Week 12 - Save and package your model for deployment

In [15]:
```
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
from sklearn.preprocessing import MinMaxScaler
from joblib import Parallel, delayed
import pickle
```

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')
```

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

In [3]:
```
train = train.rename(columns={'country': 'country_spain'})
```

In [4]:
```
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 [5]:
```
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']
```

Pre-processing

Defining our Xs and Ys

In [6]:
```
train_2 = train.copy()
test_2 = test.copy()
```

Transformation #2

For tranformation #2 we will add the date column as one of the features. For that, we will calculate the time since purchase using the month we are trying to predict on June 2016. For this transformation to make sense, we will also keep the first transformation, since the time line of purchase matters now, we will keep the duplicate clients' purchases instead of only keeping the last one

In [7]:
```
train_2['date'] = pd.to_datetime(train_2['date'], format='%Y-%m-%d')

train_2['date'] = train_2['date'].dt.to_period('M').dt.to_timestamp()

# Setting our prediction date, June 28, 2016, as the reference date
reference_date = pd.to_datetime("2016-06-28")

# Calculate time since purchase
train_2['months_since_purchase'] = (reference_date.year - train_2['date'].dt.year) * 1
                                   (reference_date.month - train_2['date'].dt.month)

print(train_2[['date', 'months_since_purchase']])
```

```
               date  months_since_purchase
0        2016-04-01                      2
1        2015-07-01                     11
2        2016-04-01                      2
3        2015-08-01                     10
4        2016-03-01                      3
...             ...                    ...
5757281  2016-05-01                      1
5757282  2015-08-01                     10
5757283  2015-11-01                      7
5757284  2016-05-01                      1
5757285  2016-01-01                      5

[5757286 rows x 2 columns]
```

In [8]:
```
# Adding feature on test dateased
test_2['date'] = pd.to_datetime(test_2['date'], format='%Y-%m-%d')
test_2['date'] = test_2['date'].dt.to_period('M').dt.to_timestamp()

test_2['months_since_purchase'] = (reference_date.year - test_2['date'].dt.year) * 12
                                  (reference_date.month - test_2['date'].dt.month)

print(test_2[['date', 'months_since_purchase']])
```

```
               date  months_since_purchase
0        2015-06-01                     12
1        2016-02-01                      4
2        2015-07-01                     11
3        2016-03-01                      3
4        2016-02-01                      4
...             ...                    ...
1236739 2016-02-01                      4
1236740 2016-02-01                      4
1236741 2015-08-01                     10
1236742 2016-05-01                      1
1236743 2016-04-01                      2

[1236744 rows x 2 columns]
```

In [9]:
```
X_train_2 = train_2.drop(['customer_code', 'date'] + products, axis=1)
y_train_2 = train_2[products]

X_test_2 = test_2.drop(['customer_code', 'date'] + products, axis=1)
y_test_2 = test_2[products]
```

Training

In [10]:
```
# Defining the best training parameter
params = {'C': 10, 'solver': 'liblinear', 'max_iter': 300}
```

Database with second transformation

In [11]:
```
# Initialize dictionary for storing metrics
metrics = defaultdict(lambda: defaultdict(dict))

trained_models = {}

# Train and evaluate the model on the 'train_2' dataset
for product in products:
    clf = LogisticRegression(**params)

    # Train data and labels for each product
    y_train_2_product = y_train_2[product].values
    y_test_2_product = y_test_2[product].values

    # Train the model
    clf.fit(X_train_2, y_train_2_product)

    # Sacing the model to the dictionary
    trained_models[product] = clf

    # Predictions
    y_train_2_pred = clf.predict(X_train_2)
    y_test_2_pred = clf.predict(X_test_2)
    y_train_2_pred_proba = clf.predict_proba(X_train_2)[:, 1]
    y_test_2_pred_proba = clf.predict_proba(X_test_2)[:, 1]

    # Calculate metrics
    metrics['train_2']['train'][product] = {
        'ROC AUC': roc_auc_score(y_train_2_product, y_train_2_pred_proba),
        'F1 Score': f1_score(y_train_2_product, y_train_2_pred),
```

```
        'Confusion Matrix': confusion_matrix(y_train_2_product, y_train_2_pred)
    }

    metrics['train_2']['test'][product] = {
        'ROC AUC': roc_auc_score(y_test_2_product, y_test_2_pred_proba),
        'F1 Score': f1_score(y_test_2_product, y_test_2_pred),
        'Confusion Matrix': confusion_matrix(y_test_2_product, y_test_2_pred)
    }
```

In [27]:
```
metrics_dict = dict(metrics)
```

In [12]:
```
# Summarize the average metrics across all products
summary_data_2 = []
for dataset in ['train', 'test']:
    avg_roc_auc = np.mean([metrics['train_2'][dataset][p]['ROC AUC'] for p in products
    avg_f1 = np.mean([metrics['train_2'][dataset][p]['F1 Score'] for p in products])
    summary_data_2.append(['train_2', dataset, avg_roc_auc, avg_f1])

# Create summary DataFrame
summary_df_2 = pd.DataFrame(summary_data_2, columns=['Dataset', 'Type', 'Avg ROC AUC',
print("\nEvaluated Model on Dataset: train_2")
print(summary_df_2.to_string(index=False))
```

```
Evaluated Model on Dataset: train_2
Dataset  Type  Avg ROC AUC  Avg F1 Score
train_2 train     0.885926      0.111536
train_2  test     0.883623      0.212467
```

Pickle the model

In [32]:
```
with open('multi_label_metrics.pkl', 'wb') as metrics_file:
    pickle.dump(metrics_dict, metrics_file)

with open('summary_df.pkl', 'wb') as summary_file:
    pickle.dump(summary_df_2, summary_file)
```

In [ ]:
```
# Load pickle files
with open('multi_label_metrics.pkl', 'rb') as metrics_file:
    loaded_metrics = pickle.load(metrics_file)

with open('summary_df.pkl', 'rb') as summary_file:
    loaded_summary_df = pickle.load(summary_file)

# Print results
print("Loaded Metrics for Individual Products:")
print(loaded_metrics)
print("\n Summary DataFrame:")
print(loaded_summary_df.to_string(index=False))
```

Loaded Metrics for Individual Products:


```
{'train_2': defaultdict(<class 'dict'>, {'train': {'savings_acct': {'ROC AUC': 0.8709
668741130299, 'F1 Score': 0.0, 'Confusion Matrix': array([[5756696,       0],
       [    590,       0]], dtype=int64)}, 'guarantees': {'ROC AUC': 0.96934762379838
98, 'F1 Score': 0.0, 'Confusion Matrix': array([[5757167,       0],
       [    119,       0]], dtype=int64)}, 'current_acct': {'ROC AUC': 0.746536503404
0649, 'F1 Score': 0.789796652125073, 'Confusion Matrix': array([[1154329, 1042230],
       [ 556769, 3003958]], dtype=int64)}, 'derivada_acct': {'ROC AUC': 0.87916666425
62761, 'F1 Score': 0.0, 'Confusion Matrix': array([[5755008,       0],
       [   2278,       0]], dtype=int64)}, 'payroll_acct': {'ROC AUC': 0.863866933804
0359, 'F1 Score': 0.0008201085725826688, 'Confusion Matrix': array([[5430634,      8
5],
       [ 326433,     134]], dtype=int64)}, 'junior_acct': {'ROC AUC': 0.9995948324888
868, 'F1 Score': 0.8910813874404896, 'Confusion Matrix': array([[5696143,    6417],
       [   5594,   49132]], dtype=int64)}, 'mas_particular_acct': {'ROC AUC': 0.84058
20430480595, 'F1 Score': 0.0, 'Confusion Matrix': array([[5710122,       0],
       [  47164,       0]], dtype=int64)}, 'particular_acct': {'ROC AUC': 0.883528473
9148358, 'F1 Score': 0.23261307014645854, 'Confusion Matrix': array([[4845165,  18581
0],
       [ 606263,  120048]], dtype=int64)}, 'particular_plus_acct': {'ROC AUC': 0.8104
332199652559, 'F1 Score': 0.0, 'Confusion Matrix': array([[5509710,       0],
       [ 247576,       0]], dtype=int64)}, 'short_term_depo': {'ROC AUC': 0.944195829
5946022, 'F1 Score': 0.0, 'Confusion Matrix': array([[5749982,       0],
       [   7304,       0]], dtype=int64)}, 'medium_term_depo': {'ROC AUC': 0.89470841
13434061, 'F1 Score': 0.0, 'Confusion Matrix': array([[5748465,       0],
       [   8821,       0]], dtype=int64)}, 'long_term_depo': {'ROC AUC': 0.9258234549
675359, 'F1 Score': 0.35640745596918333, 'Confusion Matrix': array([[5464586,   4648
8],
       [ 182741,   63471]], dtype=int64)}, 'e_acct': {'ROC AUC': 0.8589045913055696,
'F1 Score': 0.22031974482151012, 'Confusion Matrix': array([[5203809,   62009],
       [ 422949,   68519]], dtype=int64)}, 'funds': {'ROC AUC': 0.9209741654711987,
'F1 Score': 0.003905196701038596, 'Confusion Matrix': array([[5649947,     347],
       [ 106782,     210]], dtype=int64)}, 'mortgage': {'ROC AUC': 0.924928656877991
1, 'F1 Score': 0.0, 'Confusion Matrix': array([[5722937,       0],
       [  34349,       0]], dtype=int64)}, 'pension': {'ROC AUC': 0.9201483103341663,
'F1 Score': 0.004900786705234261, 'Confusion Matrix': array([[5703142,      97],
       [  53914,     133]], dtype=int64)}, 'loans': {'ROC AUC': 0.8514761840272447,
'F1 Score': 0.0, 'Confusion Matrix': array([[5743430,       0],
       [  13856,       0]], dtype=int64)}, 'taxes': {'ROC AUC': 0.8569855707144118,
'F1 Score': 0.001273281070554751, 'Confusion Matrix': array([[5437058,     211],
       [ 319813,     204]], dtype=int64)}, 'credit_card': {'ROC AUC': 0.8881335211333
022, 'F1 Score': 0.00633629585627575, 'Confusion Matrix': array([[5495504,     563],
       [ 260387,     832]], dtype=int64)}, 'securities': {'ROC AUC': 0.91213964322026
91, 'F1 Score': 0.007295732938484318, 'Confusion Matrix': array([[5609248,     521],
       [ 146975,     542]], dtype=int64)}, 'home_acct': {'ROC AUC': 0.88738763929116,
'F1 Score': 0.0, 'Confusion Matrix': array([[5734612,       0],
       [  22674,       0]], dtype=int64)}, 'pensions_2': {'ROC AUC': 0.86001630790088
69, 'F1 Score': 0.000745415414966595, 'Confusion Matrix': array([[5400571,      91],
       [ 356491,     133]], dtype=int64)}, 'direct_debt': {'ROC AUC': 0.8664512533037
506, 'F1 Score': 0.04983388284155826, 'Confusion Matrix': array([[4986281,   19766],
       [ 731537,   19702]], dtype=int64)}}, 'test': {'savings_acct': {'ROC AUC': 0.87
92745067048582, 'F1 Score': 0.0, 'Confusion Matrix': array([[1236601,       0],
       [    143,       0]], dtype=int64)}, 'guarantees': {'ROC AUC': 0.97313281805640
15, 'F1 Score': 0.0, 'Confusion Matrix': array([[1236659,      50],
       [     35,       0]], dtype=int64)}, 'current_acct': {'ROC AUC': 0.746754823223
0492, 'F1 Score': 0.7860847306282858, 'Confusion Matrix': array([[262682, 209279],
       [134018, 630765]], dtype=int64)}, 'derivada_acct': {'ROC AUC': 0.8503475574802
631, 'F1 Score': 0.0, 'Confusion Matrix': array([[1236262,       1],
       [    481,       0]], dtype=int64)}, 'payroll_acct': {'ROC AUC': 0.863992695076
8051, 'F1 Score': 0.2260781292958016 5, 'Confusion Matrix': array([[691530, 475548],
```

```
       [   181,  69485]], dtype=int64)}, 'junior_acct': {'ROC AUC': 0.999611715522814
6, 'F1 Score': 0.8859007489816477, 'Confusion Matrix': array([[1224026,    1123],
       [   1482,   10113]], dtype=int64)}, 'mas_particular_acct': {'ROC AUC': 0.84070
08440316941, 'F1 Score': 0.00019650225977598743, 'Confusion Matrix': array([[1226567,
10],
       [  10166,       1]], dtype=int64)}, 'particular_acct': {'ROC AUC': 0.883613272
1853001, 'F1 Score': 0.404206014149736, 'Confusion Matrix': array([[1002953,   7687
8],
       [  97695,   59218]], dtype=int64)}, 'particular_plus_acct': {'ROC AUC': 0.8103
149884202605, 'F1 Score': 0.0025908194874683186, 'Confusion Matrix': array([[1183548,
95],
       [  53032,      69]], dtype=int64)}, 'short_term_depo': {'ROC AUC': 0.940463514
44678, 'F1 Score': 0.201705820739189, 'Confusion Matrix': array([[1230735,    4412],
       [    923,     674]], dtype=int64)}, 'medium_term_depo': {'ROC AUC': 0.89528226
36482807, 'F1 Score': 0.0, 'Confusion Matrix': array([[1234898,       0],
       [   1846,       0]], dtype=int64)}, 'long_term_depo': {'ROC AUC': 0.9244951185
876386, 'F1 Score': 0.22064701647343335, 'Confusion Matrix': array([[820298, 363742],
       [  1063,  51641]], dtype=int64)}, 'e_acct': {'ROC AUC': 0.8572602300572404, 'F
1 Score': 0.3646591158786281, 'Confusion Matrix': array([[827410, 304160],
       [ 13898,  91276]], dtype=int64)}, 'funds': {'ROC AUC': 0.9191234975964356, 'F1
Score': 0.28292703306803707, 'Confusion Matrix': array([[1195645,   18341],
       [  15986,    6772]], dtype=int64)}, 'mortgage': {'ROC AUC': 0.923920000058336
3, 'F1 Score': 0.003270200299768361, 'Confusion Matrix': array([[1229417,      77],
       [   7238,      12]], dtype=int64)}, 'pension': {'ROC AUC': 0.9202416840890035,
'F1 Score': 0.02748344370860927, 'Confusion Matrix': array([[1224830,     260],
       [  11488,     166]], dtype=int64)}, 'loans': {'ROC AUC': 0.8326082444863335,
'F1 Score': 0.0, 'Confusion Matrix': array([[1233790,       0],
       [   2954,       0]], dtype=int64)}, 'taxes': {'ROC AUC': 0.8532171804906513,
'F1 Score': 0.29241336663675827, 'Confusion Matrix': array([[955081, 213038],
       [ 20392,  48233]], dtype=int64)}, 'credit_card': {'ROC AUC': 0.886442828879717
9, 'F1 Score': 0.24285353724199818, 'Confusion Matrix': array([[860799, 320187],
       [  3799,  51959]], dtype=int64)}, 'securities': {'ROC AUC': 0.910975865224680
4, 'F1 Score': 0.2494479079129942, 'Confusion Matrix': array([[1102383,  102991],
       [  12224,   19146]], dtype=int64)}, 'home_acct': {'ROC AUC': 0.885865315103025
1, 'F1 Score': 0.0, 'Confusion Matrix': array([[1231825,       0],
       [   4919,       0]], dtype=int64)}, 'pensions_2': {'ROC AUC': 0.86001679859522
24, 'F1 Score': 0.24403388209314042, 'Confusion Matrix': array([[689671, 470878],
       [   166,  76029]], dtype=int64)}, 'direct_debt': {'ROC AUC': 0.865676565433392
2, 'F1 Score': 0.4522344904932277, 'Confusion Matrix': array([[688859, 387173],
       [   628, 160084]], dtype=int64)}}}})

 Summary DataFrame:
Dataset  Type  Avg ROC AUC  Avg F1 Score
train_2 train     0.885926      0.111536
train_2  test     0.883623      0.212467
```