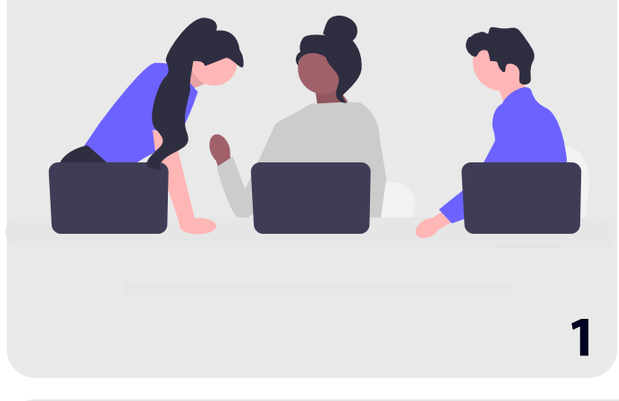
Projeto

Machine Learning

Risco de Crédito

Framework

Definir o Problema



1

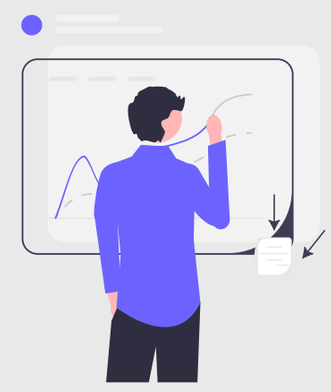
Adquirir os Dados

2

Explorar os Dados

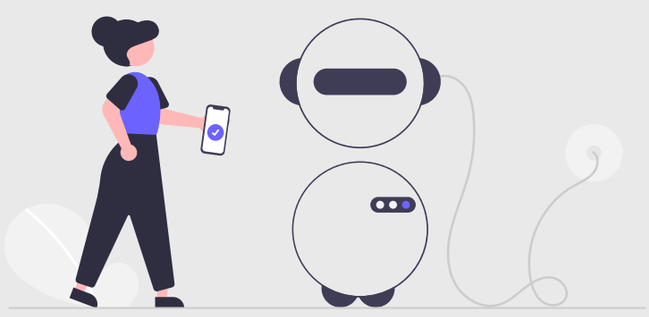
3

Preparar os Dados



4

Selecionar o Modelo

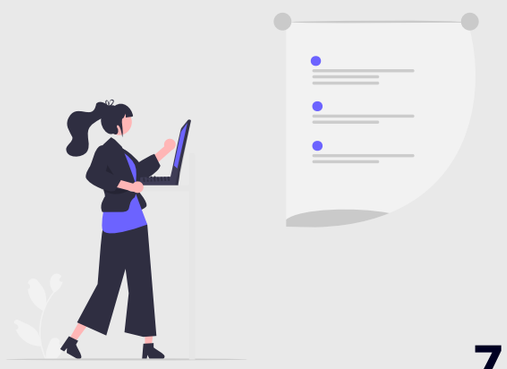


5

Ajustar o Modelo

6

Apresentar o Modelo



7

Criar um Relatório

8

1 Definir o Problema

HOME CREDIT

A Home Credit disponibilizou **246mil** registros de aplicações de empréstimos.
Cada aplicação com **122** características, entre elas:

Valor do empréstimo | **Valor das parcelas** | **Patrimônio do aplicante**

Em qual horário o empréstimo foi solicitado | **Há quanto tempo o aplicante trabalha**

+ 6 Datasets de informações de aplicações anteriores | **+ outras 118 características**

O **objetivo** é usar esses dados para construir um Modelo de Machine Learning para predizer se um aplicante terá dificuldades em pagar um empréstimo.
Cascateando:

- O primeiro foco do Modelo é distinguir entre "tipos" de aplicantes.
  No nosso caso: bons pagadores de maus pagadores.
  Em termos de métricas, buscamos um **ROC-AUC** alto.

- Como esse é um projeto de aprendizado proposto no Degree de Data Science da Ada/Let's Code, o segundo foco está em experimentar e testar diversos aspectos de **Machine Learning**.

2 Adquirir os Dados

Desafios e Soluções

Desafio
O tamanho do armazenamento dos datasets ultrapassa 1GB. Limitando, por exemplo, o armazenamento no Github.

Solução
Usar um template de pastas e desenvolver uma automação usando 'make'

Desafio
Múltiplos datasets com múltiplos tipos de relações e tipos de características

Solução
Fazer um "Pre-overview" de todos os datasets para guiar a Análise Exploratória dos Dados (EDA)

Um dos focos do projeto foi **reprodutibilidade** e **automação**.

Para reproduzir todos os passos do projeto:

git clone → **make create_environment** — Cria o ambiente virtual completo do projeto

make data — Baixa, descomprime e salva os datasets em /data/raw

make explore — Cria o datasets de exploração e decisão em data/interim

make pipeline — Cria o pipeline de processamento e salva os dados preprocessados em /data/processed

make models — Treina e salva os modelos intermediários e final em /models

3 Explorar os Dados

Exploração e Decisões

Valores Nulos

Decisão
Rejeitar colunas
> 30% de NaNs

Tipo de Dado

Categórico

Decisão
Transformar com
OrdinalEncoder
da biblioteca
feature-engine

Numérico

Ruído
Outliers (Por
IQR e Z-score)

Decisão
Rejeitar registros
com Z-score > +-3

Distribuição
Testar normalidade,
calcular Skewness
& Kurtosis e
correlação

Decisão
Agrupar colunas
relacionadas

Decisão
Combinar colunas
para criar novas
features

4 Preparar os Dados

- scikit learn
- Feature-engine
- Customizado

Pipeline

Aplicado em: ID e alto volume de NaNs
DropFeatures

Aplicado em: NaNs "escondidos" (XNA)
NanTransformer

Aplicado em: NaNs "escondidos" (Unknown)
NanTransformer

Aplicado em: Numéricos com NaNs
MeanMedianImputer
Método: Mediana

Aplicado em: Categóricas com baixo volume de NaNs
CategoricalImputer
Método: Mais Frequente

Aplicado em: Categóricas com médio volume de NaNs
RandomSampleImputer

Aplicado em: Categóricas
OrdinalEncoder

Aplicado em: Numéricas com Outliers
FunctionSampler(trim_outliers)

Criei uma função a partir da OutlierTrimmer (feature-engine) capaz de retornar X transformado e y correspondente.

Com isso, usando um imblearn wrapper, posso treinar o modelo com uma amostragem sem outliers.

Dessa forma, automatizo o teste de diferentes técnicas (manter, remover, remover baixos, remover altos, usar IQR ou Z-score) com outliers no pipeline - sempre usando X_train e y_train inalterados.

Aplicado em: Colunas de Documentos
MathFeatures
Agrupamento: Nova feature

Aplicado em: Anuidade, Crédito, Renda e Familiares
RelativeFeatures (x3)
Combinação: Novas features

Aplicado em: Numéricas
MinMaxScaler

Classificador

5 Selecionar o Modelo

Modelos Testados

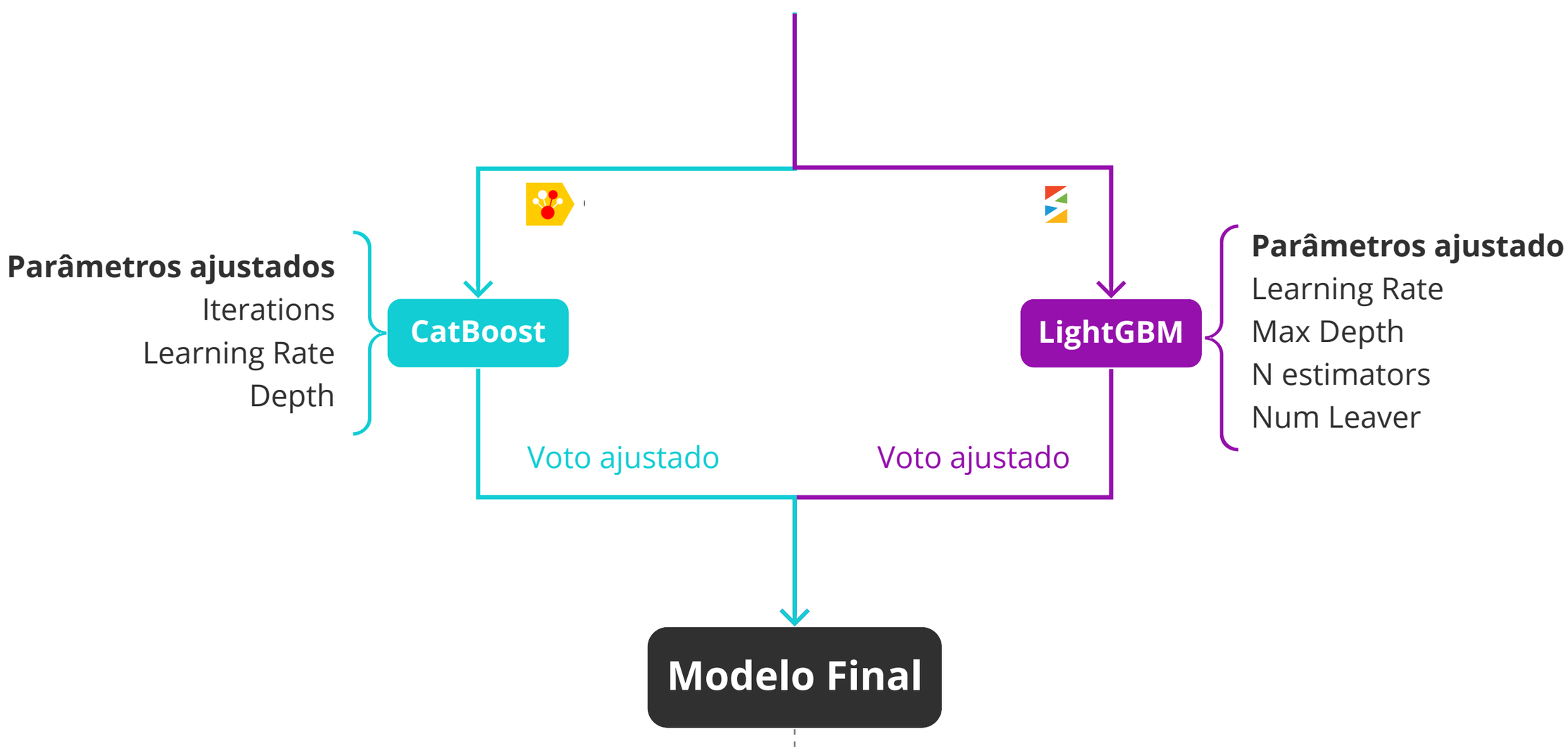
Métricas calculadas com validação cruzada

	Tempo de Treino	ROC-AUC
DummyClassifier	91s	50.0%
RegressionClassifier	105s	73.9%
KNeighborsClassifier	105s	58.1%
DecisionTreeClassifier	114s	53.7%
LinearSVC	149s	74.0%
RandomForestClassifier	185s	71.0%
AdaBoostClassifier	133s	74.4%
GradientBoostingClassifier	217s	75.1%
XGBClassifier	208s	74.7%
CatBoostClassifier	204s	75.6%
LGBMClassifier	92s	75.5%

Chosen Models for Hyperparameter tunning and combination

6 Ajustar o Modelo

Modelo Final



ROC-AUC do set de validação **75.58%**

https://github.com/ewerthonk