# INTELIGÊNCIA ARTIFICIAL APLICADA - MACHINE LEARNING

AULA 4

Prof. Antonio Willian Sousa

## CONVERSA INICIAL

Nesta aula, aprenderemos sobre outros métodos de classificação supervisionada, começando pelos classificadores bayesianos. Além disso, vamos apresentar métodos de predição de valores ao invés de valores categóricos, como vimos até então. Ao final, falaremos de um outro algoritmo de classificação que pode ser considerado clássico, o kNN.

## TEMA 1 – CLASSIFICADOR BAYESIANO

O algoritmo Naives Bayes ou Bayes Ingênuo, também conhecido como *classificador bayesiano*, faz parte da categoria de métodos supervisionados de aprendizagem. Trata-se de uma família de classificadores probabilísticos baseados no Teorema de Bayes. Esse algoritmo é muito utilizado para categorização de textos, principalmente na classificação de mensagens de correio eletrônico em *spam* e não *spam*, sendo o seu uso nesse tipo de classificação uma das razões para o seu grande sucesso e difusão.

O classificador bayesiano recebe a descrição de “ingênuo”, pois seu método desconsidera as relações que possam existir entre os atributos de um conjunto de dados. Assim, se tivermos um problema de classificação para determinar modelos de carros, o método considera que o tamanho do carro e o número de portas não apresentam nenhuma relação ou que o número de portas e a categoria (simples, luxo, especial) também não apresentam relação entre si, o que não reflete a realidade. Ainda assim, esse classificador pode oferecer bons resultados para diversas situações e permite o treinamento em conjuntos pequenos de dados.

Para compreender o funcionamento do classificador bayesiano, precisamos entender o Teorema de Bayes. De forma simplificada, o teorema é baseado na probabilidade condicional, que pode ser compreendida como a probabilidade que um evento A aconteça dado que um outro evento B já tenha ocorrido. O teorema pode ser definido da seguinte maneira:

$$P(A \mid B) = \frac{P(B \mid A)\,P(A)}{P(B)}$$

Sendo:

- P(A|B) = probabilidade *a posteriori* (aquilo que se deseja descobrir);
- P(B|A) = probabilidade de B dado que A é verdadeiro;
- P(A) = probabilidade *a priori* (probabilidade do evento A);
- P(B) = probabilidade marginal (probabilidade do evento B).

Para entendermos melhor esse conceito, imaginemos a seguinte situação em uma lista de 100 clientes (Tabela 1).

Tabela 1 – Situação 1

| GOSTA FUTEBOL | MASCULINO | FEMININO | BRASILEIRO | ESTRANGEIRO | **Total** |
|---|---|---|---|---|---|
| SIM | 40 | 30 | 40 | 30 | **70** |
| NÃO | 20 | 10 | 5 | 25 | **30** |
| **Total** | **60** | **40** | **45** | **55** | |

Fonte: o autor.

Analisando a Tabela 1, tem-se:

- 100 clientes no total;
- 70 clientes gostam de futebol (70%);
- 40 dos 70 clientes que gostam de futebol são homens (57%);
- 40 dos 70 clientes que gostam de futebol são brasileiros (57%);
- 60 dos 100 clientes são homens (60%);
- 45 dos 100 clientes são brasileiros (45%).

Com esse conjunto de dados e informações, imagine que, ao escolhermos um cliente do sexo masculino, necessitamos determinar a probabilidade de ele gostar de futebol. Para isso, podemos utilizar as informações prévias, calcular as probabilidades daquilo que conhecemos e assim estimar a probabilidade do que não conhecemos usando o Teorema de Bayes:

$$P(A \mid B) = \frac{P(B \mid A)\,P(A)}{P(B)}$$

Sendo:

- P(A | B): P(gosta de futebol | homem) = ?;
- P(B | A): P(homem | gosta de futebol) = 0,57 (ou 57%);

- P(A): P(gosta de futebol) = 0,70 (ou 70%);
- P(B): P(homem) = 0,60 (ou 60%).

Assim, nosso cálculo será:

$$P(\text{gosta de futebol} \mid \text{homem}) = \frac{P(\text{homem} \mid \text{gosta de futebol}) \times P(\text{gosta de futebol})}{P(\text{homem})}$$

$$P(\text{gosta de futebol} \mid \text{homem}) = \frac{0,57 \times 0,70}{0,60}$$

**P(gosta de futebol | homem) = 0,6650 (ou 66,50%)**

Todavia, devemos considerar que, quando estamos trabalhando com dados reais, os atributos não poderão ser analisados um a um, mas sim em todo o seu conjunto. Logo, se escolhermos um cliente, qual a probabilidade de que ele goste de futebol, considerando que ele é do sexo masculino e brasileiro? Essa é a nossa probabilidade *a posteriori*, ou aquilo que desejamos saber.

- P(gosta de futebol | homem, brasileiro) = ?
- P(homem | gosta de futebol) = 0,57 (ou 57%)
- P(brasileiro | gosta de futebol) = 0,57 (ou 57%)
- P(gosta de futebol) = 0,70 (ou 70%)
- P(homem) = 0,60 (ou 60%)
- P(brasileiro) = 0,45 (ou 45%)

Assim, nosso cálculo de P(gosta de futebol | homem, brasileiro) será:

$$\frac{P(\text{homem} \mid \text{gosta de futebol}) \times P(\text{brasileiro} \mid \text{gosta de futebol}) \times P(\text{gosta de futebol})}{P(\text{homem}) \times P(\text{brasileiro})}$$

$$P(\text{gosta de futebol} \mid \text{homem, brasileiro}) = \frac{0,57 \times 0,57 \times 0,70}{0,60 \times 0,45}$$

$$P(\text{gosta de futebol} \mid \text{homem, brasileiro}) = \frac{0,57 \times 0,57 \times 0,70}{0,60 \times 0,45}$$

**P(gosta de futebol | homem, brasileiro) = 0,8423 (ou 84,23%)**

Portanto, quando for necessário utilizar o classificador bayesiano para um conjunto grande de atributos, a probabilidade da classe (y) de uma instância dependerá de todos os atributos, como mostra a equação a seguir.

$$P\left(y \mid x_1, \cdots, x_j\right) = \frac{P\left(x_1, \cdots x_j \mid y\right)P(y)}{P\left(x_1, \cdots, x_j\right)}$$

O modelo é treinado de acordo com as classes e a definição da classe à qual uma instância pertence é dada pelo valor máximo da probabilidade *a posteriori* ou $P(y \mid x_1,\ldots,x_j)$. O uso de um modelo desse tipo apresenta algumas vantagens, pois assim como os modelos de Árvores de Decisão que já conhecemos, eles permitem uma interpretação e um entendimento fácil das decisões tomadas, bem como podem ser treinados de forma relativamente rápida e permitem trabalhar com conjuntos de dados pequenos.

A despeito das suas vantagens, os modelos baseados em classificador bayesiano podem ter problemas para classificar instâncias cuja probabilidade de algum atributo seja zero (0), pois, como vimos anteriormente, o fato de assumirmos que os atributos são independentes entre si nos leva a uma multiplicação de probabilidades e, no caso de um atributo com frequência de valor zero, a probabilidade total será zero também. Ainda a respeito da consideração que é feita de independência entre as classes, o uso de método como classificador bayesiano pode dificultar a determinação do quanto os atributos e suas correlações influenciam a escolha das classes, além de ser difícil encontrar *datasets* reais em que se possa confirmar isso.

Mesmo que apresente alguns pontos fracos, o classificador bayesiano é aplicável em diversos problemas e áreas como classificação de textos, como no caso de detecção de *spams*, em sistemas de recomendação, análise de sentimentos, entre outros.

## TEMA 2 – USANDO UM MODELO BAYESIANO

O modelo de classificação bayesiano pode ser implementado de diferentes maneiras, dependendo da biblioteca de aprendizagem de máquina usada, sendo mais comuns as seguintes implementações:

- Gaussian Naive Bayes: probabilidade *a posteriori* é calculada utilizando a densidade de probabilidade da distribuição normal;
- Multinomial Naive Bayes: usa uma distribuição multinominal parametrizada por vetores que indicam a probabilidade de cada evento

ocorrer. Nessa implementação, há uma correção para evitar que a probabilidade total seja zero;

- Bernoulli Naive Bayes: essa implantação se baseia em uma distribuição multivariada de Bernoulli que é composta por valores binários. Caso alguma característica não seja binária, ela será transformada em binária. Diferentemente do Multinomial Naive Bayes, nessa implementação há uma penalidade caso haja frequência nula para algum atributo.

Nesta seção, faremos os processos de preparação dos dados, separação dos conjuntos de treinamento e de testes e treinamento de um modelo de classificador bayesiano sobre o conjunto de dados.

## 2.1 Treinando um classificador

Um treinamento com um *dataset* desse tipo representa um problema real, com o qual vamos nos deparar ao trabalhar com dados reais. Todo o pré-processamento e a divisão dos subconjuntos de treinamento e testes foi feito nesse conjunto de dados. A Tabela 2 a seguir apresenta as informações sobre os dados.

Tabela 2 – Informações sobre os dados

| ***Dataset* de automóveis** | |
|---|---|
| Total de registros do *dataset*: | 11.913 |
| Tamanho do conjunto de treino: | 9.427 |
| Tamanho do conjunto de testes: | 2.357 |
| Quantidade de características: | 14 |
| Quantidade de classes: | 48 |

Fonte: Sousa, 2020.

A Tabela 3 exibe uma amostra do *dataset*, seus atributos e *labels*.

Tabela 3 – *Dataset* de dados de automóveis

| ano | combustivel | hp | cilindros | tipo_transmissao | tracao | num_portas | categoria | tamanho | estilo | km_por_l_estrada | km_por_l_cidade | popularidade | preco_venda | label |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2011 | gasolina premium | 335.0 | 6.0 | MANUAL | tracao_traseira | 2.0 | Audio de Fabrica,Luxo,Alta-Performance | compacto | Cupê | 26 | 19 | 3916 | 46135 | BMW 1 Series M |
| 2011 | gasolina premium | 300.0 | 6.0 | MANUAL | tracao_traseira | 2.0 | Luxo,Performance | compacto | Conversível | 28 | 19 | 3916 | 40650 | BMW 1 Series |
| 2011 | gasolina premium | 300.0 | 6.0 | MANUAL | tracao_traseira | 2.0 | Luxo,Alta-Performance | compacto | Cupê | 28 | 20 | 3916 | 36350 | BMW 1 Series |
| 2011 | gasolina premium | 230.0 | 6.0 | MANUAL | tracao_traseira | 2.0 | Luxo,Performance | compacto | Cupê | 28 | 18 | 3916 | 29450 | BMW 1 Series |
| 2011 | gasolina premium | 230.0 | 6.0 | MANUAL | tracao_traseira | 2.0 | Luxo | compacto | Conversível | 28 | 18 | 3916 | 34500 | BMW 1 Series |
| 2012 | gasolina premium | 230.0 | 6.0 | MANUAL | tracao_traseira | 2.0 | Luxo,Performance | compacto | Cupê | 28 | 18 | 3916 | 31200 | BMW 1 Series |
| 2012 | gasolina premium | 300.0 | 6.0 | MANUAL | tracao_traseira | 2.0 | Luxo,Performance | compacto | Conversível | 26 | 17 | 3916 | 44100 | BMW 1 Series |
| 2012 | gasolina premium | 300.0 | 6.0 | MANUAL | tracao_traseira | 2.0 | Luxo,Alta-Performance | compacto | Cupê | 28 | 20 | 3916 | 39300 | BMW 1 Series |
| 2012 | gasolina premium | 230.0 | 6.0 | MANUAL | tracao_traseira | 2.0 | Luxo | compacto | Conversível | 28 | 18 | 3916 | 36900 | BMW 1 Series |
| 2013 | gasolina premium | 230.0 | 6.0 | MANUAL | tracao_traseira | 2.0 | Luxo | compacto | Conversível | 27 | 18 | 3916 | 37200 | BMW 1 Series |

Fonte: Sousa, 2020.

O trecho de código a seguir mostra um processo de classificação de um conjunto de dados de automóveis utilizando o algoritmo Naives Bayes, com a implementação Gaussian Naive Bayes. O objetivo do modelo é determinar, por meio das características de um veículo, o fabricante. Assim, a coluna de *labels* contém um código para cada fabricante distinto que aparece no *dataset*.

```
import pandas as pd

from sklearn import tree
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split

# Lê o dataset de carros
carros = pd.read_csv("carros_categorical.csv")

# Seleciona apenas as colunas necessárias
X = carros.loc[:, 'ano':'preco_venda'].to_numpy()
y = carros['label'].to_numpy()

# Divide o conjunto em treinamento e teste, na proporção 80-20
X_treino, X_teste, y_treino, y_teste = train_test_split(X, y, test_size=0.20,
                                        random_state=42)
# Executa a definição e treinamento do modelo
gnb = GaussianNB()
gnb.fit(X_treino, y_treino)
```

Após o treinamento do modelo, podemos avaliar o seu desempenho de forma simplificada, verificando o quanto está acertando nas predições sobre a base de testes, e isso de forma mais aprofundada, por meio dos escores de precisão, *recall* e F1, conforme o código a seguir.

```
        import numpy as np
from sklearn.metrics import classification_report
from sklearn.naive_bayes import GaussianNB

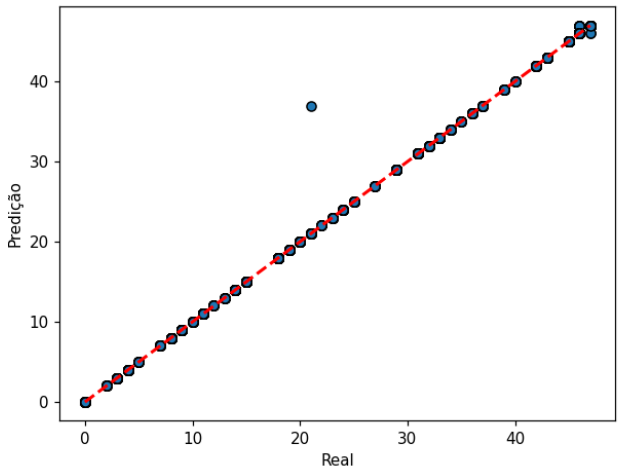
y_predicao = gnb.predict(X_teste)

# Avalia o desempenho do modelo treinado
print("Total de categorias: {}".format(len(categorias)))
cats_teste = np.unique(y_teste).tolist()
code_cats = carros['fabricante'].cat.categories
print("Total de categorias de teste: {}".format(len(code_cats)))

# Plota o grafico de resultados
fig, ax = plt.subplots()
ax.plot([y_teste.min(), y_teste.max()],
   [y_teste.min(), y_teste.max()],
   'k--', lw=2, color='red')
ax.scatter(y_teste, y_predicao, edgecolors=(0, 0, 0))
ax.set_xlabel('Real')
ax.set_ylabel('Predição')
plt.show()
print("Total de classificações erradas de um total de \
 {} instâncias : {}".format(X_teste.shape[0],
            (y_teste != y_predicao).sum()))
classification_report(y_teste, y_predicao, target_names=code_cats)
```

Assim, serão geradas as seguintes saídas (Gráfico 1):

Gráfico 1 – Saídas




Fonte: Sousa, 2020.

Classificações erradas de um total de 2.357 instâncias : 11

| classe | precision | recall | f1-score |
|---|---|---|---|
| Acura | 1.00 | 1.00 | 1.00 |
| Aston Martin | 1.00 | 1.00 | 1.00 |
| Audi | 1.00 | 1.00 | 1.00 |
| BMW | 1.00 | 1.00 | 1.00 |
| Bentley | 1.00 | 1.00 | 1.00 |
| Buick | 1.00 | 1.00 | 1.00 |
| Cadillac | 1.00 | 1.00 | 1.00 |
| Chevrolet | 1.00 | 1.00 | 1.00 |
| … | … | … | … |
| Subaru | 1.00 | 1.00 | 1.00 |
| Suzuki | 1.00 | 1.00 | 1.00 |
| Toyota | 1.00 | 1.00 | 1.00 |
| Volkswagen | 0.99 | 0.95 | 0.97 |
| Volvo | 0.87 | 0.96 | 0.92 |

Podemos ver que o modelo após o treinamento e a execução na base de testes apresenta um bom desempenho, sem que tenha sido necessária nenhuma alteração ou estudo mais aprofundado de parâmetros no momento da definição do modelo. Esta acaba sendo uma característica do uso dos modelos de classificadores bayesianos, em que o trabalho de ajuste do modelo, se necessário, de forma geral é feito alterações por meio de informações da distribuição dos dados em relação aos atributos e às classes.

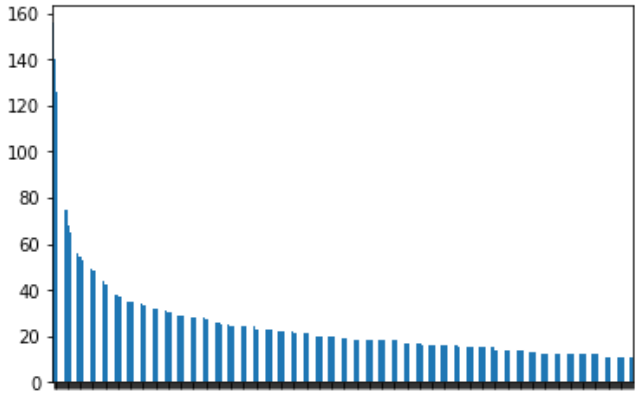
## 2.2 Melhorando o desempenho

Quando utilizamos um modelo como o classificador bayesiano, de fato estamos predizendo, além das classes de cada instância, um valor de probabilidade associada a essa escolha. Assim, para cada probabilidade calculada, há uma confiança associada. Alguns classificadores, a depender do balanceamento dos dados utilizados no treinamento, podem devolver valores de confiança excessivamente altos para algumas classes ou excessivamente baixos para outras. Para evitar esse comportamento, o classificador deve passar por um processo de calibração, que permitirá uma melhor predição e levará em consideração o balanceamento dos dados de treino.

Para ilustrar o funcionamento do processo de calibração, vamos considerar o uso do mesmo *dataset* de automóveis, mas agora estamos

interessados em predizer não os fabricantes, mas sim o modelo de um automóvel por meio dos atributos fornecidos. Com essa mudança, o problema, que originalmente possuía 48 classes, passará agora a contar com 381 classes. Essa mudança traz algumas dificuldades adicionais para o classificador, pois os dados se encontram consideravelmente desbalanceados, conforme podemos ver no Gráfico 2.

Gráfico 2 – Distribuição dos automóveis por modelo




Fonte: Sousa, 2020.

A execução de um classificador Naive Bayes para esse problema, considerando esse desbalanceamento, obterá os seguintes resultados.

**Classificações erradas de um total de 1845 instâncias : 177**

**Precisão: 0.925**
**Recall: 0.912**
**F1 Score: 0.904**

Se executarmos o processo de calibração, os resultados de classificação poderão ser melhorados, conforme podemos ver a seguir.

```
from sklearn.calibration import CalibratedClassifierCV

# Calibrated with isotonic calibration
isotonic = CalibratedClassifierCV(gnb, cv=4, method='isotonic')
isotonic.fit(X_treino, y_treino)
```

Com essa pequena alteração, os resultados do classificador são melhorados.

**Classificações erradas de um total de 1845 instâncias : 119**

**Precisão: 0.945**
**Recall: 0.938**
**F1: 0.935**

Dessa maneira, a calibração do classificador pode prover uma melhora significativa na predição, bem como há a possibilidade de escolher uma implementação diferente. No caso do classificador bayesiano, temos três implementações que podem ser utilizadas dependendo dos dados disponíveis. O processo de balanceamento pode ser executado para quaisquer umas dessas implementações.

## TEMA 3 – REGRESSÃO LINEAR E LOGÍSTICA

Nesta seção, vamos discutir o funcionamento dos métodos de regressão linear e regressão logística, seus conceitos e suas aplicações, bem como explicar suas diferenças e como cada um deles pode ser aplicado a problemas distintos.

### 3.1 Regressão linear

A regressão linear é uma técnica estatística utilizada no processo de investigação das relações existentes entre as variáveis que definem um conjunto de dados. A análise dessas relações é feita por meio de um modelo que constrói uma equação que associa as variáveis dependentes com as independentes. Dessa maneira, o modelo criado busca identificar uma relação de causa entre duas ou mais variáveis contínuas. Análises desse tipo podem ser usadas para prever valores de uma variável por meio de outras que a definem ou substituir a medição de uma variável que não seja possível de medir pela observação de mudança de valor das outras variáveis.

Regressão linear, assim como os demais modelos que vimos até agora, é uma técnica simples e que permite entender como os valores são obtidos. Além disso, esse tipo de análise permite responder a questões como:

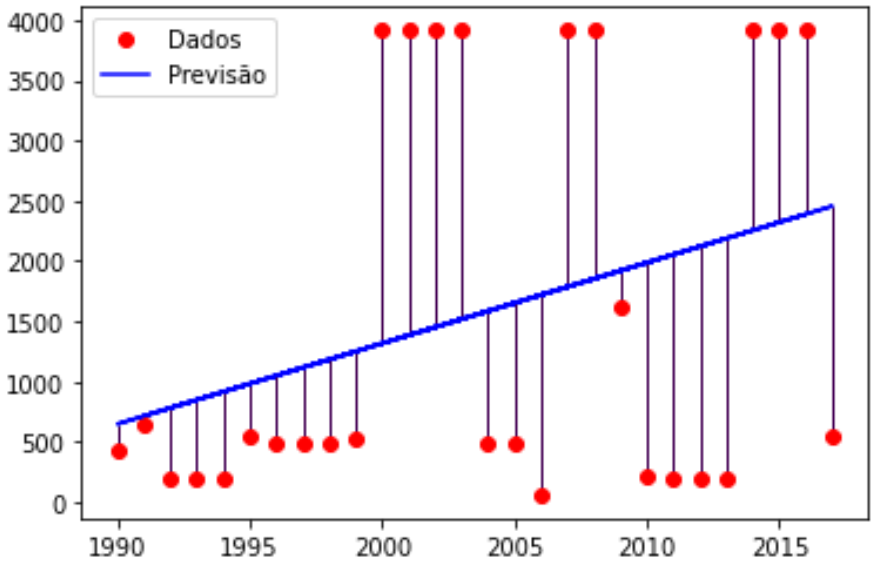
- o quão forte é o relacionamento entre as variáveis;
- quais as variáveis que contribuem para o valor final e quais não;
- qual a acurácia das previsões;
- se existe um relacionamento entre um par de variáveis.

Para entender como obter o modelo ou a equação que precisamos para analisar o relacionamento das variáveis, consideremos a situação em que temos uma variável independente com uma única variável dependente, cuja regressão linear pode ser expressa conforme a equação a seguir.

$$Y = \beta_0 + \beta_1 X$$

Nessa equação, já temos os valores tanto de **y** (variável independente), quanto de **x** (variável dependente) e queremos descobrir os valores dos coeficientes ($\beta_0$, $\beta_1$) para o nosso conjunto de dados. Para descobrir os valores dos parâmetros, consideremos uma situação em que necessitamos avaliar se existe uma relação linear entre a popularidade de um automóvel e o seu preço de venda. Assim, nossa variável independente (y) será o valor da popularidade e a variável independente o preço de venda (x). Já os valores da variável independente preditos pelo modelo (ŷ) poderão ser ligeiramente diferentes do valor real, pois o modelo dificilmente obterá uma precisão tão grande em todos os pontos do conjunto de dados. A Gráfico 3 apresenta os dados, bem como o modelo de regressão linear relacionado.

Gráfico 3 – Dados e modelo de regressão linear




Fonte: Sousa, 2020.

O modelo de regressão obtido será aquele – em casos como este, com apenas uma variável independente – que determine uma reta a qual minimiza a diferença entre o valor real e o valor da predição do modelo, sendo utilizado o valor do quadrado do erro como forma de evitar que diferenças negativas tenham influencia no valor final e como uma forma de penalizar valores de diferenças muito altas. O cálculo de cada um dos parâmetros é feito utilizando as equações a seguir:

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \qquad \beta_0 = \bar{y} - \beta_1 \bar{x}$$

Dessa maneira, os valores dos parâmetros são obtidos utilizando como referência para o cálculo as diferenças dos pontos para os valores médio, **x** e **y**. Assim, o modelo é determinado e novas predições de valores podem ser feitas. Essa forma de obtenção do modelo segue o mesmo princípio para situações em que os atributos, os quais representam as variáveis independentes, se apresentem em quantidades maiores que apenas um, como no exemplo.

## 3.2 Regressão logística

A regressão logística é um método estatístico. Diferentemente da regressão linear, que prevê valores contínuos, retorna uma predição categórica. Somente a variável dependente necessita ser categórica; as variáveis independentes podem ou não ser categóricas, de maneira que o valor estimado pelo modelo é obtido por meio das variáveis independentes, representando a probabilidade da variável dependente assumir um determinado valor.

Quando a predição do modelo pode ser definida por duas classes distintas como positivo ou negativo, aprovado ou reprovado, a análise executada é chamada de regressão logística binomial. Para situações em que temos três ou mais valores categóricos possíveis, utiliza-se a regressão logística multinomial.

Uma das principais diferenças entre a regressão logística e a regressão linear está no fato de a primeira utilizar uma função diferente para obtenção dos valores. Na regressão linear, os valores de saída podem variar de **-∞ a +∞,** porém, na regressão logística, os valores estão restritos ao intervalo de 0 a 1 porque esta prevê um valor de probabilidade. Assim, a regressão logística e o valor de probabilidade podem ser definidos conforme as seguintes equações:

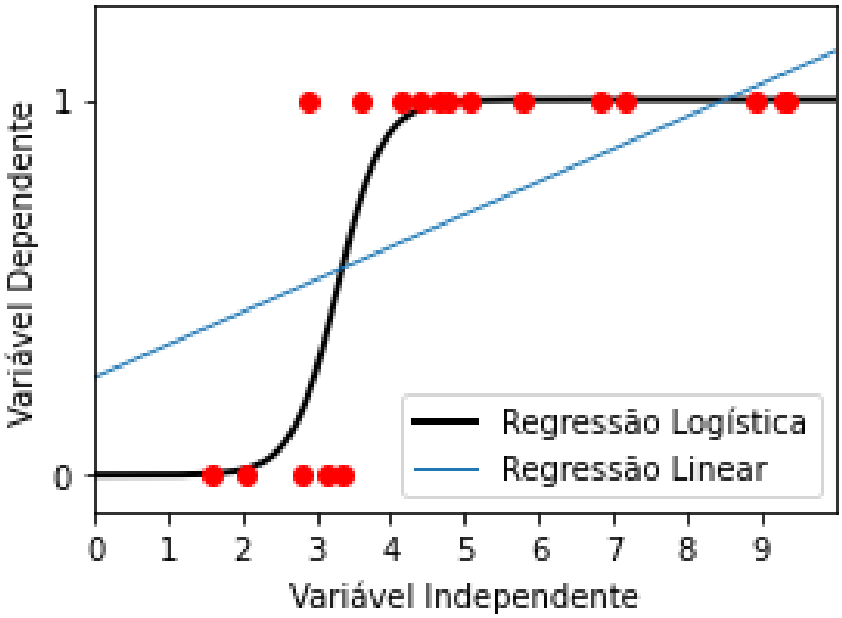
$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}. \qquad \log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X.$$

A função que representa o cálculo do valor de probabilidade é conhecida como função logística ou função sigmoide. Essa função, independentemente dos valores fornecidos, traz como resultado um valor entre 0 e 1. Isso é ideal para

situações em que seja necessário fornecer um valor de probabilidade como saída, ainda que ela apresente a limitação de conseguir lidar com uma quantidade muito grande de atributos categóricos. Mesmo assim, a regressão logística, como alguns dos classificadores que conhecemos anteriormente, apesar de simples é bastante eficiente e pode ser uma boa escolha para uma análise preliminar.

O gráfico da função sigmoide pode ser visto na Gráfico 4, em que percebemos uma clara diferença no modo de separação dos dados dos dois modelos de regressão.

Gráfico 4 – Comparação regressão linear e logística




Fone: Sousa, 2020.

A função sigmoide é importante não apenas para modelos de classificação logística, como também para outros modelos de classificação, como redes neurais. Dada a sua característica não linear, ela não exige que exista um relacionamento linear entre as variáveis dependentes e a variável independente. Contudo, pode ser necessária a transformação de atributos não lineares, bem como a escolha de atributos relevantes, uma vez que o modelo não lida bem com uma quantidade muito grande de atributos.

Dependendo da quantidade de atributos fornecida ao problema, o uso de atributos com alto grau de correlação ou irrelevantes para o modelo pode torná-lo extremamente ajustado aos dados de treinamento, com uma baixa capacidade de variação. Essa condição é chamada de *overfitting*. Em oposição a essa

condição, o modelo também pode apresentar o comportamento de não se adaptar sequer aos dados de treinamento e testes, com uma taxa de erro elevada, não conseguindo prever bem os valores de forma correta. Essa condição é chamada de *underfitting*.

## TEMA 4 – USANDO MODELOS DE REGRESSÃO

Nesta seção, vamos aprender como utilizar os diferentes modelos de regressão, logística e linear, por meio de exemplos em que cada uma delas seja a mais adequada.

### 4.1 Usando regressão logística

Com o entendimento de como funciona o modelo de regressão logística, estamos prontos para utilizá-lo em um problema de classificação real. Para exemplificar seu funcionamento, consideremos um *dataset* contendo atributos sobre automóveis, em que um atributo indica de forma binária se um automóvel é popular ou não, por meio da indicação de uma variável na qual o 0 indica não popular, e 1 indica popular. A Tabela **Erro! Fonte de referência não encontrada.** traz uma amostra dos dados do nosso *dataset*.

Tabela 4 – *Dataset* de automóveis

| ano | combustivel | hp | cilindros | tipo_transmissao | tracao | num_portas | categoria | tamanho | estilo | km_por_l_estrada | km_por_l_cidade | popularidade | preco_venda | label |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2011 | 7 | 300.0 | 6.0 | 3 | 3 | 2.0 | 68 | 0 | 4 | 28 | 19 | 3916 | 40650 | 1 |
| 2011 | 7 | 300.0 | 6.0 | 3 | 3 | 2.0 | 65 | 0 | 5 | 28 | 20 | 3916 | 36350 | 1 |
| 2011 | 7 | 230.0 | 6.0 | 3 | 3 | 2.0 | 68 | 0 | 5 | 28 | 18 | 3916 | 29450 | 1 |
| 2011 | 7 | 230.0 | 6.0 | 3 | 3 | 2.0 | 64 | 0 | 4 | 28 | 18 | 3916 | 34500 | 1 |
| 2012 | 7 | 230.0 | 6.0 | 3 | 3 | 2.0 | 68 | 0 | 5 | 28 | 18 | 3916 | 31200 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2017 | 3 | 355.0 | 8.0 | 0 | 3 | 4.0 | 49 | 1 | 3 | 23 | 16 | 549 | 57315 | 0 |
| 2017 | 3 | 355.0 | 8.0 | 0 | 0 | 4.0 | 49 | 1 | 3 | 22 | 16 | 549 | 51530 | 0 |
| 2017 | 5 | 420.0 | 8.0 | 0 | 3 | 4.0 | 70 | 1 | 3 | 22 | 15 | 549 | 65965 | 0 |
| 2017 | 3 | 355.0 | 8.0 | 0 | 0 | 4.0 | 49 | 1 | 3 | 22 | 16 | 549 | 60315 | 0 |
| 2017 | 3 | 355.0 | 8.0 | 0 | 3 | 4.0 | 49 | 1 | 3 | 23 | 16 | 549 | 48530 | 0 |

Fonte: Sousa, 2020.

Perceba que as variáveis são todas numéricas e que algumas delas – por exemplo, estilo e categoria –, descritas de forma textual, estão representadas de forma categórica. Assim, para cada valor numérico há uma descrição textual correspondente. No caso do atributo *tipo_transmissao*, a codificação das categorias pode ser feita conforme a Tabela 5 a seguir.

Tabela 5 – Codificação das categorias

| **Atributo:** tipo_transmissão | |
|---|---|
| **CÓDIGO** | **CATEGORIA** |
| 0 | AUTOMATICO |
| 1 | AUTOMATIZADO |
| 2 | DIRECT_DRIVE |
| 3 | MANUAL |
| 4 | UNKNOWN |

Fonte: Sousa, 2020.

Podemos treinar um classificador utilizando regressão logística para identificar se um dado automóvel é popular ou não, bem como se um automóvel que não consta no *dataset*, mas com especificações semelhantes, será popular ou não. Esse classificador será treinado utilizando todos os atributos das nossas instâncias e devolverá uma única saída contendo um dos dois valores, 0 ou 1, indicando “não popular” e “popular”. O código a seguir mostra como é feito processo de treinamento:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np

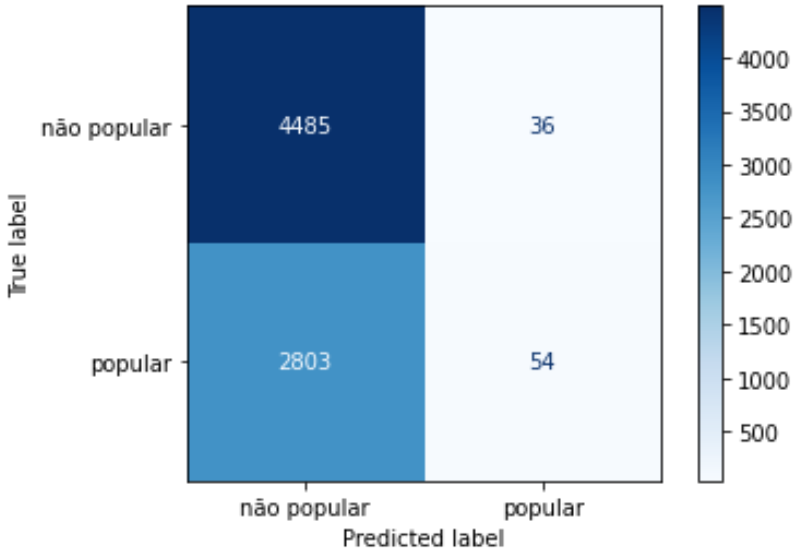
# Lê o dataset de carros
df_carros = pd.read_csv("carros_modelos_categorical.csv")
X = carros.iloc[:, :-1].to_numpy()
y = carros.iloc[:, -1].to_numpy()

# Divide o conjunto em treinamento e teste, na proporção 80-20
X_treino, X_teste, y_treino, y_teste = train_test_split(X, y, test_size=0.20,
stratify=y,  random_state=123, shuffle=True)
# Classificacao Regressao Logistica
clf_logistica = linear_model.LogisticRegression(C=1e5)
clf_logistica.fit(X_treino, y_treino)
y_pred_logistica = clf_logistica.predict(X_teste)
```

Para avaliarmos o desempenho do modelo treinado, podemos utilizar os valores das métricas de precisão, *recall* e *f1-score*, como a matriz de confusão das duas classes, mostrada na Gráfico 5.

Gráfico 5 – Matriz de confusão das classes popular e não popular

| | *precision* | *recall* | *f1-score* |
|---|---|---|---|
| **não popular** | **0.62** | **0.97** | **0.76** |
| **popular** | **0.60** | **0.08** | **0.14** |




Fonte: Sousa, 2020.

Partindo dos valores de precisão, podemos ver que o desempenho do classificador não é bom, pois, em termos de precisão, 60% ou 62% é um valor apenas um pouco melhor que uma escolha aleatória. Também podemos perceber pelos valores de *recall* que a categoria não popular está sendo detectada com um alto índice de erros. Esses valores de precisão e *recall* refletem nos valores de *f1-score*, que também pode ser utilizado para medirmos a qualidade da predição de cada uma das classes. Em geral, podemos utilizar apenas o *f1-score*, pois essa métrica incorpora informações de precisão e *recall*.

Uma vez que detectamos a necessidade de melhorar o nosso modelo, devemos iniciar um processo de reavaliação dos processos de seleção de dados ou do treinamento do modelo. No caso específico com o qual estamos lidando, por se tratar de um classificador de regressão logística, um bom indicador pode ser a forma como os atributos são entregues ao classificador.

Alguns atributos, ainda que possam ser representados de forma categórica, podem ser mais bem interpretados pelo modelo de regressão se representados de forma separada. Tomemos como exemplo o atributo tipo_transmissão do nosso *dataset* de automóveis. Ele está representado por meio de um intervalo de número de 0 a 4 para cada tipo de transmissão, conforme vimos anteriormente. Contudo, esse tipo de informação também pode

ser representado por um método chamado de *one-hot encoder*, em que, para cada categoria, ao invés de termos um número teremos um atributo que armazena a informação sobre aquela instância possuir ou não aqueles atributos.

A representação do atributo tipo_transmissão por meio de *one-hot encoder* será da forma como mostrado na Tabela 6 a seguir.

Tabela 6 – Representação do atributo tipo_transmissão em *one-hot encoder*

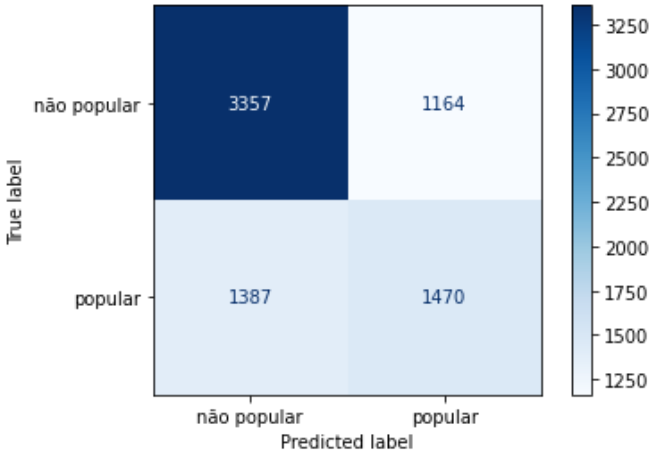
| **Atributo:** tipo_transmissão (tt) | | | | |
|---|---|---|---|---|
| **tt_automatico** | **tt_automizado** | **tt_direct_drive** | **tt_manual** | **tt_desconhecido** |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 |

Fonte: Sousa, 2020.

Assim, teremos cinco novos atributos a serem considerados no processo de treinamento. Essa alteração deverá ser feita para todas as instâncias e o treinamento do modelo deverá ser executando novamente. Os valores das métricas a seguir e a matriz de confusão mostram o novo desempenho do classificador após essas alterações.

Gráfico 6 – Novo desempenho do classificador

| | ***precision*** | ***recall*** | ***f1-score*** |
|---|---|---|---|
| **não popular** | **0.70** | **0.74** | **0.72** |
| **popular** | **0.55** | **0.50** | **0.53** |




Fonte: Sousa, 2020.

Podemos perceber uma sensível melhora nos valores das métricas e também no número de falsos positivos mostrados na matriz de confusão. Outras alterações podem ser feitas, como utilização de pesos para as categorias, exclusão de atributos que não possuam correlação, entre outras alterações que permitam um melhor desempenho do modelo. Tudo isso é feito por meio da experimentação e avaliação de resultados.

## 4.2 Usando regressão linear

Na seção anterior, utilizamos um processo de classificação com o método de regressão logística para prever se um automóvel se encaixaria nas classes popular e não popular. Além de conseguir prever resultados categóricos, o modelo de regressão também é muito útil quando queremos fazer uma previsão de um valor contínuo, por exemplo, o preço de venda de um novo automóvel.

Atributos como preço, altura, peso, entre vários outros, são difíceis de serem bem representados por categorias em decorrência da quantidade de valores possíveis. Assim, para atributos desse tipo, devemos buscar métodos que, baseados nos dados históricos, consigam sugerir um valor. Para isso, podemos utilizar o método de regressão linear.

Para avaliar o funcionamento do método de regressão logística, utilizaremos o nosso *dataset* de automóveis e tentaremos prever o valor de venda de um automóvel que não figure nos dados de treino. O código a seguir mostra o processo.

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Lê o dataset de carros
carros = pd.read_csv("carros_modelos_categorical.csv")
# Seleciona apenas as colunas necessárias
X = carros.loc[:, 'combustivel':'popularidade'].to_numpy()
y = carros['preco_venda'].to_numpy()

# Divide o conjunto em treinamento e teste, na proporção 80-20
X_treino, X_teste, y_treino, y_teste = train_test_split(X, y, test_size=0.20)
```

```
# Classificacao Regressão Linear
ols = linear_model.LinearRegression(normalize=True)
ols.fit(X_treino, y_treino.reshape((-1, 1)))
y_pred_ols = ols.predict(X_teste)
```

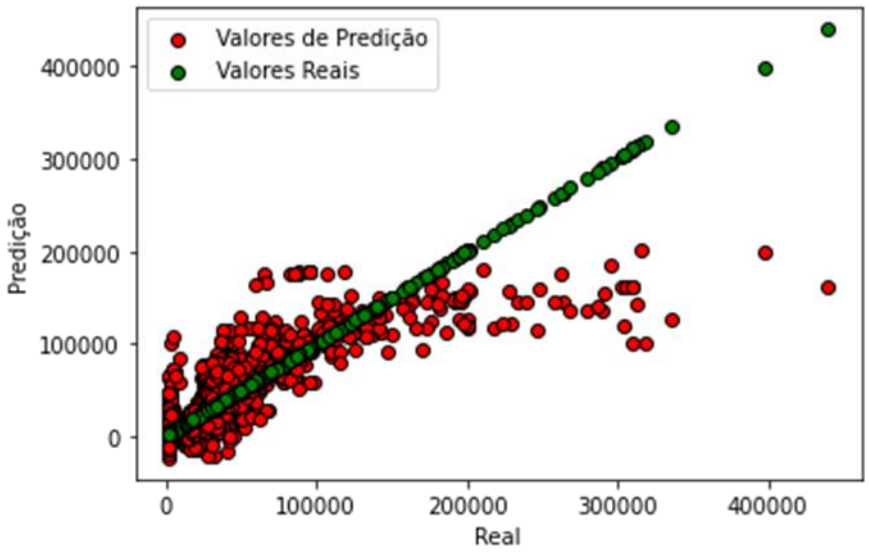
No caso da predição de um atributo contínuo, diferentemente da predição categórica, não podemos utilizar métricas como precisão, *recall* ou *f1-score*, como também não podemos utilizar matriz de confusão. Apesar de termos os valores reais dos conjuntos de testes, não podemos simplesmente contabilizar a quantidade de erros e acertos, pois devemos considerar o quanto a predição está distante do valor real.

Para fazer a avaliação do desempenho do classificador, podemos utilizar o valor do erro médio quadrático e o coeficiente de determinação, além do gráfico de plotagem dos valores de predição e dos valores reais. Dessa forma, à medida que formos modificando o modelo, podemos avaliar se houve melhora. A seguir, apresentamos os valores de erro, do coeficiente de determinação e o gráfico da predição para o modelo de predição do preço de venda de um automóvel.

Gráfico 7 – Valores de erro, do coeficiente de determinação e o gráfico da predição

**Quadrado da Média dos Erros: 675861464.37**

**Coeficiente de Determinação: 0.60**




Fonte: Sousa, 2020.

Por meio da observação desses valores, obtemos um valor de referência de modo que devemos buscar reduzir o Quadrado da Média dos Erro e aumentar o Coeficiente de Determinação. No caso da classificação, utilizando regressão

linear, sabemos que não haverá uma determinação simples de certo e errado, mas sim devemos avaliar o quanto os valores de predição estão adequados aos valores reais. Devemos também considerar as margens de erro da predição, permitindo assim uma noção mais clara do intervalo de variação da predição.

O Coeficiente de Determinação é uma medida cujo valor está compreendido entre 0 e 1. O seu valor indica o quanto os dados reais se encontram próximos à linha de regressão determinada pelo modelo, ou seja, indica o quanto o modelo está ajustado aos dados. Ainda, é também referenciado como $R^2$ ou coeficiente de determinação múltipla quando utilizado em modelos de regressão múltipla.

Já o Quadrado da Média dos Erros é uma medida obtida com base nas diferenças dos valores preditos pelo modelo e os valores reais (resíduos). Uma vez obtido, seu valor somente deverá ser comparado com valores oriundos de um novo modelo, de maneira que o primeiro valor serve de referência e deve-se buscar reduzi-lo. Este não deve, portanto, ser comparado com outras medidas como o Erro Médio Absoluto ou a Soma dos Quadrados dos Resíduos, que permite avaliar, respectivamente, a variação dos erros em termos absolutos sem penalizar os valores muito altos, e a variação da medida da variável independente.

A escolha de qual medida dos erros utilizar dependerá do que queremos avaliar com o nosso modelo. Se o fato de existirem valores discrepantes não for um problema, o uso de uma medida com o Erro Médio Absoluto também não o será. Contudo, se necessitarmos penalizar valores que se afastam muito da média (valores discrepantes), o uso do Quadrado da Média dos Erros permitirá avaliar melhor a existência desses valores no nosso conjunto de dados. Para muitos dos problemas com os quais é preciso lidar, em geral, o uso do Quadrado da Média dos Erros e do Coeficiente de Determinação tende a ser suficiente.

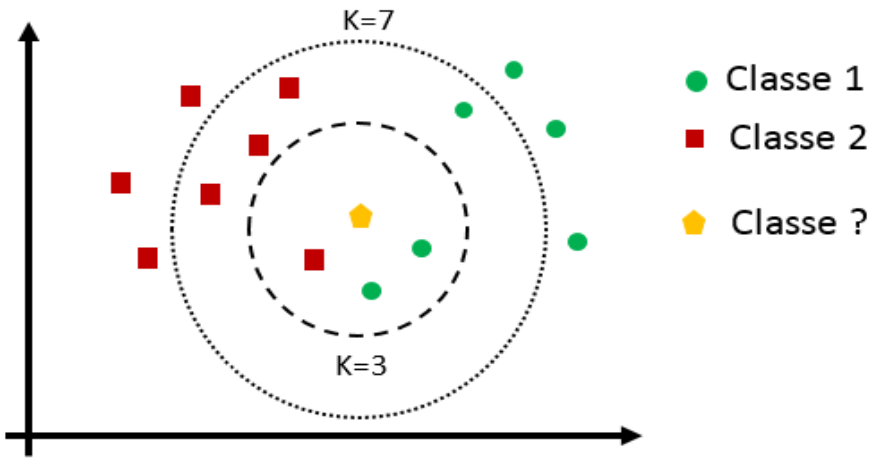
## TEMA 5 – CLASSIFICAÇÃO BASEADA EM VIZINHANÇA

Até agora, aprendemos a trabalhar com diferentes tipos de classificadores, como árvores de decisão e regressão linear. Nesta seção, será apresentado um classificador que, assim como a regressão logística, pode ser utilizado tanto para tarefas de classificação como de regressão: k-vizinhos mais próximos, do inglês *k-Nearest Neighbors* (kNN).

## 5.1 kNN

O kNN é um algoritmo de aprendizagem supervisionada no qual a predição se baseia nos dados das instâncias próximas da instância em análise, com a predição das classes feita por meio de voto. Ou seja, contabiliza-se as classes dos vizinhos e a instância em análise recebe a classe que teve maior número de votos. Imagine que, em um conjunto de dados de clientes, necessitamos determinar se um cliente gosta ou não de futebol. Para resolver essa questão, são analisados os vetores de características do nosso conjunto de dados de clientes e analisamos se os clientes próximos do nosso cliente-alvo são classificados como gostando ou não de futebol. Se a maioria dos clientes próximos do alvo gostam de futebol, essa será a categoria daquele cliente. Dessa forma, podemos perceber que o método de aprendizagem do kNN é simples e bem intuitivo. O Gráfico 8 exibe o conjunto de dados e a determinação da classe de uma nova instância baseado no kNN para 2 valores de **k** distintos.

Gráfico 8 – kNN para duas classes e duas distâncias distintas

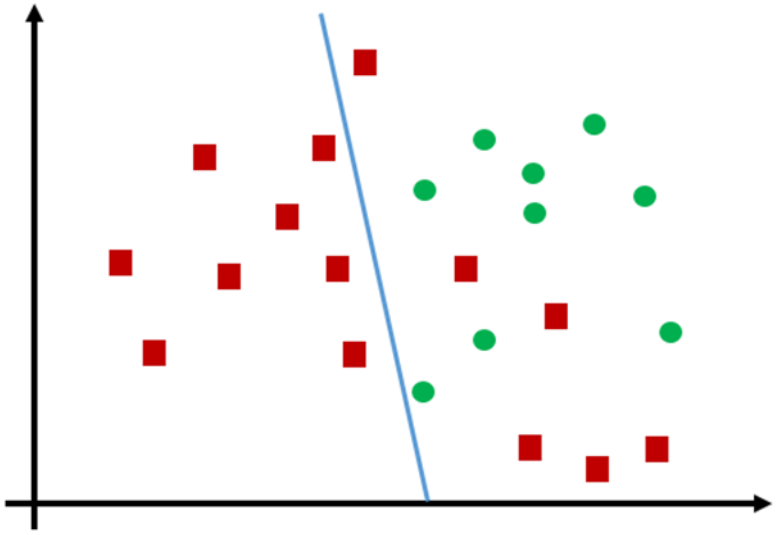



Fonte: Sousa, 2020.

No processo de predição do kNN, devemos considerar o cálculo da distância entre as instâncias, para determinar quais as **k** instâncias mais próximas, e o próprio valor de **k** também deve ser fornecido. Esses valores são cruciais no desempenho do modelo, pois valores de **k** muito baixos podem criar modelos muito específicos e como baixa capacidade de generalização – *overfitting*. Já valores de **k** muito altos podem gerar modelos que apresentam alta capacidade de generalização, mas com alta taxa de erros na predição – *underfitting*.

O *underfitting*, ou sobre-ajuste é uma situação em que o modelo obtido não consegue encontrar uma boa relação entre os dados de treino, ou seja, não serve nem para prever os valores dos dados de treino. Então, é muito provável que o seu desempenho será ruim em dados diferentes. O Gráfico 9 indica uma situação com duas classes em que o modelo obtido sofre de *underfitting*.

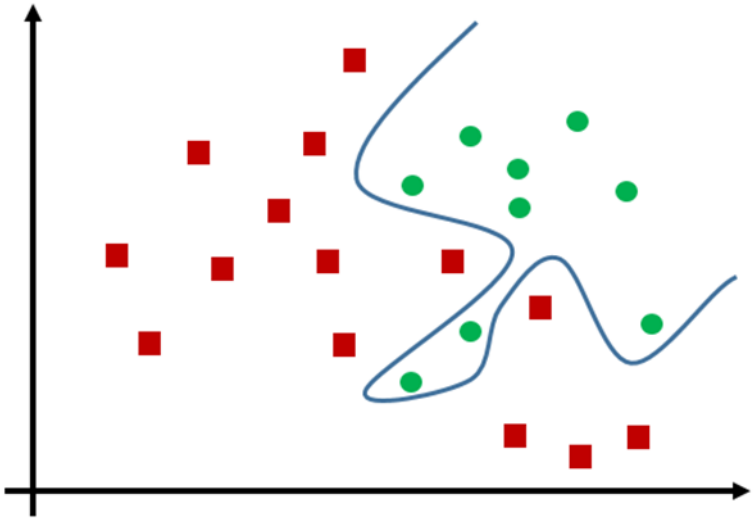
Gráfico 9 – Modelo com *underfitting*



Fonte: Sousa, 2020.

Já o *overfitting*, ou super-ajuste, ocorre quando o modelo é tão bem ajustado aos dados de treino que ele não consegue ser executado para outros dados. Assim, os resultados obtidos com base em um modelo com *overfitting* parecem ser perfeitos, pois o modelo não erra nenhuma das predições. No entanto, esse modelo somente funciona para os dados utilizados no treinamento. Assim, o acréscimo de qualquer instância que mude a distribuição dos dados afetará seriamente sua capacidade. O Gráfico 10 indica uma situação com duas classes em que o modelo obtido sofre de *overfitting*.

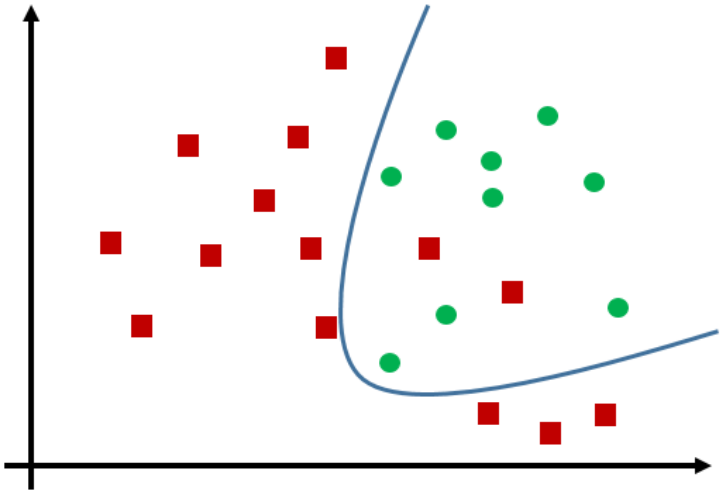
Gráfico 10 – Modelo com *overfitting*



Fonte: Sousa, 2020.

Modelos que apresentem *overfitting* e *underfitting* não são os ideais para uso. Assim, devemos buscar um modelo mais equilibrado com maior capacidade de generalização. O Gráfico 11 apresenta um modelo intermediário que consegue prever classes apesar de apresentar uma pequena taxa de erros.

Gráfico 11 – Modelo com ajuste intermediário



Fonte: Sousa, 2020.

Um modelo que consiga generalizar de forma relativamente boa não sofrerá nem de *underfitting* nem de *overfitting*. É necessário, sempre que treinarmos modelos de aprendizagem de máquina, estarmos atentos ao equilíbrio entre a variância e o bias, que influenciam a capacidade de generalização dos modelos.

## 5.2 Usando kNN

O kNN apresenta vantagens como permitir que os seus resultados sejam facilmente compreendidos e interpretados e poder ser utilizado para classificação e regressão. Como desvantagem, ele exige uma alta quantidade de memória, pois precisa carregar todas as instâncias, sendo computacionalmente caro. A predição pode ser lenta se a quantidade de distâncias a ser calculada for muito grande e a dimensão dos vetores de características for muito alta.

Para ilustrar o funcionamento do kNN, suponhamos que desejemos fazer a predição de automóveis nas categorias popular e não popular, assim como fizemos anteriormente utilizando um classificador baseado em regressão logística. O código a seguir exibe como executar o treinamento de modelo baseado em kNN considerando o número de vizinhos k=5.

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

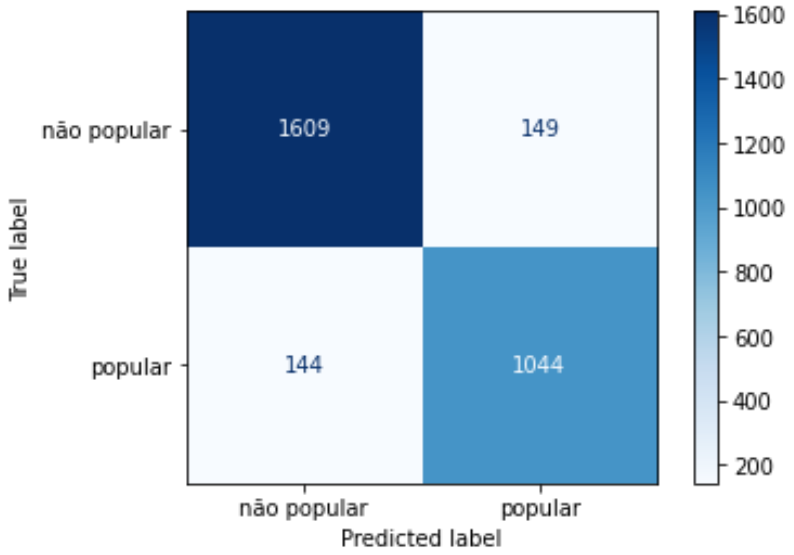
# Lê o dataset de carros
carros = pd.read_csv("carros_modelos_categorical.csv")
X = carros.iloc[:, :-1].to_numpy()
y = carros.iloc[:, -1].to_numpy()
X = StandardScaler().fit_transform(X)

# Divide o conjunto em treinamento e teste, na proporção 80-20
X_treino, X_teste, y_treino, y_teste = train_test_split(X, y, test_size=0.20, stratify=y,
                                                        random_state=123, shuffle=True)
# Treina o modelo
knn5 = KNeighborsClassifier(algorithm='ball_tree')
knn5.fit(X_treino, y_treino)
y_pred_5 = knn5.predict(X_teste)
```

Obtém, assim, os seguintes resultados indicados pela métricas e pela matriz de confusão a seguir, que mostram um bom desempenho do classificador no conjunto de dados de testes.

Gráfico 12 – Matriz de confusão para classificador kNN

| | precision | recall | f1-score |
|---|---|---|---|
| **não popular** | **0.89** | **0.89** | **0.89** |
| **popular** | **0.84** | **0.83** | **0.84** |




Fonte: Sousa, 2020.

Perceba que, no processo de treinamento desse classificador, fizemos um chamado ao método fit_transform do objeto StandardScaler. Essa chamada executa o processo de padronização das características do conjunto de dados. O processo de padronização consiste em, para cada característica de cada instância, diminuir o seu valor da média e dividir esse resultado pelo desvio-padrão. Dessa maneira, todos os valores de cada característica ficam contidos em um intervalo de 0 a 1, fazendo com que características com valores absolutos muito altos não se tornem dominantes no processo de classificação. Essa forma de padronização descrita é chamada de *z-score*.

Uma segunda condição que chama a atenção no processo de treinamento do kNN é a indicação do tipo de algoritmo que será utilizado para calcular a proximidade entre as instâncias. No exemplo anterior, utilizamos a chamada algorithm='ball_tree', porém, outros podem ser utilizados e dependerá do problema a ser resolvido, podendo ser utilizado em problemas de classificação – como fizemos – nos problemas de regressão.

O kNN também pode ser utilizado para resolver problemas de predição de valores contínuos como o valor de venda de uma instância do nosso conjunto de testes de automóveis. Para isso, o algoritmo de regressão executa uma interpolação dos valores das instâncias vizinhas da que estamos buscando

predizer. Assim como na predição de classes, também será necessário definir o número de k-vizinhos que serão considerados, como também é possível definir a utilização das distâncias dos vizinhos como pesos a serem utilizados no cálculo da regressão. O código a seguir demonstra o treinamento de um kNN para predizer o valor do preço de venda de um automóvel baseado nos dados do nosso conjunto de treinamento.

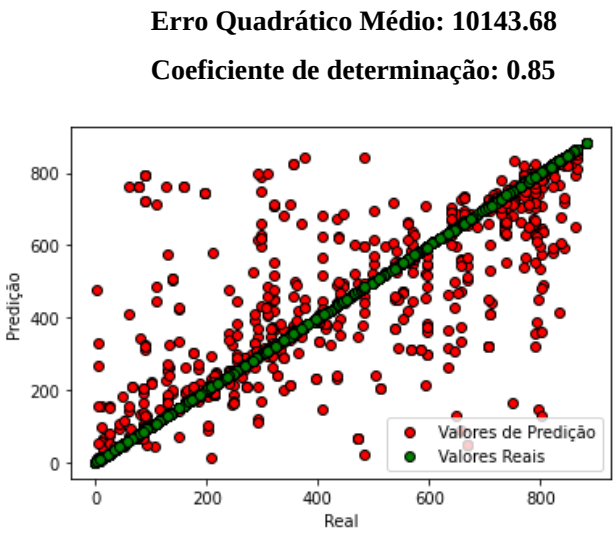
```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Lê o dataset de carros
carros = pd.read_csv("carros_modelos_categorical.csv")
X = carros.iloc[:, :-1].to_numpy()
y = carros.iloc[:, -1].to_numpy()
X = StandardScaler().fit_transform(X)

# Divide o conjunto em treinamento e teste, na proporção 80-20
X_treino,X_teste,y_treino,y_teste = train_test_split(X,y,  test_size=0.30, stratify=y,
random_state=123,shuffle=True)
knn_regr = KNeighborsRegressor(n_neighbors=3, weights='distance')
knn_regr.fit(X_treino, y_treino)
y_pred = knn_regr.predict(X_teste)
```

O desempenho do modelo pode ser avaliado por meio das medidas do Erro Quadrático Médio e do Coeficiente de Determinação, bem como por meio da plotagem dos valores de venda reais e aqueles obtidos pela predição.

Gráfico 13 – Desempenho do modelo




Fonte: Sousa, 2020.

## FINALIZANDO

Nesta aula, apresentamos métodos de classificação estatísticos, como o Naive Bayes, a Regressão Logística e a Regressão Linear. Alguns desses métodos possibilitam executar predições categóricas, bem como predições contínuas. Para entender o funcionamento e para quais problemas esses modelos são mais apropriados, é necessário entender o funcionamento dos métodos estatísticos nos quais eles se baseiam. Também é preciso praticar com diferentes tipos de dados e *datasets* distintos, bem como avaliar os resultados e como estes podem ser melhorados.

Além dos classificadores estatísticos, também vimos um classificador supervisionado cujas predições são baseadas em vizinhança: o kNN. Esse classificador também pode ser utilizado tanto para problemas de classificação quanto de regressão. O uso do classificador adequado sempre dependerá de diversos fatores que devem ser analisados com critério e cautela, mas na maioria dos casos, sempre será necessária uma exploração dos dados de treinamento, seus atributos, características e a correlação destas. Dessa maneira, é possível obter um modelo adequado à sua necessidade e com bom desempenho.

## REFERÊNCIAS


CUESTA, H. **Practical data analysis**. Birmingham: Packt Publishing Ltd, 2013.

GÉRON, A. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow**: Concepts, tools, and techniques to build intelligent systems. Sebastopol: O'Reilly Media, 2019.

MCKINNEY, W. **Python para análise de dados**: tratamento de dados com Pandas, NumPy e IPython. São Paulo: Novatec, 2019.

RICHERT, W. **Building machine learning systems with Python**. Birmingham: Packt Publishing Ltd, 2013.

RUSSEL, S. J.; NORVIG, P. **Inteligência Artificial**: uma abordagem moderna. 3. ed. Tradução de Regina Célia Simille. Rio de Janeiro: Elsevier, 2013.