Universidade do Minho
Escola de Engenharia

Trabalho Prático

Modelos de Machine Learning

Dados e Aprendizagem Automática
Mestrado em Engenharia Informática

Grupo 12

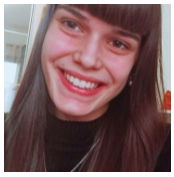
Gabriela Santos Ferreira da Cunha - pg53829

Gonçalo Semelhe Sousa Braga - pg53845

Nuno Guilherme Cruz Varela - pg54117

Rita Costa Dantas - pg51605



janeiro, 2024

Conteúdo

1 Introdução

No âmbito da unidade curricular de Dados e Aprendizagem Automática, foi-nos proposta a conceção e desenvolvimento de um projeto de *Machine Learning* utilizando, entre outros, os modelos de aprendizagem abordados ao longo do semestre.

Este projeto é dividido em duas tarefas, onde o objetivo de cada consiste em:

- Explorar, analisar e preparar um *dataset*, extraindo conhecimento relevante no contexto do problema em questão;
- Conceber e otimizar diversos modelos de *Machine Learning*;
- Realizar uma análise crítica dos resultados.

A primeira tarefa é dedicada ao *dataset* escolhido pelo grupo, "Rain in Australia", disponível no Kaggle. A segunda tarefa corresponde ao *dataset* fornecido pela equipa docente, relativo à energia solar.

2 Metodologia

Para a realização de ambas as tarefas, o grupo seguiu a metodologia CRISP-DM que oferece uma estrutura abrangente e sistemática que pode ser aplicada numa variedade de setores e contextos. Esta metodologia é constituída pelas seguintes etapas:

1. ***Bussiness understanding*:** definição dos objetivos do negócio que orientam o projeto e transformação dos mesmos em metas específicas;
2. ***Data understanding*:** recolha dos dados relevantes e exploração da natureza dos dados, identificando padrões, características e problemas potenciais;
3. ***Data preparation*:** limpeza e transformação dos dados para atender aos requisitos do modelo, incluíndo tratamento de *missing values*, *outliers*, inconsistências e *feature engineering*;
4. ***Modeling*:** escolha e aplicação das técnicas de modelação adequadas, avaliação, ajuste e validação dos modelos para otimizar o seu desempenho;
5. ***Evaluation*:** avaliação do desempenho do modelo em relação aos objetivos do negócio, iterando nesta fase e na anterior, conforme necessário;
6. ***Deployment*:** implementação do modelo e monitorização do seu desempenho em produção.

3 Tarefa 1

3.1 Contextualização

Neste trabalho direcionado ao *dataset* escolhido pelo grupo, procuramos criar um modelo que ofereça uma previsão precisa do tempo para o próximo dia na Austrália. Fundamentalmente, previsões meteorológicas precisas afetam a segurança, economia, agricultura, viagens, energia, meio ambiente e decisões pessoais de estilo de vida. A sua importância reside no seu impacto abrangente em múltiplos setores, permitindo um melhor planeamento, gestão de recursos e redução de riscos. Por exemplo:

- **Para segurança e preparação para desastres:** Previsões precisas podem salvar vidas e reduzir danos, fornecendo alertas precoces e preparação para eventos climáticos extremos, como furacões, tempestades ou ondas de calor;
- **Para agricultura e segurança alimentar:** Os agricultores dependem das previsões meteorológicas para o planeamento de culturas, irrigação e controlo de pragas. As previsões ajudam a melhorar os rendimentos e garantir a segurança alimentar.

3.2 *Dataset*

O *dataset* selecionado inclui informação sobre 10 anos de observações diárias das condições meterológicas de vários locais ao longo da Austrália, sendo constituído por 145460 registos e 23 *features* que são as seguintes:

Feature	Descrição
Date	Data da observação
Location	Localização da estação meteorológica
MinTemp	Temperatura mínima do dia em ºC
MaxTemp	Temperatura máxima do dia em ºC
Rainfall	Quantidade de precipitação ocorrida durante o dia em mm
Evaporation	Evaporação em mm
Sunshine	Número de horas de sol
WindGustDir	Direção da rajada de vento mais forte registada durante o dia, recorrendo aos pontos cardeais
WindGustSpeed	Velocidade da rajada de vento mais forte registada durante o dia em km/h
WindDir9am	Direção do vento às 09:00h
WindDir3pm	Direção do vento às 15:00h
WindSpeed9am	Velocidade do vento medida nos 10 minutos anteriores às 09:00h, em km/h
WindSpeed3pm	Velocidade do vento medida nos 10 minutos anteriores às 15:00h, em km/h
Humidity9am	Percentagem de humidade registada às 09:00h
Humidity3pm	Percentagem de humidade registada às 15:00h
Pressure9am	Pressão atmosférica em hpa reduzida ao nível médio do mar às 09:00h
Pressure3pm	Pressão atmosférica em hpa reduzida ao nível médio do mar às 15:00h
Cloud9am	Fração do céu, em oktas, coberta por nuvens às 09:00h
Cloud3pm	Fração do céu, em oktas, coberta por nuvens às 15:00h
Temp9am	Temperatura registada às 09:00h em ºC
Temp3pm	Temperatura registada às 15:00h em ºC
RainToday	Boolean que indica se houve precipitação (mm) que excedesse 1mm
RainTomorrow	Boolean que indica se houve precipitação (mm) no dia seguinte que excedesse 1mm

Figura 1: *Features* do *dataset* e respetiva descrição.

Sendo assim, o principal objetivo é prever se chove no dia seguinte, utilizando, para isso, modelos de classificação treinados que têm como *target* a variável "RainTomorrow". Para além da previsão desta classe, decidimos prever também a tempetura máxima ("TempMax") através de técnicas de regressão.

3.3 Exploração dos Dados

Quando lidamos com dados que não conhecemos, a primeira questão que surge é "quais e que tipo de dados é que estamos a lidar? ". Para podermos desenvolver os nossos modelos com a maior eficácia possível, é crucial ter uma compreensão completa dos dados do problema. Assim, começamos por efetuar uma análise sobre os dados. De notar que neste relatório estarão apenas presentes os aspetos e conclusões, a nosso ver, mais importantes. Todas as outras conclusões estão devidamente documentadas no *notebook*.

Tipicamente, começamos por extrair uma visão geral dos dados em termos do número de observações, tipos de características, *missing rate* e percentagem de observações duplicadas. Em relação às observações duplicadas, foi importante procurar particularmente por pares (Date, Location) duplicados, sendo que queremos garantir que, para uma determinada localização, apenas existe uma entrada por cada dia.

Relativamente às estatísticas gerais dos dados, estas ajudam a compreender a gama de valores típica, a variabilidade e a tendência central de vários parâmetros meteorológicos. As seguintes conclusões foram obtidas:

- ***Temperature range:*** A temperatura mínima registada varia de -8.5°C a 33.9°C, enquanto a temperatura máxima varia de -8.8°C a 8.1°C, indicando uma ampla variação de temperaturas no conjunto de dados;
- ***Precipitation:*** A precipitação média é de 2.36 mm e o máximo registrado é de 371 mm. A maioria dos valores registrados (percentil 75) está abaixo de 0.8 mm, indicando uma grande proporção de dias com pouca ou nenhuma precipitação;
- ***Evaporation and sunlight:*** Existem diferenças estatisticamente significativas entre os valores máximos de evaporação (15 mm), horas de sol (15 horas) e o percentil 75, indicando a presença de possíveis condições climáticas anormais ou extremas;
- ***Wind speed:*** Varia entre 6 km/h e 135 km/h. As velocidades médias do vento às 9:00h e 15:00h foram 13 km/h e 19 km/h, respectivamente, indicando condições médias de vento moderado;
- ***Humidity:*** Houve uma diferença significativa nos níveis de umidade entre 9:00h e 15:00h, sendo os níveis mais altos pela manhã (média de 68.9) do que à tarde (média de 51.5);
- ***Atmospheric pressure:*** As leituras médias de pressão às 9:00h e 15:00h foram 1017.6 hPa e 1015.3 hPa, respetivamente, com intervalos interquartis relativamente constantes, mostrando uma distribuição de pressão estável.

Um dos aspetos mais preponderantes na criação dos modelos de aprendizagem bem como na sua afinação, é a quantidade de *missing values* que as observações possuem. Em casos mais drásticos, os *missing values* podem até reduzir a eficiência e a capacidade de aprendizagem dos modelos de aprendizagem. Visto isto, é importante conhecermos os *missing values* que as observações realizadas possuem.

- As colunas "Evaporation", "Sunshine", "Cloud9am" e "Cloud3pm" contêm uma quantidade significativa de missing values, sugerindo potenciais lacunas na precisão dessas variáveis.
- As colunas "MinTemp", "MaxTemp", "Rainfall", etc., também têm alguns valores em falta, contudo, em menor quantidade do que as colunas mencionadas anteriormente.

Date	0.000000
Location	0.000000
MinTemp	1.020899
MaxTemp	0.866905
Rainfall	2.241853
Evaporation	43.166506
Sunshine	48.009762
WindGustDir	7.098859
WindGustSpeed	7.055548
WindDir9am	7.263853
WindDir3pm	2.906641
WindSpeed9am	1.214767
WindSpeed3pm	2.105046
Humidity9am	1.824557
Humidity3pm	3.098446
Pressure9am	10.356799
Pressure3pm	10.331363
Cloud9am	38.421559
Cloud3pm	40.807095
Temp9am	1.214767
Temp3pm	2.481094
RainToday	2.241853
RainTomorrow	2.245978

dtype: float64

(a) Percentagem de *missing values* por *feature*.

Date	0
Location	0
MinTemp	1485
MaxTemp	1261
Rainfall	3261
Evaporation	62790
Sunshine	69835
WindGustDir	10326
WindGustSpeed	10263
WindDir9am	10566
WindDir3pm	4228
WindSpeed9am	1767
WindSpeed3pm	3062
Humidity9am	2654
Humidity3pm	4507
Pressure9am	15065
Pressure3pm	15028
Cloud9am	55888
Cloud3pm	59358
Temp9am	1767
Temp3pm	3609
RainToday	3261
RainTomorrow	3267

dtype: int64

(b) Frequência de *missing values* por *feature*.

Figura 2: *Missing values*.

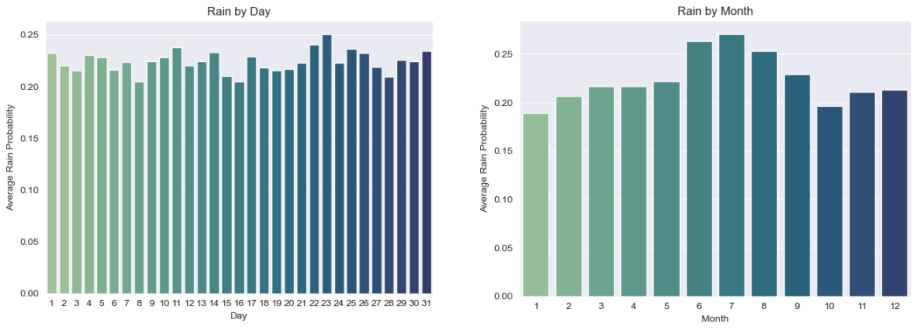
Procedendo à analise da distribuição dos dados de cada atributo numérico, apresentada na figura 13, e ao *boxplot* da figura 14, conseguimos visualizar os *outliers* de cada *feature*. Estes *outliers* terão de ser posteriormente analisados para perceber a sua natureza e potencial impacto no modelo.

Durante a análise das variáveis categóricas, foi notável um desequilíbrio significativo entre a probabilidade de ocorrência de chuva e a ausência dela (figura 15). A probabilidade de chuva foi substancialmente menor do que a probabilidade de ausência de chuva, apontando para um desequilíbrio entre as classes.

Para visualizar a correlação entre as variáveis, começamos por substituir os valores categóricos por equivalentes numéricos. Os resultados gerais encontram-se na figura 16. De notar que foram também elaborados alguns gráficos para analisar a correlação entre algumas variáveis específicas. Neste caso, podemos

visualizar correlações fortes (próximas de 1 - >0.8) entre as temperaturas máximas e mínimas diárias e as temperaturas registadas às 9h e 15h, assim como entre a pressão registada entre as 9h e 15h.

Por último, não conseguimos analisar nenhuma tendência entre a probabilidade de chover e o dia do mês. Já em relação ao mês, verificamos a sazonalidade da chuva, onde a probabilidade de chover é maior no inverno australiano (meses de junho, julho e agosto) e menor no verão.



(a) Chuva consoante o dia.

(b) Chuva consoante o mês.

Figura 3: Chuva consoante o dia e o mês.

Realçamos, ainda, que voltamos a efetuar uma pequena exploração após a preparação dos dados, especialmente para voltarmos a analisar as correlações depois do tratamento dos dados.

3.4 Preparação dos Dados

Apesar de realizarmos diversas preparações de dados, todas tiveram alguns comportamentos comuns. Neste sentido, em todas as preparações é realizada a separação da *feature* "Date", onde apenas aproveitamos o valor do mês, sendo que tanto o dia como o ano não trazem conhecimento relevante ao modelo. Para além disso, adotamos a utilização do *label encoding*, para converter as variáveis categóricas em numéricas, em relação ao *one-hot*, após observarmos que a primeira abordagem nos trazia melhores resultados. Quanto aos *outliers*, nos modelos de classificação, obtivemos melhores resultados com a inclusão destes, o que nos indica que se tratam de valores significativos para a previsão dos modelos. Já para os modelos de regressão, estes *outliers* aumentavam a variabilidade residual e exerciam uma influência desproporcional nos coeficientes, o que nos levou a efetuar a remoção dos mesmos.

3.4.1 Tratamento de *Missing Values*

Inicialmente, começamos por efetuar o tratamento dos *missing values* de *features* numéricas através da imputação de valores como a média e a mediana. Para as *features* com um menor número de *outliers*, foi imputada a média. Para

as *features* com uma distribuição *skewed*, foi utilizada a mediana, devido à elevada percentagem de *outliers*, uma vez que a média é bastante sensível a valores extremos, sendo recomendada para dados com poucos *outliers*.

Relativamente aos *missing values* das *features* categóricas ("WindGustDir", "WindDir9am" e "WindDir3pm"), utilizamos valores aleatórios extraídos da distribuição dos valores observados em cada variável, por forma a promover uma abordagem mais realista em comparação com os métodos simples de imputação, uma vez que, desta forma, preservamos a distribuição global da variável, mantendo as suas propriedades estatísticas.

Contudo, efetuamos diferentes preparações, onde o tratamento dos *missing values* é feito com a imputação de dados recolhidos de uma API - OpenMeteo Historical Weather API. Para isto, efetuamos várias *calls*, englobando as *locations* presentes no nosso *dataset*, e conseguimos recolher dados meteorológicos sobre cada dia entre o início de 2007 e o fim de 2017: a temperatura máxima e mínima, a quantidade de precipitação, a velocidade e direção das rajadas de vento, o número de horas de sol e a evaporação. Para além disso, foram recolhidos dados por cada hora de cada dia, para preencher os *missing values* relativos à porção de céu coberto por nuvens, pressão, humidade e temperatura que são apresentadas no nosso *dataset* ao nível da hora (9am e 3pm) e não do dia.

Contudo, para adaptar os dados recolhidos aos nossos foi necessário efetuar alguma manipulação sobre estes:

- substituição dos *location ids* pelo nome da localização em questão;
- renomeação de certas colunas para estarem de acordo com as do *dataset* principal;
- remoção dos dados relativos a horas diferentes das 09:00h ou 15:00h, sendo que a informação recolhida ao nível da hora é apenas necessária para esses horários;
- transformação de segundos para horas, sendo que a *feature* "Sunshine" apresenta o número de horas de sol por dia e a API apresentava esta medição em segundos;
- conversão da percentagem do céu coberta por nuvens para oktas;
- conversão das direções de graus para pontos cardeais.

Por último, após o tratamento efetuado sobre os dados da API, pudemos, então, juntar estes dados através de um *left merge* pelas *features* "Date" e "Location", preenchendo os valores em falta.

3.4.2 Desbalanceamento na *Target Feature*

Como foi possível observar na exploração dos dados, existe um grande desbalanceamento das classes presentes na *target feature* das técnicas de classificação, sendo este um ponto importante a resolver durante a preparação, acabando por

se tornar no maior desafio desta fase. De notar que este tratamento apenas foi necessário para as preparações realizadas para os modelos de classificação, mantendo-se o desbalanceamento nos problemas de regressão. Este desbalanceamento da *target feature* causa uma incapacidade do modelo conseguir prever a classe minoritária, algo que não é desejável. Deste modo, foram abordadas diferentes técnicas - *undersampling* da classe maioritária e *oversampling* da classe minoritária.

O *undersampling* descarta instâncias da classe maioritária. Por um lado é reduzida a quantidade de dados, o que pode acelerar o processo de treino e reduzir o *overfit*, especialmente em grandes conjuntos de dados. Todavia, esta abordagem pode resultar na perda de informações relevantes e a quantidade de dados mantida pode não ser suficiente para o modelo aprender, resultando numa menor *performance* deste perante dados desconhecidos. Para a implementação do *undersampling*, recorremos ao RandomUnderSampler devido à sua simplicidade que acaba também por beneficiar o tempo total de treino do modelo.

O *oversampling* cria cópias adicionais das instâncias da classe minoritária, aumentando a sua representação nos dados. A grande vantagem desta técnica é que nenhuma informação é perdida, pois continuam a ser usados todos os exemplos existentes da classe minoritária. Contudo, é preciso ter em conta os dados que são inseridos, uma vez que grandes variações nas propriedades estatísticas de cada variável podem prejudicar a aprendizagem do modelo. Para a implementação do *oversampling*, recorremos ao SMOTE que cria novas instâncias interpolando entre exemplos da classe minoritária existentes, o que é vantajoso em relação a outras técnicas mais simples, sendo que o simples aumento do número de instâncias sem diversificação pode levar a um ajuste excessivo a exemplos específicos, i.e *overfitting*.

Por último, nas seguintes tabelas, resumimos as diferentes preparações de dados realizadas:

Preparação	***Missing Values***		**Desbalanceamento**
	Numéricas	**Categóricas**	
1	Média/Mediana	Imputação Aleatória	-
2	API	API	-
3	Média/Mediana	Imputação Aleatória	Undersampling
4	API	API	Undersampling
5	Média/Mediana	Imputação Aleatória	Oversampling
6	API	API	Oversampling

Tabela 1: Preparações de dados para classificação.

Preparação	***Missing Values***	
	Numéricas	**Categóricas**
1	Média/Mediana	Imputação Aleatória
2	API	API

Tabela 2: Preparações de dados para regressão.

3.5 Modelos Desenvolvidos

Relativamente à modelação, utilizamos partições de 70% e 30% para treino e teste respetivamente para ambas as colunas que desejavamos prever. De maneira a encontrar os melhores parâmetros para o modelo, recorremos à técnica do *GridSearchCV* com 5 *folds* para *cross validation*. Assim, para esta fase do trabalho prático criamos os seguinte modelos: **Classificação** -> Decision Tree, Neural Network, Random Forest, XGBoost, entre outros; **Regressão** -> Decision Tree, Neural Network, Random Forest, XGBoost, entre outros;

Nas secções seguintes, iremos abordar os modelos que deram melhores resultados e consideramos mais relevantes.

3.5.1 Classificação

Decision Tree

Neste modelo utilizamos o grid search para otimização de hiperparâmetros. Para a grid dos parâmetros, utilizamos os seguintes:

```
param_grid = {
    'criterion': ['gini', 'entropy', 'log_loss'],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

De seguida criou-se o *estimator* do Decision Tree Classifier com `random_state` de 2023. Por fim, criamos o modelo passando o *estimator*, a *grid* dos parâmetros, o *refit* a True, com $cv = 5$ e o *score* do modelo calculado com base na métrica "f1".

```
    clf = DecisionTreeClassifier(random_state=2023)
    grid_search = GridSearchCV(estimator=clf,
    param_grid=param_grid, refit=True,
    verbose=3, cv=5,
    scoring='f1', n_jobs=-1)
```

Random Forest

Neste modelo utilizamos o *grid search* para otimização de hiperparâmetros. Para a *grid* dos parâmetros, utilizamos os seguintes:

```
param_grid = {
    'n_estimators': [50, 100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

De seguida, criou-se o *estimator* do Random Forest Classifier com `random_state` de 2023. Por fim, criamos o modelo passando o *estimator*, a *grid* dos parâmetros, o *refit* a True, com $cv = 5$ e o *score* do modelo calculado com base na métrica "f1".

```
    model = RandomForestClassifier(random_state=2023)
    grid_search = GridSearchCV(estimator=model,
    param_grid=param_grid,
    refit=True, verbose=2,
    cv=5, scoring="f1",
    n_jobs=-1)
```

XGBoost

Neste modelo utilizamos o *grid search* para otimização de hiperparâmetros. Para a *grid* dos parâmetros, utilizamos os seguintes:

```
param_grid = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.1, 0.01],
    'max_depth': [3, 4, 5],
    'gamma': [0, 0.1, 0.01],
}
```

De seguida criou-se o *estimator* do XGBClassifier com `random_state` de 2023. Por fim, criamos o modelo passando o *estimator*, a *grid* dos parâmetros, o *refit* a True, com $cv = 5$ e o *score* do modelo calculado com base na métrica "f1".

```
    model = XGBClassifier(random_state=2023)

    grid_search = GridSearchCV(estimator=model,
    param_grid=param_grid,
    refit=True, verbose=3,
    cv=5, scoring='f1',
    n_jobs=-1)
```

Neural Network - Multilayer Perceptron

De forma a prever a *target feature* "RainTomorrow", procedemos também à construção de uma rede neuronal *multilayer perceptron*. Primeiramente, começamos por normalizar todos os dados por forma a extrair o máximo rendimento da rede neuronal. Como estamos perante uma problema de classificação binário, a última camada da rede terá de possuir um neurónio com a função de ativação *sigmoid*. Desta forma, construímos a rede da seguinte forma:

```
def build_model(activation = 'relu', learning_rate = 0.001):
    model = Sequential()
    model.add(Dense(16, input_dim = X.shape[1], activation = activation))
    model.add(Dense(8, activation = activation))
    model.add(Dense(1, activation = 'sigmoid')) # output

    #Compile the model
    model.compile(
        loss = 'binary_crossentropy',
        optimizer  = tf.optimizers.Adam(learning_rate),
        metrics = ['accuracy'])
    return model
```

Após a construção da rede, prosseguimos à utilização de um GridSearchCV com 5 *folds* de *cross validation* com o objetivo de encontrar o melhor *optimiser* para a rede. Por vezes, o processo de afinar uma rede neuronal torna-se difícil pelo facto de possuir muitos parâmetros que podem ser afinados. Apesar do grupo ter tentado cumprir este requisito, é um dos pontos que poderia ser melhorado.

3.5.2 Regressão

Decision Tree

Neste modelo utilizamos o *grid search* para otimização de hiperparâmetros. Para a *grid* dos parâmetros, utilizamos os seguintes:

```
param_grid = {
    'max_depth': [5, 10, 15, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2', None]
}
```

De seguida, criou-se o *estimator* do Decision Tree Regressor com `random_state` de 2023. Por fim, criamos o modelo passando o *estimator*, a *grid* dos parâmetros, o *refit* a True, com $cv = 5$ e o *score* do modelo calculado com base na métrica "neg_mean_squared_error".

```
    model = DecisionTreeRegressor(random_state=2023)
    grid_search = GridSearchCV(model, param_grid,
    cv=5, verbose=2,
    refit=True,
    scoring='neg_mean_squared_error',
    n_jobs=-1)
```

Random Forest

Neste modelo utilizamos o *grid search* para otimização de hiperparâmetros. Para a *grid* dos parâmetros, utilizamos os seguintes:

```
param_grid = {
    'n_estimators': [50, 100, 150, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

De seguida, criou-se o *estimator* do Random Forest Regressor com `random_state` de 2023. Por fim, criamos o modelo passando o *estimator*, a *grid* dos parâmetros, o *refit* a True, com $cv = 5$ e o *score* do modelo calculado com base na métrica "neg_mean_squared_error".

```
    model = RandomForestRegressor(random_state=2023)
    grid_search = GridSearchCV(model,
    param_grid, cv=5,
    verbose=2, refit=True,
    scoring='neg_mean_squared_error',
    n_jobs=-1)
```

XGBoost

Neste modelo utilizamos o *grid search* para otimização de hiperparâmetros. Para a *grid* dos parâmetros, utilizamos os seguintes:

```
param_grid = {
    'n_estimators': [50, 100, 150, 200],
    'learning_rate': [0.1, 0.01, 0.001],
    'max_depth': [3, 5],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0],
}
```

De seguida, criou-se o *estimator* do XGBRegressor com `random_state` de 2023. Por fim, criamos o modelo passando o *estimator*, a *grid* dos parâmetros, o *refit* a True, com $cv = 5$ e o *score* do modelo calculado com base na métrica "neg_mean_squared_error".

```
model = XGBRegressor(random_state=2023)
grid_search = GridSearchCV(model, param_grid,
cv=5, verbose=2,
refit=True,
scoring='neg_mean_squared_error',
n_jobs=-1)
```

Neural Network - Multilayer Perceptron

Tal como para o problema de classificação, também realizamos uma rede neuronal *multilayer perceptron* para prever a *feature* "MaxTemp". A principal diferença na construção deste rede e a rede do problema de classificação reside na última camada em que, neste caso, a função de ativação do neurónio será *relu*. Desta forma, a rede foi construída da seguinte forma:

```
def build_model(activation = 'relu', learning_rate = 0.001):
    model = Sequential()
    model.add(Dense(16,input_dim = 22, activation = activation))
    model.add(Dense(8, activation = activation))
    model.add(Dense(1, activation = activation)) # output

    #Compile the model
    model.compile(
        loss = 'mae',
        optimizer  = tf.optimizers.Adam(learning_rate),
        metrics = ['mae', 'mse'])
    return model
```

3.6 Dicussão de Resultados

Através dos modelos desenvolvidos acima referidos, retiramos os seguintes resultados, tanto para o problema de classificação como de regressão, para cada uma das 6 preparações.

Pré-Processamento	Modelo	Precision		Recall		FScore		Accuracy
		0	1	0	1	0	1	
1 - Mal distribuído - média e mediana	Decision Tree Learner	0,87	0,69	0,94	0,5	0,9	0,58	0,84
	Logistic Regression	0,87	0,72	0,94	0,5	0,91	0,59	0,85
	Random Forest Learner	0,88	0,77	0,96	0,52	0,91	0,62	0,86
	GradientBoosting	0,88	0,74	0,95	0,53	0,91	0,62	0,86
	XGBoost	0,88	0,76	0,95	0,55	0,91	0,63	0,86
	Naive Bayes	0,89	0,55	0,86	0,61	0,87	0,58	0,81
	k-Nearest Neighbours	0,87	0,71	0,94	0,52	0,91	0,6	0,85
	Neural Networks	0,88	0,7	0,93	0,56	0,91	0,62	0,85
	k-Means	0,92	0,32	0,48	0,85	0,63	0,46	0,56
	Média	0,88	0,71	0,93	0,54	0,90	0,61	0,85
2 - Mal distribuído - API	Decision Tree Learner	0,87	0,69	0,93	0,52	0,9	0,59	0,84
	Logistic Regression	0,87	0,72	0,94	0,52	0,91	0,6	0,85
	Random Forest Learner	0,88	0,77	0,96	0,53	0,91	0,62	0,86
	GradientBoosting	0,88	0,75	0,95	0,54	0,91	0,63	0,86
	XGBoost	0,88	0,75	0,95	0,56	0,92	0,64	0,86
	Naive Bayes	0,89	0,58	0,88	0,6	0,88	0,59	0,82
	k-Nearest Neighbours	0,87	0,73	0,95	0,52	0,91	0,61	0,85
	Neural Networks	0,86	0,78	0,96	0,45	0,91	0,57	0,85
	k-Means	0,92	0,32	0,49	0,85	0,64	0,46	0,57
	Média	0,88	0,72	0,94	0,53	0,91	0,61	0,85
3 - UnderSampling - média e mediana	Decision Tree Learner	0,76	0,78	0,78	0,76	0,77	0,77	0,77
	Random Forest Learner	0,8	0,8	0,8	0,8	0,8	0,8	0,8
	Logistic Regression	0,77	0,79	0,8	0,77	0,78	0,78	0,78
	GradientBoosting	0,79	0,8	0,8	0,79	0,8	0,8	0,8
	XGBoost	0,8	0,81	0,81	0,8	0,8	0,81	0,81
	Naive Bayes	0,72	0,79	0,81	0,7	0,76	0,74	0,75
	k-Nearest Neighbours	0,77	0,8	0,81	0,77	0,79	0,78	0,79
	Neural Networks	0,78	0,8	0,8	0,77	0,79	0,78	0,79
	k-Means	0,33	0,27	0,37	0,23	0,35	0,25	0,3
	Média	0,77	0,80	0,80	0,77	0,79	0,78	0,79

Figura 4: Resultados obtidos na classificação - parte 1.

4 - UnderSampling - API	Decision Tree Learner	0,8	0,74	0,71	0,83	0,75	0,78	0,77
	Random Forest Learner	0,81	0,8	0,8	0,81	0,8	0,81	0,81
	Logistic Regression	0,78	0,8	0,81	0,78	0,8	0,79	0,79
	GradientBoosting	0,8	0,81	0,8	0,8	0,8	0,8	0,8
	XGBoost	0,81	0,81	0,81	0,81	0,81	0,81	0,81
	Naive Bayes	0,74	0,79	0,81	0,71	0,77	0,75	0,76
	k-Nearest Neighbours	0,78	0,8	0,81	0,78	0,8	0,79	0,79
	Neural Networks	0,77	0,82	0,83	0,76	0,8	0,79	0,79
	k-Means	0,74	0,67	0,62	0,79	0,67	0,72	0,7
	Média	0,79	0,80	0,80	0,79	0,79	0,79	0,79
5 - OverSampling - média e mediana	Decision Tree Learner	0,85	0,84	0,83	0,85	0,84	0,84	0,84
	Random Forest Learner	0,91	0,9	0,9	0,91	0,91	0,91	0,91
	Logistic Regression	0,78	0,79	0,8	0,77	0,79	0,78	0,78
	GradientBoosting	0,87	0,9	0,91	0,86	0,89	0,88	0,88
	XGBoost	0,88	0,93	0,93	0,87	0,91	0,9	0,9
	Naive Bayes	0,74	0,78	0,8	0,71	0,77	0,74	0,76
	k-Nearest Neighbours	0,97	0,83	0,8	0,98	0,88	0,9	0,89
	Neural Networks	0,8	0,82	0,82	0,8	0,81	0,81	0,81
	k-Means	0,33	0,26	0,38	0,22	0,35	0,24	0,3
	Média	0,85	0,85	0,85	0,84	0,85	0,85	0,85
6 - OverSampling - API	Decision Tree Learner	0,85	0,83	0,82	0,85	0,84	0,84	0,84
	Random Forest Learner	0,91	0,89	0,89	0,91	0,9	0,9	0,9
	Logistic Regression	0,79	0,8	0,8	0,78	0,79	0,79	0,79
	GradientBoosting	0,87	0,88	0,89	0,87	0,88	0,88	0,88
	XGBoost	0,89	0,92	0,92	0,88	0,9	0,9	0,9
	Naive Bayes	0,75	0,78	0,8	0,73	0,77	0,75	0,76
	k-Nearest Neighbours	0,98	0,83	0,8	0,98	0,88	0,9	0,89
	Neural Networks	0,83	0,8	0,79	0,84	0,81	0,82	0,81
	k-Means	0,74	0,67	0,62	0,79	0,67	0,73	0,7
	Média	0,86	0,85	0,85	0,86	0,85	0,85	0,85

Figura 5: Resultados obtidos na classificação - parte 2.

Da tabela acima apresentada, podemos concluir que no geral os modelos mais robustos são o XGBoost e o Random Forest Learner, algo expectável visto que são algoritmos que usufruem de técnicas de *ensemble* como o *boosting* e *bagging*, respetivamente. Como previámos anteriormente, as preparações de dados sem balanceamento das classes revelam uma incapacidade de prever a classe minoritária, algo que é facilmente pelos baixos valores de *recall* para a classe "1". Relativamente, às preparações com balanceamento conseguimos observar que os modelos têm mais capacidade para prever a clase minoritária como era expectável. O método de balancemeanto que revelou ser mais eficaz foi o *oversampling*, SMOTE, apresentando valores elevados de *accuracy* e *f-score*. Isto acontece devido ao facto do *oversampling* utilizar mais dados para treinar, uma vez que é feito um aumento dados e, desta forma, generaliza melhor comparativamente com o *undersampling*.

Pré-Processamento	Modelo	R^2	MAE	MSE	RMSE	Observações
1 - Split date - Mean and Median - Label Encoding - Drop outliers - Random value imputation	Decision Tree Learner	0,9794	0,6575	0,8957	0,9464	
	Linear Regression	0,9800	0,6549	0,8705	0,9330	
	Polynomial Regression	0,9855	0,5647	0,6316	0,7948	
	Random Forest	0,9844	0,5708	0,6761	0,8222	
	Gradient Boosting	0,9853	0,5639	0,6387	0,7992	
	K-Nearest Neighbors	0,9527	1,0751	2,0549	1,4335	
	XGBoost	0,9863	0,5398	0,5937	0,7705	
	Neural Network	0,9805	0,6288	0,8472	0,9204	
	Média	0,98	0,66	0,90	0,93	
2 - Split date - API values for missing values - Drop outliers - Label Encoding	Decision Tree Learner	0,9775	0,6814	0,9826	0,9912	
	Linear Regression	0,9768	0,7014	1,0163	1,0081	
	Polynomial Regression	0,9821	0,6246	0,7822	0,8844	
	Random Forest	0,9823	0,6118	0,7759	0,8809	
	Gradient Boosting	0,9823	0,6159	0,7757	0,8808	
	K-Nearest Neighbors	0,9509	1,1028	2,1475	1,4654	
	XGBoost	0,9839	0,5857	0,7037	0,8389	
	Neural Network	0,9793	0,6363	0,9051	0,9514	
	Média	0,98	0,72	1,07	1,01	

Figura 6: Resultados obtidos na regressão.

Relativamente aos resultados obtidos para o problema de regressão, podemos verificar pela figura 6 que o modelo que apresenta menores valores de erro e melhores coeficientes de determinação é o XGBoost. A melhor preparação de dados para este problema é a primeira, em que é realizada a imputação da média e mediana nos *missing values*.

Mais uma vez, verificamos a dominância do XGBoost em relação aos restantes algoritmos de *machine learning*. O XGBoost é um algoritmo que usufrui da técnica de *ensemble* de *boosting* em que aprende priorizando os erros cometidos nas suas iterações anteriores. Para além disso, é bastante robusto ao *overfitting*, utilizando técnicas como a regularização que o permite evitar.

4 Tarefa 2

4.1 Contextualização

A energia solar é uma das principais fontes de energias renováveis, desempenhando não só um papel fundamental na transição para fontes de energia limpa e renovável, mas também na promoção da sustentabilidade ambiental. Nesse contexto, além de ser crucial otimizar o uso da energia solar, a relação entre o gasto e a produção energética é essencial para permitir um planeamento eficaz do consumo energético e a integração harmoniosa de sistemas de energia solar em redes elétricas existentes. Este é, portanto, um problema de previsão de energia com impacto significativo na eficiência energética, mas também na redução das emissões de gases com efeito estufa e na promoção da sustentabilidade.

4.2 *Dataset*

O *dataset* colecionado contém diversos dados referentes à produção energética de determinados painéis solares na cidade de Braga, cobrindo um período que

vai desde setembro de 2021 até abril de 2023. Contudo, encontra-se dividido em 6 ficheiros: metade relativa aos dados de energia, cada um para um ano diferente, e a outra metade relativa à meteorologia, sendo novamente cada um para um ano diferente.

A nível de *features*, os *datasets* de energia são compostos por:

Feature	Descrição
Data	O timestamp associado ao registo, ao dia
Hora	A hora associada ao registo
Normal (kWh)	A 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)
Horário Económico (kWh)	A quantidade de energia eléctrica consumida, em kWh e proveniente da rede eléctrica, num período considerado económico em ciclos bi-horário diários (horas de vazio)
Autoconsumo (kWh)	A quantidade de energia eléctrica consumida, em kWh, proveniente dos painéis solares
Injeção na rede (kWh)	A quantidade de energia eléctrica injectada na rede eléctrica, em kWh, proveniente dos painéis solares

Figura 7: *Features* do *dataset* e respetiva descrição.

Relativamente aos *datasets* meteorológicos, estes são compostos pelas seguintes variáveis:

Feature	Descrição
dt	O timestamp associado ao registo
dt_iso	A data associada ao registo, ao segundo
city_name	O local em causa
temp	Temperatura em ºC
feels_like	Sensação térmica em ºC
temp_min	Temperatura mínima sentida em ºC
temp_max	Temperatura máxima sentida em ºC
pressure	Pressão atmosférica sentida em atm
sea_level	Pressão atmosférica sentida ao nível do mar em atm
grnd_level	Pressão atmosférica sentida à altitude local em atm
humidity	Humidade em percentagem
wind_speed	Velocidade do vento em metros por segundo
rain_1h	Valor médio de precipitação
clouds_all	Nível de nebulosidade em percentagem
weather_description	Avaliação qualitativa do estado do tempo

Figura 8: *Features* do *dataset* e respetiva descrição.

4.3 Preparação Prévia dos *Datasets*

Como referimos anteriormente, o *dataset* encontra-se divido em vários *datasets*, sendo necessário juntar os dados. Os dados dos anos de 2021 e 2022 correspondem aos dados utilizados para treinar os modelos. Os restantes relativos ao ano de 2023 são dados sem o *target*, uma vez que serão utilizados para o modelo prever e, posteriomente, submeter os resultados na plataforma Kaggle.

Portanto, de forma a obtermos o *dataset* de treino concatenamos os *datasets* dos anos 2021 e 2022 tanto para a meteorologia como para produção energética.

De seguida, realizamos um *merge* da meteorologia e energia pela data e hora de ambos os *datasets*. O mesmo procedimento foi aplicado para o *dataset* de teste.

Aos *datasets* resultantes da preparação anterior, adicionamos ainda várias colunas relativas a radiações através da OpenMeteo Historical Weather API, uma vez que consideramos que poderia trazer informação relevante para o modelo conseguir uma melhor previsão. As colunas acrescentadas ao *dataset* inicial são as seguintes: shortwave_radiation, direct_radiation, diffuse_radiation, direct_normal_irradiance, terrestrial_radiation, shortwave_radiation_instant, direct_radiation_instant, diffuse_radiation_instant, direct_normal_irradiance_instant e terrestrial_radiation_instant. Todas estas colunas são relativas a diferentes tipos de radiação, sendo umas calculadas a partir da média da radiação ao longo de uma hora e outras obtidas num determinado instante dentro daquela hora.

Para além disso, nesta preparação prévia dos *datasets*, estão também a ser preenchidos os *missing values* do *dataset* meteorológico de teste com recurso a dados recolhidos da OpenWeatherMap API.

4.4 Exploração dos Dados

Como já foi referido na primeira tarefa, quando lidamos com dados que não conhecemos, é crucial ter uma compreensão completa dos dados do problema. Desta forma, foi realizada uma análise exploratória antes de se começar a manipulação sobre os dados.

Para perceber o domínio de problema, devemos conhecer as *features* do problema. Assim para cada um dos *datasets*, visualizamos as *features* que os constituem bem como o seu formato. O nosso *dataset* tem agora 11016 entradas e 31 *features*.

A estatística quando relacionada aos dados, permite-nos obter informações de como os dados se encontram distribuídos no *dataset*. Assim, observamos a correlação entre *features* na figura 9.

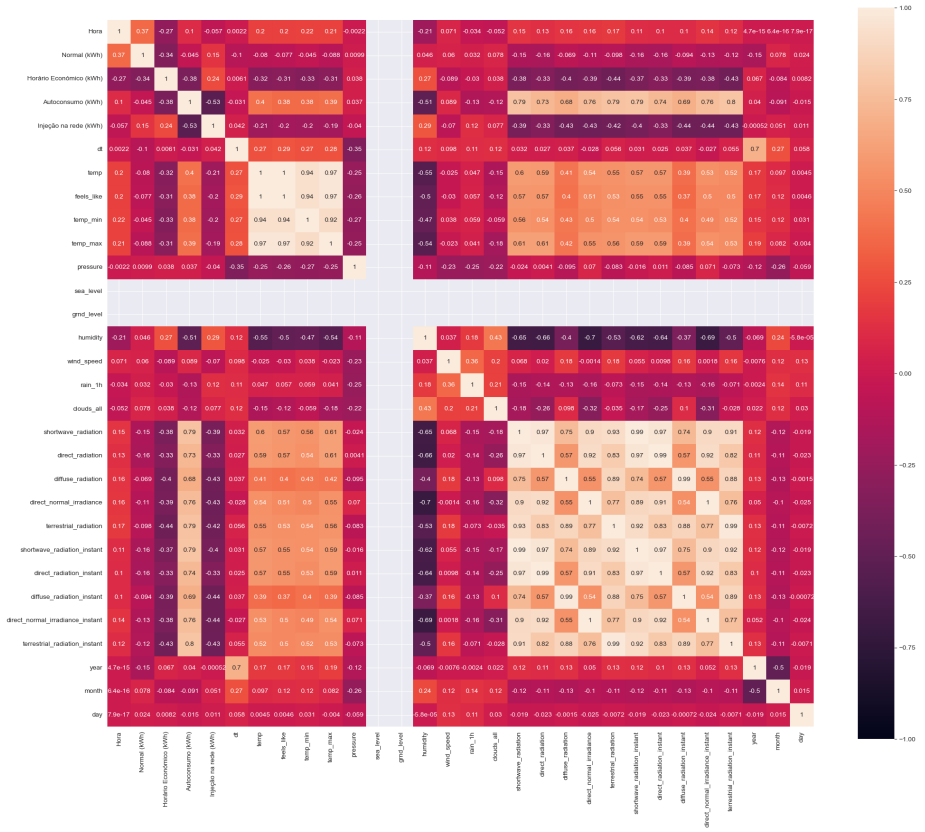


Figura 9: Matriz de correlação.

Com base no gráfico acima, conseguimos perceber que o "sea_level" e "grnd_level" possuem uma elevada quantidade de *missing values*. Conseguimos perceber também que as *features* "temp", "feels_like", "temp_min" e "temp_max" estão muito correlacionadas.

Quando estamos a analisar um conjunto de dados, convém sempre saber quais os valores que cada *feature* possui, bem como saber qual a distribuição desses mesmos valores, isto é, saber quantos tipos de valores diferentes entre si possuímos. Desta forma, realizamos a seguinte análise tendo obtido os respetivos valores. Para a *feature* 'weather_description' possuímos os seguintes valores discretos: "overcast clouds", "broken clouds", "few clouds", "sky is clear", "scattered clouds", "light rain", "moderate rain" e "heavy intensity rain"

Relativamente às estatísticas globais, conseguimos obter valores para a média, somatório, desvio padrão, mínimo, máximo, quartil 25%, quartil 50% e quartil 75% relativos a cada *feature*. Por exemplo, para as *features* "Hora", "Normal (kWh)", "Horário Económico (kWh)" e "Autoconsumo (kWh)" possuímos o seguinte resultado:

	Hora	Normal (kWh)	Horário Económico (kWh)	Autoconsumo (kWh)	\
count	11016.000000	11016.000000	11016.000000	11016.000000	
mean	11.500000	0.202278	0.159714	0.117314	
std	6.922501	0.349478	0.271792	0.176762	
min	0.000000	0.000000	0.000000	0.000000	
25%	5.750000	0.000000	0.000000	0.000000	
50%	11.500000	0.000000	0.000000	0.000000	
75%	17.250000	0.314000	0.288000	0.227000	
max	23.000000	3.251000	6.978000	1.192000	

Figura 10: Estatísticas globais.

Num conjunto de dados, é importante analisar a falta de dados existentes nos conjuntos de observações. Para tal, é feita a análise dos *missing values*.

Data	0
Hora	0
Normal (kWh)	0
Horário Económico (kWh)	0
Autoconsumo (kWh)	0
Injeção na rede (kWh)	7777
dt	0
dt_iso	0
city_name	0
temp	0
feels_like	0
temp_min	0
temp_max	0
pressure	0
sea_level	11016
grnd_level	11016
humidity	0
wind_speed	0
rain_1h	8732
clouds_all	0
weather_description	0
shortwave_radiation	0
direct_radiation	0
diffuse_radiation	0
direct_normal_irradiance	0
terrestrial_radiation	0
shortwave_radiation_instant	0
direct_radiation_instant	0
diffuse_radiation_instant	0
direct_normal_irradiance_instant	0
terrestrial_radiation_instant	0
dtype: int64	

Figura 11: *Missing values*.

Para visualizarmos os *outliers* que o conjunto de dados possui, a distribuição dos valores de cada *feature* e a distribuição de valores da injeção na rede, criamos vários gráficos para uma mais fácil visualização (figuras 19, 20 e 21, respetivamente).

4.5 Preparação dos Dados

4.5.1 *Feature Engineering*

Dá-se o nome de *feature engineering* ao processo de extração de novas *features* a partir das *features* já existentes no *dataset*. No caso de estudo, foram criadas novas *features* tais como "Ano", "Mês" e "Dia", a partir da *feature* "Data" presente no *dataset*.

4.5.2 Remoção de Colunas

Para que exista um ganho de informação, por parte do modelo de *machine learning*, em relação a cada um das *features* do *dataset*, estas devem possuir diferentes dados.

Como neste caso, a *feature* "city_name" tinha sempre o mesmo valor, então decidimos removê-la pois não apresenta ganho de conhecimento para o modelo. Depois de realizarmos vários testes com diferentes preparações de dados, conseguimos perceber que as *features* "Dia", "Mês" e "Ano" não se traduzem num ganho de informação relevante, retirando-as também do conjunto de dados. As *features* "dt" e "dt_iso" seguem a mesma linha de raciocínio das *features* anteriores e, desta forma, também foram removidas.

Relativamente aos dados da radiância inseridos adicionalmente, estas *features* constituem um aumento da informação conhecida relativamente ao domínio do problema. Ainda assim, existe um tipo de radiação que não deve pertencer ao conjunto de dados, que são os parâmetros das radiações medidos instantaneamente, pois possuem uma unidade de medida de tempo (instante) que é desproporcional às outras unidades de medida de tempo (hora, dia). Além disto, este parâmetros apresentam uma elevada correlação entre si, ou seja, devem ser removidos para não prejudicarem a generalização dos modelos de *machine learning*, uma vez que acabam por transmitir a mesma informação de *features* já existentes. Desta forma, todos os parâmetros ligados à radiação instantânea foram devidamente removidos.

Todas as *features* que apresentam uma elevada correlação entre elas, influenciam de uma forma negativa o modelo de aprendizagem. A correlação forte entre atributos pode resultar em multicolinearidade, onde um atributo pode ser linearmente previsto a partir de outros. O XGBoost, modelo utilizado, pode ser propenso a *overfitting*, especialmente quando há muitas *features* correlacionadas pois a multicolinearidade pode levar a árvores mais profundas, aumentando a probabilidade de o modelo se ajustar demais aos dados de treino. Este fenómeno acontece com as *features* "feels_like", "temp_min", "temp_max", que acabaram também por ser removidas.

4.5.3 *Label Encoding*

As máquinas que suportam e executam os modelos de *machine learning*, foram criadas para trabalhar de forma mais capaz com valores númericos. Desta forma, quando uma *feature* importante para o contexto do problema possui um tipo diferente de númerico, o seu tipo tem de ser alterado para um tipo mais "*user-friendly*" da máquina. Assim os valores que a *feature* "weather_description" possui foram alterados para o seguinte:

```
replace_map = {
    'heavy intensity rain': 0,
    'moderate rain': 1, 'few clouds': 2,
    'scattered clouds': 3, 'broken clouds': 4,
```

```
    'light rain': 5,'overcast clouds': 6,
    'sky is clear': 7, 'clear sky': 7
}
```

4.5.4 Tratamento de *Missing Values*

Os *missing values* têm uma elevada preponderância no desempenho final dos modelos. Os *missing values* de um conjunto de dados correspondem à falta de informação que um determinado conjunto de dados possui sobre o domínio do problema em que está inserido. Assim, se colocarmos um modelo de *machine learning* a utilizar esses dados, eles podem fazer com que o modelo não consiga generalizar para casos mais globais do que aqueles que se encontram no conjunto de dados, surgindo assim o **overfitting**.

Com base no que foi explicado em cima, foram removidos do conjunto de dados as *features* "sea_level", "grnd_level" e "rain_1h", visto que possuíam uma elevada quantidade de *missing values*, como foi averiguado na exploração dos dados.

4.5.5 Manipulação de Valores

Tendo em atenção o método utilizado para a leitura dos *datasets*, este faz com que dados em falta seja considerados *'Not a Number'*(NaN). Na *feature* "Injeção na rede (kWh)" esses dados em falta têm um significado conhecido, significam None. Deste modo, tivemos de substituir todos os valores "NaN" por "None" na *feature* referida, para que estes não fossem considerados *missing values*.

4.6 Modelos Desenvolvidos e Discussão de Resultados

Relativamente à previsão da injeção na rede, utilizamos essencialmente algoritmos de *ensemble learning*, como o XGBoost e Random Forest, e redes neuronais, de modo a obter melhores resultados. Realizamos, ainda, técnicas de MaxVoting e Stacking por forma a combinar vários modelos. O grupo decidiu focar-se mais no modelo XGBoost, uma vez que ao longo do trabalho foi o que demonstrou obter melhores resultados na competição.

Ao longo da competição, utilizamos diferentes abordagens relativas à conceção dos modelos. Numa primeira fase, fazíamos um *split* de 80/20 no *dataset* de treino com o objetivo de termos mais um indicador, partição de 20% para teste, para avaliar se o modelo produzido era ou não robusto. Desta forma, complementávamos as *accuracies* obtidas no Kaggle e nesta partição de 20% com vista a avaliar o modelo. Posteriormente, utilizamos o *dataset* inteiro para treinar, dado que, dete modo, o modelo iria treinar com mais dados e poderia revelar-se mais robusto, generalizando melhor. Os modelos desta última abordagem eram avaliados tendo em conta a *accuracy* obtida no *cross validation* utilizado pelo GridSearchCV.

Relativamente ainda à modelação, os melhores hiperparâmetros eram encontrados com recurso a diferentes graus de *cross validation*: 5, 10 e 15 *folds*. É importante realçar que utilizamos diferentes preparações e recolhemos os dados com o objetivo de encontrar o modelo mais robusto por forma a obtermos uma boa classificação na competição.

Assim, os resultados obtidos, seguindo a segunda metodologia apresentada, materializam-se na seguinte tabela:

Preparação	Modelo	CV5		CV10		CV15		CV20	
		Treino	Kaggle	Treino	Kaggle	Treino	Kaggle	Treino	Kaggle
Sem radiações	XGBoost	0,862748	---	0,865291	---	0,867113	---	---	---
Todas as radiações	XGBoost	0,861203	---	0,873004	---	0,874823	---	---	---
Todas as radiações; Drop WD e Humidity	XGBoost	0,859751	---	0,873095	---	0,874460	---	---	---
Todas as radiações; Drop WD, Humidity e Mês	XGBoost	0,869374	---	0,874004	---	0,874510	---	---	---
Radiações sem instant; Drop mês	XGBoost	0,871644	0.90236	0,875366	0,90976	0,877095	---	0,87899	0,90236
Radiações sem instant; Drop mês e wind_speed	XGBoost	0,871190	---	0,874367	---	0,876822	---	---	---
Apenas direct_radiation; Drop mês	XGBoost	0,875003	---	0,877545	0.89792	0,878458	0,88609	---	---

Figura 12: Resultados obtidos na tarefa 2.

Estes resultados foram retirados com a *grid* abaixo apresentada. De notar que optamos por utilizar um valor baixo para `max_depth`, visto que limitar a profundidade das árvores de decisão previne o *overfit*, um dos objetivos do grupo. Para além disso, após vários testes com outras grelhas de parâmetros, reparamos que com um número de estimadores igual a 200, 300 ou 400, o modelo estava a generalizar pior tendo em conta a pontuação obtida na classificação pública. Por isso, decidimos utilizar 100 estimadores como parâmetro do XGBoost, algo que nos permitiu melhorar a *accuracy* sobre a partição de dados de teste pública.

```
param_grid = {
    'max_depth': [4],
    'min_child_weight': [1, 2, 3],
    'subsample': [0.5, 0.6, 0.7],
    'colsample_bytree': [0.6, 0.8, 1],
    'learning_rate': [0.01, 0.1],
    'n_estimators': [100],
}
```

Da análise da tabela de resultados apresentada acima, conseguimos concluir que a introdução das colunas da radiação resultou numa mais-valia para o desempenho dos modelos. A preparação que revelou resultar em modelos mais robustos foi a preparação apresentada na secção 4.5, obtendo *accuracies* acima dos 90% na competição pública. A preparação que resultou em valores ligeiramente mais elevados de *accuracy* durante o treino foi a preparação com apenas uma coluna da radiação, o que nos fez acreditar que estavamos perante um

modelo melhor. No entanto, o modelo não generalizou tão bem quanto esperavamos, obtendo piores resultados do que a preparação atual na competição pública.

Com esta abordagem retiramos ainda mais resultados, presentes na figura 18, com adição de mais colunas meteorológicas ao *dataset* para além das radiações. Porém, não obtivemos os resultados esperados pelo que mantivemos apenas as radiações.

Relativamente à primeira abordagem efetuada, em que realizamos partições de 80/20 no *dataset* de treino, anexamos algumas tabelas, presentes na figura 17, com resultados obtidos aquando dessa arbordagem. De notar que a *grid* de parâmetros utilizada possuía valores de `max_depth` e `estimators` variáveis, ao contrário da abordagem seguida nos resultados acima.

5 Conclusão

Este trabalho prático teve como objetivo explorar, analisar e preparar 2 *datasets* para criar modelos de aprendizagem automática. O processo de desenvolvimento dos modelos apresentados exigiu bastante reflexão por parte do grupo e, por vezes, a reconstrução dos modelos ou do tratamento dos dados. De uma forma geral, o grupo considerou que a existência de uma competição impulsionou uma motivação adicional para a realização deste trabalho, levando-nos a investir e dedicar mais tempo e esforço na aprimoração dos modelos. Deste modo, consideramos que os objetivos propostos foram alcançados com sucesso.

6 Anexos

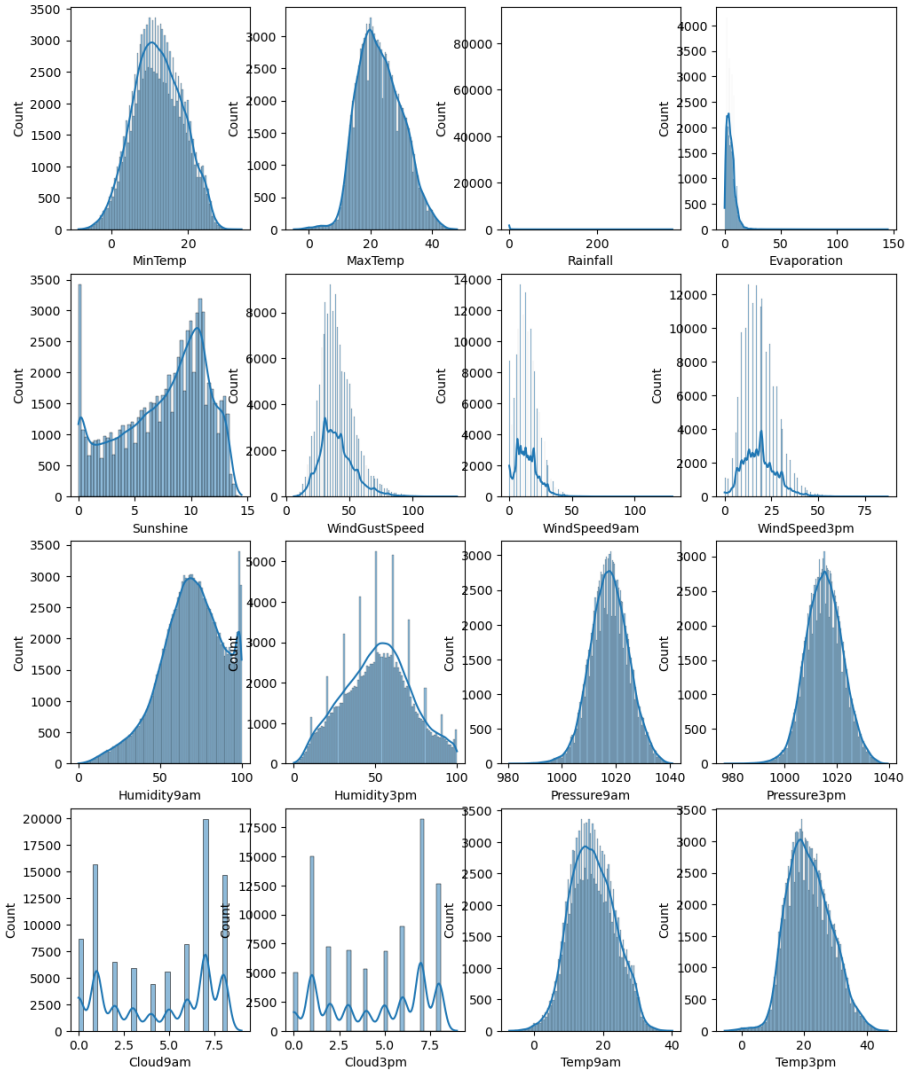


Figura 13: Distribuição dos atributos numéricos.

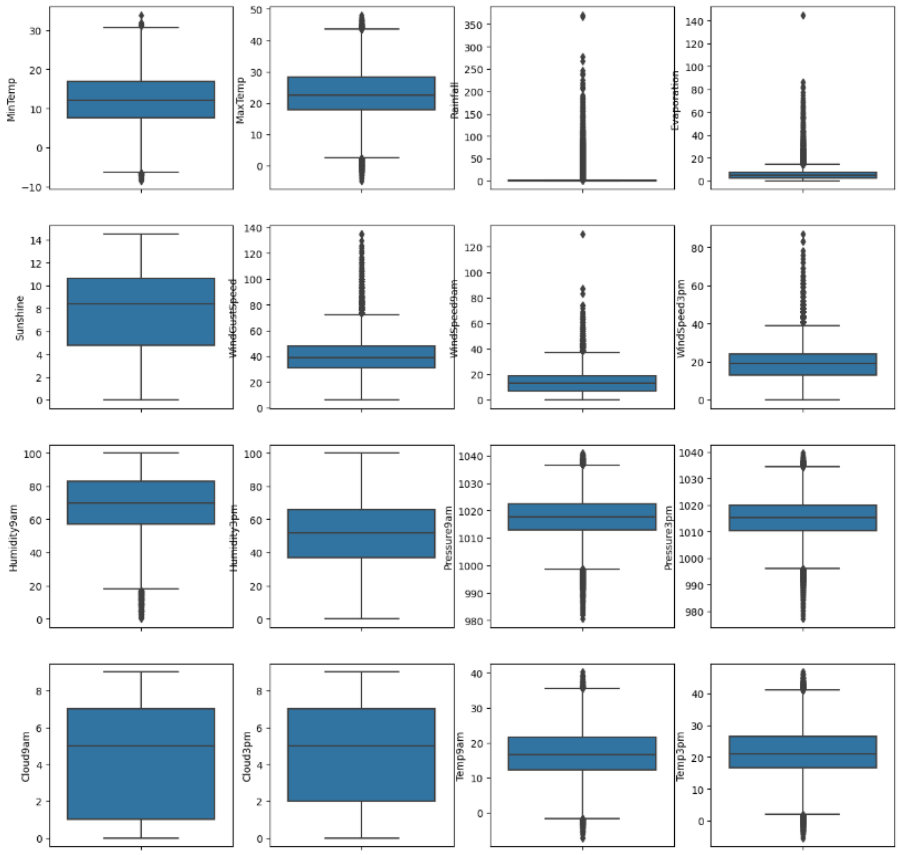


Figura 14: *Boxplot*.

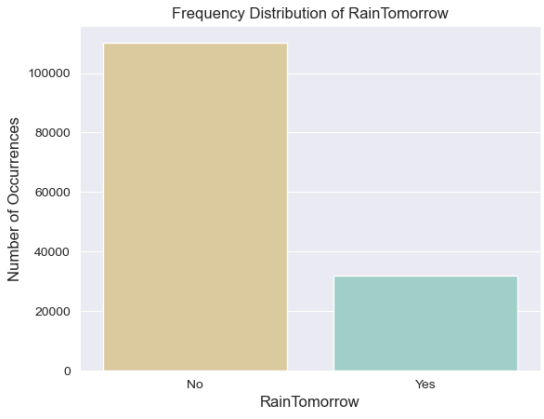


Figura 15: Distribuição da probabilidade de chover no dia seguinte.

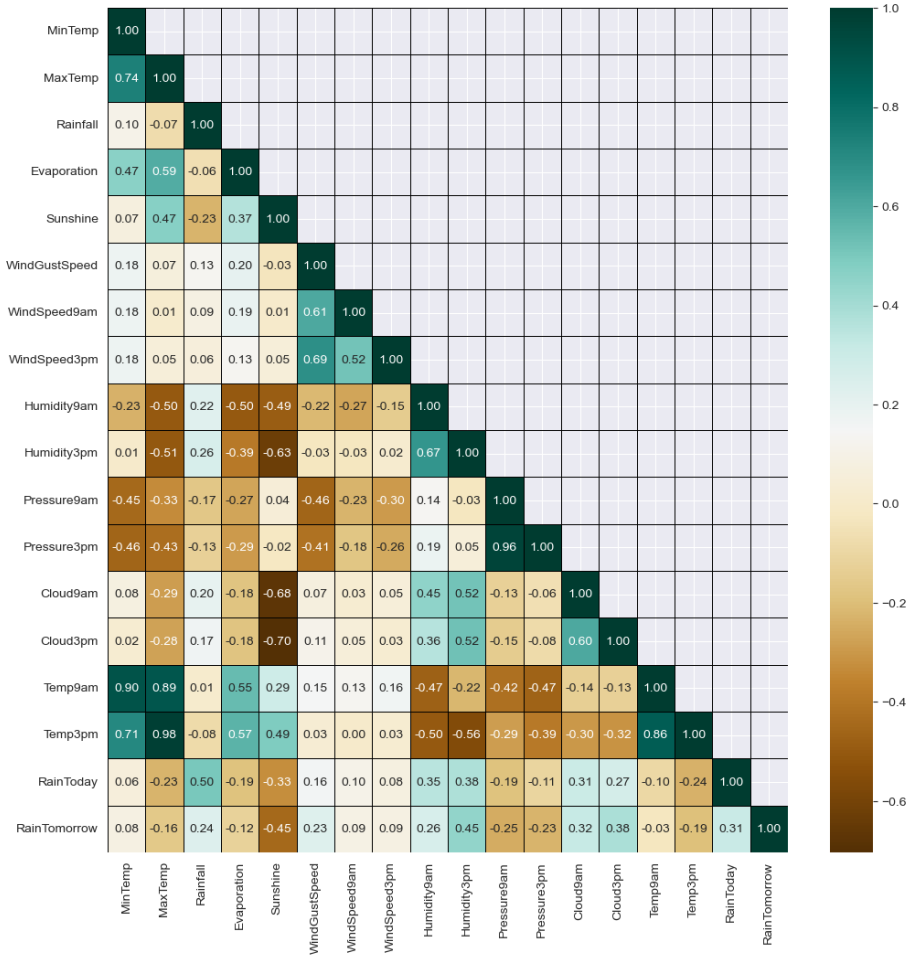


Figura 16: Correlações entre os atributos.

	Split 80 - 20 | Stratified									
	Modelo	CV5			CV10			CV15		
		Treino	Teste	Kaggle	Treino	Teste	Kaggle	Treino	Teste	Kaggle
Todas radiações; WD One hot encoding	XGBoost	0.892080	0.897005	---	0.892534	0.897005	0.893490	0.894235	0.897005	0.89349
Todas radiações; Drop direct_radiation, instant, temp_min/max e feels_like	XGBoost	0.887880	0.893829	---	0.890150	0.893829	---	---	---	---
Apenas direct_normal_irradiance	XGBoost	0.883682	0.892015	---	0.893895	0.896552	---	---	---	---

Figura 17: Resultados obtidos na tarefa 2 com a primeira abordagem.

Split 100 - 0			
	Modelo	CV10	
		Treino	Kaggle
Radiações; Drop rain, surface_pressure, sealevel_pressure, sunshine_duration e LabelEncoding	XGBoost	0,874458	0,88905
Radiações; Drop rain, surface_pressure, sealevel_pressure, sunshine_duration, month e LabelEncoding	XGBoost	0,875910	0,89497
Radiações; Drop rain, surface_pressure, sealevel_pressure, sunshine_duration e OneHotEncoding	XGBoost	0,873913	---
Radiações; Drop rain, surface_pressure, sealevel_pressure, sunshine_duration, month e OneHotEncoding	XGBoost	0,875729	---
Radiações; Drop rain, surface_pressure, sealevel_pressure, sunshine_duration, month, wind_speed e OneHotEncoding	XGBoost	0,876727	0,89644
Radiações; Drop rain, surface_pressure, sealevel_pressure, sunshine_duration, month, wind_speed, pressure e OneHotEncod	XGBoost	0,876546	---
Apenas direct_radiation; Drop rain, surface_pressure, sealevel_pressure, sunshine_duration, month, wind_speed, pressure e C	XGBoost	0,878089	0,8713
Radiações s/ instant; Drop rain, surface_pressure, sealevel_pressure, sunshine_duration, month, wind_speed, pressure e Onel	XGBoost	0,875456	---
Radiações; Drop rain, surface_pressure, sealevel_pressure e LabelEncoding	XGBoost	0,875638	---
Radiações; Drop rain, surface_pressure, sealevel_pressure, sunshine_duration e weather	XGBoost	0,874095	---

Figura 18: Resultados obtidos na tarefa 2 com a segunda abordagem e mais colunas.

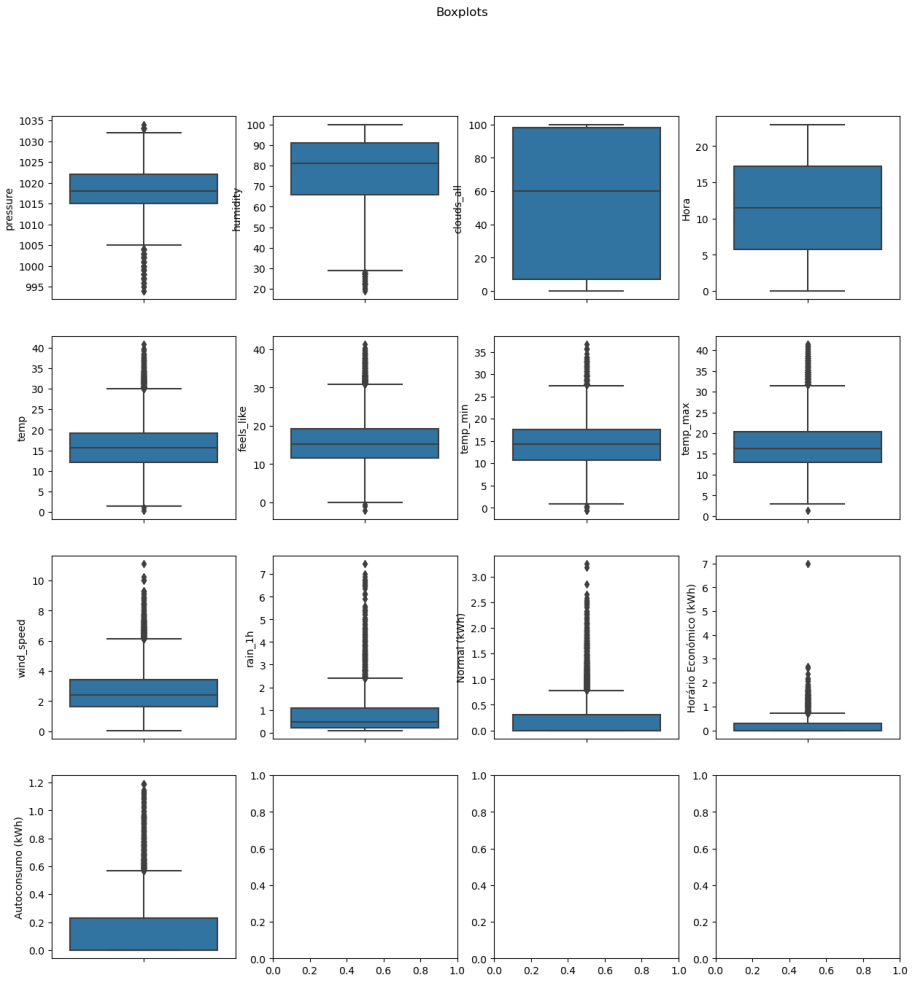


Figura 19: *Outliers*.

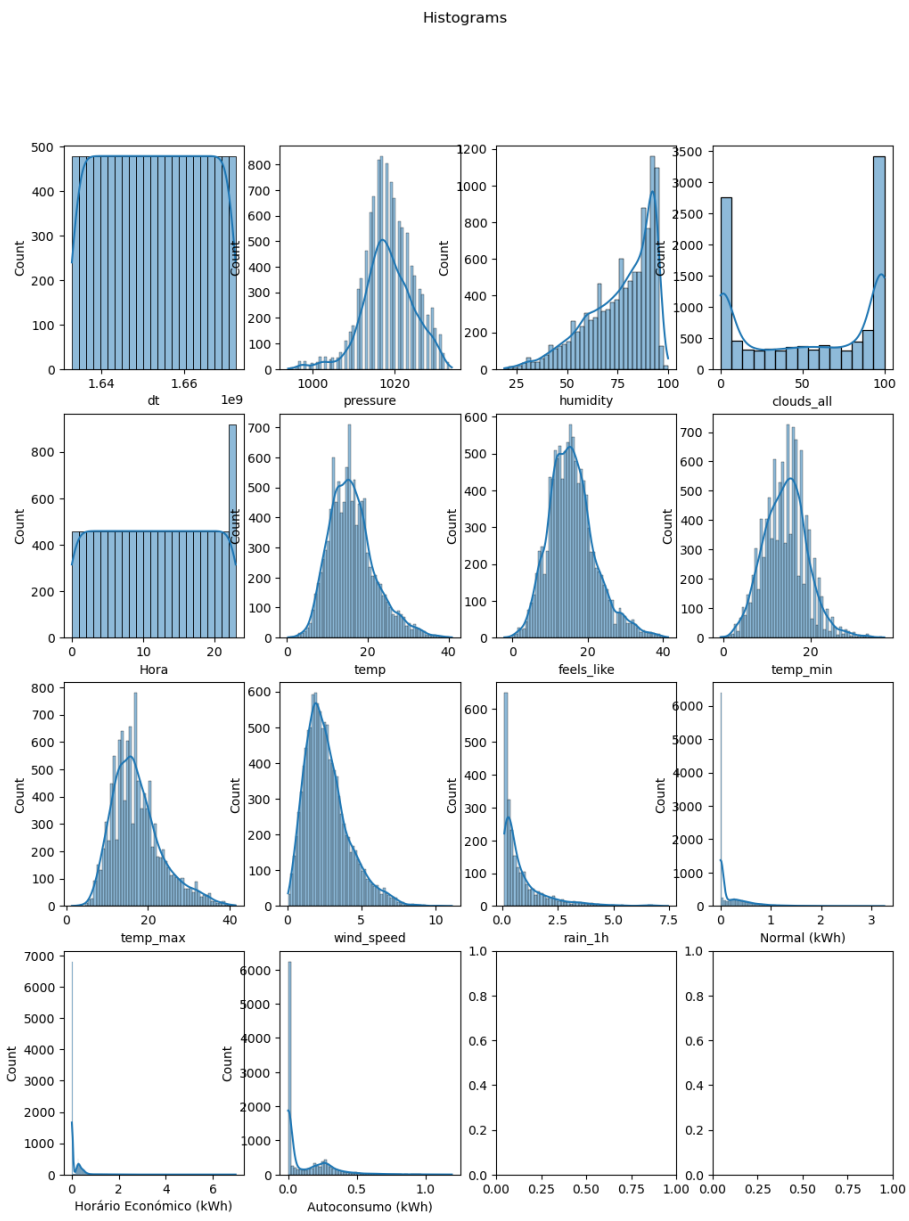


Figura 20: Dispersão estatística.

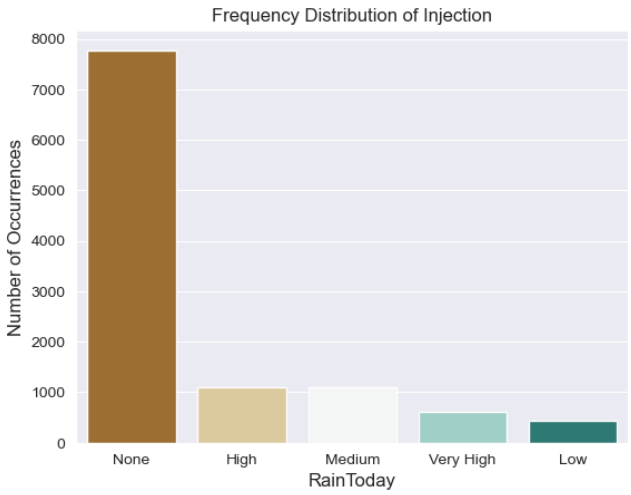


Figura 21: Frequência da distribuição da *feature* "Injeção na rede (kWh)".