# Projeto final das UC
# Modelos de Machine Learning e Análise de Dados Exploratória

INSTITUTO SUPERIOR DE CONTABILIDADE E ADMINISTRAÇÃO DO PORTO
POLITÉCNICO DO PORTO

*David Emanuel Maia de Carvalho*
*demcarvalho98@gmail.com*
*Rui Filipe Pereira Parada*
*ruiparada.2003@gmail.com*
*ISCAP, Instituto Politécnico do Porto*

**Resumo**

*O presente projeto aplica conceitos de Análise de Dados Exploratória (ADE) e Machine Learning (ML) para explorar e adaptar o dataset "bank_data". Inicialmente, realizamos uma análise estatística dos dados, identificando padrões e relações entre as variáveis. Em seguida, tratamos valores ausentes, outliers e duplicações, além de remover colunas altamente correlacionadas. Para o modelo preditivo, processamos os dados e testamos nove algoritmos de classificação, avaliando o desempenho com métricas como accuracy, recall e F1-score. Utilizamos Cross Validation e GridSearch para otimização dos modelos e analisamos a importância das variáveis nos algoritmos baseados em árvores. Concluímos qual o modelo ideal para utilizar no projeto em questão.*

# Introdução

A tomada de decisão baseada em dados tornou-se essencial em diversas áreas, incluindo o setor bancário, onde a análise de informações pode melhorar a eficiência de campanhas e a relação com os clientes. Neste projeto, exploramos um conjunto de dados do setor bancário fornecido pelos docentes para compreender padrões de comportamento e construir um modelo preditivo que avalia a propensão dos clientes a aderirem a uma campanha.

O trabalho conjuga as temáticas abordadas nas duas UC, Análise Exploratória de Dados (ADE) e Modelos de Machine Learning (MML). Pretendemos, inicialmente, analisar cuidadosamente cada variável e as relações que as mesmas estabelecem entre si, para, então, proceder ao tratamento dos dados que percebermos necessário. Para responder ao problema de classificação inicial do projeto, pretendemos testar 9 modelos de ML: *KNN, Logistic Regression, Linear SVC, RBF SVC, Decision Tree, Random Forest, Gradient Boosting, XGBoost e LightGBM*. Por fim, com base no estudo de diversas métricas dos modelos de classificação, argumentaremos qual o modelo ideal a ser escolhido.

O presente relatório visa ser um auxílio à interpretação do jupyter notebook, que permita acompanhar o raciocínio e o desenvolvimento do projeto.

# Metodologia

## Estrutura do jupyter notebook

**1. Imports bib** - importação das bibliotecas necessárias para o código

**2. Leitura do dataset** - código para o upload do ficheiro csv "bank_data"

**3. Análise de dados exploratória** - código para analisar o dataset inicial, *df.shape* para verificar o número de linhas e de colunas e *df.dtypes* para obter o tipo de dados de cada coluna. Realização de análise univariada, análise bivariada e análise multivariada.

**4. Tratamento dos dados - missing values, outliers e duplicados | df_teste1** - criação de uma cópia do dataset original para preservar a sua integridade. Após uma análise inicial dos dados, tratamento de missing values, outliers, duplicados e colunas altamente correlacionadas.

**5. Processamento de dados e defs para o modelo preditivo** - definição das funções process_data, split_data, apply_model, model_evaluation e variables_importance. Criação de df_teste2

**6. Modelos preditivos** - teste de 9 modelos de ML. Teste com 10% e com 100% dos dados.

## Análise de dados exploratória

Para que o objetivo do trabalho seja concluído com sucesso é necessária a realização de um estudo e uma análise completa dos dados e posteriormente o seu tratamento. Numa primeira fase para a resolução deste trabalho, começamos pelo reconhecimento do problema que é proposto e a interpretação da base de dados.

Primeiramente é realizada uma análise à forma do dataset e identificação da tipologia de dados presentes no mesmo. Das 21 colunas presentes no dataset, verificamos que 10 são relativas a variáveis numéricas e 11 são variáveis categóricas, sendo uma destas últimas a nossa variável a prever, 'y'.

Nesta primeira análise verificamos a presença de *missing values* nas mais variadas colunas. No que respeita à variável numérica 'pdays', a codificação com o valor de 999, representa os *missing values* desta variável. Nas variáveis categóricas 'job', 'marital', 'education', 'default', 'housing' e 'loan', os *missing values* estavam codificados com a etiqueta de "unknown", já para a 'poutcome' a codificação era "nonexistent".

## Análise Univariada

Para melhor compreender as variáveis de forma individual e as suas variações ao longo do dataset, efetuamos uma análise univariada das variáveis numéricas e categóricas.

A análise das variáveis numéricas foi efetuada com recurso à visualização das suas estatísticas descritivas (média, desvio-padrão, min, máx, quartis) e ainda de gráficos, histograma com curva de distribuição e boxplot, apresentados no notebook. Com base nestes recursos, foi possível verificar a existência de simetrias ou assimetrias de cada uma das variáveis, sobre as suas curtoses e ainda sobre a dispersão dos dados e possível consequente presença de *outliers* (para mais detalhe sobre assimetria e curtose, Tabela 1, em anexo) .

No que respeita ao conjunto de variáveis 'age', 'duration', 'campaign' e 'previous', apresentam todas elas assimetrias positivas significando por isso que a maioria dos seus valores está abaixo da média na respetiva variável. Destas, apenas a 'age' é considerada uma assimetria positiva moderada, sendo as restantes consideradas assimetrias severas. Quanto à curtose, a tendência repete-se, uma vez que, apenas no caso da 'age' esta é moderada e não severa como nas restantes, significando por isso que todas apresentam um pico alto e estreito e vários valores extremos, identificáveis também pela visualização dos vários boxplot.

Pelos gráficos da variável 'pdays', é possível verificar que os seus *missing values* provocam uma assimetria negativa severa nos dados, tendo por isso os *missing values* um grande impacto nos seus registos, tornando possível ainda a identificação de vários valores muito baixos nessa variável.

Quanto às distribuições do conjunto de variáveis 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m' e 'nr.employed' verificamos que estas são distribuições consideradas multimodais por apresentarem vários picos. No que respeita à análise do conjunto 'cons.price.idx' e 'cons.conf.idx', verificamos simetria nas suas distribuições, estando desta forma os dados igualmente distribuídos. Quanto ao conjunto 'emp.var.rate', 'euribor3m' e 'nr.employed', são identificáveis assimetrias negativas, justificadas pela maioria dos valores estar localizada acima da média, tendo, ainda assim, alguns valores muito baixos.

Para a análise das variáveis categóricas, recorremos à contagem de valores únicos e ainda gráficos Barplot, possibilitando efetuar a comparação da contagem desses registos dentro de cada variável. Com esta análise, verificamos ainda o real impacto que cada categoria representa em determinada variável, bem como a quantidade de *missing values* para essa mesma variável.

Para o conjunto de variáveis 'job', 'marital', 'education', 'default', 'housing', 'loan' e 'poutcome', foi possível visualizar o impacto dos *missing values* no conjunto de registos de cada variável.

No que respeita às variáveis 'job' e 'marital', verificamos que os dados não estão apenas distribuídos por uma só categoria, mas sim pelas várias categorias presentes em cada uma das variáveis. Desta forma, podemos notar ainda que a quantidade de dados em falta representa quantidades muito baixas, sendo esta residual quando comparada com as restantes categorias que representam um peso praticamente total e que será explorado em detalhe na secção seguinte, **secção Tratamento de Dados**.

Já para o caso da variável 'education', a dimensão dos *missing values* pelo facto de ser superior e apresentar uma maior significância, terá um tratamento diferenciado relativamente às anteriores, explorado mais à frente.

No que concerne à variável 'default', esta apresenta apenas 3 diferentes categorias, sendo elas "no", "unknown" e "yes". Verificamos, contudo, que a quantidade de "yes" é desprezável (apenas 3 registos) quando comparada com os 'no' que predominam e os 'unknown' que representam a segunda maior fatia dentro desta variável, o que permite inferir que esta tem uma variância de praticamente zero.

Aquando da análise de 'housing' e 'loan' identificamos uma mesma quantidade de valores 'unknown' para ambas. Contudo, na variável 'housing', a distribuição entre as duas categorias "yes" e "no" é muito equilibrada, e uma possível definição dos registos "unknown" poderia ter um elevado impacto na distribuição dos registos pela variável. Para 'loan', esta distribuição pelas categorias já não é equilibrada, e uma possível definição dos *missing values* não teria um impacto tão elevado quanto a da variável 'housing'.

Relativamente à 'poutcome' a quantidade identificada como valores em falta representa praticamente a totalidade dos registos, havendo apenas uma minoria de registos distribuídos entre as categorias "failure" e "success". Embora a partir desses poucos registos se possa concluir que a categoria "failure" predomina face à "success" e que exista por isso uma certa tendência, devido à elevada quantidade de *missing values*, qualquer retirada de conclusões e definição desses valores pode enviesar os dados.

Quanto às variáveis 'contact', 'month', 'day_of_week' e 'y'(*target*), estas não apresentam nenhum valor em falta, podendo ser analisadas individualmente sem ter esse fator em consideração. Primeiramente, por análise do 'contact' verifica-se que a maioria dos contatos foram efetuados para "celular", ficando o "telephone" como categoria minoritária. É possível perceber pelo 'month' que as campanhas de *marketing* tiveram o seu período mais forte em "may" mas no mês de registos seguinte "jul" já se verificou um decréscimo elevado na frequência de contactos. Ao longo dos meses posteriores, percebe-se ainda que houve um contínuo decréscimo na quantidade de contactos efetuados. Analisando a variável 'day_of_week', percebemos um determinado equilíbrio entre os dias da semana e o número de

contactos efetuados. Por fim, por análise da variável 'y', *target*, conseguimos concluir que os contatos efetuados tiveram sucesso numa parte minoritária dos contactados, representando cerca de 11%.

## Análise Bivariada

Com vista a obter uma melhor compreensão do dataset, recorremos também a uma análise bivariada das várias combinações de variáveis, numérica-numérica, categórica-categórica e ainda numérica-categórica. Começamos por estabelecer quais os métodos mais adequados para analisar cada uma das combinações acima mencionadas.

No que diz respeito à combinação numérica-numérica, com vista ao uso de um método que mais corretamente se adequasse aos tipos de relações que existem no nosso dataset, começamos por definir que o método mais indicado a usar seria o da correlação de *Spearman*, uma vez que este contempla relações sejam elas lineares ou não. Desta forma, e com recurso a uma matriz de correlação de *Spearman*, conseguimos medir a força e direção das várias correlações entre as variáveis.

Partindo para a análise da matriz de correlação, foi-nos possível identificar as combinações de variáveis com correlações mais fortes. Assim, tendo elas uma correlação de 0.94, 0.94 e 0.93, as combinações de variáveis seguintes são, respetivamente, as mais altas identificadas, 'emp.var.rate'-'euribor3m', 'emp.var.rate'-'nr.employed' e 'nr.employed'-'euribor3m'. Relativamente às restantes correlações, todas elas se encontram abaixo dos 0.66 tendo por isso correlações muito mais fracas quando comparadas com as supramencionadas.

Quanto à combinação categórica-categórica, decidimos recorrer ao uso do método da informação mútua. Este método foi o definido uma vez que pode ser adaptado à nossa tipologia de dados e possibilita-nos avaliar a informação compartilhada entre as variáveis. Assim, uma vez calculadas todas as informações mútuas para todas as relações, conseguimos identificar quais as combinações de variáveis que se mostraram com mais informação partilhada.

Das relações entre variáveis categórica-categórica, algumas destacam-se face às restantes como sendo as que compartilham mais informação entre elas. Das referidas, fazem parte as relações 'education'-'job', 'contact'-'month' e 'housing'-'loan', apresentando, respetivamente, valores de correlação de 0.4061, 0.2077 e 0.1178. As restantes relações apresentam todas valores inferiores a 0.0529, sendo por isso relações mais fracas quando comparadas com as supramencionadas. Relativamente às relações entre esta tipologia de variável e a variável 'y' (*target*), de notar que apresentam todas elas relações fracas quando comparadas com as anteriores.

No que concerne à combinação categórica-numérica, com recurso ao método da informação mútua foi possível avaliarmos a informação partilhada entre as variáveis numéricas e a variável 'y' (*target*). Destas, conseguimos identificar as variáveis seguintes como sendo as que apresentam relações mais fortes, a 'duration', 'euribor3m', 'cons.price.idx', 'cons.conf.idx', 'nr.employed' e 'emp.var.rate', apresentadas de forma decrescente de informação partilhada com a nossa variável *target*. As restantes apresentam relações com valores de informação mais baixos quando comparadas com as mencionadas acima.

## Análise Multivariada

Com o objetivo de obter uma melhor visualização sobre as relações entre os pares de variáveis numéricas e uma 3ª dimensão adicionada que é a nossa variável *target*, recorremos ao tipo de gráfico pairplot. A partir do mesmo, foi-nos possível validar as relações obtidas na análise bivariada entre numérica-numérica. Contudo, a adição do parâmetro *hue* tinha o objetivo de poder visualizar se haveria mais algum padrão associado e que apenas fosse visível com o cruzamento destas variáveis com o nosso 'y'. Relativamente a esta pretensão, a mesma não se verificou, uma vez que não foi detetada nenhuma relação relevante com a adição desta 3ª dimensão, dado que todas as visualizações se parecem bastante heterogéneas. Apesar disso, e depois de verificarmos pela análise bivariada que a 'campaign' é a numérica com menos informação partilhada com o 'y', foi possível verificarmos, pelo pairplot, que o aumento da 'campaign', ou seja, o aumento de nº de contactos que o cliente recebeu durante a campanha, parece ter uma relação negativa com o 'y', independentemente do par numérico com que esteja relacionada. Assim, um maior número de contactos estabelecidos com o cliente, parece não indicar que haja uma relação com a adesão ao depósito bancário.

# Tratamento dos dados

Com o intuito de preservar a integridade do dataset inicial (df), criamos a cópia df_teste1 para o tratamento de *missing values*, *outliers*, linhas duplicadas e colunas altamente correlacionadas.

## 1. Missing values

Verificamos que, tal como descrito no enunciado do projeto, algumas features categóricas possuíam *missing values* codificados com a etiqueta "unknown". Para o tratamento destes casos, substituímos a etiqueta por NaN, com recurso à biblioteca *numpy*, o que possibilita o recurso ao *.dropna* e ao *.fillna*. A soma de *missing values* totalizava 12718 valores, distribuídos pelas colunas 'job', 'marital', 'education', 'default', 'housing' e 'loan'.

A coluna 'job' continha 330 *missing values*, ao passo que a 'marital' possuía 80, o que representava 0,8% e 0,2% do total de linhas do dataset, respetivamente. Dada a baixa percentagem que estas linhas representavam para o df_teste1, optamos por eliminar as linhas destes registos, com recurso à função *.dropna*. O mesmo sucedeu para o tratamento das colunas 'housing' e 'loan'. Ambas apresentavam duas categorias, "yes" e "no", pelo que averiguamos a percentagem de cada uma para estudar a possibilidade de tratar dos *missing values* com recursos com a média ou a moda. No entanto, verificamos que, principalmente na *feature* 'housing', não existia uma discrepância relevante que nos possibilitasse, por exemplo, aplicar novamente a moda para preencher os NaN. Por o número de valores em falta representar apenas 2,5% do total de registos de cada coluna, decidimos que seria mais seguro apagar essas linhas do que tentar supor o seu valor. Pelo número de *missing values* da coluna 'education' ser mais elevado (1731) decidimos substituir os NaN pela moda das variáveis da coluna, com recurso ao *.fillna*.

Para tratar da coluna 'default', descartamos, inicialmente, a possibilidade de utilizar a função *.dropna*, já que isso significaria abdicar de mais de 8000 registos, que representavam cerca de 20% dos registos do dataset. Contudo, quando equacionamos usar novamente a moda, constatamos que das linhas com valores "yes" ou "no", apenas 3 das mais de 30000 tinham o valor "yes" (todas as restantes contavam com "no"). Assim, tratando-se estes 3 registos como *outliers*, decidimos apagar não as linhas, mas a coluna 'default', já que com variância zero nada acrescentaria ao modelo preditivo.

Para além da coluna 'default', também a 'poutcome' e a 'pdays' foram apagadas. Nestes dois casos essa decisão foi tomada já que ambas contavam com uma percentagem muito baixa de valores realmente significativos para o modelo preditivo, 'poutcome' tinha cerca de 35000 linhas com o valor "nonexistent", também tratado como um *outlier*, e 'pdays' apresentava "999" na grande maioria das suas linhas, interpretado como "não contacto".

## 2. Outliers

Para além da análise dos *outliers* com base na visualização dos *boxplots* executados na análise univariada, estudamos a distância interquartil para as colunas 'age', 'duration', 'campaign' e 'cons.conf.idx'. Testamos a definição de k=1.5 e k=3 e avançamos com a análise com k=3, para uma análise mais flexível. Em seguida verificamos a quantidade de valores extremos, neste caso para lá do limite superior, já que os *outliers* estavam todos à direita de Q3.

- Quantidade de registos onde 'age' > 92: 4
- Quantidade de registos onde 'duration' > 974: 999
- Quantidade de registos onde 'campaign' > 9: 1051
- Quantidade de registos onde 'cons.conf.idx' > -17: 0 (com k=3, nº de *outliers* passa a 0)

Tomamos a decisão de não eliminar os *outliers* das colunas 'age' devido ao insignificante número de valores extremos. A mesma decisão foi aplicada à coluna 'duration', porém por um motivo diferente. Fizemos uma comparação da relação da coluna 'duration' com o target nos valores inferiores a 974 e superiores a 974 e percebemos uma inversão do resultado de 'y'. A nossa leitura sugere que nos valores extremos residem mais "yes" do que "no", ao contrário do que acontece com valores inferiores a 974 na coluna 'duration', **figura 10** (em anexo). Interpretamos esta informação como importante a ser mantida, de forma a não enviesar tanto a análise como o modelo. Executamos a mesma comparação com 'y' para a coluna 'campaign': a relação com valores inferiores e superiores a 9 mantinha-se, sempre com mais "no" do que "yes", **figura 11** (em anexo). A acrescentar, os 1051 registos representam 2,6% do dataset, logo procedemos à eliminação destas linhas.

## 3. Linhas duplicadas

A existência de linhas duplicadas em nada acrescentaria ao modelo preditivo. Como tal, aplicamos *.drop_duplicate()* para apagar os 14 registos duplicados.

## 4. Colunas altamente correlacionadas

Descrevemos acima a elevada correlação entre as colunas 'euribor3m', 'nr.employed' e 'emp.var.rate', verificada com base na matriz de correlação – de *Spearman*. A existência de variáveis altamente correlacionadas pode implicar que o modelo dê importância a mais a estas, já que as mesmas acabam por ser redundantes. Assim, para prevenir este impacto negativo sobre a eficácia do modelo, decidimos apagar as colunas 'nr.employed' e 'emp.var.rate'.

Como resultado do tratamento dos dados com os processos acima desenvolvidos, o df_teste1 reduziu de 41188 para 38738 linhas e de 21 para 16 colunas.

# Processamento dos dados e defs para o modelo preditivo

## 1. Funções para o modelo preditivo

Para garantir a organização e reutilização do código, definimos funções que estruturam as principais etapas do processamento e avaliação do modelo. Dessa forma, conseguimos manter um fluxo claro e eficiente.

### 1.1. Processamento dos dados (process_data)

Esta função tem como objetivo processar os dados antes do treino do nosso modelo preditivo. Inclui a normalização das variáveis numéricas, o tratamento de variáveis categóricas através de *one-hot encoding* e o tratamento do target ('y') com recurso ao *label encoding*. Com estas transformações, garantimos que os dados estão devidamente preparados para o modelo e facilitamos a sua interpretação. De ressaltar ainda que decidimos aplicar discretização na *feature* 'age', agrupando as idades em 5 *bins* definidos por nós. Com esta alteração, 'age' foi substituída por 'age_disc' e passou a incorporar as variáveis categóricas.

### 1.2. Divisão dos dados (split_data)

Após o processamento dividimos os dados em treino e teste. Decidimos definir test_size=test_portion para que a percentagem possa ser ajustada abaixo, quando chamamos a função split_data. Adiantamos que neste projeto aplicamos a divisão 80% para treino e 20% para teste.

### 1.3. Aplicação do modelo (apply_model)

Nesta função, definimos os hiperparâmetros e reguladores para os nove modelos que decidimos testar: *KNN, Logistic Regression, Linear SVC, RBF SVC, Decision Tree, Random Forest, Gradient Boosting, XGBoost e LightGBM*. Para cada modelo, estabelecemos intervalos de valores para os principais parâmetros, permitindo a otimização através de uma busca sistemática. Utilizamos o *GridSearchCV* para encontrar a melhor combinação de parâmetros dentro desses intervalos, garantindo um ajuste eficiente. Além disso, aplicamos *cross-validation* com cv=10, assegurando que cada modelo seja avaliado com 10 divisões diferentes dos dados, aumentando a robustez dos resultados. Esta função também armazena e apresenta o melhor conjunto de parâmetros (*best parameters*) encontrados para cada um dos modelos, o que facilita a nossa análise.

### 1.4. Avaliação do modelo (model_evaluation)

Em seguida, para medir a performance dos modelos que queríamos testar, criámos uma função dedicada à avaliação dos resultados. Calculamos métricas relevantes para problemas de classificação com recurso ao *classification_report*, que fornece dados como a *accuracy*, *precision*, *recall* e *F1-score*. A acrescentar, executamos o display de duas matrizes de confusão (uma para o treino e outra para o teste) para cada modelo testado. A função model_evaluation, diretamente ligada com a função *apply_model*, permite-nos realizar uma análise clara do desempenho de cada modelo preditivo e facilita a comparação entre diferentes abordagens.

### 1.5. Importância das Variáveis (variables_importance)

Por fim, desenvolvemos a função variables_importance para calcular a importância das variáveis nos modelos baseados em árvores de decisão: *Decision Trees*, *Random Forest*, *Gradient Boosting*, *XGBoost* e *LightGBM*. Esta análise fornece insights sobre quais variáveis mais influenciam as previsões do modelo, auxiliando na interpretação dos resultados e numa possível otimização do conjunto de dados.

Estas funções foram fundamentais para estruturar o projeto de forma clara e eficiente, garantindo que cada etapa do pipeline de Machine Learning fosse bem definida e facilmente replicável.

## 2. Preparação dos dados

Iniciamos a preparação dos dados pela definição das variáveis target e *independent_vars*, assim como a separação das variáveis numéricas e categóricas. Em seguida, numa outra célula, decidimos criar df_teste2, uma cópia de df_teste1 que seleciona apenas 10% dos dados (frac=0.1), de modo a otimizar o tempo de execução dos modelos durante a nossa análise e ajuste de hiperparâmetros. Acrescentamos ainda random_state=42 para selecionar sempre os mesmos 10% de dados, o que contribui para uma análise precisa e consistente. De referir que para testar os modelos com 100% dos dados basta alterar a fração da cópia de df_teste2 de frac=0.1 para frac=1.0. Por fim, definimos o X como as *independent_vars* (todas as colunas menos o nosso target) e o y como o *target* e "chamamos" as funções process_data e split_data, aplicando-as ao nosso dataset. Adicionamos, ainda, uma variável que junta o nome original das colunas para ser utilizada no gráfico de importância de variáveis. Tal deve-se ao facto de o nome (e número) das colunas ser alterado com a utilização do *one-hot encoding*.

## Modelos preditivos | 100% dos dados do dataset

| | CV Score | Treino / Teste | (target) | Precision | Recall | F1-score | Accuracy |
|---|---|---|---|---|---|---|---|
| **KNN** | 0.9 | **Treino** | **0** | 0.92 | 0.98 | 0.95 | 0.91 |
| | | | **1** | 0.70 | 0.34 | 0.45 | |
| | | **Teste** | **0** | 0.92 | 0.98 | 0.95 | 0.90 |
| | | | **1** | 0.64 | 0.31 | 0.42 | |
| **Logistic Regression** | 0.904679 | **Treino** | **0** | 0.92 | 0.97 | 0.95 | 0.91 |
| | | | **1** | 0.65 | 0.38 | 0.48 | |
| | | **Teste** | **0** | 0.92 | 0.97 | 0.95 | 0.91 |
| | | | **1** | 0.66 | 0.38 | 0.48 | |
| **Linear SVC** | 0.902452 | **Treino** | **0** | 0.92 | 0.98 | 0.95 | 0.90 |
| | | | **1** | 0.65 | 0.34 | 0.44 | |
| | | **Teste** | **0** | 0.92 | 0.97 | 0.95 | 0.90 |
| | | | **1** | 0.63 | 0.33 | 0.44 | |
| **RBF SVC** | 0.904421 | **Treino** | **0** | 0.93 | 0.98 | 0.95 | 0.91 |
| | | | **1** | 0.73 | 0.41 | 0.52 | |
| | | **Teste** | **0** | 0.92 | 0.98 | 0.95 | 0.91 |
| | | | **1** | 0.66 | 0.37 | 0.47 | |

## Modelos preditivos | 100% dos dados do dataset

| | CV Score | Treino / Teste | (target) | Precision | Recall | F1-score | Accuracy |
|---|---|---|---|---|---|---|---|
| **Decision Tree** | 0.906680 | **Treino** | **0** | 0.95 | 0.95 | 0.95 | 0.91 |
| | | | **1** | 0.60 | 0.59 | 0.59 | |
| | | **Teste** | **0** | 0.95 | 0.95 | 0.95 | 0.91 |
| | | | **1** | 0.59 | 0.59 | 0.59 | |
| **Random Forest** | 0.886609 | **Treino** | **0** | 0.89 | 1.00 | 0.94 | 0.89 |
| | | | **1** | 0.87 | 0.02 | 0.04 | |
| | | **Teste** | **0** | 0.89 | 1.00 | 0.94 | 0.89 |
| | | | **1** | 0.88 | 0.02 | 0.03 | |
| **Gradient Boosting** | 0.912003 | **Treino** | **0** | 0.95 | 0.96 | 0.95 | 0.92 |
| | | | **1** | 0.68 | 0.58 | 0.62 | |
| | | **Teste** | **0** | 0.94 | 0.96 | 0.95 | 0.91 |
| | | | **1** | 0.65 | 0.56 | 0.60 | |
| **XGBoost** | 0.912907 | **Treino** | **0** | 0.95 | 0.96 | 0.95 | 0.92 |
| | | | **1** | 0.67 | 0.58 | 0.62 | |
| | | **Teste** | **0** | 0.94 | 0.96 | 0.95 | 0.92 |
| | | | **1** | 0.65 | 0.56 | 0.60 | |

## Modelos preditivos | 100% dos dados do dataset

| | CV Score | Treino / Teste | (target) | Precision | Recall | F1-score | Accuracy |
|---|---|---|---|---|---|---|---|
| **LightGBM** | 0.912294 | **Treino** | **0** | 0.95 | 0.96 | 0.96 | 0.92 |
| | | | **1** | 0.69 | 0.60 | 0.64 | |
| | | **Teste** | **0** | 0.95 | 0.96 | 0.95 | 0.91 |
| | | | **1** | 0.64 | 0.58 | 0.61 | |

Após o teste dos 9 modelos com 100% dos dados de *df_teste2*, podemos comparar as métricas de cada um para interpretar qual o modelo a ser escolhido para responder ao problema de classificação inicial.

Numa avaliação inicial das métricas *precision*, *recall*, *F1-score*, *accuracy* e *best mean cross-validation score*, podemos perceber que sobressaem os modelos *Logistic Regression*, *Decision Tree*, *Gradient Boosting*, *XGBoost* e *LightGBM*.

*Logistic Regression*, *Decision Tree* e *XGBoost* têm um ponto bastante importante em comum. A *accuracy* não varia entre treino e teste, o que é um ótimo ponto indicativo de que o modelo não sofre de *overfitting*. Por outro lado, *Gradient Boosting*, *XGBoost* e *LightGBM* são os modelos que apresentam *accuracy* mais alta no treino do modelo, com 0.92 (no caso do *XGBoost*, apresenta 0.92 quer no treino quer no teste). O modelo comum a ambos os pontos citados é o *XGBoost*.

De entre os modelos referidos, é também no *XGBoost* que encontramos menor diferença entre treino e teste para nas métricas *precision*, *recall* e *F1-score*. Para além disso, este modelo apresenta o valor mais elevado para *best mean cross-validation score*, o que indicia alta adaptabilidade do mesmo. Por fim, o modelo *XGBoost* é, ainda, um dos modelos com tempo de execução mais rápido, cerca de 5 minutos com 100% dos dados de *df_teste2*.

Embora os modelos referidos não apresentem variações extremamente significativas em termos de performance, se a escolha se deve resumir a um modelo preditivo, então decidimos optar pelo *XGBoost*.

# Desafios e oportunidades

Com o desenvolvimento deste projeto deparamo-nos com vários desafios e dificuldades. Inicialmente para a parte mais dedicada a ADE, durante o tratamento dos dados foram várias as situações perante as quais a qualidade dos dados estava colocada em causa. Assim, tivemos de analisar as variáveis e as suas relações e ainda entender e testar variados modos para conseguirmos identificar os que mais se pudessem adequar para realizarmos o melhor tratamento destes dados. Durante esse tratamento, para além de termos de tratar valores em falta e *outliers* nas diferentes colunas, tivemos ainda de realizar uma seleção de variáveis. Este foi o processo que mais desafios nos trouxe uma vez que foram realizadas várias tentativas distintas para as diferentes formas de tratar os dados. Para este processo, lidamos de modo distinto para as diferentes variáveis, recorrendo à limpeza de algumas linhas em determinados casos, noutros casos remoção de colunas, noutra situação realizamos a imputação de dados com base na moda e noutros casos decidimos manter os dados que poderiam eventualmente ser considerados outliers. Foi realizada ainda a remoção dos valores duplicados que após estes tratamentos anteriores passaram a existir.

No que respeita aos modelos, a quantidade de dados do dataset foi também um desafio pelo facto de levar muito tempo à execução dos mesmos. Dessa forma, executamos todos os modelos inicialmente para apenas 10% dos dados com vista a reduzir esse tempo de execução e validar se os mesmos estavam funcionais. Outro desafio foi o ajuste e afinação dos hiperparâmetros com vista na otimização do desempenho dos diferentes modelos. Para além desta afinação, o momento de interpretação dos resultados foi o que mais nos desafiou por termos vários modelos com valores comparativos muito aproximados (3 modelos). Desta forma, foi necessária uma interpretação exaustiva das métricas de avaliação dos modelos, tanto para treino como para teste, de modo a garantirmos que a escolha do modelo seria a mais acertada, entre um modelo com a melhor performance, que consiga controlar o overfitting e que tenha um tempo de execução adequado para o pretendido.

Como oportunidades são vários os cenários que podem ser retirados deste projeto. Após a execução do mesmo, foi-nos possível cimentar os conhecimentos adquiridos e entender a aplicabilidade dos mesmos num caso real de como poderíamos usar a análise exploratória de dados e os modelos de *machine learning* para ajudar a otimizar e automatizar tarefas com um grande volume de dados. Conseguimos entender e perceber a dimensão e o impacto que tem este projeto, pelos *insights* que conseguimos retirar de todas as análises e conclusões e como isto pode ser valioso para as empresas. A partir desta fase é-nos possível compreender que é

algo escalável e aplicável aos mais variados domínios e que os benefícios da aplicação destes algoritmos são vastos.

Acreditamos que seja valioso a oportunidade de melhoria deste nosso algoritmo, a adaptação e otimização do mesmo para outras variáveis, dentro deste domínio das finanças e até mesmo aplicabilidade a outros domínios.

# Conclusões

Este projeto teve como objetivo explorar diferentes modelos de ML para resolver um problema de classificação, analisando a influência das variáveis e comparando o desempenho de cada abordagem. Desde a preparação dos dados até a avaliação final dos modelos, foram aplicadas técnicas essenciais para garantir resultados robustos e interpretáveis.

A análise exploratória permitiu identificar padrões relevantes e realizar o tratamento adequado das variáveis, assegurando uma base de dados consistente. O tratamento de dados incluiu o tratamento de *missing values*, de *outliers*, de linhas duplicadas e de variáveis altamente correlacionadas. A acrescentar, a normalização de variáveis numéricas, a aplicação de *one-hot encoding* para variáveis categóricas e de *label encoding* para a variável target ('y'). Esses passos foram fundamentais para garantir a coerência dos dados antes do treino dos modelos.

Após o teste dos 9 modelos com 100% dos dados de df_teste2, pudemos comparar as métricas de cada um para interpretar qual o modelo a ser escolhido para responder ao problema de classificação inicial. Com base na avaliação inicial das métricas *precision*, *recall*, *F1-score*, *accuracy* e *best mean cross-validation score*, podemos perceber que sobressaiam os modelos *Logistic Regression*, *Decision Tree*, *Gradient Boosting*, *XGBoost* e *LightGBM*. Num estudo pormenorizado e ponderado, decidimos sublinhar o modelo *XGBoost* como modelo indicado para o desafio inicial e central do trabalho.

Em suma, este projeto permitiu consolidar conhecimentos sobre a Análise Exploratória de Dados, a sua importância para identificar relações entre variáveis e para obter um dataset consistente, e o fluxo de trabalho de Machine Learning, com destaque para a importância da escolha adequada de modelos e parâmetros. Como trabalhos futuros, seria interessante explorar outras técnicas de otimização de hiperparâmetros, testar novos algoritmos e expandir a análise para um conjunto de dados maior ou mais complexo.

# Anexos

## *Assimetria* e *Curtose* para as variáveis numéricas

| Variáveis | Assimetria | Curtose |
|---|---|---|
| 'age' | 0.78 | 0.79 |
| 'duration' | 3.26 | 20.25 |
| 'campaign' | 4.76 | 36.98 |
| 'pdays' | -4.92 | 22.23 |
| 'previous' | 3.83 | 20.11 |
| 'emp.var.rate' | -0.72 | -1.06 |
| 'cons.price.idx' | -0.23 | -0.83 |
| 'cons.conf.idx' | 0.30 | -0.36 |
| 'euribor3m' | -0.71 | -1.41 |
| 'nr.employed' | -1.04 | -0.004 |

**Tabela 1.** Valores de assimetria e curtose para as respetivas variáveis numéricas

# Análise pré-tratamento de *missing values*

**Figura 1.** Verificar distribuição das categorias de "job"

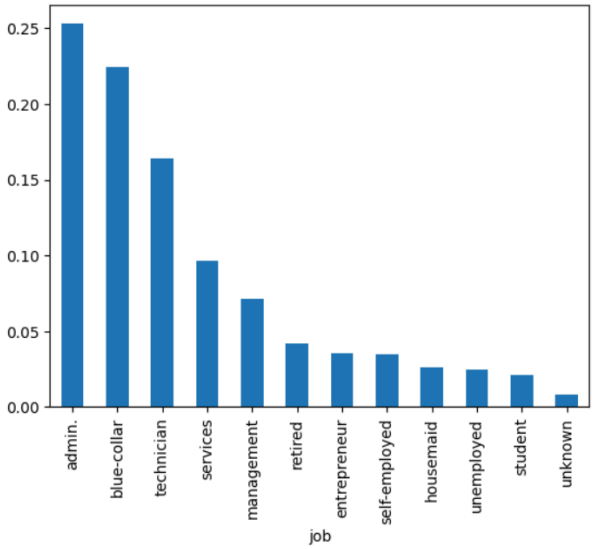



**Figura 2.** Verificar distribuição das categorias de "martital"

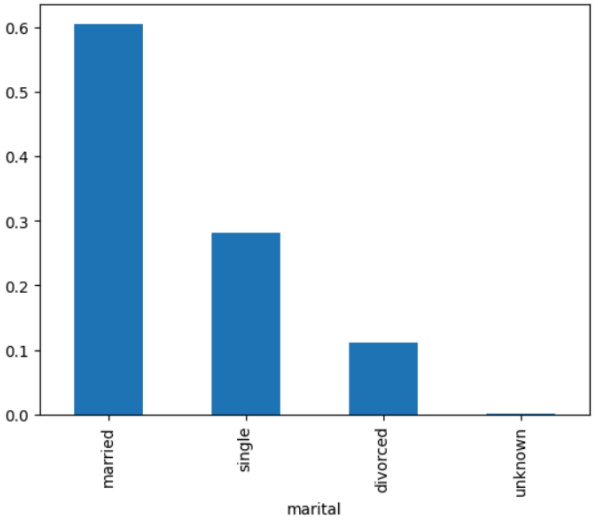



**Figura 3.** Verificar distribuição das categorias de "housing"

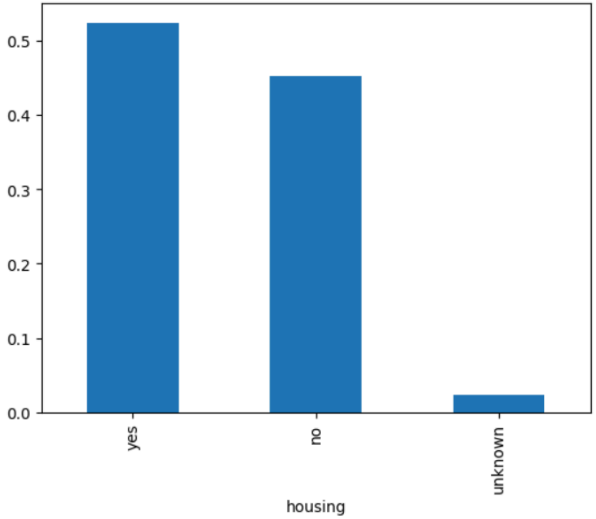



**Figura 4.** Verificar distribuição das categorias de "loan"

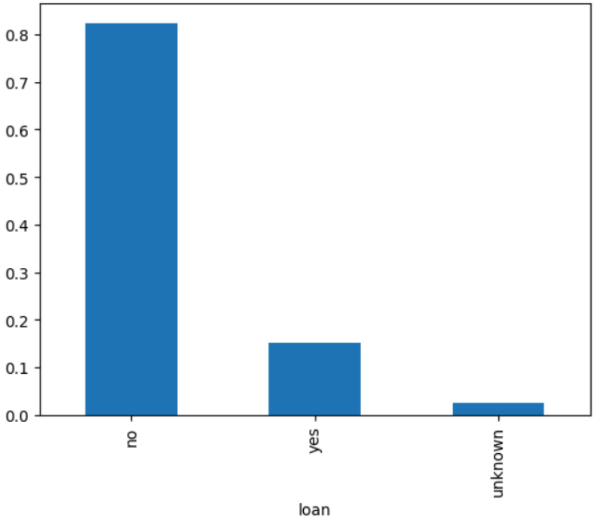


**Figura 5.** Verificar distribuição das categorias de "default"

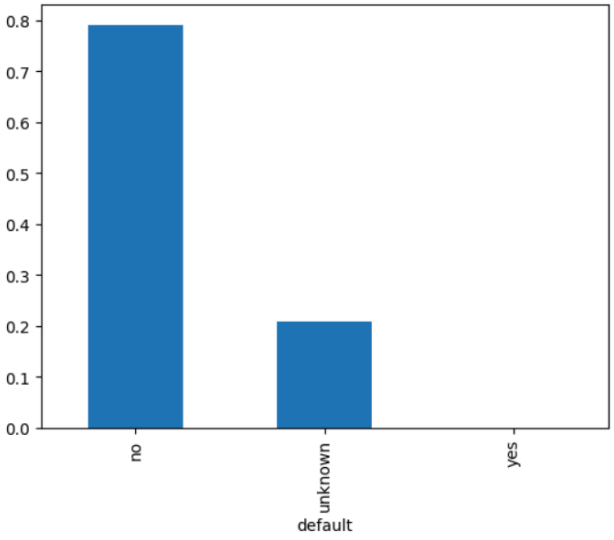



**Figura 6.** Verificar distribuição das categorias de "poutcome"

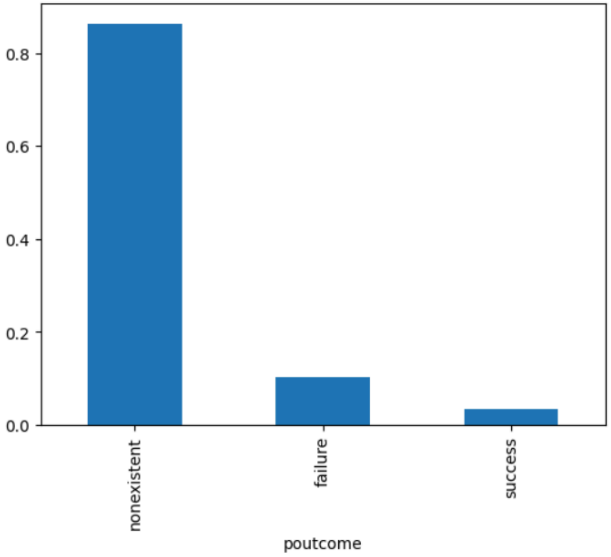
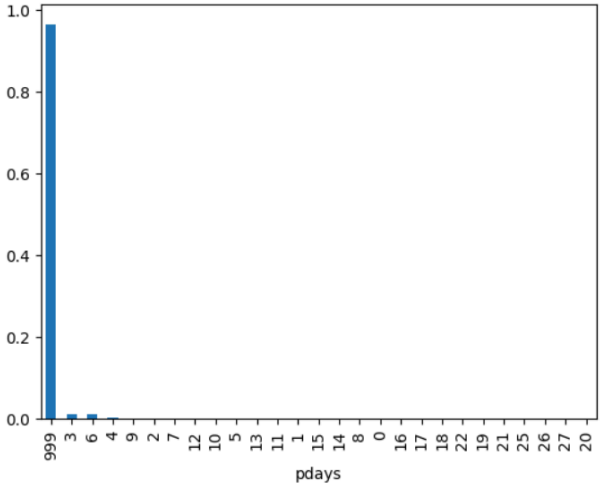



**Figura 7.** Verificar distribuição das categorias de "pdays"



## Análise pré-tratamento de *outliers*

```python
# Distância interquartil

# Lista com as colunas que queremos avaliar
colunas_dist_interquartil = ['age', 'duration', 'campaign', 'cons.conf.idx']



# Ajustar o valor de k para detectar outliers
k = 3



# Loop pelas colunas para calcular IQR e limites para outliers
for coluna in colunas_dist_interquartil:
    # Calcular Q1, Q3 e IQR
    Q1 = df_teste1[coluna].quantile(0.25)
    Q3 = df_teste1[coluna].quantile(0.75)
    IQR = Q3 - Q1

    # Definir os limites para outliers
    limite_inferior = Q1 - k * IQR
    limite_superior = Q3 + k * IQR

    # Exibir os limites para cada coluna
    print(f"Coluna: {coluna}")
    print(f"Limite inferior: {limite_inferior}")
    print(f"Limite superior: {limite_superior}")
    print("-" * 40)
```

```
Coluna: age
Limite inferior: -13.0
Limite superior: 92.0
----------------------------------------
Coluna: duration
Limite inferior: -546.0
Limite superior: 973.0
----------------------------------------
Coluna: campaign
Limite inferior: -5.0
Limite superior: 9.0
----------------------------------------
Coluna: cons.conf.idx
Limite inferior: -61.600000000000016
Limite superior: -17.499999999999986
----------------------------------------
```

**Figura 8.** Código e resultado da distância interquartil das colunas 'age', 'duration', 'campaign' e 'cons.conf.idx'

```python
# Condições dos outliers
outlier_age = df_teste1['age'] > 92
outlier_duration = df_teste1['duration'] > 974
outlier_campaign = df_teste1['campaign'] > 9
outlier_conf = df_teste1['cons.conf.idx'] > -17



# Contar registros individuais
count_age_outliers = outlier_age.sum()
count_duration_outliers = outlier_duration.sum()
count_campaign_outliers = outlier_campaign.sum()
count_confiança_outliers = outlier_conf.sum()



# Contar registros com pelo menos um outlier
outliers = outlier_age | outlier_duration | outlier_campaign | outlier_conf
count_all_outliers = outliers.sum()



# Resultados
print(f"Qt de registos 'age' > 92: {count_age_outliers}")
print(f"Qt de registos 'duration' > 974: {count_duration_outliers}")
print(f"Qt de registos 'campaign' > 9: {count_campaign_outliers}")
print(f"Qt de registros 'cons.conf.idx' > -17: {count_confiança_outliers}")
print(f"Qt de registos com pelo menos um outlier: {count_all_outliers}")
```

```
Qt de registos 'age' > 92: 4
Qt de registos 'duration' > 974: 1031
Qt de registos 'campaign' > 9: 1094
Qt de registros 'cons.conf.idx' > -17: 0
Qt de registos com pelo menos um outlier: 2105
```

**Figura 9.** Código e resultado da contagem de *outliers* das colunas 'age', 'duration', 'campaign' e 'cons.conf.idx'

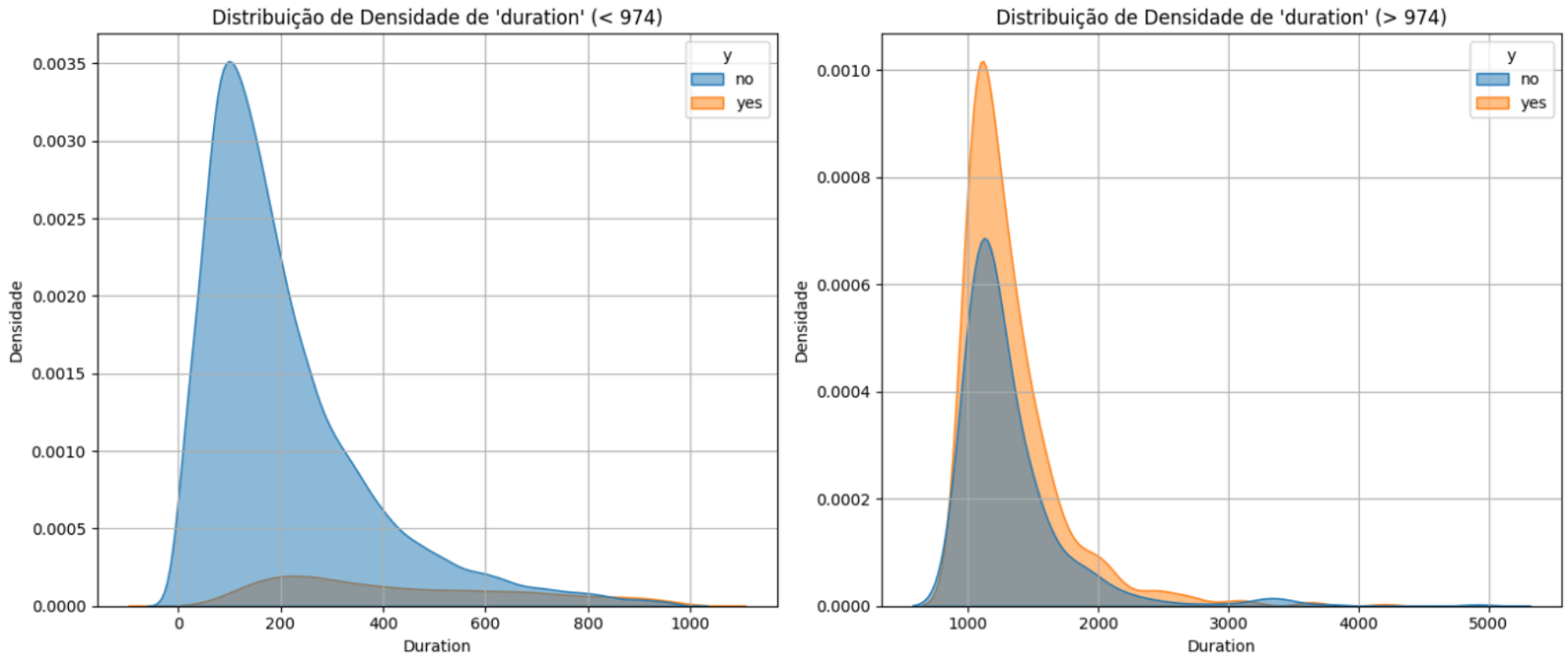



**Figura 10.** Gráficos de comparação da distribuição do target quando 'duration' está abaixo e acima do Q3

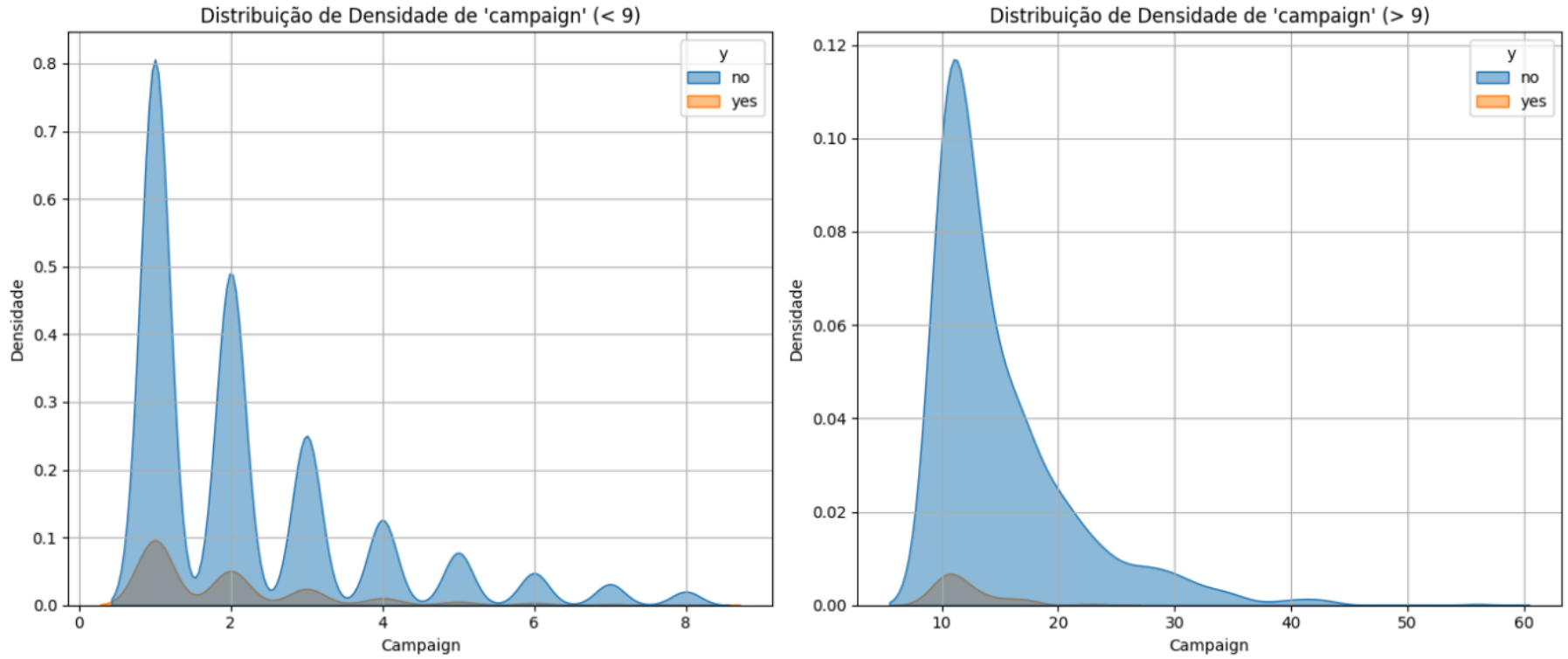



**Figura 11.** Gráficos de comparação da distribuição do target quando 'campaign' está abaixo e acima do Q3

# Matrizes de confusão dos modelos com 100% dos dados de df_teste2

**Figura 12.** KNN - treino

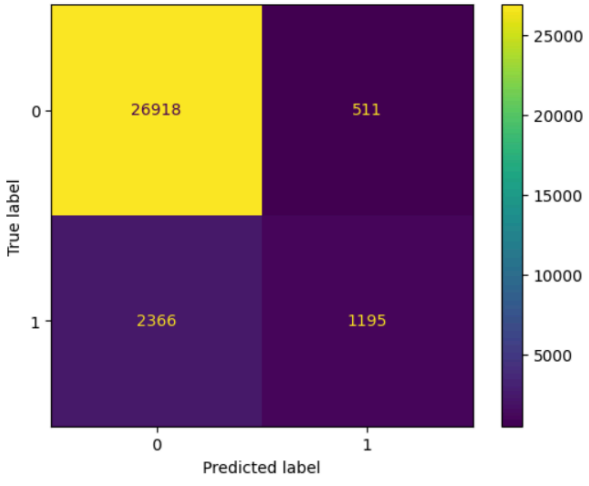



**Figura 14.** Logistic Regression - treino

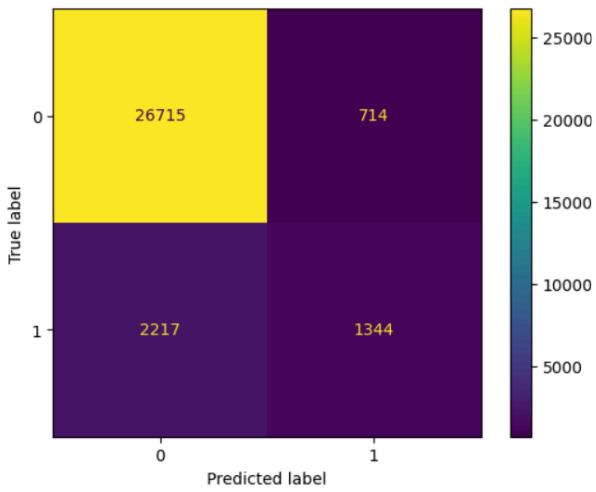



**Figura 16.** Linear SVC - treino

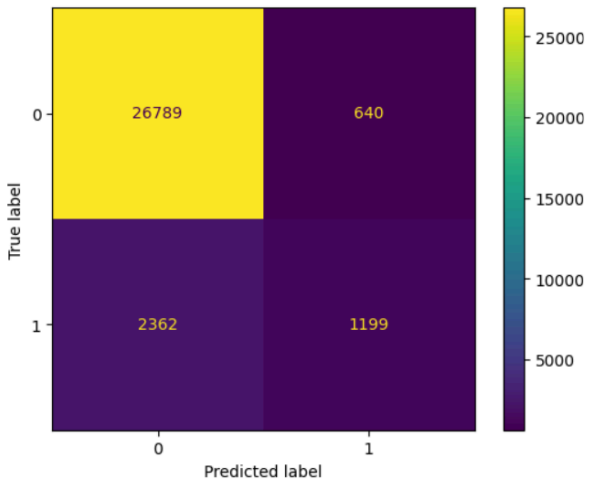



**Figura 13.** KNN - teste

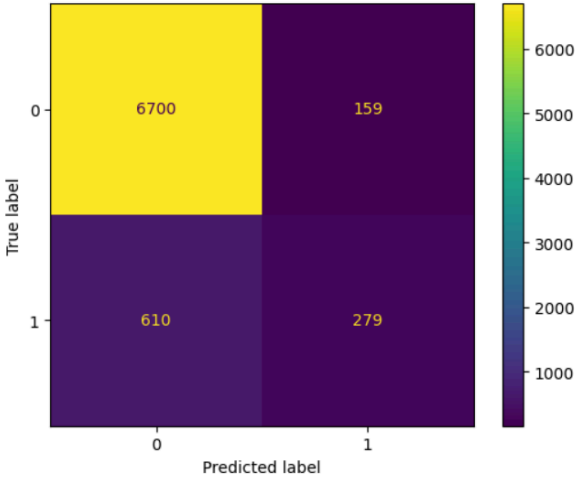



**Figura 15.** Logistic Regression - teste

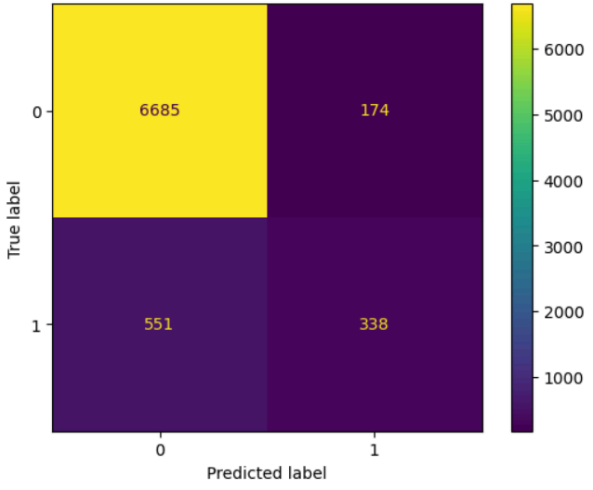



**Figura 17.** Linear SVC - teste

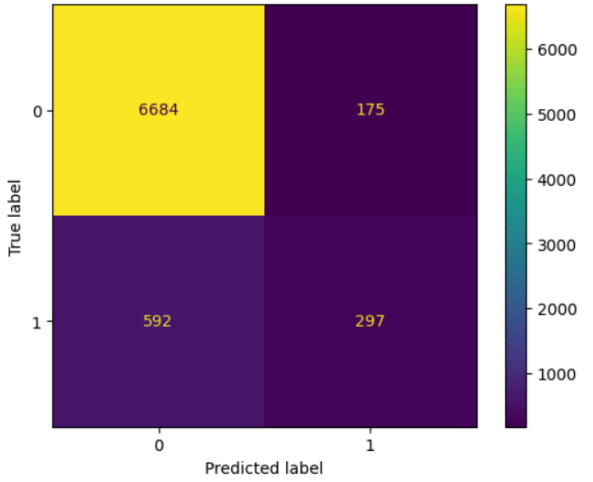


**Figura 18.** RBF SVC - treino

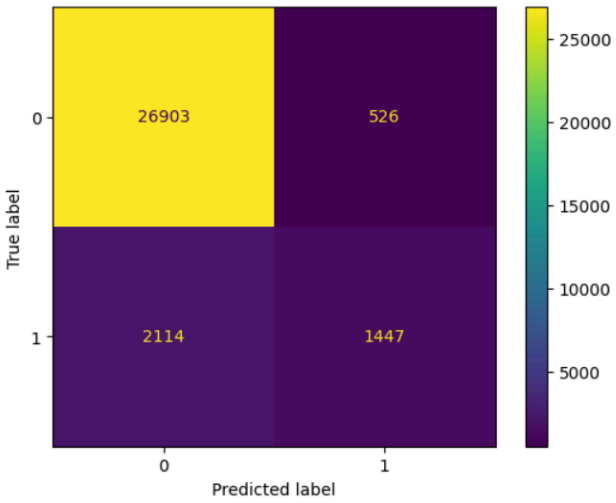



**Figura 19.** RBF SVC - teste

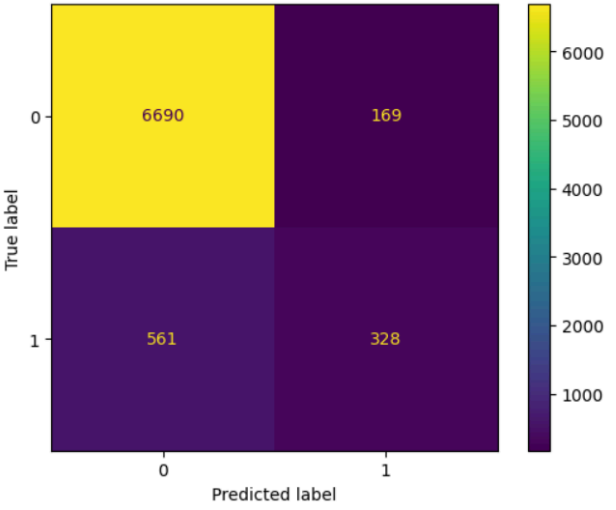



**Figura 20.** Decision Tree - treino

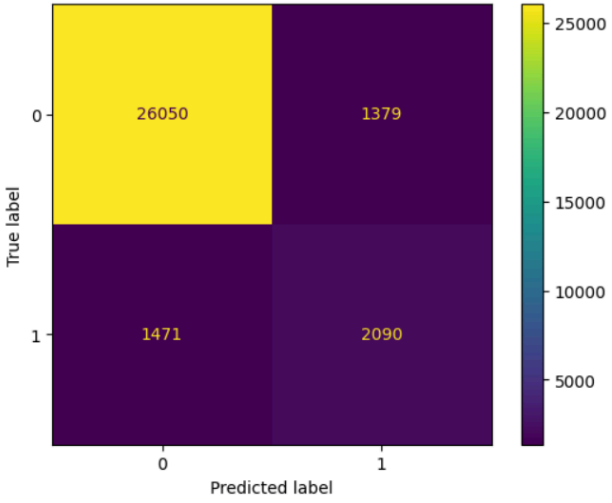



**Figura 21.** Decision Tree - teste

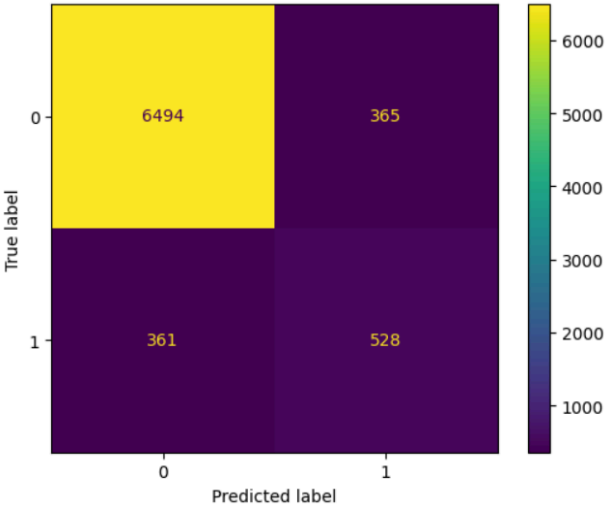



**Figura 22.** Random Forest - treino

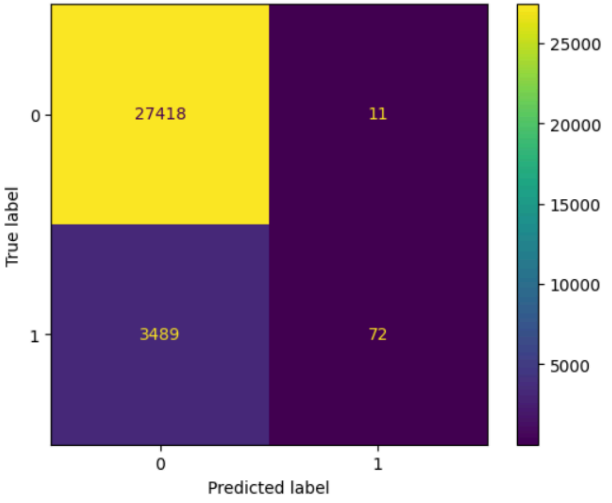



**Figura 23.** Random Forest - teste

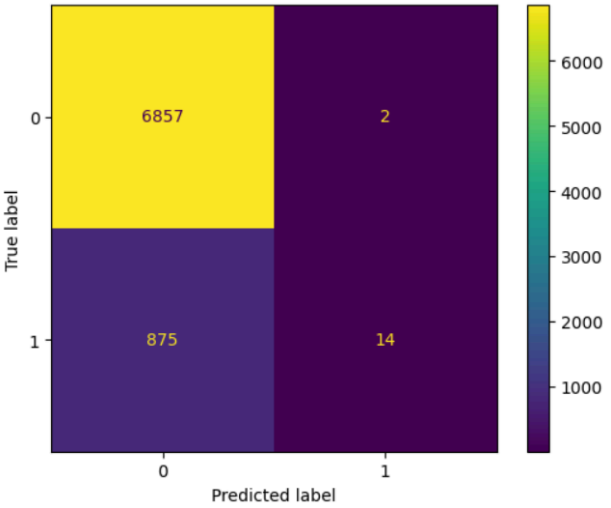


**Figura 24.** Gradient Boosting - treino

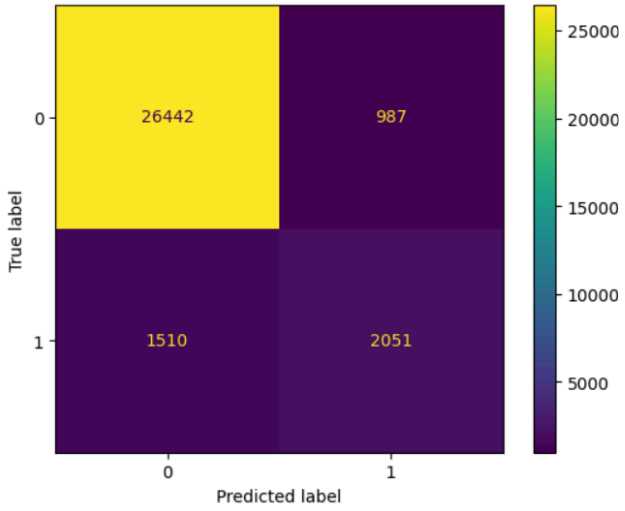



**Figura 25.** Gradient Boosting - teste

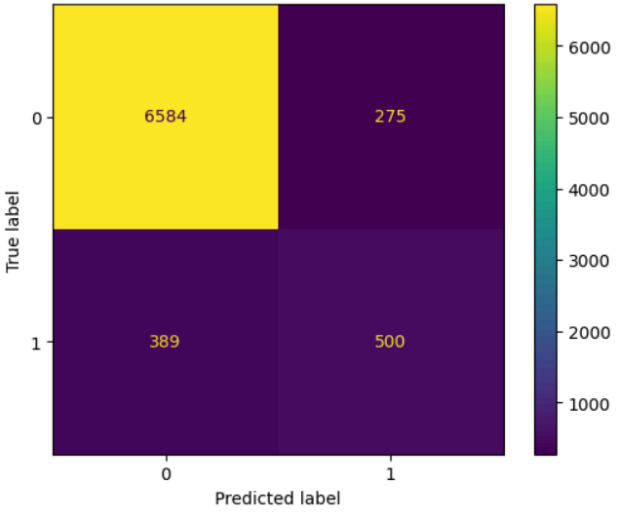



**Figura 26.** XGBoost - treino

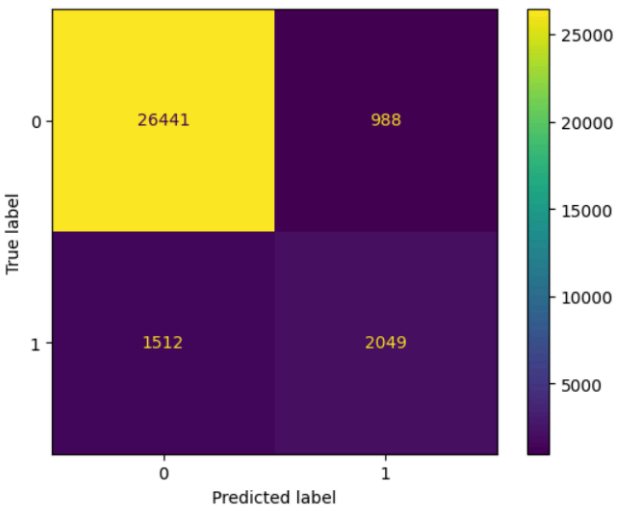



**Figura 27.** XGBoost - teste

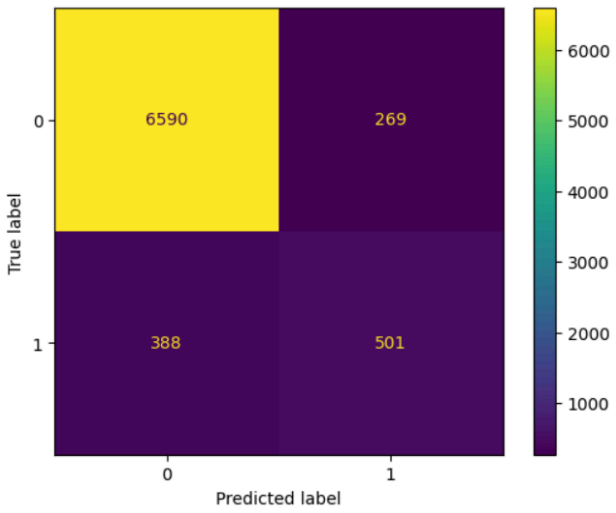



**Figura 28.** LightGBM - treino

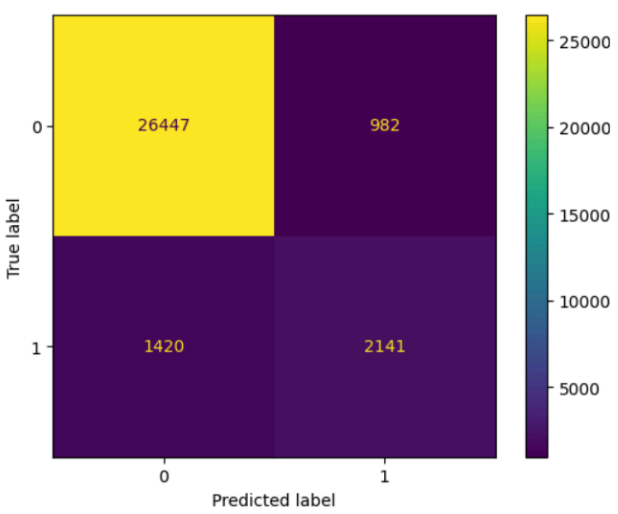
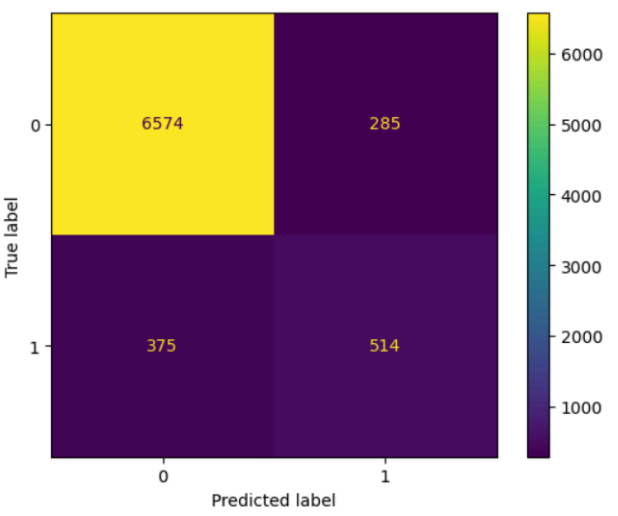



**Figura 29.** LightGBM - teste



# Função apply_model antes da afinação de hiperparâmetros

```python
# Função de apply model

problem_type = 'classification'

def apply_model(algorithm, problem_type, X_train, X_test, y_train, y_test):
  if 'KNN' == algorithm:
    param_grid = {'n_neighbors': np.arange(2, 20 , 2)}
    grid = GridSearchCV(KNeighborsClassifier(), param_grid, cv=10)
  elif 'logistic_regression' == algorithm:
    param_grid = {'C': [0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 10.0]}
    grid = GridSearchCV(LogisticRegression(), param_grid, cv=10)
  elif 'linear_SVC' == algorithm:
    param_grid = {'C': [0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 10.0]}
    grid = GridSearchCV(SVC(kernel='linear', max_iter=10000), param_grid, cv=10)
  elif 'rbf_SVC'== algorithm:
    param_grid = {'C': [0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 10.0],
                  'gamma': [0.01, 0.1, 1, 5, 10, 100]}
    grid = GridSearchCV(SVC(kernel='rbf'), param_grid, cv=10)
  elif 'DT' == algorithm:
      param_grid = {'max_depth': range(2, 6), 'max_leaf_nodes': range(2, 10), 'criterion': ['gini', 'entropy']}
      grid = GridSearchCV(DecisionTreeClassifier(), param_grid, cv=10).fit(X_train, y_train)
  elif 'RF' == algorithm:
      param_grid = {'max_depth': np.arange(2, 7), 'max_leaf_nodes': np.arange(5, 7), 'n_estimators': [100, 200, 300, 400, 500]}
      grid = GridSearchCV(RandomForestClassifier(), param_grid, cv=10).fit(X_train, y_train)
  elif 'GB' == algorithm:
    param_grid = {'max_depth': np.arange(2, 7), 'learning_rate': [0.05, 0.1, 0.2, 0.30.05]}
    grid = GridSearchCV(GradientBoostingClassifier(n_estimators=100), param_grid, cv=10).fit(X_train, y_train)
  elif 'xgboost' == algorithm:
    param_grid = {'max_depth': np.arange(1, 7), 'learning_rate': [0.05, 0.1, 0.2, 0.3]}
    grid = GridSearchCV(XGBClassifier(n_estimators=100), param_grid, cv=10).fit(X_train, y_train)
  elif 'lightgbm' == algorithm:
    param_grid = {'max_depth': np.arange(4, 11), 'learning_rate': [0.05, 0.1, 0.2, 0.3]}
    params = {'n_estimators': 100,
            'verbose': -1,
            'importance_type': 'gain'
        }
    grid = GridSearchCV(LGBMClassifier(**params), param_grid, cv=10).fit(X_train, y_train)
  else:
    print('Escolha um modelo de classificação válido.')
```

**Figura 30.** Função apply_model antes da afinação de hiperparâmetros, com as listas de parâmetros a testar