Instituto Superior de Tecnologias Avançadas de Lisboa

Robótica e Inteligência Artificial

Relatório de Projeto - Ciência de Dados

Lisboa

Junho 2024

André Costa Orta - 073

Duarte Moreno Martins de Lage Arjones - 314

Ricardo Cristino da Costa Lima – 316

Samuel dos Santos Fidalgo de Freitas - 312

Índice

Introdução

Este projeto foi realizado no âmbito da disciplina de Ciência de dados e tem como principal objetivo perceber o desempenho e eficácia de alguns dos algoritmos de Machine Learning através do uso de técnicas de manipulação, interpretação e visualização de dados.

Devido ao crescimento exponencial de aplicações e plataformas que utilizam quantidades de dados em grande escala, torna-se essencial perceber o que realmente podemos extrair dos mesmos e de que forma é que através da sua análise, tratamento e aplicabilidade de técnicas avançadas de manipulação dos mesmos, conseguimos retirar insights valiosos e tornar processos muito mais eficientes. Por esses mesmos motivos, foi feito o estudo de dois algoritmos de Machine Learning e de várias técnicas de análise de dados aplicados a um conjunto de dados relacionados a preços de carros, a fim de perceber qual revela um melhor desempenho e uma melhor previsão para o futuro preço dos demais carros.

Assim, este projeto tem como objetivos:

- Perceber a nível técnico a aplicabilidade das técnicas de Ciência de Dados, de forma a aprofundar o conhecimento na área;

- Compreender o papel ativo e importante que a Ciência de Dados desempenha nos dias de hoje na evolução dos demais setores das organizações e empresas;

- Entender o impacto e as implicações da área no futuro da humanidade, e inspirar a sua reflexão;

Fundamentação técnico-científica

Visual Studio Code

O Visual Studio Code é um IDE (ambiente de desenvolvimento integrado) leve e poderoso, que suporta várias linguagens de programação assim como outro tipo de tecnologias. Ainda, inclui várias extensões que aumentam o seu desempenho, o que o torna numa ferramenta excecional para o desenvolvimento de software.

Python

O Python é uma linguagem de programação de alto nível conhecida pela sua simplicidade e sintaxe clara. É muito utilizada em áreas como o desenvolvimento web, automação, análise de dados e ciência de dados.

Ciência de Dados

A Ciência de Dados é o campo científico que utiliza métodos estatísticos, processos e sistemas informáticos para extrair conhecimento e insights de dados estruturados e não estruturados. Para isso, são utilizadas técnicas de Machine Learning, visualização de dados e big data.

Machine Learning

O Machine Learning é o subcampo da Ciência de dados focado no desenvolvimento de algoritmos que permitem que os computadores aprendam a partir de dados e façam previsões ou decisões baseadas nos mesmos.

Gradient Boosting

O Gradient Boosting é um dos algoritmos de Machine Learning que consiste na junção de vários algoritmos mais fracos num só (assemble), de forma a torná-lo otimizado e eficiente. Cada semi-modelo corrige os erros dos anteriores, resultando num super-algoritmo robusto e preciso.

Random Forest

O Random Forest é um dos algoritmos de Machine Learning que utiliza múltiplas árvores de decisão. Cada árvore é treinada com um subconjunto aleatório dos dados. As previsões são feitas com base na média ou maioria das previsões das árvores, aumentando a precisão e reduzindo o risco de overfitting.

Sobre o Dataset

O " Vehicle Sales Data" compila uma vasta gama de informações sobre transações de venda de veículos. Este conjunto inclui detalhes como ano, marca, modelo, versão, tipo de carroçaria, tipo de transmissão, VIN (Número de Identificação do Veículo), estado de registo, classificação de condição, leitura do odómetro, cores exterior e interior, informações do vendedor, valores do Manheim Market Report (MMR), preços de venda e datas de venda.

Principais Características:

Informações do Veículo: Contém detalhes específicos de cada veículo, como a marca, o modelo, a versão e o ano de fabricação.

Dados da Transação: Inclui informações sobre as transações de venda, como preços de venda e datas em que ocorreram.

Análise de Mercado: Os valores MMR proporcionam uma estimativa do valor de mercado de cada veículo, possibilitando a análise de tendências e flutuações de mercado.

Condição e Quilometragem: Fornece dados sobre a condição dos veículos e as leituras dos odómetros, permitindo analisar como esses fatores afetam os preços de venda.

Possíveis Aplicações:

Estudo de Mercado: Investigadores e analistas podem utilizar este conjunto de dados para estudar as tendências no mercado automóvel, incluindo variações de preços com base na condição e quilometragem dos veículos.

Modelos Preditivos: Cientistas de dados podem usar este conjunto para criar modelos que estimem os preços dos veículos com base em diferentes características.

Insights de Negócios: Profissionais da indústria automóvel, concessionários e instituições financeiras podem obter informações sobre as preferências dos consumidores, a procura no mercado e estratégias de preços.

Formato:

O conjunto de dados é apresentado em formato tabular (CSV), onde cada linha representa uma transação de venda de veículo e cada coluna representa um atributo diferente associado a essa transação.

Desenvolvimento

O nosso projeto está dividido em seis partes, estas são: análise de dados e variáveis, visualização dos dados, limpeza dos dados, preparação dos dados, aplicação dos modelos e por último a visualização dos resultados.

Primeiro Passo – Importação de bibliotecas e análise de dados

Demos início ao nosso projeto com a importação das principais bibliotecas, classes e funções, como podemos observar na imagem abaixo (figura 1):

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import GradientBoostingRegressor
```

Figura 1

Após a importação, passamos então à leitura e à exibição dos dados do dataset que escolhemos para trabalhar. Na imagem abaixo (figura 2) podemos observar as linhas de código que usamos para que esta leitura e exibição dos dados do dataset seja possível.

```
df = pd.read_csv("car_prices.csv")

print(df.head())
print(df.info())
print(df.describe())

print("\nValores nulos ou ausentes por coluna:")
print(df.isnull().sum())
```

Figura 2

Nesta imagem (figura 3) podemos observar a estrutura do nosso dataset:

```
   year   make                model        trim   body transmission  \
0  2015    Kia              Sorento          LX    SUV    automatic
1  2015    Kia              Sorento          LX    SUV    automatic
2  2014    BMW             3 Series  328i SULEV  Sedan    automatic
3  2015  Volvo                  S60          T5  Sedan    automatic
4  2014    BMW  6 Series Gran Coupe        650i  Sedan    automatic

                vin state  condition  odometer  color interior  \
0  5xyktca69fg566472    ca        5.0   16639.0  white    black
1  5xyktca69fg561319    ca        5.0    9393.0  white    beige
2  wba3c1c51ek116351    ca       45.0    1331.0   gray    black
3  yv1612tb4f1310987    ca       41.0   14282.0  white    black
4  wba6b2c57ed129731    ca       43.0    2641.0   gray    black

                                  seller      mmr  sellingprice  \
0                  kia motors america  inc  20500.0       21500.0
1                  kia motors america  inc  20800.0       21500.0
2  financial services remarketing (lease)  31900.0       30000.0
3                  volvo na rep/world omni  27500.0       27750.0
4  financial services remarketing (lease)  66000.0       67000.0

                         saledate
0  Tue Dec 16 2014 12:30:00 GMT-0800 (PST)
1  Tue Dec 16 2014 12:30:00 GMT-0800 (PST)
2  Thu Jan 15 2015 04:30:00 GMT-0800 (PST)
3  Thu Jan 29 2015 04:30:00 GMT-0800 (PST)
4  Thu Dec 18 2014 12:30:00 GMT-0800 (PST)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 558837 entries, 0 to 558836
Data columns (total 16 columns):
#   Column        Non-Null Count   Dtype
---  ------        --------------   -----
0   year          558837 non-null  int64
1   make          548536 non-null  object
2   model         548438 non-null  object
3   trim          548186 non-null  object
4   body          545642 non-null  object
5   transmission  493485 non-null  object
6   vin           558833 non-null  object
7   state         558837 non-null  object
8   condition     547017 non-null  float64
9   odometer      558743 non-null  float64
10  color         558088 non-null  object
11  interior      558088 non-null  object
12  seller        558837 non-null  object
13  mmr           558799 non-null  float64
14  sellingprice  558825 non-null  float64
15  saledate      558825 non-null  object
```

Figura 3

Após isto podemos então passar à análise das variáveis, como podemos verificar na seguinte imagem (figura 4), os nomes das variáveis, o tipo, e as suas descrições.

Nome	Tipo	Descrição
year	int64	Ano de fabricação do veículo
make	object	Marca do veículo
model	object	Modelo do veículo
trim	object	Versão do modelo do veículo
body	object	Tipo de carroceria do veículo
transmission	object	Tipo de transmissão do veículo
vin	object	Número de Identificação do Veículo (VIN)
state	object	Estado de registro do veículo
condition	float64	Classificação da condição do veículo
odometer	float64	Quilometragem do veículo
color	object	Cor exterior do veículo
interior	object	Cor interior do veículo
seller	object	Informação do vendedor
mmr	float64	Valores do Manheim Market Report
sellingprice	float64	Preço de venda do veículo
saledate	object	Data de venda do veículo

Figura 4

Com isto damos por terminada a primeira etapa do nosso projeto. Passamos então ao segundo passo (visualização dos dados).

Segundo Passo – Visualização de Dados

Começamos por visualizar a contagem de vendas por ano. Para que isso seja possível executamos as linhas de código presentes na seguinte imagem (figura 5):

```
df.groupby("year").size().plot(kind="bar")
plt.title("Contagem de Vendas por Ano")
plt.xlabel("Ano")
plt.ylabel("Contagem de Vendas")
plt.show()
```

Figura 5

Estas linhas de código permitem-nos visualizar um gráfico de barras (figura 6), que nos mostra o número de vendas de carros (ou registos no dataset) para cada ano presente na coluna *year*. Podemos também verificar que a maior parte das vendas foram realizadas entre 2012 e 2014.

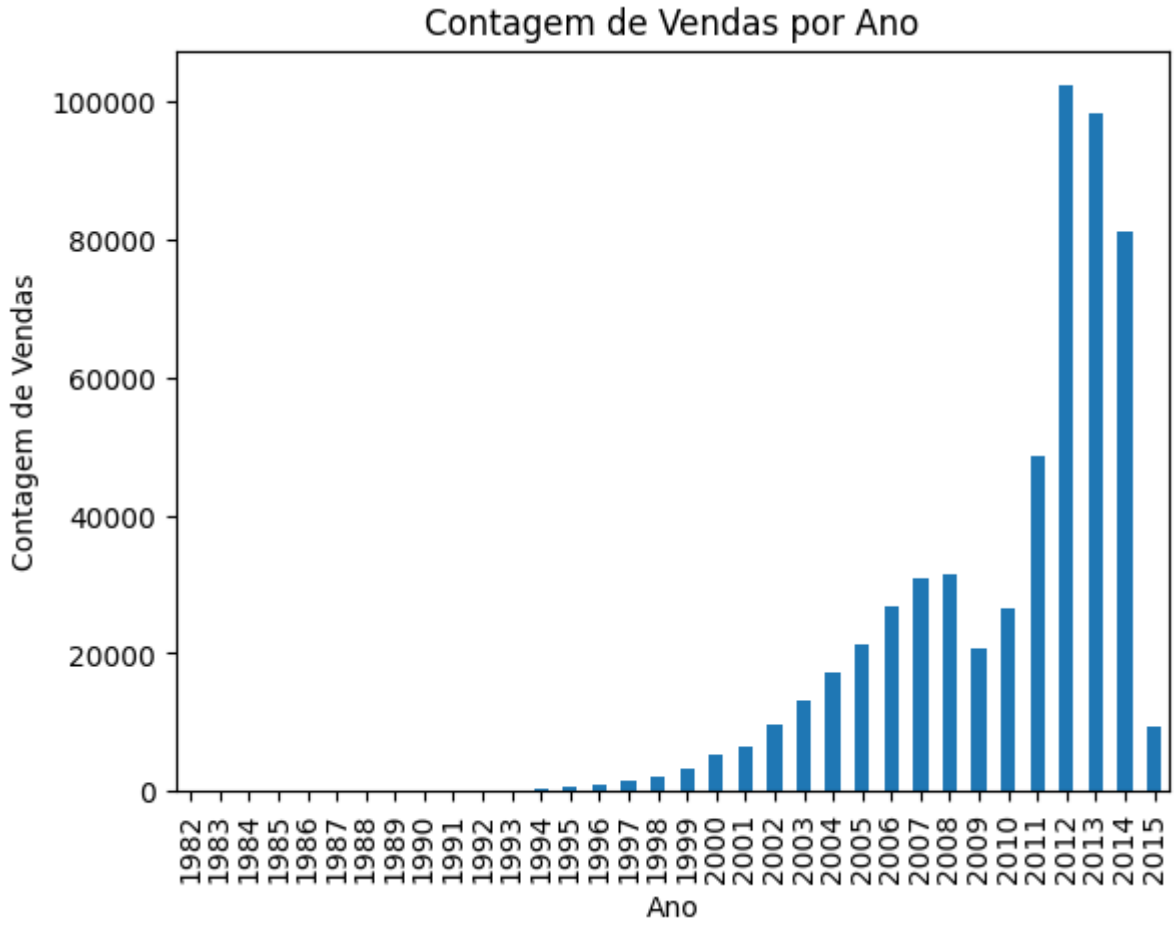


Figura 6

Após a visualização da contagem de vendas por ano, visualizamos a distribuição dos preços de venda dos carros.

Para que este gráfico seja visualizado com sucesso executamos as seguintes linhas de código (figura 7):

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

plt.scatter(df.index, df["sellingprice"], alpha=0.5)
plt.title("Distribuição dos Preços de Venda dos Carros")
plt.xlabel("Índice do Registro")
plt.ylabel("Preço de Venda")
plt.show()
```

Figura 7

O código presente na imagem acima (figura 7) fornece-nos um gráfico de dispersão que mostra a distribuição dos preços de venda dos carros ao longo dos registos no dataset.

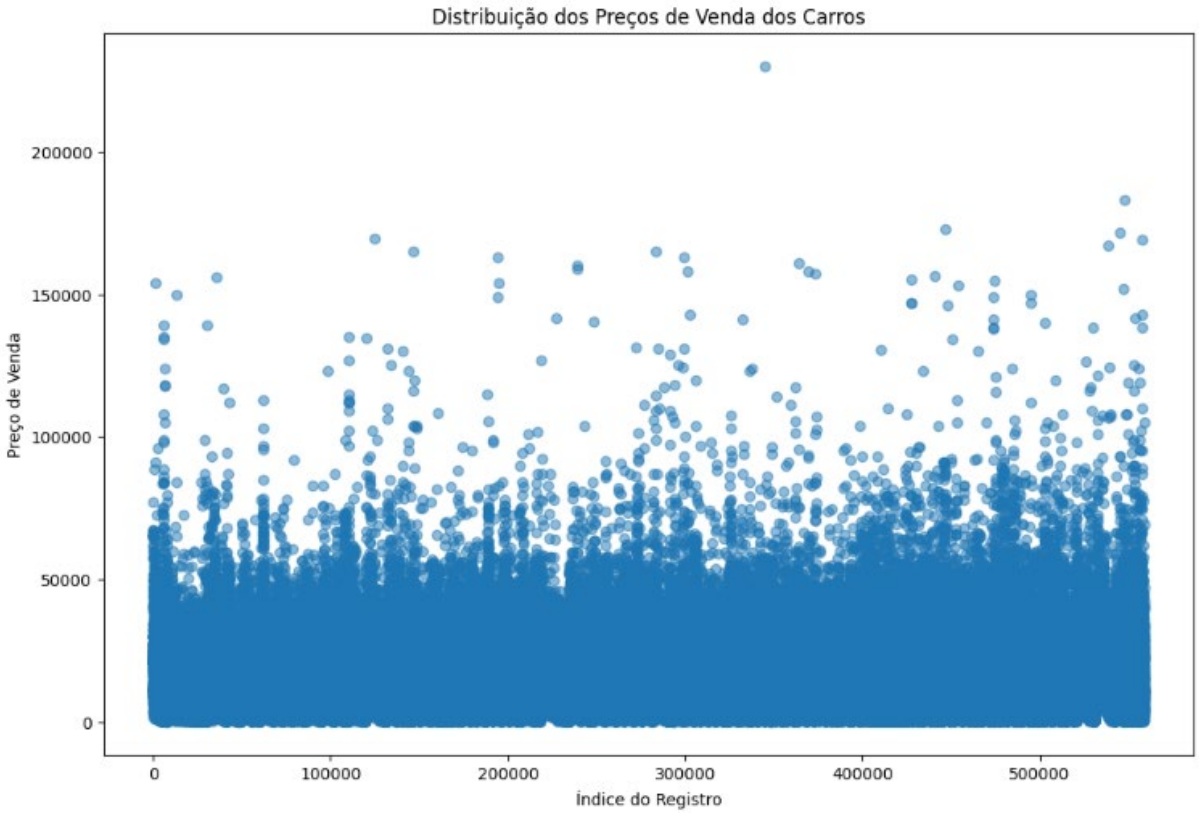


Figura 8

Este gráfico mostra-nos a distribuição dos preços de venda dos carros ao longo dos registos no dataset.

Através deste gráfico podemos concluir que a maior parte dos carros foram vendidos na faixa do 0 aos 50k.

Depois de visualizarmos a distribuição dos preços de venda dos carros e a contagem de vendas por ano, visualizamos então a distribuição das categorias de preço por tipo de transmissão.

Para proceder á visualização executamos as seguintes linhas de código (figura 9):

```
# Filtrar o DataFrame para incluir apenas 'manual' e 'automatic' na coluna 'transmission'
df_filtered = df[df["transmission"].isin(["manual", "automatic"])].copy()

# Categorizar os preços de venda em 5 categorias com intervalos de 15.000
bins = [0, 15000, 30000, 45000, 60000, float("inf")]
labels = ["0-15k", "15k-30k", "30k-45k", "45k-60k", "60k+"]
df_filtered["price_category"] = pd.cut(df_filtered["sellingprice"], bins=bins, labels=labels, include_lowest=True)

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

# Contar as combinações de categorias de preços e tipos de transmissão
counts = df_filtered.groupby(["transmission", "price_category"], observed=False).size().reset_index(name="count")

sns.barplot(x="count", y="price_category", hue="transmission", data=counts, palette="viridis")
plt.title("Distribuição das Categorias de Preço por Tipo de Transmissão")
plt.xlabel("Contagem")
plt.ylabel("Categoria de Preço")
plt.legend(title="Tipo de Transmissão")
plt.show()
```

Figura 9

Com este código podemos observar um gráfico de barras (figura 10) onde conseguimos observar a distribuição das categorias de preço por tipo de transmissão. O código inclui apenas os carros com transmissão *manual* e *automatic*, categoriza os preços de venda em 5 intervalos (60k, 45k-60k, 30k-45k, 15k-30k, 0-15k).

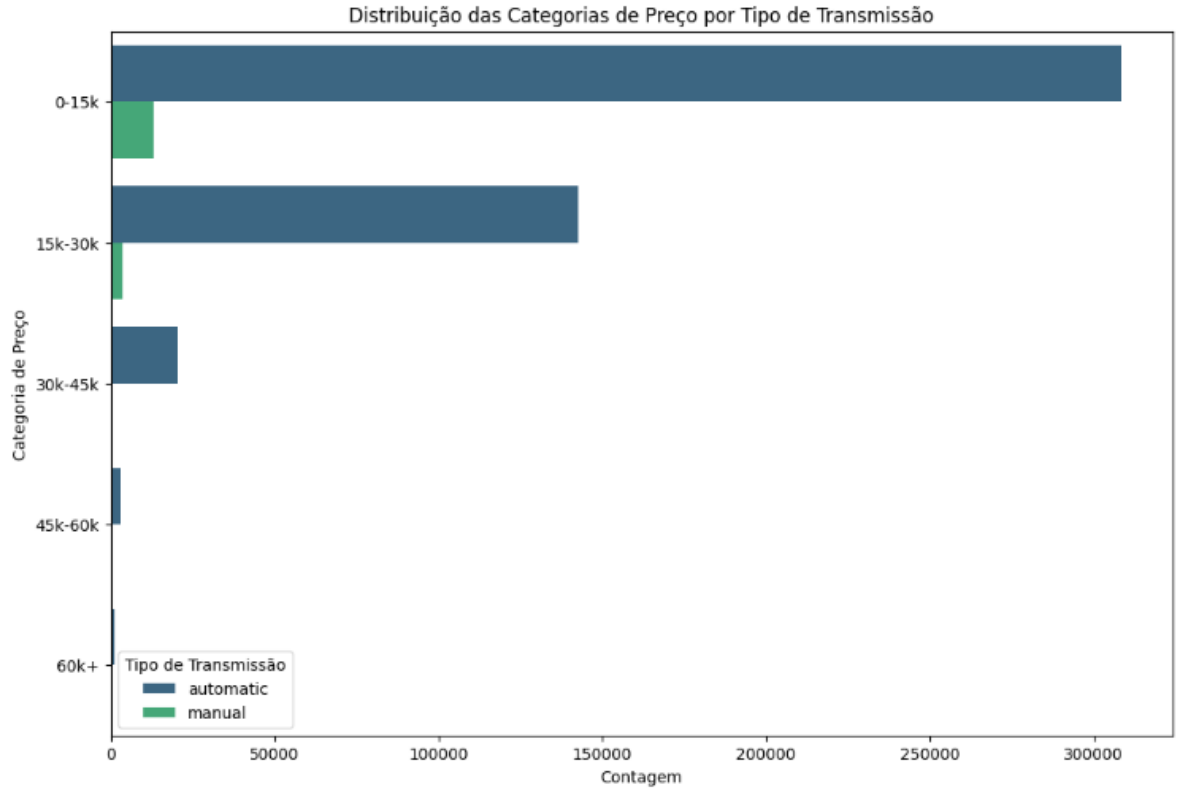


Figura 10

Neste gráfico (figura 10) podemos verificar que a maior parte dos carros vendidos são automáticos no intervalo dos 0-15k.

Finalizamos a visualização de dados com uma matriz de correlação.

Esta permite-nos identificar relações entre variáveis, isto é, correlações fortes (valores próximos de 1 ou -1 indicam uma forte relação linear positiva ou negativa) e correlações fracas (valores próximos de 0 indicam pouca ou nenhuma relação linear).

Executamos as linhas de código abaixo (figura 11) para que consigamos observar a nossa matriz de correlação:

```
# Seleciona apenas colunas numéricas
numeric_df = df.select_dtypes(include=[np.number])

# Calcula a matriz de correlação
correlation_matrix = numeric_df.corr()

# Visualização da matriz de correlação
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Matriz de Correlação")
plt.show()
```

Figura 11

Resultado da nossa matriz de correlação (figura 12):

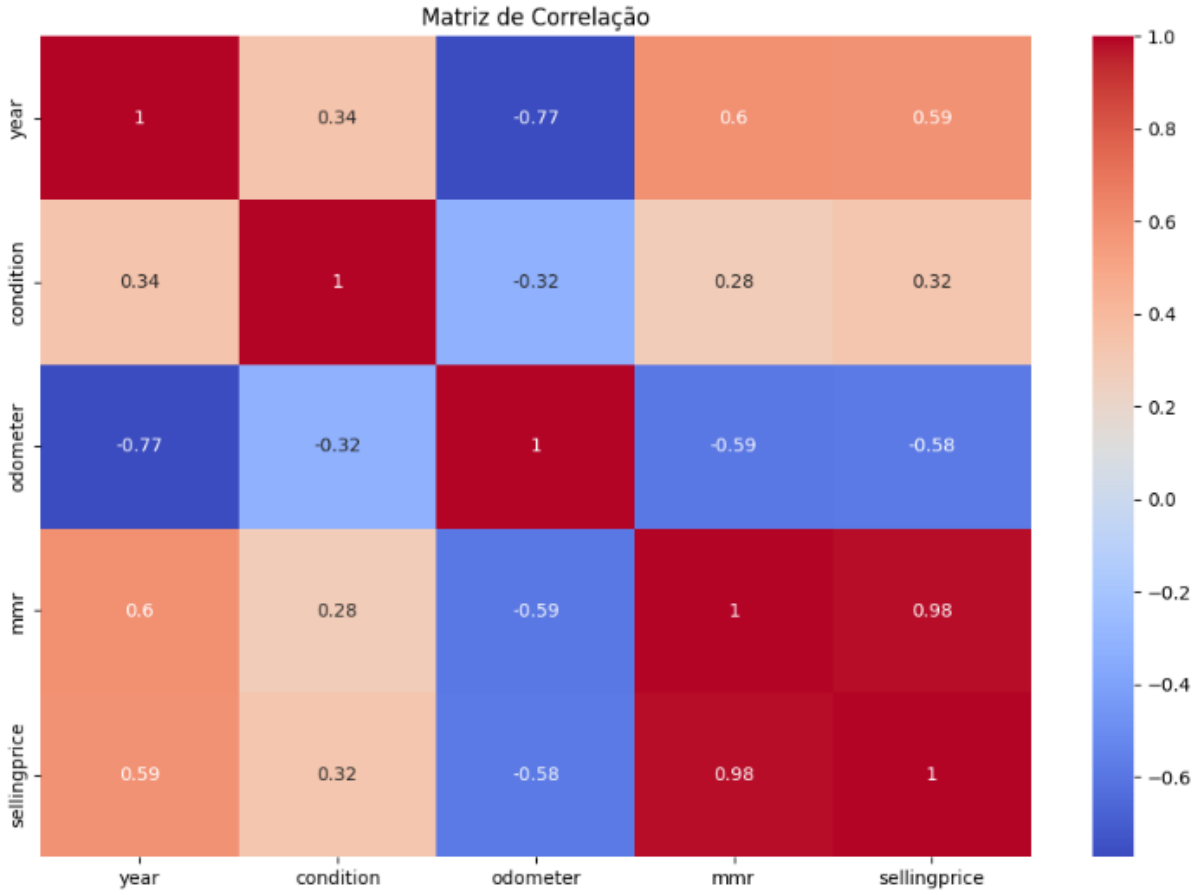


Figura 12

Através da visualização da nossa matriz de correlação (figura 12) conseguimos verificar que existem variáveis com elevados níveis de correlação, existindo também variáveis com correlação positiva e negativa.

Podemos também entender influências no preço de venda (*sellingprice*), Influências tais como:

- Correlação Forte com *mmr* (0.98): Preço de revenda influencia fortemente o preço de venda.
- Correlação Moderada com *year* (0.59): Carros mais novos tendem a ter preços de venda mais altos.
- Correlação Moderada com *odometer* (-0.58): Carros com mais quilometragem tendem a ter preços de venda mais baixos.
- Correlação Moderada com *condition* (0.32): A condição do carro tem uma influência positiva moderada no preço de venda.

Damos assim por terminada a segunda etapa do nosso projeto (visualização de dados), e damos início ao terceiro passo do nosso projeto (limpeza dos dados).

Terceiro Passo – Limpeza dos Dados

Nesta etapa utilizamos a função "dropna" da biblioteca *pandas* para remover linhas do dataset que contem valores ausentes em colunas específicas, ou seja, conterá apenas as linhas em que as colunas *condition*, *odometer* e *sellingprice* têm valores não ausentes (ou seja, valores válidos e preenchidos).

```
# Limpeza das colunas condition, odometer e sellingprice através da eliminação de linhas com valores em falta
df = df.dropna(subset=["condition", "odometer", "sellingprice"])
```

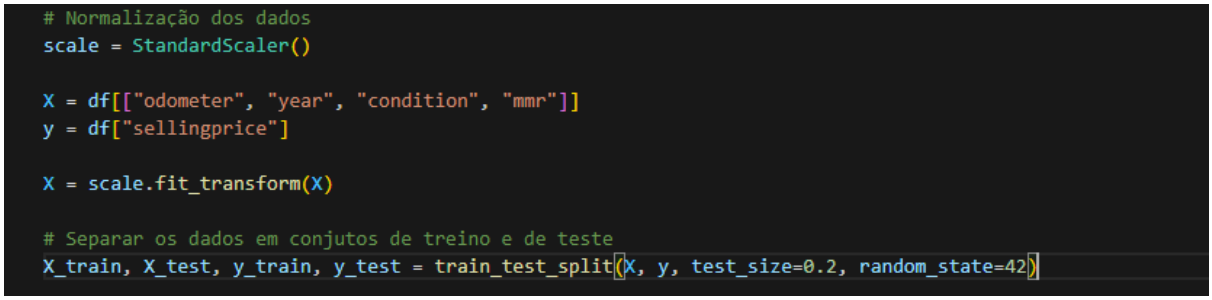
Figura 13

Concluímos assim a terceira etapa do nosso projeto e iniciamos a quarta etapa (preparação dos dados).

Quarto Passo – Preparação dos Dados

A preparação dos dados é essencial, com a finalidade de organizar os dados de maneira adequada para modelos de aprendizagem da máquina, melhorando assim a precisão do modelo ao garantir que os dados estejam normalizados e bem separados nos conjuntos de treino e teste.

Ao observarmos as linhas de código da figura 14, concluímos que:



```
# Normalização dos dados
scale = StandardScaler()

X = df[["odometer", "year", "condition", "mmr"]]
y = df["sellingprice"]

X = scale.fit_transform(X)

# Separar os dados em conjutos de treino e de teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figura 14

Este código permite-nos normalizar os dados das colunas *odometer*, *year*, *condition* e *mmr* utilizando a classe *StandardScaler*.

De seguida separa o dataset, *df*, em variáveis independentes X e em variável dependente y.

Por último divide os dados normalizados em conjuntos de treino e teste, com 80% dos dados usados para treino e 20% para teste.

Damos assim por terminada a quarta etapa do projeto e damos assim início à penúltima etapa do projeto (aplicação de modelos).

Quinto Passo – Aplicação dos Modelos

Nesta fase do projeto, aplicámos dois modelos de regressão - Random Forest e Gradient Boosting - para fazer as previsões dos valores alvo.

Random Forest

Primeiro, utilizámos o modelo de Random Forest Regressor. O código abaixo (figura 15) mostra como configurámos e treinámos o modelo, assim como a avaliação das previsões.

```
rf_model = RandomForestRegressor(n_estimators=200, random_state=42, max_depth=5)
rf_model.fit(X_train, y_train)

y_pred = rf_model.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred)
r2_rf = r2_score(y_test, y_pred)

print(f"Random Forest MSE: {mse_rf}")
print(f"Random Forest R^2: {r2_rf}")
```

✓ 1m 13.1s

```
Random Forest MSE: 3022753.3306974447
Random Forest R^2: 0.9681755867619253
```

Figura 15

Os resultados obtidos foram:

- MSE (Erro Quadrático Médio): 3022753.3306974447
- R^2 (Coeficiente de Determinação): 0.9681755867616295

Podemos observar que a métrica MSE não é muito elevada e que a métrica R^2 está próxima de 1, indicando que o modelo tem uma boa capacidade preditiva.

Gradient Boosting

Em seguida, aplicámos o modelo de Gradient Boosting. A configuração, treino e avaliação do modelo estão apresentados na figura 16.

```
gb_model = GradientBoostingRegressor(n_estimators=250, learning_rate=0.05, max_depth=3, random_state=42)
gb_model.fit(X_train, y_train)

y_pred = gb_model.predict(X_test)
mse_gb = mean_squared_error(y_test, y_pred)
r2_gb = r2_score(y_test, y_pred)

print(f"Gradient Boosting MSE: {mse_gb}")
print(f"Gradient Boosting R^2: {r2_gb}")
```

✓ 1m 21.9s

```
Gradient Boosting MSE: 2234953.315118946
Gradient Boosting R^2: 0.9764697710706882
```

Figura 16

Os resultados obtidos foram:

- MSE (Erro Quadrático Médio): 2234953.315118946
- R^2 (Coeficiente de Determinação): 0.976469771076882

Neste caso, verificámos que as métricas apresentaram melhores resultados, com um MSE ainda mais reduzido e um R^2 mais próximo de 1, mostrando uma performance superior do modelo de Gradient Boosting em relação ao Random Forest.

Após treino dos modelos, vamos visualizar os resultados para melhor compreensão.

Sexto Passo – Visualização de Resultados

Neste passo, focamo-nos na visualização dos resultados obtidos pelos modelos de Random Forest e Gradient Boosting.w

Gráficos de Dispersão: Valores Preditos vs. Valores Reais

Para avaliar a performance dos nossos modelos, desenhámos gráficos de dispersão dos valores previstos em comparação com os valores reais. A linha vermelha representa o caso ideal onde os valores previstos seriam iguais aos valores reais.

```
# Previsões dos modelos
y_pred_gb = gb_model.predict(X_test)
y_pred_rf = rf_model.predict(X_test)

plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_rf, color='green', alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red', linestyle='--', linewidth=2)
plt.xlabel('Valores Reais')
plt.ylabel('Valores Previstos')
plt.title('Random Forest: Valores Previstos vs. Valores Reais')
plt.show()

plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_gb, color='blue', alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red', linestyle='--', linewidth=2)
plt.xlabel('Valores Reais')
plt.ylabel('Valores Previstos')
plt.title('Gradient Boosting: Valores Previstos vs. Valores Reais')
plt.show()
✓ 1.6s
```

Figura 17

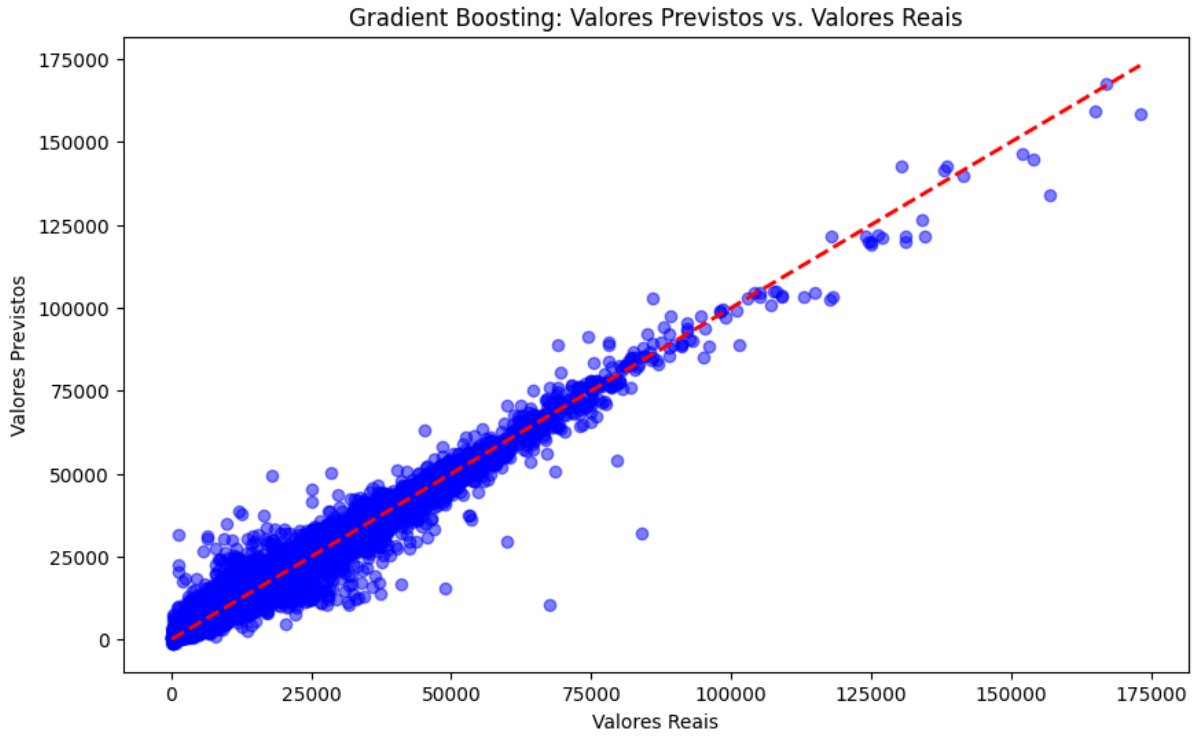


Figura 18

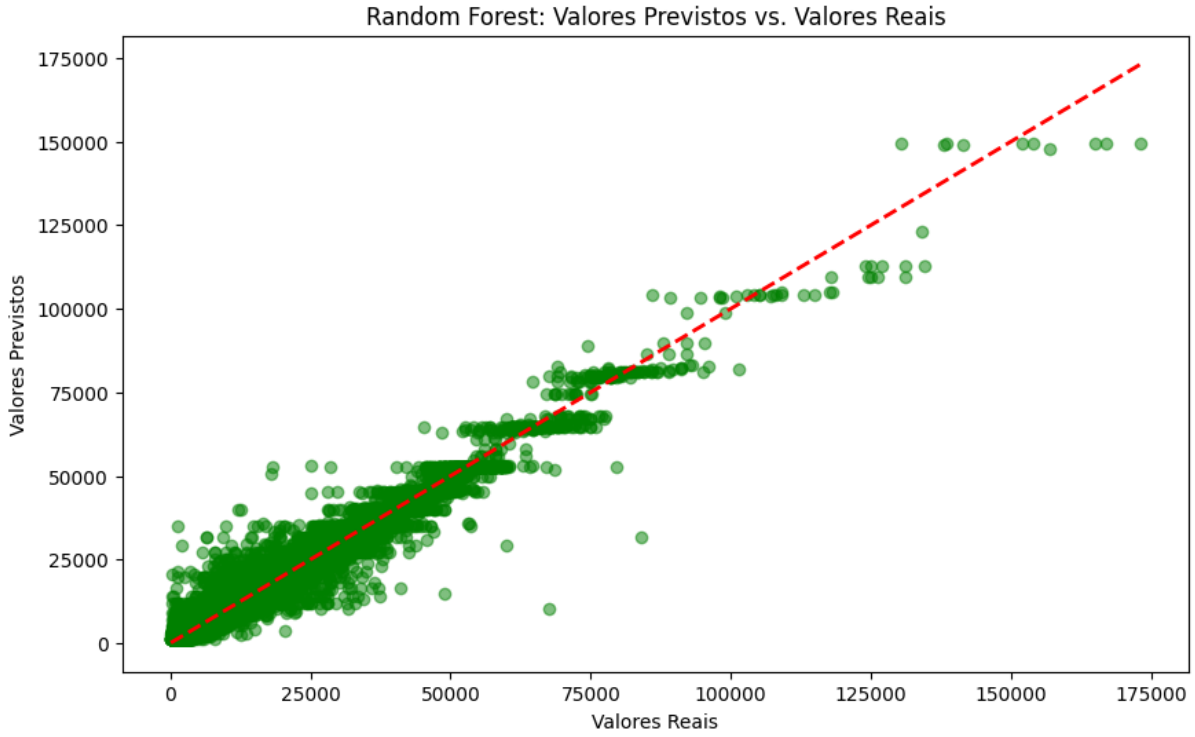


Figura 19

Podemos observar que os dados previstos estão muito próximos dos dados reais para ambos os modelos, indicando que as previsões estão a fazer sentido e a ter precisão.

Gráficos de Importância das Variáveis

Também visualizámos a importância das variáveis para entender que características são mais influentes nos modelos.

```
original_columns = ["odometer", "year", "condition", "mmr"]

# Importância das variáveis para Random Forest
feat_importances_rf = pd.Series(rf_model.feature_importances_, index=original_columns)
feat_importances_rf.nlargest(10).plot(kind='barh', color='green', figsize=(10, 6))
plt.title('Importância das Variáveis - Random Forest')
plt.xlabel('Importância Relativa')
plt.ylabel('Variável')
plt.show()

# Importância das variáveis para Gradient Boosting
feat_importances_gb = pd.Series(gb_model.feature_importances_, index=original_columns)
feat_importances_gb.nlargest(10).plot(kind='barh', color='blue', figsize=(10, 6))
plt.title('Importância das Variáveis - Gradient Boosting')
plt.xlabel('Importância Relativa')
plt.ylabel('Variável')
plt.show()
```

✓ 0.1s

Figura 20

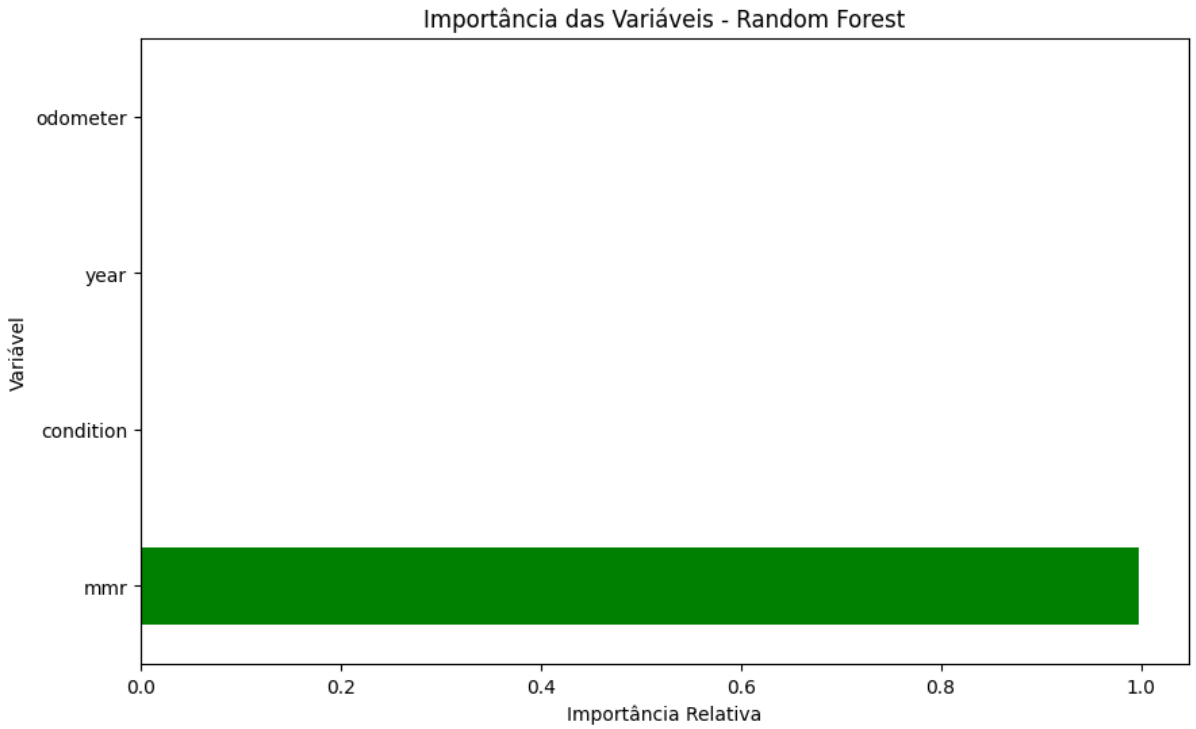


Figura 21

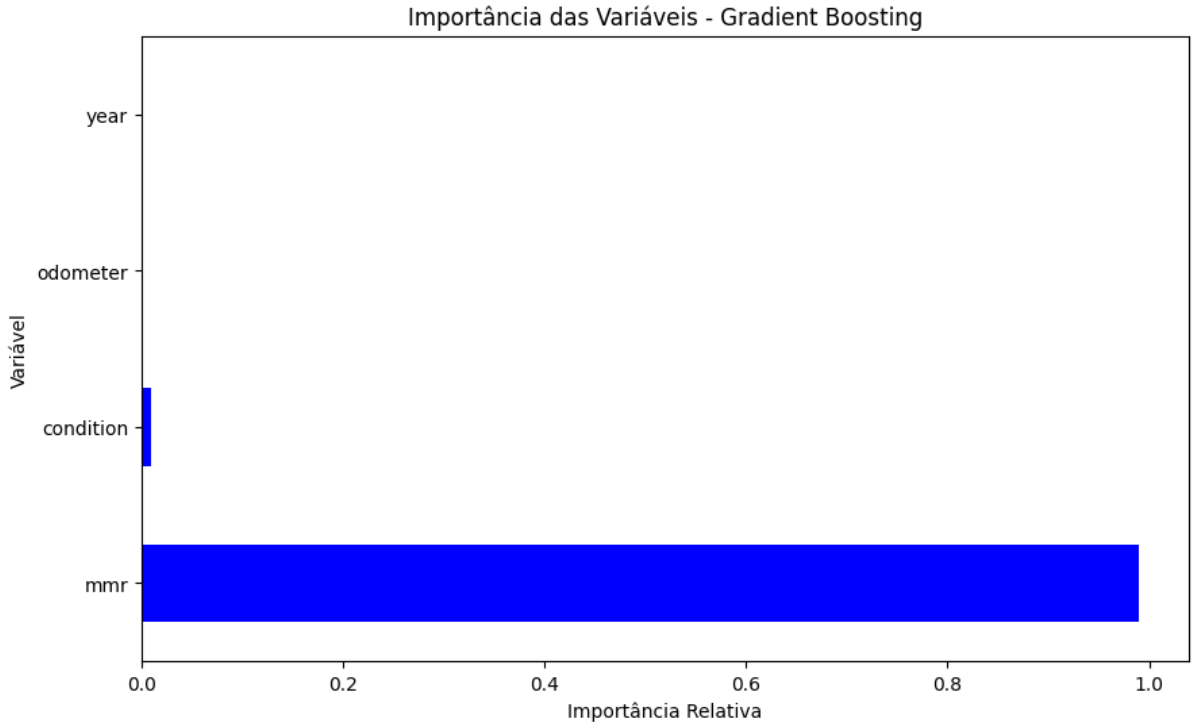


Figura 22

Estes gráficos mostram a relevância de cada variável para os modelos, ajudando-nos a compreender quais as características que mais influenciam as previsões. Concluímos então que a variável de maior importância é a variável *mmr*, seguida da variável *condition*.

Comparação de Métricas: MSE e R²

Para além dos gráficos de dispersão e de importância das variáveis, também comparámos diretamente as métricas MSE (Mean Squared Error) e R² (Coeficiente de Determinação) dos dois modelos. Esta comparação permite-nos avaliar qual modelo apresenta melhor performance.

```python
# Criar DataFrame com os dados
metrics = pd.DataFrame({
    'Modelo': ['Random Forest', 'Gradient Boosting'],
    'MSE': [mse_rf, mse_gb],
    'R^2': [r2_rf, r2_gb]
})

# Normalizar MSE para a escala de 0 a 1
metrics['MSE_Normalized'] = metrics['MSE'] / metrics['MSE'].max()

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))  # Reduzindo o tamanho da figura

width = 0.4
x = range(len(metrics['Modelo']))

# Plotar MSE normalizado no primeiro subplot (ax1)
ax1.bar(x, metrics['MSE_Normalized'], width, label='MSE (Normalizado)\n (Menor é melhor)', color='tab:blue')
ax1.set_xlabel('Modelo')
ax1.set_ylabel('Valor Normalizado')
ax1.set_title('Comparação de MSE - Random Forest vs. Gradient Boosting')
ax1.set_xticks([p for p in x])
ax1.set_xticklabels(metrics['Modelo'])
ax1.legend(loc='best')

# Plotar R^2 no segundo subplot (ax2)
ax2.bar(x, metrics['R^2'], width, label='R^2\n(Maior é melhor)', color='tab:green')
ax2.set_xlabel('Modelo')
ax2.set_ylabel('R^2')
ax2.set_title('Comparação de R^2 - Random Forest vs. Gradient Boosting')
ax2.set_xticks([p for p in x])
ax2.set_xticklabels(metrics['Modelo'])
ax2.legend(loc='best')

plt.tight_layout()  # Garante que não haja sobreposição entre os subplots
plt.show()
✓ 0.1s
```

Figura 23

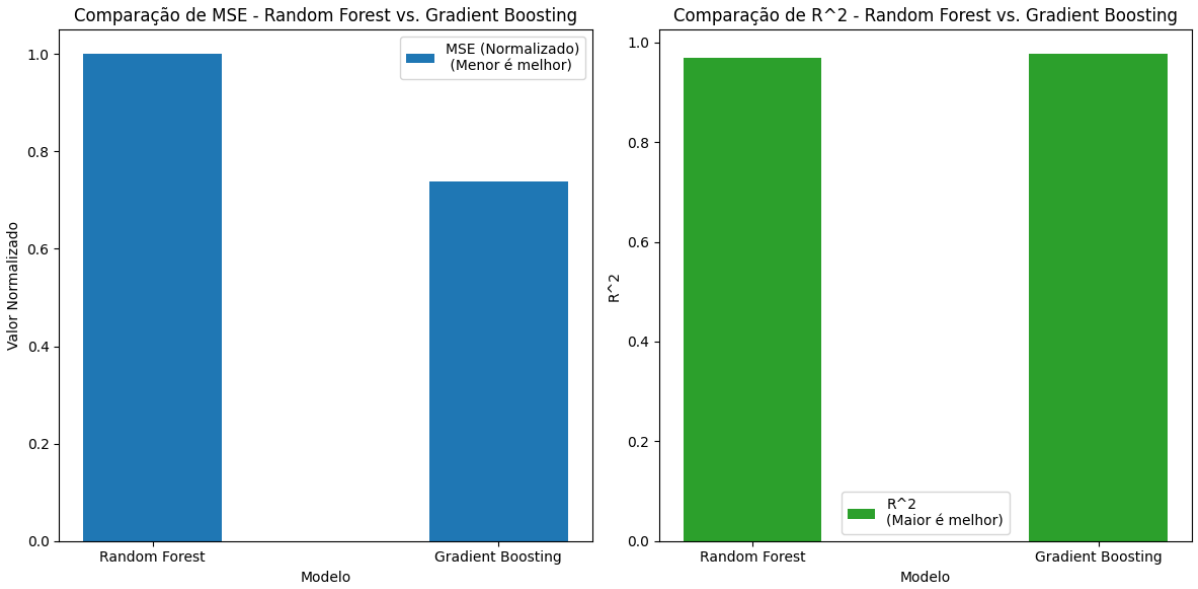


Figura 24

Os gráficos acima (figura 24) mostram a comparação das métricas MSE (normalizado) e R^2 entre os modelos de Random Forest e Gradient Boosting. Observamos que o Gradient Boosting apresenta um MSE menor e um R^2 maior, indicando uma melhor performance em comparação com o Random Forest.

Conclusão

Este projeto visou desenvolver e avaliar modelos preditivos para estimar os preços de venda de veículos usados, utilizando técnicas de ciência de dados. Os modelos utilizados, Random Forest e Gradient Boosting, foram comparados utilizando as métricas de Erro Quadrático Médio (MSE) e Coeficiente de Determinação (R^2).

Resultados

O Gradient Boosting demonstrou um desempenho superior, com maior precisão na previsão dos preços.

Implicações

Os resultados indicam que o Gradient Boosting é mais eficaz devido à sua capacidade de corrigir erros sequencialmente, resultando numa melhor adaptação aos dados. A análise confirmou a eficiência deste modelo em explicar a variância observada.

Considerações Finais

O projeto evidenciou a eficácia das técnicas de aprendizagem automática na previsão de preços de carros usados, identificando variáveis-chave como ano, marca, modelo, condição e quilometragem. Este trabalho demonstrou a relevância da ciência de dados na resolução de problemas complexos, fornecendo uma base sólida para estudos futuros e aplicações no setor empresarial.

Bibliografia

https://www.kaggle.com/datasets/syedanwarafridi/vehicle-sales-data

OpenAI. ChatGPT, 3.5/4, junho de 2024

https://www.python.org

https://code.visualstudio.com

https://aws.amazon.com/pt/what-is/data-science/

https://www.ibm.com/topics/machine-learning

https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/

https://www.ibm.com/topics/random-forest
https://statisticsbyjim.com/regression/mean-squared-error-mse/

https://statisticsbyjim.com/regression/interpret-r-squared-regression/

https://scikit-learn.org/stable/supervised_learning.html