335
income_to_age              0.056505
```

```
seniority_in_months        0.043584
age                        0.043224
active_cust                0.041302
join_channel_encoded       0.021970
province_name_encoded      0.017967
Name: pensions_2, dtype: float64


Top features for direct_debt:
total_products             0.347449
active_cust                0.114795
income_to_product_ratio    0.106846
customer_code_encoded      0.080131
income                     0.075471
income_to_age              0.067558
age                        0.054427
seniority_in_months        0.050823
join_channel_encoded       0.031087
province_name_encoded      0.027168
Name: direct_debt, dtype: float64
```