

---

# Dados e Aprendizagem Automática

---



Universidade do Minho  
Escola de Engenharia

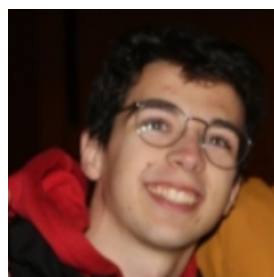
## TRABALHO REALIZADO POR:

GONALO MARTINS DOS SANTOS  
RICARDO LOPES LUCENA  
NUNO MIGUEL LEITE DA COSTA  
JOAO PEDRO MACHADO RIBEIRO

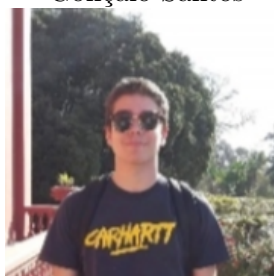
## Grupo 23



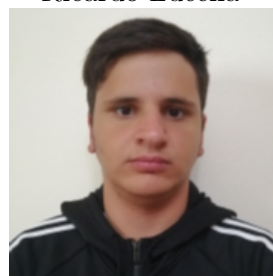
pg53842  
Gonalo Santos



pg54187  
Ricardo Lucena



pg54121  
Nuno Costa



a95719  
Joao Ribeiro

---

# Índice

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Metodologia</b>	<b>2</b>
<b>3</b>	<b>Dataset de Grupo</b>	<b>2</b>
3.1	Exploração do Dataset . . . . .	2
3.2	Preparação dos Dados . . . . .	8
3.3	Modelação e Avaliação . . . . .	9
<b>4</b>	<b>Dataset de competição</b>	<b>11</b>
4.1	Compreensão do contexto . . . . .	11
4.2	Preparação dos datasets . . . . .	11
4.3	Exploração dos Dados . . . . .	12
4.3.1	Relação das várias features com o target . . . . .	13
4.4	Tratamento dos Dados . . . . .	15
4.4.1	Tratamento geral . . . . .	15
4.4.2	Missing values . . . . .	15
4.4.3	Feature Engineering . . . . .	16
4.4.4	Codificação das features categóricas . . . . .	16
4.4.5	Correlação com o <i>target</i> . . . . .	17
4.4.6	Balanceamento do dataset . . . . .	17
4.5	Modelação e Avaliação . . . . .	18
4.5.1	Performance dos diferentes modelos . . . . .	19
4.6	Conclusão . . . . .	19

---

# 1 Introdução

Neste trabalho prático, encaramos o desafio de explorar e manipular datasets, analisando e transformando as informações neles contidas. Foi-nos proposto, adicionalmente, a construção de modelos de previsão de resultados com base em algoritmos de *Machine Learning*, que foram estudados ao longo do semestre. Desta forma, o projeto concentra-se em dois datasets: um selecionado pela equipa docente, que se baseia em prever a quantidade de energia produzida por certos painéis solares da cidade de Braga, e outro escolhido pelo grupo, centrado no preço de carros usados em relação um conjunto de características. Neste relatório, iremos abordar todas as tarefas realizadas pelo nosso grupo, desde a metodologia utilizada para cada conjunto de dados, passando pela exploração e tratamento de cada dataset, até à construção de modelos de *Machine Learning* e concluindo com uma análise dos resultados obtidos.

## 2 Metodologia

A elaboração deste projeto começou por determinar a metodologia a ser utilizada para extrair conhecimento. Optar por uma metodologia envolve a adoção de um conjunto de etapas e passos que o projeto de extração de conhecimento deve seguir para resolver diversos problemas, permitindo um planeamento e gestão eficaz do projeto, bem como a obtenção de melhores resultados. Neste sentido, o grupo optou por escolher a metodologia **CRISP-DM** (**C**Ross **I**ndustry **S**tandard **P**rocess for **D**ata **M**ining).

Depois de escolhida a metodologia, é importante seguir as suas seis etapas (**Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, Deployment**) para avançar no desenvolvimento de uma solução. Este processo nem sempre é contínuo, havendo momentos em que é necessário retroceder a passos anteriores para alcançar uma resposta mais eficaz para o problema em questão.

## 3 Dataset de Grupo

O conjunto de dados utilizado pela nosso grupo é constituído por informações de automóveis usados provenientes de várias regiões do Paquistão e da Índia. A escolha deste conjunto de dados tem como objetivo principal analisar de que forma diferentes características dos carros usados influenciam os seus preços no mercado.

Link para o conjunto de dados: <https://www.kaggle.com/datasets/mohidabdulrehman/ultimate-car-price-prediction-dataset>

### 3.1 Exploração do Dataset

Nesta secção, serão abordados os métodos que o nosso grupo utilizou para explorar o conjunto de dados. Inicialmente, realizámos uma exploração mais superficial, onde analisámos os Data Types, Missing Values, e características dos atributos. Em seguida, procedemos a uma exploração mais detalhada, utilizando várias representações visuais para comparar os atributos com a **target feature**, a partir das quais retirámos conclusões.

Assim, a nossa exploração começou considerando que o conjunto de dados é composto por **46.022 linhas** e **14 features**, que são as seguintes:

1. **Id**: Um identificador único para cada carro;
2. **Company Name**: O nome da companhia que construiu o carro;

3. **Model Name**: O nome do modelo do carro;
4. **Model Year**: O ano do modelo do carro;
5. **Location**: A localização onde os carros estão expostos para venda;
6. **Mileage**: A quantidade de milhas percorridas pelos carros;
7. **Engine Type**: O tipo de motor (Petrol, Hybrid e Diesel);
8. **Engine Capacity**: Pontência do motor do carro;
9. **Color**: Cor do carro;
10. **Assembly**: Carro local ou importado;
11. **Body Type**: Tipo de carro;
12. **Transmission Type**: Tipo de transmissão do carro (Manual e Automatic);
13. **Registration Status**: Carro registrado ou não;
14. **Price**: Preço de venda do carro (coluna target).

Posteriormente verificamos os *Data Types* que estamos a trabalhar e a ocorrência de *Missing Values* no nosso conjunto de dados.

cars.dtypes	
Id	int64
Company Name	object
Model Name	object
Model Year	int64
Location	object
Mileage	int64
Engine Type	object
Engine Capacity	int64
Color	object
Assembly	object
Body Type	object
Transmission Type	object
Registration Status	object
Price	int64
dtype:	object

cars.isna().any()	
Id	False
Company Name	False
Model Name	False
Model Year	False
Location	False
Mileage	False
Engine Type	False
Engine Capacity	False
Color	False
Assembly	False
Body Type	False
Transmission Type	False
Registration Status	False
Price	False
dtype:	bool

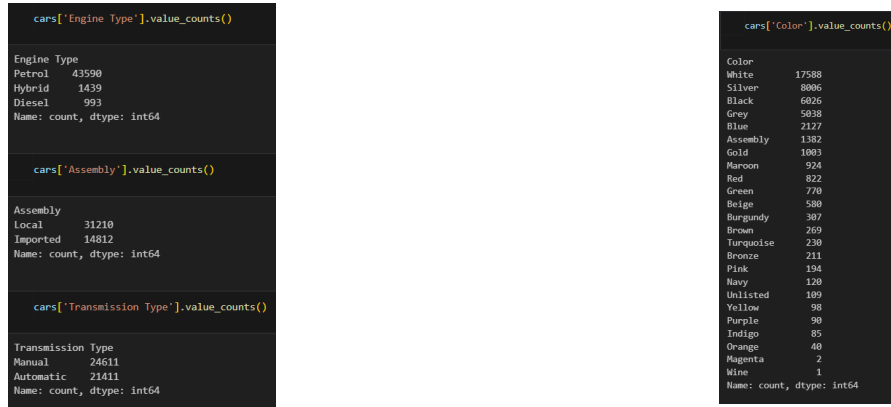
Com esta análise, concluímos os seguintes Tipos de Dados dos atributos:

- **object** - Company Name, Model Name, Location, Engine Type, Color, Assembly, Body Type, Transmission Type e Registration Status.
- **int64** - Id, Model Year, Mileage, Engine Capacity e Price.

cars.describe()					
	Id	Model Year	Mileage	Engine Capacity	Price
count	46022.000000	46022.000000	46022.000000	46022.000000	4.602200e+04
mean	23010.526205	2011.035374	90965.128243	1313.115575	2.014153e+06
std	13285.595581	6.399403	63656.656034	614.690832	2.939071e+06
min	0.000000	1990.000000	1.000000	16.000000	1.110000e+05
25%	11505.250000	2007.000000	48899.500000	1000.000000	8.500000e+05
50%	23010.500000	2013.000000	80000.000000	1300.000000	1.450000e+06
75%	34515.750000	2016.000000	120000.000000	1500.000000	2.300000e+06
max	46022.000000	2019.000000	999999.000000	6600.000000	7.750000e+07

Figure 1: Perceção quantitativa dos Data Types int64

Para além dos Data Types, constatamos a ausência direta de missing values. No entanto, ao analisar os valores de cada atributo (conforme exemplificado em algumas imagens seguintes), o grupo identificou que no atributo **Color** existiam duas cores, *Assembly* e *Unlisted*, consideradas inválidas. Dessa forma, o grupo decidiu tratar esses dois valores como se fossem missing values.



Numa abordagem mais detalhada, explorámos as ocorrências dos valores de cada atributo através de representações visuais, bem como as suas relações com a coluna de Price. As análises visuais e as conclusões obtidas são as seguintes:

**Company Name and Model Name** - Concluimos que há uma considerável influência de três marcas provenientes do continente asiático, nomeadamente a Toyota, Suzuki e Honda. Esta tendência deve-se ao facto de todas as três marcas oferecerem uma qualidade notável e, adicionalmente, a importação de carros do mesmo continente ser mais económica em comparação com outras opções.

Company Name	Model Name	Price
Adam	Revo	2.100000e+05
Audi	A3	5.861667e+06
	A4	6.702318e+06
	A5	1.118810e+07
	A6	1.137738e+07
	...	
Toyota	Voxy	4.018182e+06
	Wish	2.110769e+06
	iQ	1.446667e+06
United	Bravo	8.597031e+05
Volvo	S40	1.275000e+06

Figure 2: Company Name e Model Names

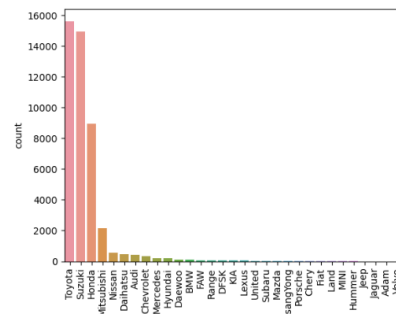


Figure 3: Count Values do Company Name

**Color** - Agrupamos as cores nos seguintes conjuntos: Dark, Bright, Warm, Cool. Ao ver a correlação entre as cores e o preço dos carros vimos uma tendência entre carros com cores mais claras e escuras.

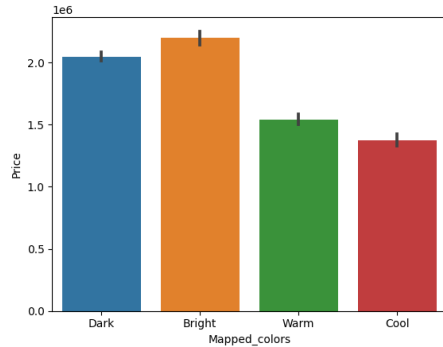


Figure 4: Colors Binned comparado com o Price

**Engine Capacity** - Através da observação da imagem abaixo vemos que apesar de não haver muita correlação entre ambos os atributos, o Price tende a subir consoante o aumento da Engine Capacity.

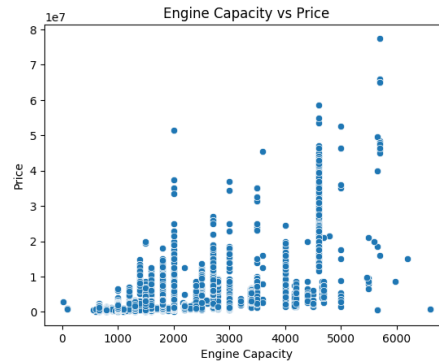
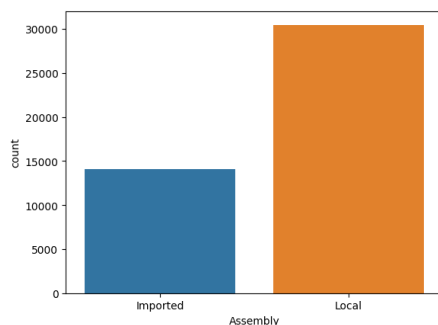
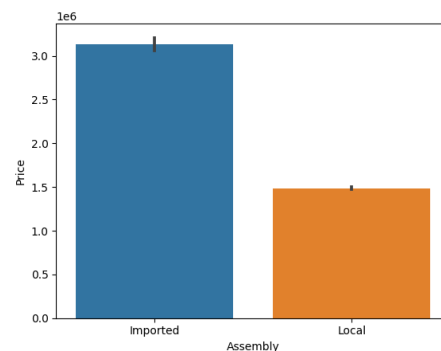


Figure 5: Engine Capacity comparado com o Price

**Assembly** - Apesar do valor "Imported" ter um menor número de ocorrências, é o que é normalmente associado a preços mais altos.

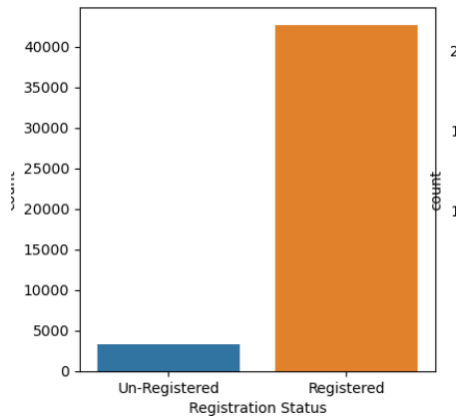


Count values do Assembly

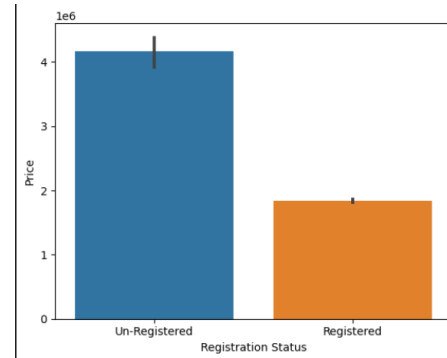


Assembly comparado com o Price

**Registration Status** - Chegamos à conclusão de que veículos não registrados têm um preço menos acessível em comparação com veículos registrados, o que resulta em um maior número de ocorrências para os carros registrados.



Count values do Registration Status



Registration Status comparado com o Price

**Body Type** - Após analisar esta feature, o nosso grupo concluiu que os tipos *SUV* e *Cross-Over* são os mais caros, tornando-os menos acessíveis à maioria da população e, como resultado, apresentam um número menor de ocorrências. Em contraste, os tipos *Hatchback* e *Sedan*, devido aos preços mais acessíveis, tornam-se escolhas predominantes para a maioria da população, refletindo assim um maior número de ocorrências. Quanto aos tipos *Van* e *Mini Van*, destinados não ao uso pessoal, mas sim às indústrias e atividades comerciais, apresentam um menor número de ocorrências.

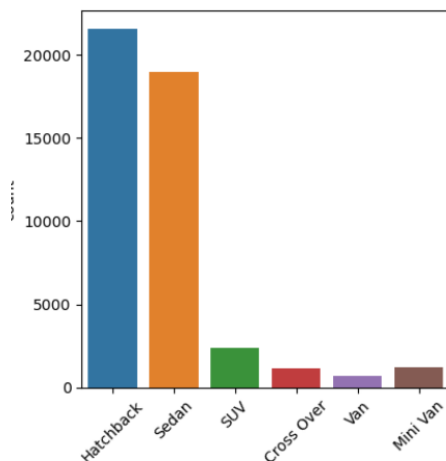


Figure 6: Count values do Body Type

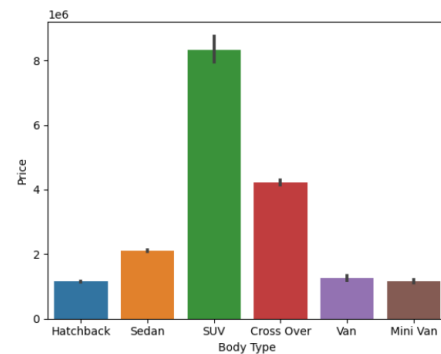


Figure 7: Body Type comparado com o Price

**Location** - Com uma análise a esta feature, o nosso grupo concluiu que a província de Punjab registra um grande número de ocorrências devido à sua atividade econômica robusta, que supera até mesmo a capital Islamabad. Isso se deve ao fato de Punjab ser um importante polo agrícola do país. Por outro lado, os preços são mais elevados na província de Islamabad, uma vez que esta é a capital do país e um centro de administração pública.

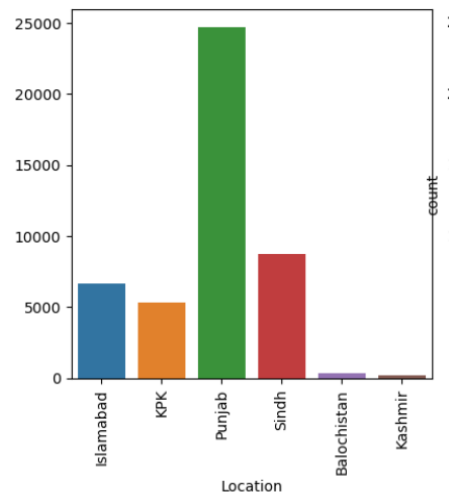


Figure 8: Count values do Location

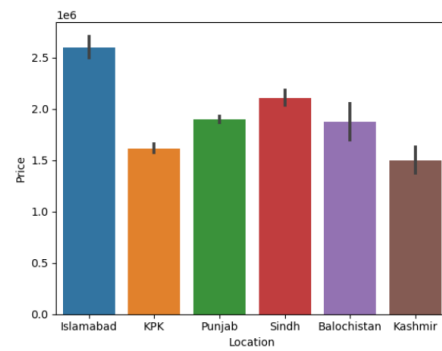


Figure 9: Location comparado com o Price

**Engine Type** - Ao explorar o Engine Type, percebemos que a categoria "Petrol" destaca-se, registrando um maior número de ocorrências. Isso ocorre devido aos carros movidos a gasolina apresentarem preços mais acessíveis, o que influencia diretamente nessa tendência.

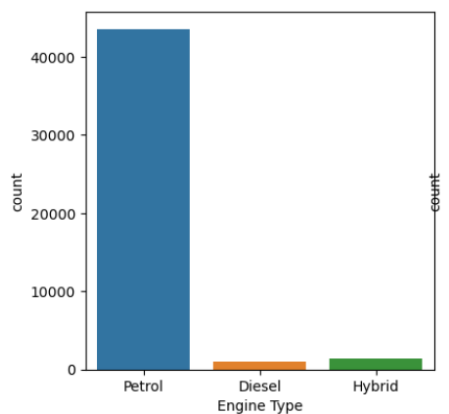


Figure 10: Count values do Engine Types

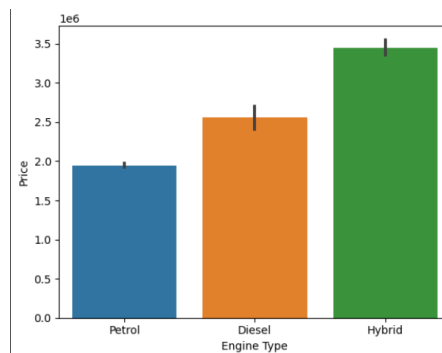


Figure 11: Engine Type comparado com o Price

**Kilometers** - Observamos que quantos menos quilômetros um carro tiver, maior o seu preço.



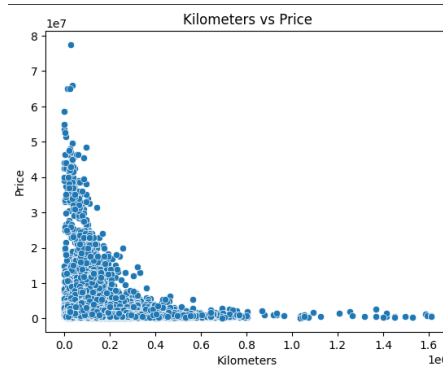


Figure 12: Kilometers comparado com o Price

**Model Year** - Concluimos que os carros mais recentes apresentam um custo mais elevado em relação aos carros antigos e que a frota automóvel é recente.

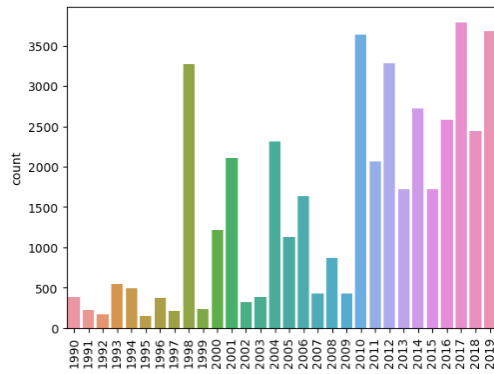


Figure 13: Count values do Model Year

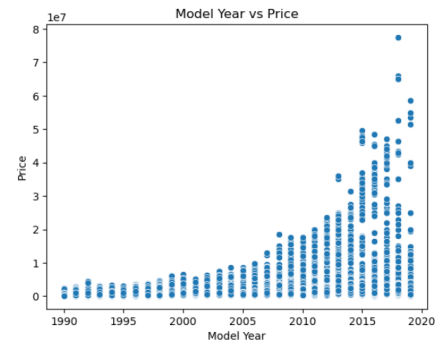


Figure 14: Model Year comparado com o Price

**Transmission Type** - Concluimos que os carros do tipo Automático são mais caros que os carros do tipo Manual devido à tecnologia implementada neste tipo de carros.

### 3.2 Preparação dos Dados

Numa fase inicial da preparação de dados efetuamos as seguintes alterações:

- Remoção da coluna ID já que não tem qualquer correlação com o target.
- Conversão das unidades da coluna *mileage* para kms.
- Remoção dos missing values da coluna *Color* (valores com *Assembly* ou *Unlisted*) e binning da mesma tendo em conta o tom das cores.
- Troca das cidades representadas na coluna *Location* pelo seu respetivo país.
- Realizamos encode através do label encoder nas variáveis categóricas, nomeadamente Company Name, Model Name, Location, Engine Type, Color, Assembly, Body Type, Transmission Type, Registration Status, Mapped<sub>Color</sub> e Mapped<sub>Locations</sub>.

Prosseguimos a efetuar uma matriz de correlação:



```
cars[cars.columns[1:]].corr()['Price'][:]
```

```
✓ 0.0s
```

Model Name	0.050729
Model Year	0.217971
Location	-0.036827
Kilometers	-0.188354
Engine Type	-0.069132
Engine Capacity	0.646900
Color	-0.020958
Assembly	-0.264244
Body Type	0.155299
Transmission Type	-0.340638
Registration Status	0.204849
Price	1.000000
Mapped_colors	-0.048264
Mapped_locations	0.040825

Name: Price, dtype: float64

Após observar os resultados da matriz, decidimos remover os atributos que menor correlação tinham com o target. Alguns desses atributos incluem a *Color*, *Location* e *Mapped\_locations*.

### 3.3 Modelação e Avaliação

Depois de efetuar toda a exploração e tratamento do dataset prosseguimos para a modelação do mesmo. Começamos por efetuar um *one-hot encoding* nas seguintes colunas do dataset: *Company Name*, *Model Name*, *Engine Type*, *Assembly*, *Body Type*, *Transmission Type*, *Registration Status*, *Mapped\_colors*, de forma a representar todos os atributos categóricos em formato numérico.

Após esse processo ter terminado testamos vários modelos no nosso dataset através do uso de *Cross Validation*. O grupo procurou obter o *MSE*, *MAE*, *R2*, e *RMSE* de forma a poder determinar quais os melhores modelos que utilizamos. Na imagem representada em baixo é possível visualizar todos os resultados que obtemos:

Modelo	MSE	MAE	R2	RMSE
Linear Regression	3474524859815.89	853524.68	0.5872	1863278.32
Decision Tree Regression	643671463730.34	214282.74	0.9235	795937.50
Random Forest Regression	398373454581.18	178256.48	0.9526	623467.99
ExtraTrees	385339071273.02	185253.96	0.9541	615424.91
GradientBoosting	528258720785.15	256537.47	0.9371	723234.04
XGBoost	393329869364.02	179118.35	0.9533	621673.20
LightGBM	447983127367.79	195587.32	0.9466	662218.36
CatBoost	329608044664.47	175870.97	0.9607	569247.27

Table 1: Resultados das Métricas de Desempenho para Diferentes Modelos

Procuramos implementar dois modelos que não eram baseados em árvores, no entanto, enfrentamos dificuldades, visto que não conseguimos atingir a convergência para ambos.

Interpretando os resultados obtidos, tal como evidenciado na tabela acima, concluímos que os três melhores modelos são o *CatBoost*, seguido pelo *XGBoost* e o *Random Forest Regression*. Posteriormente, optamos por criar um notebook dedicado a cada um destes três modelos, permitindo uma análise detalhada de cada um e a otimização dos melhores hiperparâmetros específicos para cada caso.

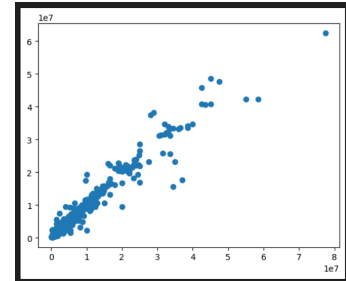
Após aplicarmos um Grid Search para obter os melhores parâmetros obtemos os seguintes resultados:

---

## Random Forest

```
Fitting 5 folds for each of 51 candidates, totalling 255 fits  
(0.9359812961206183, {'max_depth': 17, 'n_estimators': 50})
```

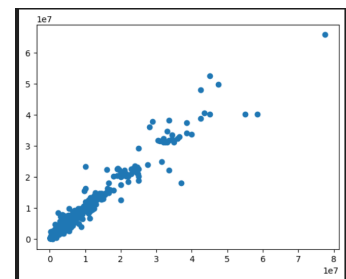
```
MAE: 174066.6206584888  
MSE: 376488777029.1872  
RMSE: 613586.8129524845
```



## XGBoost

```
Fitting 5 folds for each of 135 candidates, totalling 675 fits  
(0.9482115015219016,  
 {'learning_rate': 0.3, 'max_depth': 4, 'n_estimators': 100})
```

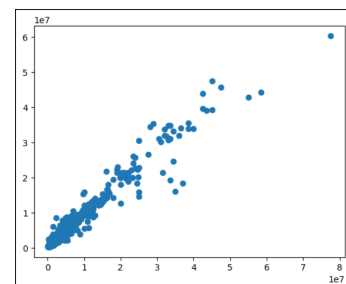
```
MAE: 200046.5646419249  
MSE: 353580343902.98236  
RMSE: 594626.222010922
```



## CatBoost

```
(0.9447583485762706, {'depth': 10, 'iterations': 50, 'learning_rate': 0.1})
```

```
MAE: 214617.50332635117  
MSE: 410888318351.52936  
RMSE: 641005.7085171156
```



Pelas imagens acima expostas, o modelo que teve melhor performance foi o que utilizou o XGBoost. Como podemos reparar os valores obtidos após a primeira vez que

---

se executou o modelo (para descobrir os 3 melhores modelos) foram melhores. O grupo suspeita que este decréscimo nos resultados tem a ver com o menor overfitting o que acaba por ser algo positivo sendo que o nosso modelo está mais apto para prever o preço de novos dados.

## 4 Dataset de competição

### 4.1 Compreensão do contexto

Com o âmbito de catapultar a influência das energias renováveis no dia a dia da nossa sociedade, o dataset de competição baseava-se na previsão da energia produzida por painéis solares na cidade de Braga com base em várias características meteorológicas e próprias dos painéis.

### 4.2 Preparação dos datasets

Os datasets que nos foram disponibilizados não estavam preparados da melhor maneira para começar a analisar os seus dados.

Por isso, primeiramente a equipa teve que juntar os 6 datasets disponibilizados em 2 datasets, um de treino e outro de teste.

O dataset de treino seria o resultado de juntar os seguintes datasets:

1. energia\_202109-202112.csv
2. energia\_202201-202212.csv
3. meteo\_202109-202112.csv
4. meteo\_202201-202212.csv

O dataset de teste seria o resultado de juntar os seguintes datasets:

1. energia\_202301-202304.csv
2. meteo\_202301-202304.csv

Esta junção deve-se à presença da **target feature** (Injeção na rede (kWh)) nos primeiros 2 datasets de energia e à falta desta feature no último dataset de energia.

A equipa, através de uma **API** de meteorologia disponível na internet, preencheu os missing values das features *sea\_level* e *grnd\_level* de ambos os datasets. Ainda tiramos proveito desta ferramenta para adicionar novas features importantes para um treino completo dos modelos construídos em fases posteriores, enumerando: *snowfall (cm)*, *wind gusts (km/h)*, *soil temperature (°C)*, *soil moisture (m³/m³)*, *short-wave radiation (W/m²)*, *direct radiation (W/m²)*, *diffuse radiation (W/m²)*, *direct normal irradiance (W/m²)*, *terrestrial radiation (W/m²)*, *shortwave radiation instant (W/m²)*, *direct radiation instant (W/m²)*, *diffuse radiation instant (W/m²)*, *direct normal irradiance instant (W/m²)*, *terrestrial radiation instant (W/m²)*. Por fim, usamos a API também para preencher dados de meteorologia em falta no dataset de teste entre as datas de **15/03/2023** até **04/04/2023**.

Com isto, tínhamos agora disponíveis 2 datasets prontos para as próximas tarefas: **treino.csv** e **teste.csv**.

Referência para a API mencionada: <https://open-meteo.com/en/docs/historical-weather-api>

**NOTA:** as features mencionadas que foram acrescentadas aos datasets originais não são foco do trabalho e, por isso, não vão ser discutidas no resto do relatório.

---

### 4.3 Exploração dos Dados

Nesta secção, serão abordados os métodos que o nosso grupo utilizou para explorar o conjunto de dados. Inicialmente, realizámos uma exploração mais superficial, onde analisámos os Data Types, Missing Values e características dos atributos. Em seguida, procedemos a uma exploração mais detalhada, utilizando várias representações visuais para comparar os atributos com a **target feature**, a partir das quais retirámos conclusões.

Assim, a nossa exploração começou considerando que o conjunto de dados é composto por **11.016 linhas e 35 features**. Como referido anteriormente, algumas destas features foram adicionadas com recurso à API, vamos apenas focar-nos nas features originais dos datasets fornecidos. Estas features podem ser divididas em 6 features energéticas e 15 features meteorológicas.

Features energéticas:

1. **Data** - o timestamp associado ao registo, ao dia;
2. **Hora** - a hora associada ao registo;
3. **Normal (kWh)** - quantidade de energia eléctrica consumida, em kWh e proveniente da rede eléctrica, num período considerado normal em ciclos bi-horário diários (horas fora de vazio);
4. **Horário Económico (kWh)** - quantidade de energia eléctrica consumida, em kWh e proveniente da rede eléctrica, num período considerado económico em ciclos bi-horário diários (horas de vazio);
5. **Autoconsumo (kWh)** - quantidade de energia eléctrica consumida, em kWh, proveniente dos painéis solares;
6. **Injeção na rede (kWh)** - quantidade de energia eléctrica injectada na rede eléctrica, em kWh, proveniente dos painéis solares.

Features meteorológicas:

1. **dt** - o timestamp associado ao registo;
2. **dt\_iso** a data associada ao registo, ao segundo;
3. **city\_name** - o local em causa;
4. **temp** - temperatura em °C;
5. **feels\_like** - sensação térmica em °C;
6. **temp\_min** - temperatura mínima sentida em °C;
7. **temp\_max** - temperatura máxima sentida em °C;
8. **pressure** - pressão atmosférica sentida em atm;
9. **sea\_level** - pressão atmosférica sentida ao nível do mar em atm;
10. **grnd\_level** - pressão atmosférica sentida à altitude local em atm;
11. **humidity** - humidade em percentagem;
12. **wind\_speed** - velocidade do vento em metros por segundo;

13. **rain\_1h** - valor médio de precipitação;
14. **clouds\_all** - nível de nebulosidade em percentagem;
15. **weather\_description** - avaliação qualitativa do estado do tempo.

Posteriormente verificamos os *Data Types* com que estávamos a trabalhar e a ocorrência de *Missing Values* no nosso conjunto de dados.

treino.dtypes	
dt	int64
dt_iso	object
city_name	object
temp	float64
feels_like	float64
temp_min	float64
temp_max	float64
pressure	int64
sea_level	float64
grnd_level	float64
humidity	int64
wind_speed	float64
rain_1h	float64
clouds_all	int64
weather_description	object
Data	object
Hora	int64
Normal (kWh)	float64
Horario Economico (kWh)	float64
Autoconsumo (kWh)	float64
Injecao na rede (kWh)	object

treino.isna().any()	
dt	False
dt_iso	False
city_name	False
temp	False
feels_like	False
temp_min	False
temp_max	False
pressure	False
sea_level	False
grnd_level	False
humidity	False
wind_speed	False
rain_1h	True
clouds_all	False
weather_description	False
Data	False
Hora	False
Normal (kWh)	False
Horario Economico (kWh)	False
Autoconsumo (kWh)	False
Injecao na rede (kWh)	False

Com esta análise, concluímos os seguintes Tipos de Dados dos atributos:

- **object** - Data, Injeção na rede (kWh), dt\_iso, city\_name, weather\_description;
- **int64** - dt, pressure, humidity, clouds\_all;
- **float64** - Hora, Normal (kWh), Horario Economico (kWh), Autoconsumo (kWh), temp, feels\_like, temp\_min, temp\_max, sea\_level, grnd\_level, wind\_speed, rain\_1h.

Para além disto, constatamos a presença de missing values apenas na feature **rain\_1h**.

#### 4.3.1 Relação das várias features com o target

**Normal (kWh)** - Pela imagem abaixo, podemos concluir que quanto mais baixo o valor deste atributo, mais alto tende a ser a Injeção na rede.

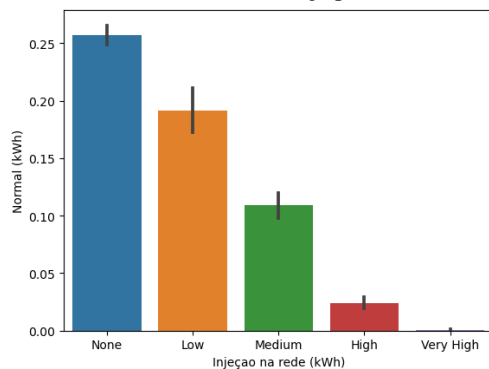


Figure 15: Normal kWh comparado com Injecao na Rede (kWh)

**temp** - Vendo a imagem observamos que os valores do atributo Injeção de Rede mais altos são associados a valores de *temp* altos e vice-versa.

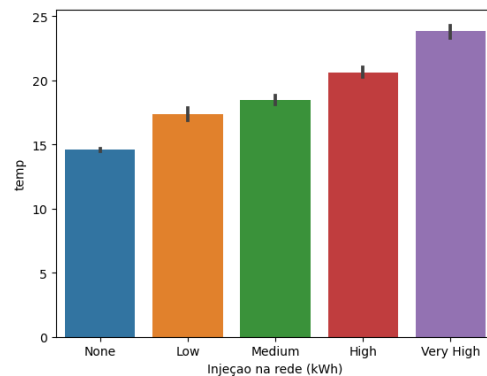


Figure 16: temp (temperatura) comparado com Injecao na Rede (kWh)

**AutoConsumo (kWh)** - Consoante o aumento do "Autoconsumo", a Injeção na rede aumenta, com exceção do valor "Very High" onde este valor não é o mais alto registado e tem valores ligeiramente mais altos do que os valores associados a uma injeção na rede de valor "Low".

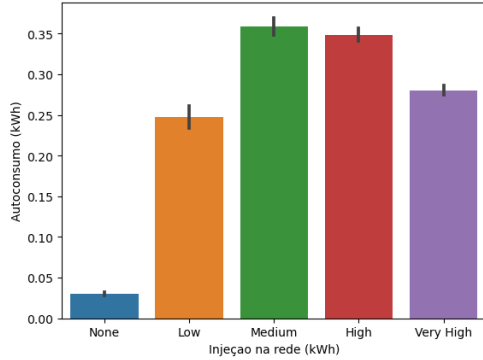


Figure 17: Autoconsumo kWh comparado com Injecao na Rede (kWh)

**Horario Economico (kWh)** - Podemos observar que os valores altos de Horario Economico são associados a um valor de "None" na Injeção na rede. O resto dos valores da Injeção na rede estão associados a um horario económico de "0" (com exceção do valor "Low").

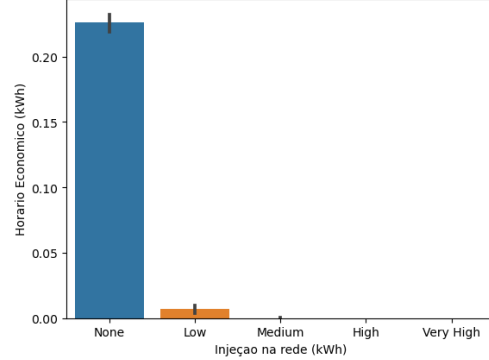


Figure 18: Horario Economico comparado com Injecao na Rede (kWh)

**clouds\_all** - Consoante o decréscimo do valor de clouds\_all, mais alto o valor da Injeção na rede.

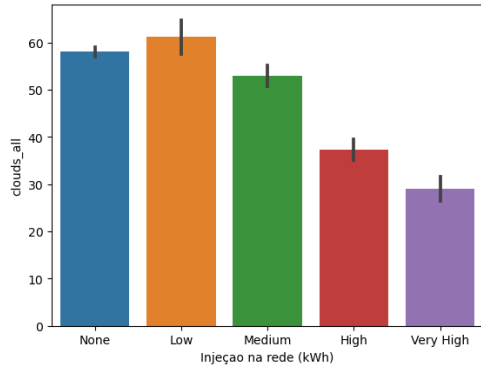


Figure 19: clouds\_all comparado com Injecao na Rede (kWh)

**humidity** - Consoante o decréscimo do valor de humidity, mais alto o valor da Injeção na rede.

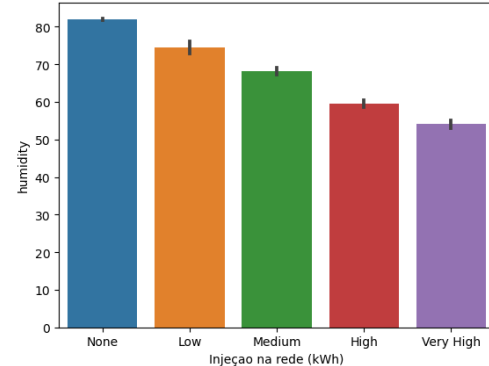


Figure 20: Humidity comparado com Injecao na Rede (kWh)

Selecionamos os gráficos que mais informação revelam sobre a target feature.

---

## 4.4 Tratamento dos Dados

Nesta secção, serão abordados os métodos que o nosso grupo utilizou para tratar o conjunto de dados. Tal como na exploração de dados, esta fase foi dividida em duas, numa fizemos o tratamento de missing values, criação de novas features (feature engineering), etc. na outra fizemos a remoção de colunas pouco correlacionadas com o **target** e o balanceamento do próprio dataset.

**Nota:** é importante reparar que alguns dos valores que vão ser apresentados podem ser diferentes do expectável por termos usado a **API** referida anteriormente para preencher vários missing values dos datasets.

### 4.4.1 Tratamento geral

A coluna **dt** é um timestamp, ou seja, representa um id para o tempo o que torna esta coluna inútil para a aprendizagem automática. Decidimos remover esta coluna. Assim como a feature **dt**, a feature **dt\_iso** é inútil, não pela natureza da feature em si, mas porque temos as features **Data** e **Hora** que podem substituir a mesma. Decidimos retirar esta feature.

A coluna **city\_name** tem apenas um valor (local) o que a torna numa feature inútil. Decidimos remover esta coluna.

### 4.4.2 Missing values

No início, o estado dos missing values era o seguinte:

treino.isna().sum()		teste.isna().sum()	
dt	0	dt	504
dt_iso	0	dt_iso	0
city_name	0	city_name	504
temp	0	temp	0
feels_like	0	feels_like	0
temp_min	0	temp_min	0
temp_max	0	temp_max	0
pressure	0	pressure	504
sea_level	0	sea_level	0
grnd_level	0	grnd_level	0
humidity	0	humidity	0
wind_speed	0	wind_speed	0
rain_1h	8732	rain_1h	2050
clouds_all	0	clouds_all	0
weather_description	0	weather_description	504
Data	0	Data	0
Hora	0	Hora	0
Normal (kWh)	0	Normal (kWh)	0
Horario Economico (kWh)	0	Horario Economico (kWh)	0
Autoconsumo (kWh)	0	Autoconsumo (kWh)	0
Injecao na rede (kWh)	0		

A feature **rain\_1h** apresenta imensos missing values em ambos os datasets representando pelo menos 80% de ambos os datasets. A maneira como decidimos corrigir estes missing values foi analisando a sua relação com a feature **weather\_description**.

treino[treino['rain_1h'].notnull()]['weather_description'].value_counts()		treino[treino['rain_1h'].isnull()]['weather_description'].value_counts()	
✓ 0.0s		✓ 0.0s	
weather_description		weather_description	
light rain	1784	sky is clear	3160
moderate rain	587	overcast clouds	2591
heavy intensity rain	64	broken clouds	1528
Name: count, dtype: int64		scattered clouds	1213
		few clouds	761
		Name: count, dtype: int64	

Através destas imagens conseguimos reparar que, sempre que não há missing values na feature **rain\_1h** (imagem da esquerda) choveu (porque a palavra "rain" está presente na feature **weather\_description**) e que sempre que há um missing value (imagem da direita) não choveu (a feature **weather\_description** apresenta um valor sem "rain" no nome). Ou seja, podemos partir do pressuposto que os missing values da coluna **rain\_1h** representam a falta de chuva.

Decidimos então preencher os missing values da feature **rain\_1h** com o valor 0.



---

Para tratar dos missing values da coluna *weather\_description* do dataset de teste, decidimos conceber um modelo de árvores de decisão para prever os mesmos. O código implementado foi o seguinte:

```
from sklearn.tree import DecisionTreeClassifier

label_encoder_wd = preprocessing.LabelEncoder()

teste_wd = teste.copy()

teste_wd['Data'] = label_encoder_wd.fit_transform(teste_wd['Data'])
teste_wd['Season'] = label_encoder_wd.fit_transform(teste_wd['Season'])
teste_wd['Month'] = label_encoder_wd.fit_transform(teste_wd['Month'])

treino_wd = teste_wd.dropna()
teste_wd = teste_wd[teste_wd['weather_description'].isnull()]

X_treino_wd = treino_wd.drop('weather_description', axis=1)
y_treino_wd = treino_wd['weather_description']

X_teste_wd = teste_wd.drop('weather_description', axis=1)

clf = DecisionTreeClassifier(random_state=183)
clf.fit(X_treino_wd, y_treino_wd)
predictions = clf.predict(X_teste_wd)

# Fill the missing values with the predictions
indices_of_missing = teste[teste['weather_description'].isnull()].index # list of the mis
for fill_index, dataframe_index in enumerate(indices_of_missing):
    teste.loc[dataframe_index, 'weather_description'] = predictions[fill_index]

predictions
```

#### 4.4.3 Feature Engineering

A partir da feature Data podemos criar duas novas features: *Season* e *Month*.

A ideia por detrás da criação destas features é a averiguação da correlação entre elas e a target feature.

Teoria: estações mais frias e com menos radiação solar (Inverno e Outono) podem implicar menos Injeção na rede e, pelo contrário, os meses mais quentes e com mais radiação solar (Verão e Primavera) podem implicar injeções na rede maiores. Para os meses o raciocínio é similar.

#### 4.4.4 Codificação das features categóricas

A codificação de features categóricas em features numéricas é vital para praticamente todo o tipo de modelos de classificação. Por esta razão, todas as features categóricas devem ser codificadas, com a exceção da **target feature**.

A codificação foi feita da seguinte maneira:

1. Codificação manual da feature *Season* feita através do uso de um map;
2. Codificação da feature *Data* com o recurso a **Label encoding**;

---

### 3. One-hot encoding da feature *weather\_description*.

#### 4.4.5 Correlação com o *target*

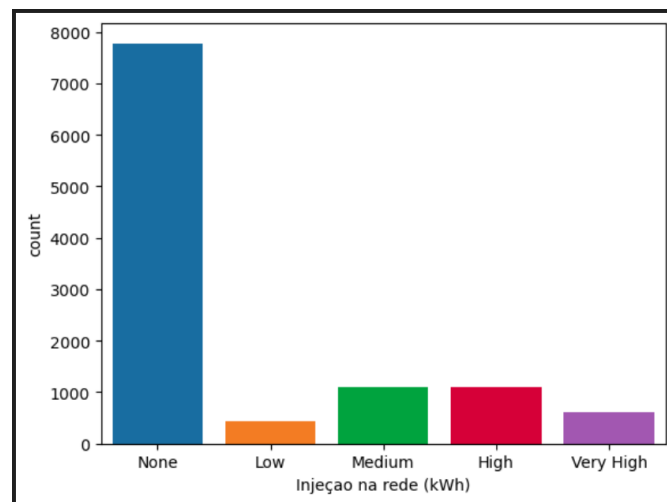
feels_like	0.451258	shortwave_radiation (W/m <sup>2</sup> )	0.858935
temp_min	0.444381	direct_radiation (W/m <sup>2</sup> )	0.836365
temp_max	0.485924	diffuse_radiation (W/m <sup>2</sup> )	0.640145
pressure	0.033233	direct_normal_irradiance (W/m <sup>2</sup> )	0.808014
sea_level	0.026055	terrestrial_radiation (W/m <sup>2</sup> )	0.797994
grnd_level	0.067209	shortwave_radiation_instant (W/m <sup>2</sup> )	0.872372
humidity	-0.564490	direct_radiation_instant (W/m <sup>2</sup> )	0.847506
wind_speed	0.046878	diffuse_radiation_instant (W/m <sup>2</sup> )	0.654760
rain_1h	-0.123628	direct_normal_irradiance_instant (W/m <sup>2</sup> )	0.814405
clouds_all	-0.200856	terrestrial_radiation_instant (W/m <sup>2</sup> )	0.811783
Data	-0.004499	Season	0.083415
Hora	0.081695	Month	-0.084943
Normal (kWh)	-0.260828	broken clouds	0.036119
Horario Economico (kWh)	-0.347107	few clouds	0.012001
Autoconsumo (kWh)	0.699685	heavy intensity rain	-0.039006
Injeção na rede (kWh)	1.000000	light rain	-0.100003
snowfall (cm)	NaN	moderate rain	-0.098663
wind_gusts_10m (km/h)	0.157285	overcast clouds	-0.078621
soil_temperature_0_to_7cm (°C)	0.386188	scattered clouds	0.031912
soil_moisture_0_to_7cm (m <sup>3</sup> /m <sup>3</sup> )	-0.224945	sky is clear	0.152949

As features *Data*, *Hora*, *wind\_speed*, *Month*, *Season*, *snowfall (cm)*, *sea\_level*, *grnd\_level* e *pressure* apresentam uma relação e correlação super fraca com o target, por esta razão decidimos retirar estas features.

A feature "snowfall (cm)" apresenta uma correlação de NaN porque esta coluna apresenta apenas um valor: 0.0. Ou seja, no período de registo dos datasets, 2021 até 2023, nunca nevou em Braga.

#### 4.4.6 Balanceamento do dataset

O desbalanceamento deste dataset era evidente e um problema grave desde o início. Podemos observar claramente este desbalanceamento com a próxima imagem:



Para corrigir este problema, o grupo decidiu seguir uma estratégia híbrida de under e over sampling.

O *undersampling* consiste em remover linhas da classe mais abundante (neste caso da classe **None**) com o objetivo de nivelar a abundância entre as classes.

O *oversampling* consiste em aumentar a quantidade de observações das classes menos abundantes.

---

Ambas as estratégias apresentam os seus problemas: no undersampling perdemos informação que pode ser vital e no oversampling acrescentamos informação que pode levar ao overfit dos modelos.

Tendo isto em mente, o grupo achou por bem seguir uma estratégia mista usando ambos os métodos referidos a cima. Ao usarmos uma estratégia híbrida reduzimos os riscos de cada estratégia usada e maximizamos os seus benefícios.

O código que implementa esta estratégia é o seguinte:

```
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline

# Sampling Pipeline
over_strategy={0:7777, 1:750, 2:1500, 3: 1500, 4:750}
under_strategy={0:6000, 1: 750, 2: 1500, 3: 1500, 4:750}
over = SMOTE(random_state=183, sampling_strategy=over_strategy)
under = RandomUnderSampler(random_state=183, sampling_strategy=under_strategy)
steps = [('over', over), ('under', under)]
pipeline = Pipeline(steps=steps)

X_treino = treino.drop('Injecao na rede (kWh)', axis=1)
y_treino = treino['Injecao na rede (kWh)']

X_treino_res, y_treino_res = pipeline.fit_resample(X_treino, y_treino)
```

## 4.5 Modelação e Avaliação

Nesta secção vamos abordar os vários modelos construídos pelo grupo para prever os dados que faltavam (Injeção na rede (kWh)) no dataset de teste.

Antes de mais, o nosso problema é um de classificação visto que a target feature *Injeção na rede (kWh)* é uma variável categórica. O que o grupo decidiu fazer foi explorar vários modelos de classificação (os mais conhecidos e estudados na UC) e comparar resultados.

Os modelos explorados foram:

1. **Decision tress;**
2. **Extra trees;**
3. **Random forest;**
4. **Extreme Gradient Boosting (XGBoosting);**
5. **Light Gradient Boosting Machine (LightGBM);**
6. **Logistic regression;**
7. **Redes neuronais;**
8. **Support vector machine.**

De notar que, diferentes modelos necessitam de diferentes preparações de dados e que por isso cada modelo foi desenvolvido num **Jupyter Notebook** diferente.









Em específico, modelos baseados em árvores como: decision trees, extra trees, random forest, XGBoosting e LightGBM, não necessitam de normalização de dados.

Por outro lado, modelos que são baseados no calculado de distâncias como: support vector machines, redes neuronais e logistic regression, necessitam da normalização dos dados para obterem modelos com melhor performance e convergências mais rápidas.

A questão da normalização dos dados é a única diferença entre os modelos de resto cada modelo faz a separação dos datasets (X\_treino, y\_treino e X\_teste) e treinam os modelos usando um grid search para chegarmos aos melhores hiperparâmetros de cada um. No final, são previstos os resultados.

#### 4.5.1 Performance dos diferentes modelos

Para analisar a performance dos diferentes modelos o melhor a fazer é analisar os resultados que obtivemos na plataforma **Kaggle**.

 <b>svmPrediction.csv</b> Complete (after deadline) · GoncaloMS · now	<b>0.79177</b>	<b>0.80177</b>	<input type="checkbox"/>
 <b>redesNeuronaisPrediction.csv</b> Complete (after deadline) · GoncaloMS · 18s ago	<b>0.76582</b>	<b>0.77958</b>	<input type="checkbox"/>
 <b>logRegPrediction.csv</b> Complete (after deadline) · GoncaloMS · 1m ago	<b>0.81455</b>	<b>0.82692</b>	<input type="checkbox"/>
 <b>lightBGMPrediction.csv</b> Complete (after deadline) · GoncaloMS · 2m ago	<b>0.86962</b>	<b>0.88313</b>	<input type="checkbox"/>
 <b>extraTreesPrediction.csv</b> Complete (after deadline) · GoncaloMS · 2m ago	<b>0.84493</b>	<b>0.85355</b>	<input type="checkbox"/>
 <b>randomForestPrediction.csv</b> Complete (after deadline) · GoncaloMS · 3m ago	<b>0.86708</b>	<b>0.87278</b>	<input type="checkbox"/>
 <b>dtPrediction.csv</b> Complete (after deadline) · GoncaloMS · 1h ago	<b>0.8481</b>	<b>0.85946</b>	<input type="checkbox"/>
 <b>xgbPrediction.csv</b> Complete (after deadline) · GoncaloMS · 1h ago	<b>0.86898</b>	<b>0.87721</b>	<input type="checkbox"/>

Comparando os vários modelos, podemos observar que o modelo que apresentou os melhores resultados foi o LightGBM (no print aparece com uma gralha).

Podemos ainda observar que, os 3 modelos de cima, modelos baseados em distâncias, foram os que apresentaram piores resultados e que os melhores modelos foram os baseados em árvores (os 5 modelos de baixo).

Refletindo sobre o trabalho desenvolvido para este dataset, como os dados que nos foram fornecidos não foram os melhores apresentando missing values, começos de datasets desfasados, features com pouco impacto no target, etc. a preparação dos datasets e tratamento de dados foi de vital importância para obter modelos com uma performance satisfatória. Podemos então concluir que tivemos uma preparação de datasets e dados muito boa.

## 4.6 Conclusão

Com a conclusão do projeto prático, o grupo considera ter realizado um trabalho satisfatório. Para além de aplicar os conhecimentos adquiridos durante as aulas, o que contribuiu para uma consolidação mais robusta do conteúdo estudado, o grupo reconhece a importância do projeto em destacar a necessidade de prestar atenção a detalhes que poderiam ser negligenciados anteriormente. Houve a percepção de que, em alguns momentos, a tentativa de adicionar informações úteis e remover o que era considerado desnecessário resultava, na realidade, num overfitting do modelo.

No entanto, o grupo reconhece a valiosa lição de que é benéfico experimentar coisas novas, e que a prática, tentativa e persistência são fundamentais para alcançar a perfeição. Ao encerrar este projeto, o grupo acredita ter consolidado de maneira significa-

---

tiva o conteúdo abordado ao longo do semestre na disciplina de Dados e Aprendizagem Automática. Além disso, destacam a oportunidade de desenvolver habilidades interpessoais e enfrentar desafios e adversidades, contribuindo assim para um crescimento pessoal contínuo.