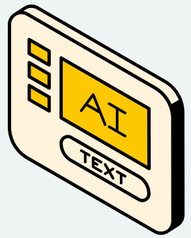
PREVISÃO DE CAMPANHA BANCÁRIA

GUILHERME CASTILHO



AGENDA

- Introdução
- Objetivo
- Variáveis do Dataset
- EDA
- Pré – Processamento dos dados
- Avaliação do Modelo

INTRODUÇÃO

O marketing do banco X está visando captar novos depósitos a prazo.

https://www.kaggle.com/datasets/janiobachmann/bank-marketing-dataset/data

OBJETIVO

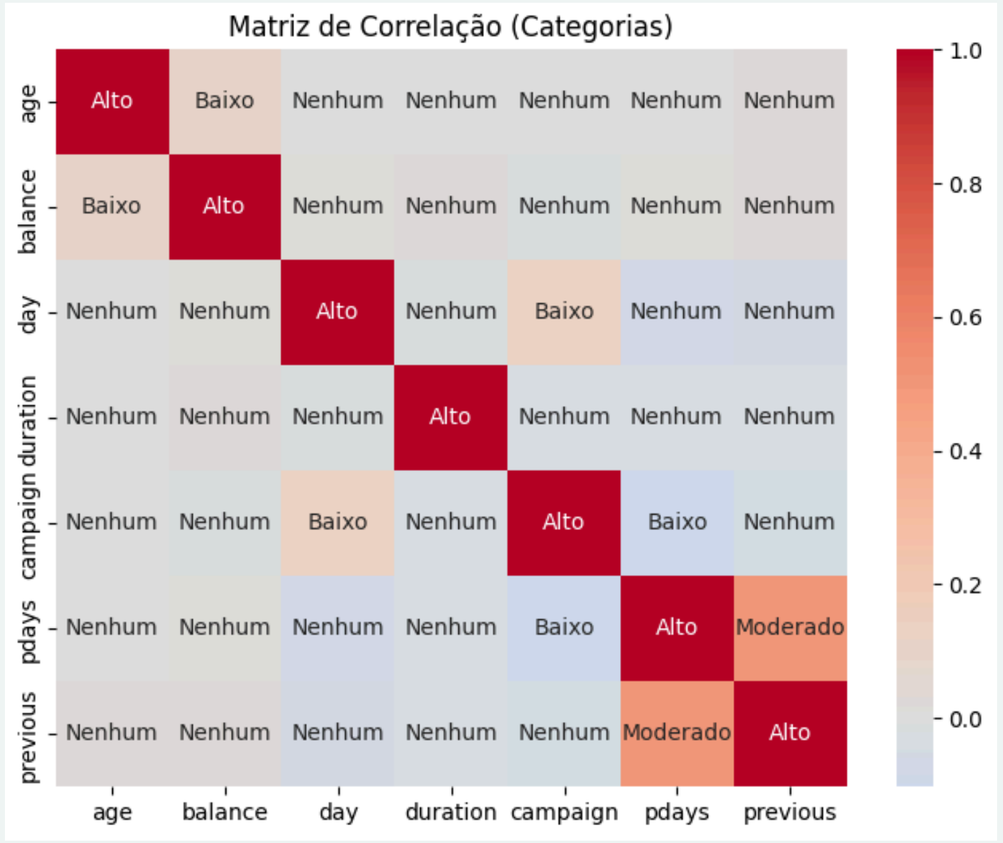
“Prever se um cliente aceitaria ou não uma campanha de depósito a prazo, a partir de características demográficas e informações de contato."

VARIÁVEIS DO DATASET

Age	Idade do cliente
Job	Ocupação do cliente
Marital	Estado civil do cliente
Education	Nível de educação do cliente
Default	Possui crédito em inadimplência?
Housing	Possui financiamento habitacional?
Loan	Possui empréstimo pessoal?
Balance	Saldo individual do cliente
Contact	Tipo de contato (meio de comunicação)
Month	Mês do último contato no ano
Day	Dia do último contato na semana
Duration	Duração do último contato, em segundos
Campaign	Número de contatos realizados durante esta campanha
Pdays	Dias desde o último contato anterior
Previous	Número de contatos realizados antes desta campanha
Poutcome	Resultado da campanha de marketing anterior
Deposit	O cliente subscreveu o depósito a prazo? (VARIÁVEL RESPOSTA)

EDA

Correlação entre as variáveis numéricas



PRÉ-PROCESSAMENTO E MODELAGEM

One-Hot Encoding

```
categ_nomes = ['job', 'marital', 'education', 'contact', 'month', 'poutcome']
df_encoded = pd.get_dummies(df, columns=categ_nomes, drop_first=True)
```

```
encoder = OneHotEncoder(handle_unknown='ignore').fit(X_train[vars_cat])
```

Divisão de Dados: train_test_split separa em 80% treino e 20% teste.

```
X = df.drop('deposit', axis=1)
y = df['deposit']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

Padronização StandardScaler para deixar as variáveis numéricas compáraveis

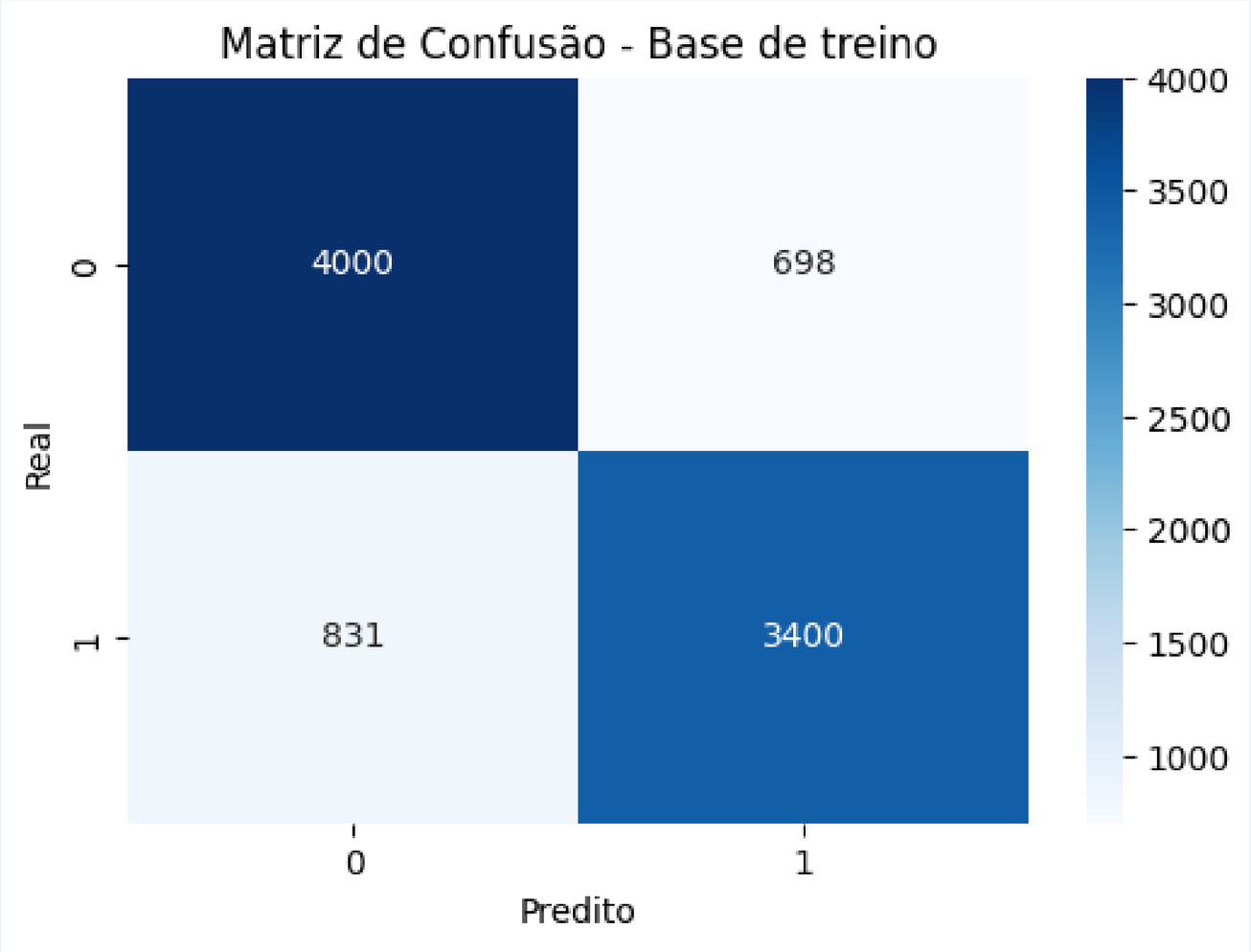
```
[53] num_cols = ['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous']

     scaler = StandardScaler()
     X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
     X_test[num_cols] = scaler.transform(X_test[num_cols])
```

Algoritmo: Regressão Logística

```
model = LogisticRegression(max_iter=1000, random_state=42)
model.fit(X_train, y_train)
```

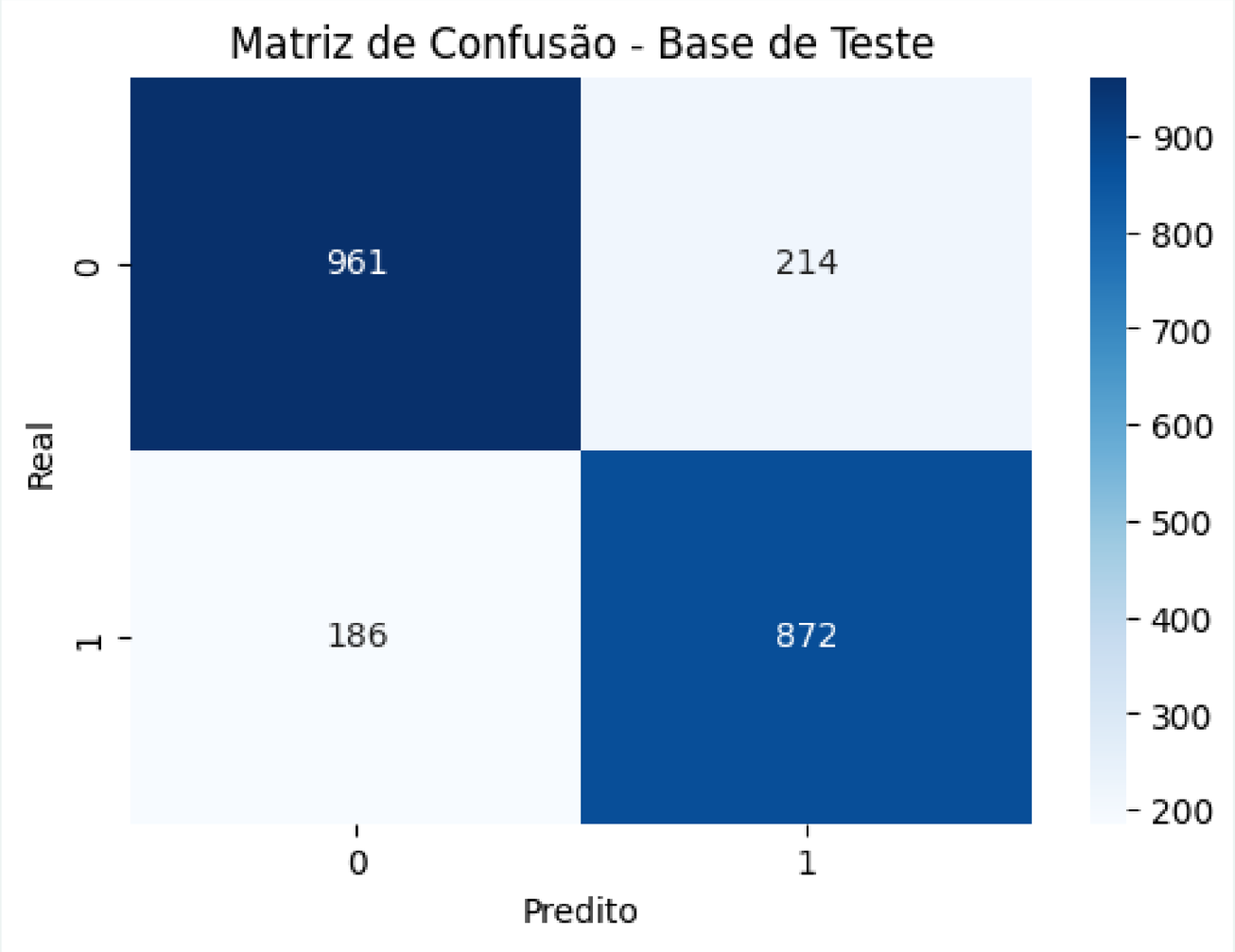
AVALIAÇÃO DO MODELO (BASE DE TREINO)

Métricas	
Recall	~80.36%
Acurácia	~82.88%
Precisão	~82.97%
F1-Score	~81.64%



AVALIAÇÃO DO MODELO (BASE DE TESTE)

Métricas	
Recall	~82.42%
Acurácia	~82.09%
Precisão	~80.29%
F1-Score	~81.34%



MELHORIAS

- Testar outros hiperparâmetros
- Melhorar balanceamento
- Testar outros modelos