**Modelo performático**

Modelo criado com o intuito de ser um parâmetro comparativo com o modelo de Regressão Logística

* Algoritmo utilizado: XGBoost (eXtreme Gradient Boosting)
* Pipeline do desenvolvimento:
  + Obtenção das bases de dados
  + Feature Engineering: criação de novas features em cada base de dados
  + Importação de cada base de dados, em diferentes modelos
  + Entendimento dos dados: volumetria, metadados, visualização de instâncias
  + Preparação dos dados: eliminação de features com alto percentual de nulos, imputação em valores nulos, codificação de variáveis categóricas
  + Feature Selection: utilização do algoritmo XGBoosting para seleção de variáveis através da Feature Importance
  + Nova preparação de dados: filtragem da base somente com features importantes e codificação de variáveis categóricas
  + Modelagem: modelo baseline, modelo tunado, visualização de métricas, visualização de ordenação
  + Salvamento dos artefatos do modelo: listas de variáveis selecionadas, modelo, encoder
  + Escoragem de tabela de teste: aplicação das transformações através dos artefatos do modelo, submissão ao Kaggle
* Base de dados utilizada no modelo de melhor desempenho: Application\_train\_v03
* Features selecionadas na etapa de Feature Selection:

'APP\_NEW\_PAYMENT\_RATE', 'EXT\_SOURCE\_3', 'REG\_CITY\_NOT\_LIVE\_CITY', 'OCCUPATION\_TYPE', 'FLOORSMAX\_MEDI', 'NAME\_EDUCATION\_TYPE', 'EXT\_SOURCE\_2', 'EXT\_SOURCE\_1', 'LIVINGAREA\_MEDI', 'NAME\_INCOME\_TYPE'

* Treinamento do modelo: o modelo foi treinado recebendo como entrada as features selecionadas. Foi gerado um grid de hiperparâmetros e utilizada a função GridSearchCV, do módulo sklearn, para a procura automática da combinação que traga as melhores métricas.
  + Grid de parâmetros utilizado:

grid\_param = { 'objective': ['binary:logistic'], 'learning\_rate': [0.1 , 0.00001],

'max\_depth': [3], 'min\_child\_weight': [4], 'subsample': [0.5 , 1.0],

'colsample\_bytree': [0.5 , 1], 'n\_estimators': [200], 'scale\_pos\_weight': [11.5],

'reg\_lambda':[0.5, 1.5 , 3], 'reg\_alpha':[0.5, 1.5 , 3] }

* + Parâmetros selecionados no treinamento

{ 'colsample\_bytree': 0.5, 'learning\_rate': 0.1, 'max\_depth': 3,

'min\_child\_weight': 4, 'n\_estimators': 200, 'objective': 'binary:logistic',

'reg\_alpha': 3, 'reg\_lambda': 0.5, 'scale\_pos\_weight': 11.5, 'subsample': 1.0 }

* Métricas do modelo tunado (dados de treino / dados de teste)
  + Acurácia: 0.70 / 0.70
  + Precision: 0.17 / 0.17
  + Recall: 0.70 / 0.68
  + AUC-ROC: 0.76 / 0.75
  + Gini: 0.53 / 0.51
  + KS: 0.39 / 0.38