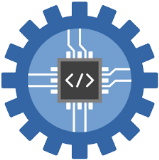


**UNIVERSIDADE FEDERAL DO PARÁ** 

FACULDADE DE ENGENHARIA DA COMPUTAÇÃO

CAMPUS UNIVERSITÁRIO DE TUCURUÍ – CAMTUC

José Henrique Rodrigues Crispim; (202033840045)

João Gomes Siqueira Neto; (201933840027)

Tendência (predição) de preços de veículos.

TUCURUÍ – PA 2024

Tendência (predição) de preços de veículos.

Relatório referente ao trabalho final sobre um modelo que preveja os preços de venda de veículos com base em suas características. Na disciplina de Mineração de Dados, ministrada pelo professor Dr. Iago Medeiros no curso de engenharia de computação.

TUCURUÍ – PA 2024

# **Introdução**

Já se perguntou como as lojas de veículos estimam o valor de um carro usado? Não é apenas um palpite — elas frequentemente usam algoritmos complexos e modelos de machine learning para prever o preço com base em diversos fatores como marca, modelo, ano, quilometragem e estado. Neste trabalho, vamos explorar como você pode construir seu próprio modelo de previsão de preço de carro usando um conjunto de dados e algumas técnicas populares de Machine Learning.

# **Exploração e Preparação de Dados**

Começaremos analisando um conjunto de dados de preços de carros usados que inclui informações como:

* Marca e modelo do carro
* Ano de fabricação
* Quilometragem
* Condição (excelente, bom, regular)
* Preço de venda

Usando bibliotecas Python como pandas e matplotlib, podemos explorar os dados para entender sua distribuição, identificar valores ausentes e visualizar relações entre diferentes características.

# **Construindo o Modelos de Previsão**

Agora vem a parte divertida — construir o modelo de machine learning! Vamos experimentar diferentes algoritmos como:

**Regressão Linear:** Este é um modelo linear simples que aprende a relação entre características e preço.

**Rede Neural:** Este modelo complexo com várias camadas de neurônios interconectados pode aprender relações não lineares complexas.

# **Importação e leitura da base dados**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

from sklearn.model\_selection import RandomizedSearchCV

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

from sklearn.neural\_network import MLPRegressor

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import mean\_squared\_error

from sklearn.pipeline import Pipeline

#--------------------------

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

# Lendo a base de dados

df = pd.read\_csv('carbase3.csv', encoding='latin1')

# **Visualização da base de dados**

# Visualizando os 5 primeiros registros

df.head()



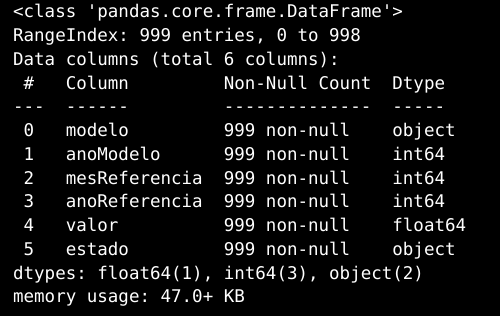
# Verificando os 5 últimos registros

df.tail()



# Informação sobre a tabela

df.info()



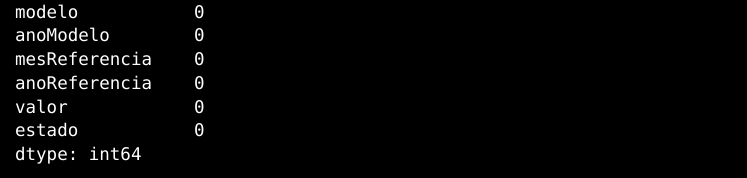
# Breve descrição da tabela

df.describe()



# Fazendo a soma dos valores nulos

df.isnull().sum()



# **Gráficos para análise**

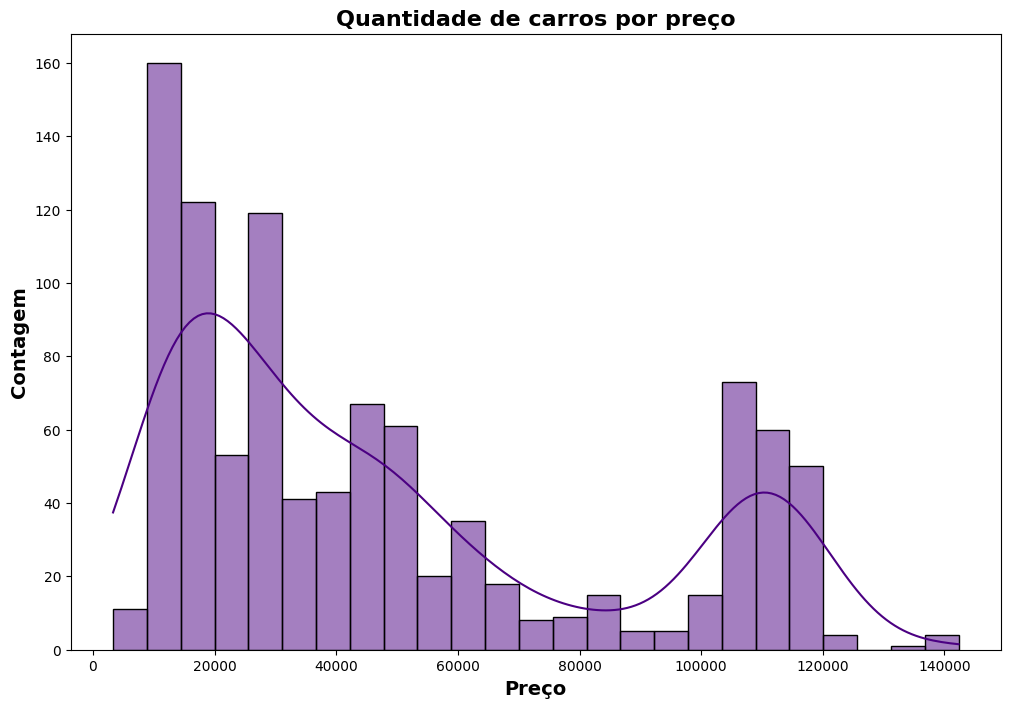
# Distribuição de preço

plt.figure(figsize=(12, 8))

sns.histplot(df['valor'], kde=True, bins=25, color='indigo')

plt.title('Quantidade de carros por preço', fontsize=16, fontweight='bold')

plt.xlabel('Preço', fontsize=14, fontweight='bold')

The he****

# Distribuição por anos

plt.figure(figsize=(10, 8)) # criando e definindo o tamanho da figura

df['anoModelo'].plot(kind='hist', bins=25, color= 'navy') # frequência da coluna 'year' e definindo cor

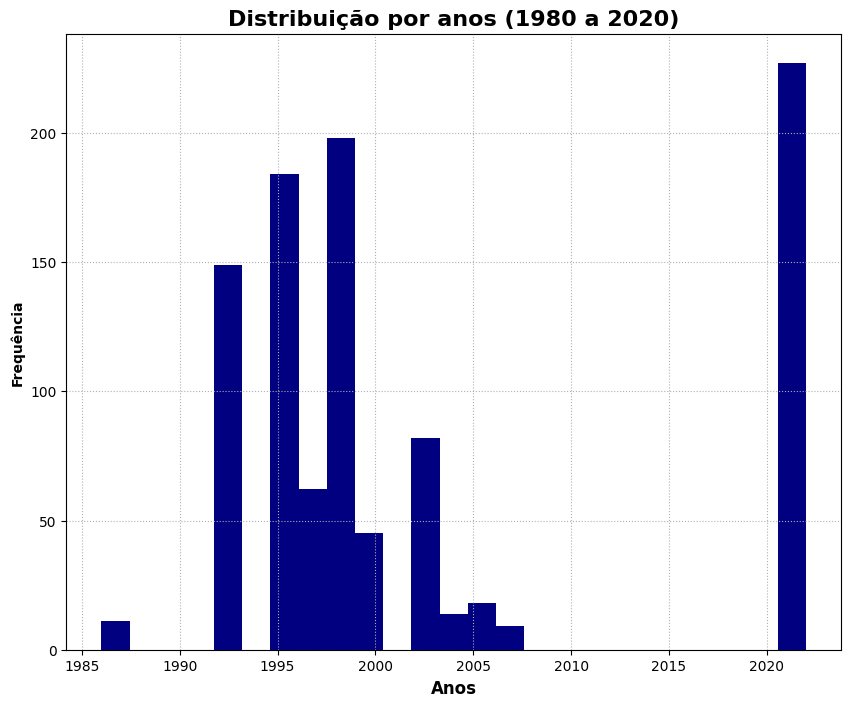
plt.title('Distribuição por anos (1980 a 2020)', fontsize= 16, fontweight='bold') # título

plt.grid(True, linestyle=':') # linha de divisão do gráfico

plt.ylabel('Frequência', fontweight= 'bold') # legenda y

plt.xlabel('Anos', fontweight= 'bold', fontsize= 12) # legenda x

plt.show(); # plotando o gráfico



# Gráfico scatter, Ano x Year, adicional = condição

plt.figure(figsize= (10, 8))

sns.scatterplot(data=df, x='anoReferencia', y='valor', hue='estado') # passando os parâmetros de visualização e comparação

plt.title('Relação Ano e Preço', fontsize = 16, fontweight= 'bold')

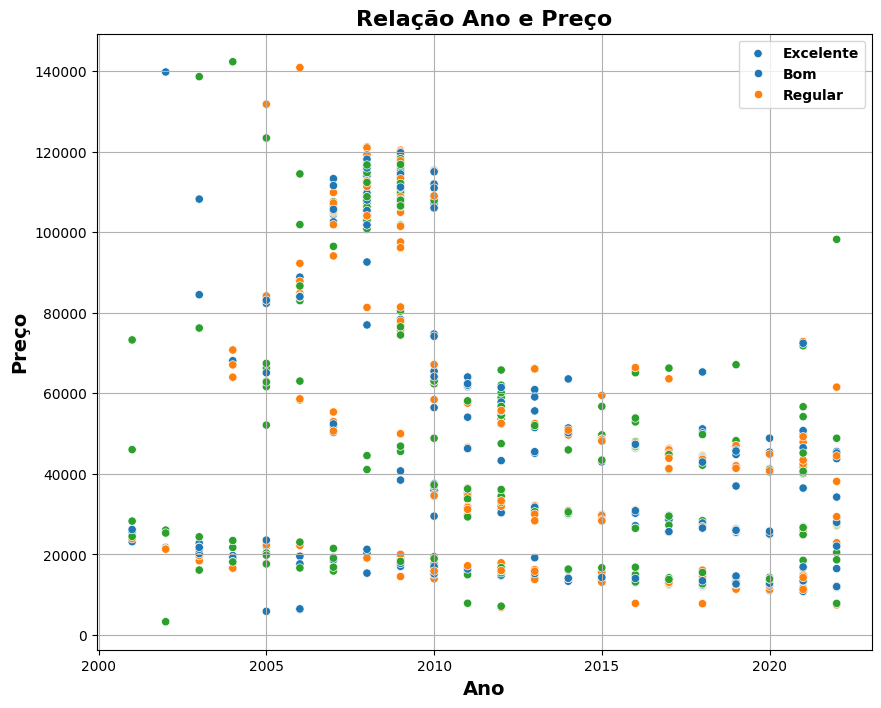
plt.xlabel('Ano', fontsize= 14, fontweight= 'bold')

plt.ylabel('Preço', fontsize= 14, fontweight= 'bold')

plt.legend(labels=['Excelente', 'Bom', 'Regular'], loc='upper right', fontsize= 'medium', prop={'weight':'bold'}) # legenda das condições

plt.grid(True, linestyle= '-') # mostrando e alterando estilo da figura

plt.show()



# Contagem da condição dos veículos

contagem = df['estado'].value\_counts() # variável que recebe a contagem da condição dos veículos

cores = ['darkslateblue', 'firebrick', 'forestgreen']

fig, ax = plt.subplots(figsize=(10, 8))

barras = ax.bar(contagem.index, contagem, color=cores) # criando as barras

plt.title('Contagem da condição dos veículos', fontsize=16, fontweight='bold')

plt.xlabel('Estado', fontsize=14, fontweight='bold')

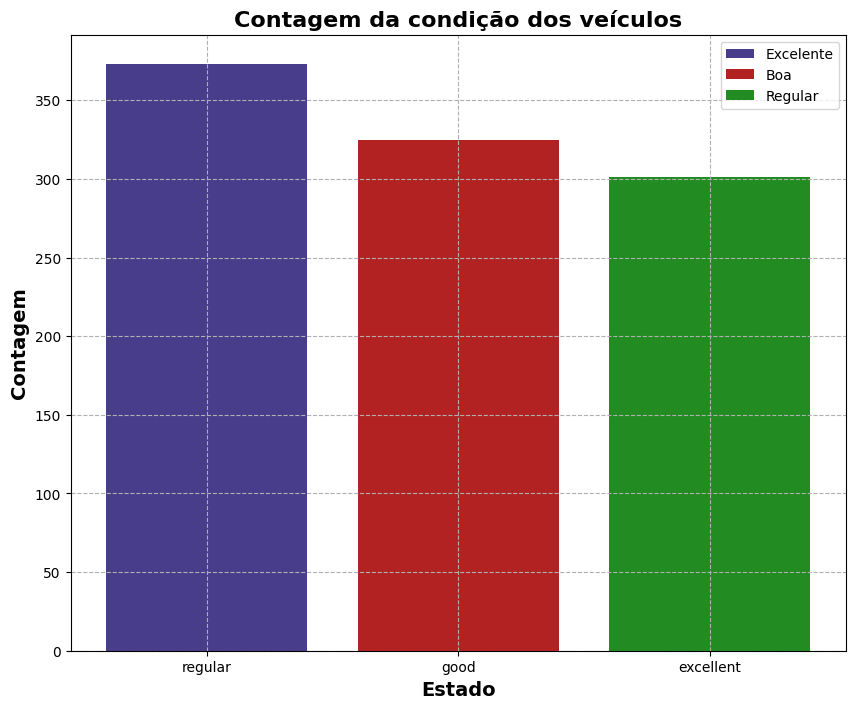
plt.ylabel('Contagem', fontsize=14, fontweight='bold')

legendas = ['Excelente', 'Boa', 'Regular']

plt.legend(barras, legendas, loc='upper right', fontsize='medium') # descrição

plt.grid(True, linestyle='--') # mostrando e defindo estilo do grid

plt.show();



# Verificar e tratar valores faltantes

df = df.dropna()

# Separar features e target

X = df.drop(columns=['valor', 'modelo'])

y = df['valor']

# Definir pré-processamento para variáveis categóricas e numéricas

categorical\_features = ['estado']

numeric\_features = ['anoReferencia', 'anoModelo']

# Criar transformers para pré-processamento

numeric\_transformer = StandardScaler()

categorical\_transformer = OneHotEncoder()

# Combinar transformers em um ColumnTransformer

preprocessor = ColumnTransformer(

transformers=[

('num', numeric\_transformer, numeric\_features),

('cat', categorical\_transformer, categorical\_features)

])

# Criar pipeline com pré-processador e modelo de Regressão Linear

model=Pipeline(steps=[('preprocessor', preprocessor),('regressor', LinearRegression())])

# Dividir os dados em conjuntos de treino e teste

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Treinar o modelo

model.fit(X\_train, y\_train)

# Fazer previsões

y\_pred = model.predict(X\_test)

# Avaliar o modelo

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = mse \*\* 0.5

print(f'Root Mean Squared Error: {rmse}')



# **Pré-processamento dos Dados**

Antes de alimentar os dados em nossos modelos de machine learning, precisamos prepará-los de forma adequada. Isso pode envolver:

* Lidar com valores ausentes: Podemos imputar valores ausentes com estimativas razoáveis ou remover linhas com informações ausentes.
* Codificar variáveis categóricas: Características categóricas como marca e modelo precisam ser convertidas em valores numéricos que os algoritmos podem entender. Isso pode ser feito usando técnicas como codificação one-hot.
* Escalar características numéricas: Podemos precisar escalar características numéricas como quilometragem e ano para uma faixa comum para evitar que dominem o processo de treinamento.

# **# Pré-processamento dos dados**

# **# Codificar colunas categóricas**

# **label\_encoder = LabelEncoder()**

# **df['modelo'] = label\_encoder.fit\_transform(df['modelo'])**

# **df['estado'] = label\_encoder.fit\_transform(df['estado'])**

# **# Selecionar features e target**

# **X = df[['modelo', 'anoModelo', 'mesReferencia', 'anoReferencia', 'estado']]**

# **y = df['valor']**

# **# Dividir os dados em treino e teste**

# **X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

# **# Escalar os dados**

# **scaler = StandardScaler()**

# **X\_train = scaler.fit\_transform(X\_train)**

# **X\_test = scaler.transform(X\_test)**

# **# Construir a rede neural**

# **model = Sequential()**

# **model.add(Dense(64, input\_dim=X\_train.shape[1], activation='relu'))**

# **model.add(Dense(32, activation='relu'))**

# **model.add(Dense(1))**

# **# Compilar o modelo**

# **model.compile(optimizer='adam', loss='mean\_squared\_error')**

# **# Treinar o modelo**

# **history = model.fit(X\_train, y\_train, epochs=100, batch\_size=32, validation\_split=0.2)**

# **# Avaliar o modelo**

# **loss = model.evaluate(X\_test, y\_test)**

# **print(f'Mean Squared Error on Test Set: {loss}')**

# **# Fazer previsões**

# **y\_pred = model.predict(X\_test)**

# **# Avaliação do desempenho**

# **from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score**

# **mae = mean\_absolute\_error(y\_test, y\_pred)**

# **mse = mean\_squared\_error(y\_test, y\_pred)**

# **r2 = r2\_score(y\_test, y\_pred)**

# **print(f'Mean Absolute Error: {mae}')**

# **print(f'Mean Squared Error: {mse}')**

# **print(f'R^2 Score: {r2}')**

# **Epoch 1/100**

# **/home/henrique/.local/lib/python3.10/site-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input\_shape`/`input\_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.**

# **super().\_\_init\_\_(activity\_regularizer=activity\_regularizer, \*\*kwargs)**

# **2024-07-03 07:58:38.747678: E external/local\_xla/xla/stream\_executor/cuda/cuda\_driver.cc:282] failed call to cuInit: CUDA\_ERROR\_UNKNOWN: unknown error**

# **2024-07-03 07:58:38.747740: I external/local\_xla/xla/stream\_executor/cuda/cuda\_diagnostics.cc:134] retrieving CUDA diagnostic information for host: henrique-Aspire-A515-54G**

# **2024-07-03 07:58:38.747747: I external/local\_xla/xla/stream\_executor/cuda/cuda\_diagnostics.cc:141] hostname: henrique-Aspire-A515-54G**

# **2024-07-03 07:58:38.747828: I external/local\_xla/xla/stream\_executor/cuda/cuda\_diagnostics.cc:165] libcuda reported version is: 390.157.0**

# **2024-07-03 07:58:38.747844: I external/local\_xla/xla/stream\_executor/cuda/cuda\_diagnostics.cc:169] kernel reported version is: 390.157.0**

# **2024-07-03 07:58:38.747848: I external/local\_xla/xla/stream\_executor/cuda/cuda\_diagnostics.cc:248] kernel version seems to match DSO: 390.157.0**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m1s[0m 7ms/step - loss: 3737837056.0000 - val\_loss: 3774401280.0000**

# **Epoch 2/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3549829888.0000 - val\_loss: 3774235392.0000**

# **Epoch 3/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3557674240.0000 - val\_loss: 3773947136.0000**

# **Epoch 4/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3711117312.0000 - val\_loss: 3773463040.0000**

# **Epoch 5/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3948166912.0000 - val\_loss: 3772676864.0000**

# **Epoch 6/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3591357952.0000 - val\_loss: 3771486976.0000**

# **Epoch 7/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 4093284096.0000 - val\_loss: 3769742080.0000**

# **Epoch 8/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3746003200.0000 - val\_loss: 3767325184.0000**

# **Epoch 9/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3651518208.0000 - val\_loss: 3764046336.0000**

# **Epoch 10/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3934388224.0000 - val\_loss: 3759639808.0000**

# **Epoch 11/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3746252032.0000 - val\_loss: 3754132224.0000**

# **Epoch 12/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3583279872.0000 - val\_loss: 3747452928.0000**

# **Epoch 13/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3660228864.0000 - val\_loss: 3739202304.0000**

# **Epoch 14/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3650897152.0000 - val\_loss: 3729186048.0000**

# **Epoch 15/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3697358848.0000 - val\_loss: 3717614080.0000**

# **Epoch 16/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3502410240.0000 - val\_loss: 3704233216.0000**

# **Epoch 17/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3687860480.0000 - val\_loss: 3688832512.0000**

# **Epoch 18/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3715737344.0000 - val\_loss: 3671466752.0000**

# **Epoch 19/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3708703744.0000 - val\_loss: 3651949056.0000**

# **Epoch 20/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3828226304.0000 - val\_loss: 3630436864.0000**

# **Epoch 21/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3583979776.0000 - val\_loss: 3606805248.0000**

# **Epoch 22/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3483200512.0000 - val\_loss: 3580611328.0000**

# **Epoch 23/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3427299072.0000 - val\_loss: 3551730176.0000**

# **Epoch 24/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3533619712.0000 - val\_loss: 3520170752.0000**

# **Epoch 25/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3672698368.0000 - val\_loss: 3486385664.0000**

# **Epoch 26/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3576204288.0000 - val\_loss: 3450423808.0000**

# **Epoch 27/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3352524288.0000 - val\_loss: 3412840704.0000**

# **Epoch 28/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3457146368.0000 - val\_loss: 3371068672.0000**

# **Epoch 29/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3251451392.0000 - val\_loss: 3327988224.0000**

# **Epoch 30/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3097169920.0000 - val\_loss: 3281952000.0000**

# **Epoch 31/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3159642368.0000 - val\_loss: 3232616960.0000**

# **Epoch 32/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3085700864.0000 - val\_loss: 3181745664.0000**

# **Epoch 33/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2986487040.0000 - val\_loss: 3128128768.0000**

# **Epoch 34/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3151283968.0000 - val\_loss: 3070854656.0000**

# **Epoch 35/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2987121408.0000 - val\_loss: 3012699392.0000**

# **Epoch 36/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 3106669568.0000 - val\_loss: 2951694336.0000**

# **Epoch 37/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2789952256.0000 - val\_loss: 2889050624.0000**

# **Epoch 38/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2806384896.0000 - val\_loss: 2825292544.0000**

# **Epoch 39/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2750510592.0000 - val\_loss: 2758617088.0000**

# **Epoch 40/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2671890688.0000 - val\_loss: 2690732800.0000**

# **Epoch 41/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2686213888.0000 - val\_loss: 2621048064.0000**

# **Epoch 42/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2688249856.0000 - val\_loss: 2548362240.0000**

# **Epoch 43/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2352349440.0000 - val\_loss: 2478119424.0000**

# **Epoch 44/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2300858112.0000 - val\_loss: 2404741632.0000**

# **Epoch 45/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2459266816.0000 - val\_loss: 2328184064.0000**

# **Epoch 46/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2415030528.0000 - val\_loss: 2252056832.0000**

# **Epoch 47/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2161583616.0000 - val\_loss: 2177231616.0000**

# **Epoch 48/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2322291456.0000 - val\_loss: 2100945152.0000**

# **Epoch 49/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2136200320.0000 - val\_loss: 2024665472.0000**

# **Epoch 50/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 2069087360.0000 - val\_loss: 1949015680.0000**

# **Epoch 51/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 1925572480.0000 - val\_loss: 1873554432.0000**

# **Epoch 52/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 1740545024.0000 - val\_loss: 1797879808.0000**

# **Epoch 53/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 1667874432.0000 - val\_loss: 1723275264.0000**

# **Epoch 54/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 1605421056.0000 - val\_loss: 1648681088.0000**

# **Epoch 55/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 3ms/step - loss: 1520181120.0000 - val\_loss: 1575549952.0000**

# **Epoch 56/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 1468735872.0000 - val\_loss: 1501881088.0000**

# **Epoch 57/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 1422382848.0000 - val\_loss: 1431345280.0000**

# **Epoch 58/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 1389005312.0000 - val\_loss: 1361026432.0000**

# **Epoch 59/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 1335721344.0000 - val\_loss: 1292102912.0000**

# **Epoch 60/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 1246709888.0000 - val\_loss: 1226293120.0000**

# **Epoch 61/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 1157934592.0000 - val\_loss: 1163656192.0000**

# **Epoch 62/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 1071623616.0000 - val\_loss: 1101500544.0000**

# **Epoch 63/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 1063613376.0000 - val\_loss: 1040754368.0000**

# **Epoch 64/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 1044875200.0000 - val\_loss: 982785536.0000**

# **Epoch 65/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 933779584.0000 - val\_loss: 928492864.0000**

# **Epoch 66/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 851507264.0000 - val\_loss: 875865088.0000**

# **Epoch 67/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 3ms/step - loss: 837794240.0000 - val\_loss: 825753280.0000**

# **Epoch 68/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 813768896.0000 - val\_loss: 777744960.0000**

# **Epoch 69/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 731737984.0000 - val\_loss: 733060480.0000**

# **Epoch 70/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 696515264.0000 - val\_loss: 691173760.0000**

# **Epoch 71/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 669611776.0000 - val\_loss: 652096192.0000**

# **Epoch 72/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 644487360.0000 - val\_loss: 614028992.0000**

# **Epoch 73/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 599890560.0000 - val\_loss: 578483200.0000**

# **Epoch 74/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 530948608.0000 - val\_loss: 546837632.0000**

# **Epoch 75/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 536259232.0000 - val\_loss: 516607072.0000**

# **Epoch 76/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 499311040.0000 - val\_loss: 488718080.0000**

# **Epoch 77/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 522445120.0000 - val\_loss: 463233216.0000**

# **Epoch 78/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 431224160.0000 - val\_loss: 440249088.0000**

# **Epoch 79/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 396168128.0000 - val\_loss: 419104352.0000**

# **Epoch 80/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 383510368.0000 - val\_loss: 399566752.0000**

# **Epoch 81/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 384459744.0000 - val\_loss: 381387104.0000**

# **Epoch 82/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 364914560.0000 - val\_loss: 365293312.0000**

# **Epoch 83/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 321753248.0000 - val\_loss: 350900384.0000**

# **Epoch 84/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 350609856.0000 - val\_loss: 337574848.0000**

# **Epoch 85/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 365327136.0000 - val\_loss: 325704608.0000**

# **Epoch 86/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 335656160.0000 - val\_loss: 314720928.0000**

# **Epoch 87/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 319131392.0000 - val\_loss: 305357888.0000**

# **Epoch 88/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 322016192.0000 - val\_loss: 296713152.0000**

# **Epoch 89/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 293271296.0000 - val\_loss: 288578496.0000**

# **Epoch 90/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 291116800.0000 - val\_loss: 281442144.0000**

# **Epoch 91/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 250845088.0000 - val\_loss: 275318688.0000**

# **Epoch 92/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 256908272.0000 - val\_loss: 269326304.0000**

# **Epoch 93/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 3ms/step - loss: 247337408.0000 - val\_loss: 264139312.0000**

# **Epoch 94/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 234877584.0000 - val\_loss: 259256880.0000**

# **Epoch 95/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 239009664.0000 - val\_loss: 254760656.0000**

# **Epoch 96/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 275011712.0000 - val\_loss: 250703520.0000**

# **Epoch 97/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 259171616.0000 - val\_loss: 246997968.0000**

# **Epoch 98/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 257827360.0000 - val\_loss: 243508992.0000**

# **Epoch 99/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 246279232.0000 - val\_loss: 240364112.0000**

# **Epoch 100/100**

# **[1m20/20[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 2ms/step - loss: 229585968.0000 - val\_loss: 237348608.0000**

# **[1m7/7[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 892us/step - loss: 304191488.0000**

# **Mean Squared Error on Test Set: 289766304.0**

# **[1m7/7[0m [32m━━━━━━━━━━━━━━━━━━━━[0m[37m[0m [1m0s[0m 5ms/step**

# **Mean Absolute Error: 11915.088658447266**

# **Mean Squared Error: 289766300.0999239**

# **R^2 Score: 0.7809809184506453**

# 

# 

# **Conclusão**

Prever o preço de carros com machine learning é uma aplicação fascinante dessas técnicas poderosas. Ao seguir os passos descritos neste trabalho, você pode obter informações valiosas sobre preços de carros e até mesmo construir seu próprio modelo de previsão!