In [11]:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import os
import zipfile
from sklearn.base import BaseEstimator, TransformerMixin
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline, FeatureUnion
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
import warnings
warnings.filterwarnings('ignore')

import os
import pandas as pd

#DATA_DIR = "DATA_DIR"   # you already set this earlier

def load_data_light(name, nrows=None, show_head=True):
    path = os.path.join(f"{name}.csv")
    df = pd.read_csv(path, nrows=nrows)
    print(f"{name}: shape is {df.shape}")
    if show_head:
        display(df.head())
    return df

datasets = {}

# Load the core tables completely
ds_names = ['application_train'
            #,'application_test','bureau_balance','bureau','credit_card_balance','ins
           ]
for name in ds_names:
    datasets[name] = load_data_light(name)

y = datasets['application_train']["TARGET"]
X = datasets['application_train'].drop("TARGET", axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
```

```
application_train: shape is (307511, 122)
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALT
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
2	100004	0	Revolving loans	M	Y	
3	100006	0	Cash loans	F	N	
4	100007	0	Cash loans	M	N	

5 rows × 122 columns

In [12]:

```
# Create a class to select numerical or categorical columns
# since Scikit-Learn doesn't handle DataFrames yet
class DataFrameSelector(BaseEstimator, TransformerMixin):
    def __init__(self, attribute_names):
        self.attribute_names = attribute_names
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        return X[self.attribute_names].values
# Identify the numeric features we wish to consider.
num_attribs = [
    'AMT_INCOME_TOTAL',  'AMT_CREDIT','DAYS_EMPLOYED','DAYS_BIRTH','EXT_SOURCE_1',
    'EXT_SOURCE_2','EXT_SOURCE_3']

num_pipeline = Pipeline([
        ('selector', DataFrameSelector(num_attribs)),
        ('imputer', SimpleImputer(strategy='mean')),
        ('std_scaler', StandardScaler()),
    ])
# Identify the categorical features we wish to consider.
cat_attribs = ['CODE_GENDER', 'FLAG_OWN_REALTY','FLAG_OWN_CAR','NAME_CONTRACT_TYPE',
               'NAME_EDUCATION_TYPE','OCCUPATION_TYPE','NAME_INCOME_TYPE']

# Notice handle_unknown="ignore" in OHE which ignore values from the validation/test
# do NOT occur in the training set
cat_pipeline = Pipeline([
        ('selector', DataFrameSelector(cat_attribs)),
        #('imputer', SimpleImputer(strategy='most_frequent')),
        ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
        #('ohe', OneHotEncoder(sparse=False, handle_unknown="ignore"))
        ('ohe', OneHotEncoder(handle_unknown="ignore"))

    ])

data_prep_pipeline = FeatureUnion(transformer_list=[
        ("num_pipeline", num_pipeline),
        ("cat_pipeline", cat_pipeline),
    ])
```

In [ ]:

```
# Logistic Regression Model Pipeline
#%%time
np.random.seed(42)
full_pipeline_with_predictor = Pipeline([
        ("preparation", data_prep_pipeline),
        ("logistic", LogisticRegression(penalty='l2', class_weight='balanced', random
        )
    ])
model_logistic = full_pipeline_with_predictor.fit(X_train, y_train)

# Random Forest Model Pipeline
#%%time
from sklearn.ensemble import RandomForestClassifier
np.random.seed(42)
full_pipeline_with_predictor = Pipeline([
        ("preparation", data_prep_pipeline),
        ("random_forest", RandomForestClassifier(class_weight='balanced', random_stat
    ])
model_rf = full_pipeline_with_predictor.fit(X_train, y_train)
 # XGBoost Model Pipeline
#%pip install xgboost
from xgboost import XGBClassifier


np.random.seed(42)
full_pipeline_with_predictor = Pipeline([
        ("preparation", data_prep_pipeline),
        ("xgb", XGBClassifier(random_state=42,
                              use_label_encoder=False,
                              eval_metric='logloss',scale_pos_weight=11.47))
    ])
model_xgb = full_pipeline_with_predictor.fit(X_train, y_train)

# LightGBM Model Pipeline
#%pip install lightgbm
from lightgbm import LGBMClassifier


#%%time
np.random.seed(42)
full_pipeline_with_predictor = Pipeline([
        ("preparation", data_prep_pipeline),
        ("lgbm", LGBMClassifier(scale_pos_weight=11.47,random_state=42)) # Scale weig
    ])
model_lgbm = full_pipeline_with_predictor.fit(X_train, y_train)

# CatBoost Model Pipeline
#%pip install catboost
import catboost

#%%time

np.random.seed(42)
full_pipeline_with_predictor = Pipeline([
        ("preparation", data_prep_pipeline),
        ("catboost", catboost.CatBoostClassifier(verbose=0,
        auto_class_weights='Balanced'))
    ])
model_cat = full_pipeline_with_predictor.fit(X_train, y_train)
```

In [ ]: