

Universidade Federal do Rio Grande do Sul  
Instituto de Informática



INF01017  
Aprendizado de Máquina

---

## **Trabalho Prático Final**

**Predição de consumo de combustível utilizando  
aprendizado de máquina**

---

Luís Filipe Martini Gastmann (00276150)  
Pedro Lubaszewski Lima (00341810)  
Vinícius Boff Alves (00335551)

Turma U

9 de novembro de 2024

# Sumário

1.1	Definição do Problema e Coleta de Dados . . . . .	2
2.1	Análise Exploratória e Pré-processamento dos Dados . . . . .	3
2.1.1	Análise Exploratória dos Dados . . . . .	3
2.1.2	Pré-processamento dos Dados . . . . .	4
3.1	Abordagem, Algoritmos e Estratégias de Avaliação . . . . .	6
4.1	<i>Spot-checking</i> de Algoritmos . . . . .	7
4.1.1	Revisitando os Dados após o Pré-processamento . . . . .	7
4.1.2	Sumarização dos Resultados . . . . .	8

## 1.1 Definição do Problema e Coleta de Dados

O objetivo deste trabalho é prever o consumo médio de combustível de um carro através de algumas das suas características e origens de fabricação. Alguns atributos das instâncias são a marca, a quantidade de cilindros, o porte do veículo etc.

O conjunto de dados utilizado para desenvolver este trabalho foi obtido da seguinte página do Kaggle: Explore Car Performance: Fuel Efficiency Data. Essa tarefa contará com diversas técnicas de preparação dos dados para posteriormente iniciar a seleção e avaliação de modelos para essa tarefa.

## 2.1 Análise Exploratória e Pré-processamento dos Dados

### 2.1.1 Análise Exploratória dos Dados

Este *dataset* possui 550 instâncias, com 11 atributos preditores e 1 atributo alvo. Esse último torna a tarefa dos modelos em regressão, visto que o objetivo aqui é prever o consumo médio de combustível de carros. No conjunto de dados, esse atributo predito se chama *combination mpg*.

Dos atributos preditores, observa-se que há 5 atributos numéricos (*city mpg*, *cylinders*, *displacement*, *highway mpg* e *year*). Além deles, os 6 atributos restantes são categóricos: *class*, *drive*, *fuel type*, *make*, *model* e *transmission*. Com isso em mente, criou-se alguns gráficos para analisar correlações e distribuições dos dados. A seguir, o *violin plot* do atributo alvo pelo número de veículos:

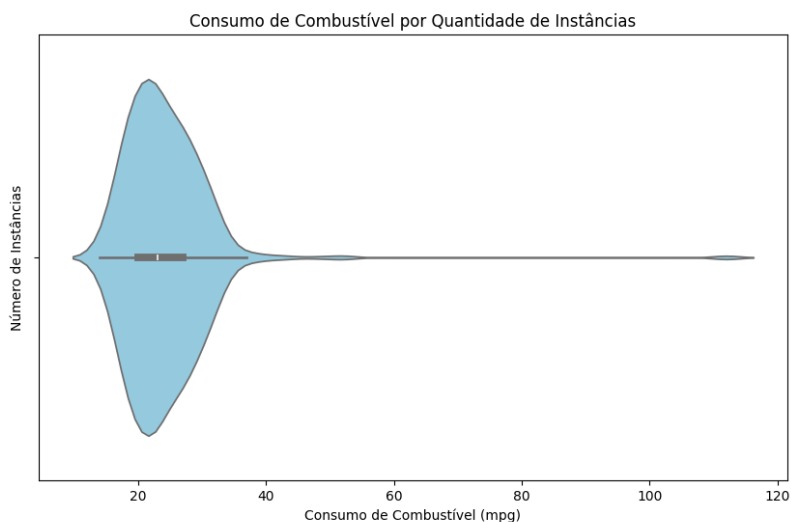


Figura 1: *Violin Plot* de Consumo Médio

Nesse ponto, já se observa que há instâncias problemáticas, claramente *outliers* que devem ser removidas do conjunto de dados. Ademais, a maior concentração dos veículos apresenta consumo médio entre 15 e 30mpg, algo que pode guiar bastante os modelos. Após isso, analisou-se o atributo de classes de veículos para ter uma ideia da sua distribuição em um gráfico de setores.

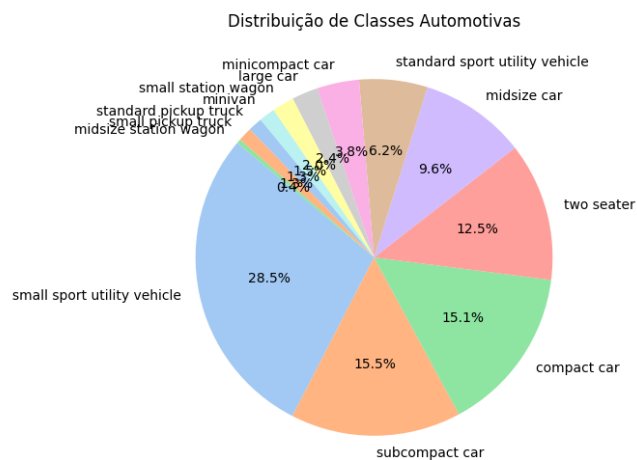


Figura 2: Distribuição de Classes de Veículos

Com ele, é perceptível que talvez seja necessário agrupar as classes de veículos e outros

atributos que contenham classes com muito poucos representantes, como *midsize station wagon*, por exemplo, em uma classe geral chamada *others*. Porém, isso será melhor analisado, se necessário, na segunda etapa do trabalho. No entanto, algo que é necessário é a codificação dos atributos categóricos em numéricos para padronizar a entrada numérica dos modelos.

Por fim, analisou-se a Correlação de Pearson entre os atributos numéricos através do *heatmap* abaixo:

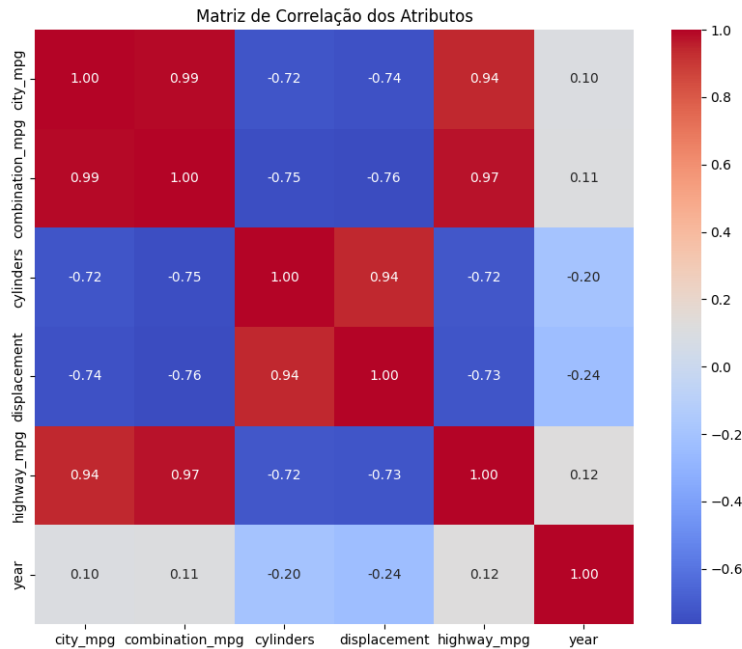


Figura 3: *Heatmap* de Correlações entre Atributos Numéricos

Olhando para o gráfico acima, percebe-se que há diversos atributos preditores que apresentam alto nível de correlação com a saída *combination mpg* e entre si. Ambos os atributos *highway mpg* e *city mpg* serão descartados do *dataset* por terem alta correlação entre si e com o valor da saída do modelo, facilitando demais a tarefa dos modelos. Além disso, observa-se que *displacement* e *cylinders* também apresentam alta correlação entre si. No entanto, a informação de *cylinders* ou cilindros de um carro é diferente da informação de *displacement* ou cilindradas desse. As cilindradas representam o volume total dos cilindros de um motor. Por conta disso, são calculados a partir dos cilindros, porém acrescentam um dado a novo ao problema. Ou seja, vale a pena mesmo assim manter ambos os atributos no *dataset*.

Isto não foi ilustrado pelos gráficos, porém os dados numéricos apresentam diversas escalas diferentes, exigindo técnicas de normalização após a separação de conjuntos de treinamento/validação e de testes.

### 2.1.2 Pré-processamento dos Dados

Como já discutido na seção acima e analisando alguns trabalhos realizados com esse conjunto de dados, esses sendo de Lunovian [5], da Ayşe [1] e do Priyanshu [9], implementou-se algumas estratégias de pré-processamento nos dados deste problema.

Primeiramente, realizou-se a remoção dos atributos discutidos anteriormente. Esses sendo o *highway mpg*, *city mpg* e *displacement*, visto que apresentam altos níveis de correlação entre si e com o valor da saída do modelo. Após essa remoção, filtrou-se as instâncias com muitos atributos faltantes (em geral, carros elétricos que funcionam diferentemente dos carros que funcionam com combustíveis fósseis).

Depois disso, foi realizada a limpeza de instâncias cujo atributo alvo representava um *outlier* no gráfico de distribuição de consumo médio dos veículos. Essa estratégia é necessária para

evitar que os modelos sejam treinados com instâncias que não representam bem a grande maioria dos veículos. Utilizou-se a medida padrão de 1.5 vezes o IQR para identificar *outliers*.

Sobre a codificação dos atributos, o grupo utilizou a codificação padrão de categórico para numérico que é o *one-hot encoding*. A justificativa para isso é que os atributos categóricos não apresentam nenhuma ordem específica que justifique utilizar uma codificação em números inteiros. Além disso, em relação à normalização, baseado no trabalho do Jason Brownlee [4] e do Matheus Vasconcelos [6], optou-se por utilizar a padronização por *z-score*, visto que os dados aproximam uma distribuição normal. Tanto a codificação, quanto a normalização foram aplicadas após a separação de conjuntos de treinamento/validação e de testes.

Um outro problema encontrado consiste na possibilidade de não ser visto algum modelo de carro (*model*) no treinamento. Isso pode acontecer pois há uma quantidade muito grande de modelos diferentes, alguns deles com poucas instâncias. Para solucionar esse problema, “forçou-se” a haver pelo menos uma instância de cada modelo de veículo no conjunto de treinamento, antes de realizar a separação de conjunto de dados para teste. Para realizar essa última separação, utilizou-se a função `train_test_split()` com um `test_size` de 15%. Essa ação corresponde à estratégia de *holdout* vista em aula para a primeira divisão dos dados.

### 3.1 Abordagem, Algoritmos e Estratégias de Avaliação

Seguindo as ideias discutidas nos itens anteriores, este problema consiste em uma tarefa de regressão. Por conta disso, o grupo buscou no conteúdo programático da disciplina algoritmos para essa tarefa. Com isso, separou-se os seguintes algoritmos de aprendizado para serem avaliados: ***k-Nearest Neighbors***, ***Random Forest***, **Regressão Linear**, **Redes Neurais** e **SVM**. A escolha desses modelos também foi inspirada nos trabalhos de Neslihan Avsar [7], de James McCaffrey [3] e de isitapol2002 [2].

Definidos esses algoritmos e as estratégias de pré-processamento, definiu-se como estratégia de validação desses algoritmos o *k-fold cross-validation*. Essa é a *magnum opus* das estratégias de validação de aprendizado supervisionado, então foi a adotada pelo grupo. O valor de  $k = 13$  foi definido experimentalmente. A justificativa mora na quantidade de instâncias que sobraram para a validação após a separação de modelos de carros únicos para o treinamento e do conjunto de testes. O número final de elementos por *fold* beira 18 instâncias (quantidade não muito grande, porém proporcionalmente aceita para a quantidade total de instâncias do *dataset*), gerando 13 medições de erro para cada modelo.

Em cada iteração do *k-fold cross-validation*, realizou-se a normalização e a codificação dos atributos correspondentes através dos métodos adequados do `scikit learn`. Foi fixada um `random.state = 42` para padronizar os resultados dos testes. Os hiperparâmetros utilizados nesta etapa foram todos os padrões de cada modelo da biblioteca do `scikit learn`.

A métrica de avaliação utilizada foi o *Mean Squared Error* (MSE). O grupo optou por sumarizar os resultados do *spot-checking* com apenas esta métrica (descartando o *Mean Absolute Error*) porque são métricas de natureza muito similar, já foi realizada a limpeza de *outliers* que poderiam prejudicar os resultados dessas medições e essa é uma métrica que apresenta convergência mais rápida (Nirajan Acharya [8]). Além disso, foram feitas diversas representações dessa métrica para obter maior confiança nos resultados obtidos.

## 4.1 *Spot-checking* de Algoritmos

### 4.1.1 Revisitando os Dados após o Pré-processamento

Com toda a metodologia explicada, voltar-se-á ao conjunto de dados pré-processados para realizar uma breve revisão dos dados de treinamento/validação. Primeiramente, após a remoção de *outliers*, redução de dimensionalidade e exclusão de instâncias de carros elétricos, gerou-se o seguinte gráfico de distribuição da saída a ser predita:

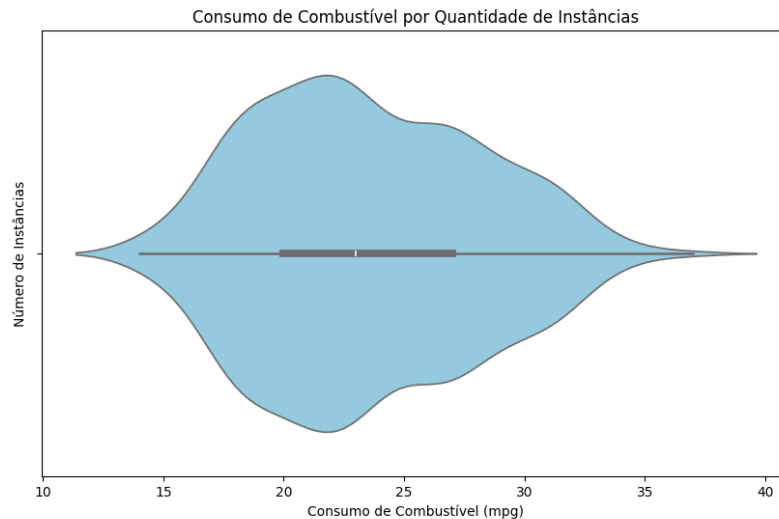


Figura 4: *Violin Plot* de Consumo Médio Pré-processado

Agora observa-se que os dados quase se distribuem em formato de uma gaussiana. Além disso, continua observando-se a concentração de dados entre 15 e 30mpg de consumo médio. O gráfico de correlação também é refeito a seguir:

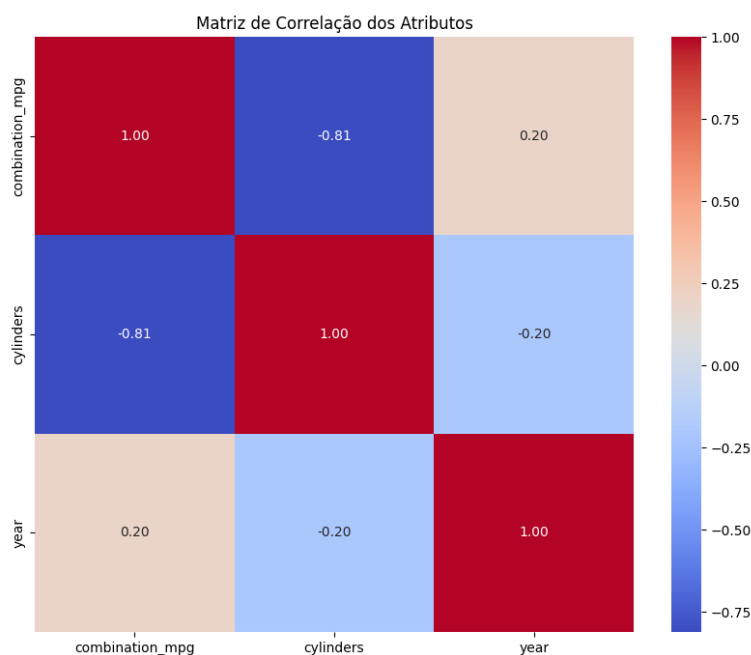


Figura 5: *Heatmap* de Correlações entre Atributos Numéricos Pré-processados

Com a remoção dos atributos citados anteriormente, observa-se uma boa redução na



correlação entre os atributos numéricos restantes. Algo que ajuda bastante a evitar a maldição da dimensionalidade.

#### 4.1.2 Sumarização dos Resultados

Com esses dados revisitados, a seguir seguem os resultados obtidos pelos modelos:

Modelo	Média dos MSE	Desvio Padrão dos MSE
kNN	5,4293	2,3616
<i>Random Forest</i>	1,9517	1,1847
Regressão Linear	1,6631	0,9758
Redes Neurais	1,8377	1,0418
SVM	3,3739	1,3368

Tabela 1: Médias e Desvios Padrões dos MSE

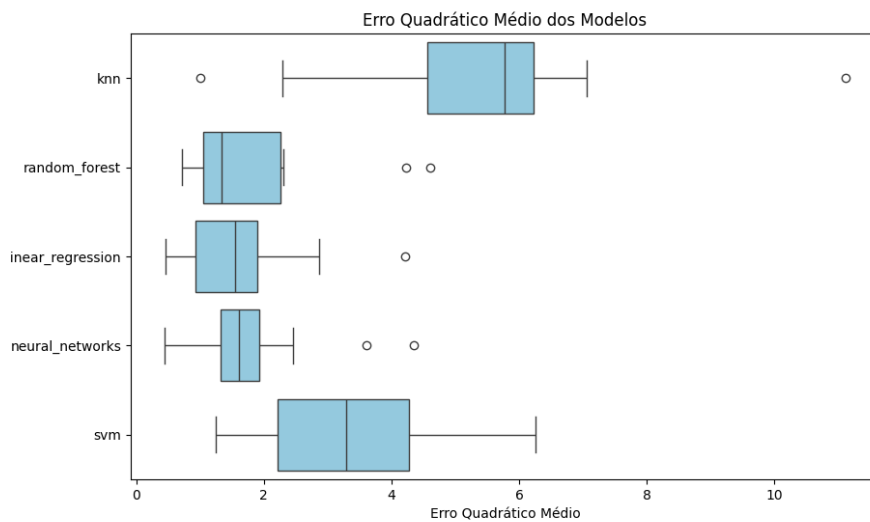


Figura 6: *Box Plot* dos MSE

Por fim, com esses resultados, fica claro que os 3 modelos mais promissores foram a ***Random Forest***, a **Regressão Linear** e as **Redes Neurais**. Como nenhum atributo foi otimizado, pode ser que os outros dois modelos se dessemenhassem melhor, porém, com as configurações básicas de hiperparâmetros, esses foram os três mais promissores.

Dentre esses, é perceptível que os mais eficientes nesta tarefa foram o algoritmo de Regressão Linear e o modelo de Redes Neurais, visto que o viés da *Random Forest* é mais elevado quando comparado com esses últimos dois. Mesmo que a Regressão Linear apresente média e desvio padrão melhores do que as Redes Neurais, o *box plot* acima mostra que o viés do erro das Redes Neurais é mais concentrado, próximo de um valor estável. Por conta dessas razões, serão aprofundados as Florestas Aleatórias, o Regressor Linear e as Redes Neurais, com ênfase nos últimos dois durante a próxima etapa do projeto.

# Bibliografia

- [1] Ayşe Nur Durmaz. OruntuTanima-Perceptron. Website available: <https://www.kaggle.com/code/ayenurdurmaz/oruntutanima-perceptron>. 2024.
- [2] isitapol2002. Multiple Linear Regression With scikit-learn. Website available: <https://www.geeksforgeeks.org/multiple-linear-regression-with-scikit-learn/>. 2022.
- [3] James McCaffrey. Regression Using a scikit MLPRegressor Neural Network. Website available: <https://visualstudiomagazine.com/Articles/2023/05/01/regression-scikit.aspx>. 2023.
- [4] Jason Brownlee. How to Use the ColumnTransformer for Data Preparation. Website available: <https://machinelearningmastery.com/columntransformer-for-numerical-and-categorical-data/>. 2020.
- [5] Lunovian. Which Cars Contribute to a Greener Future? Website available: <https://www.kaggle.com/code/anmatngu/which-cars-contribute-to-a-greener-future>. 2024.
- [6] Matheus Vasconcelos. Distribuição normal e a classe Standard Scaler da biblioteca Scikit-learn... Website available: <https://medium.com/@mhvasconcelos/distribuio-normal-e-a-biblioteca-standard-scaler-em-python-f21c52070c6b>. 2023.
- [7] Neslihan Avsar. Building a Linear Regression model with Scikit-Learn — Part — 1. Website available: <https://medium.com/@neslihannavsar/building-a-linear-regression-model-with-scikit-learn-part-1-eed4c04f53f9>. 2022.
- [8] Nirajan Acharya. Choosing Between Mean Squared Error and Mean Absolute Error. Website available: <https://medium.com/@nirajan.acharya777/choosing-between-mean-squared-error-mse-and-mean-absolute-error-mae-in-regression-a-deep-dive-c16b4eeee603>. 2023.
- [9] Priyanshu shukla. Analysis on Car performance dataset. Website available: <https://www.kaggle.com/code/priyanshu594/analysis-on-car-performance-dataset>. 2024.