# Dados e Aprendizagem Automática

**Grupo 1**
Francisca Lemos pg52693
Nelson Almeida pg52697
Nuno Costa pg52698
José Martins pg53968

Janeiro 2024

# Conteúdo

# 1 Introdução

Neste relatório irão ser abordados e exploradas as maneiras como foram encarados os dois *datasets*, entre eles a análise e exploração dos dados, o tratamento destes, os modelos usados para treina-los e por fim a análise critica dos resultados. De forma a fazer o melhor uso dos conceitos aprendidos optamos por fazer dois tipos de modelos. Um de regressão no *dataset* escolhido pelo grupo e um de classificação usado no *dataset* de competição.

# 2 Dataset grupo - Students Performance in Exams

Após analisar vários datasets, o grupo optou por escolher um adaptado à nossa realidade. O dataset escolhido trata das notas obtidas pelos estudantes nas disciplinas de matemática e português, sendo disponibilizado através da plataforma Kaggle. A escolha deste conjunto é justificada pela sua relevância, permitindo-nos compreender a influência de diversas características, tais como o apoio dos pais, a frequência de saídas, género, entre outras, nas notas finais de cada aluno. O objetivo final é utilizar essas informações para auxiliar alunos que possam se beneficiar de intervenções personalizadas com base em suas particularidades.

## 2.1 Análise e exploração dos dados

Este dataset está dividido em dois *csv*, um com a informação da disciplina de matemática e outro com a disciplina de português. Inicialmente, demos merge aos dois e a partir daí começamos a analisar os dados. Este dataset contém 32 colunas, com 1045 linhas. A target feature irá ser a feature 'G3', ou seja, a nota final do 3º período.

### 2.1.1 Valores em falta

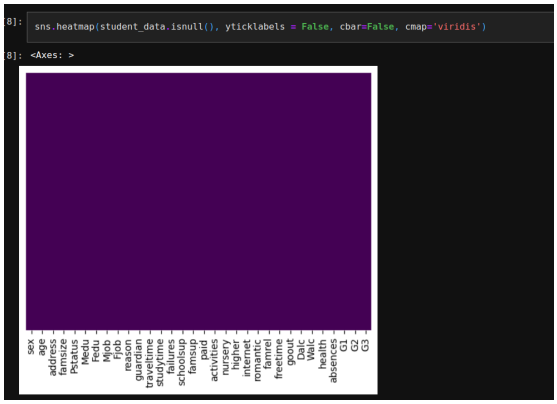


Figura 1: Valores em falta

Como podemos observar, não há valores em falta neste dataset.

### 2.1.2 Outliers

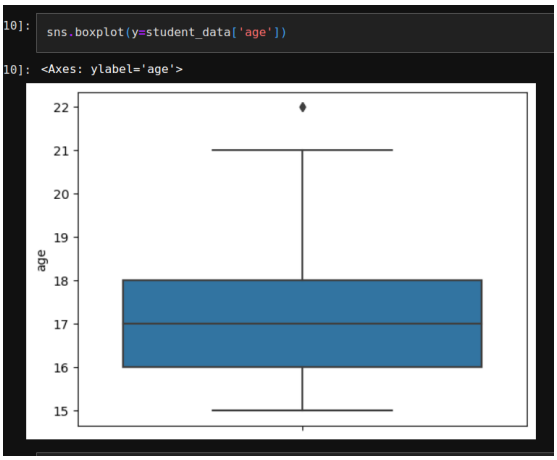


Figura 2: Outliers na feature 'age'

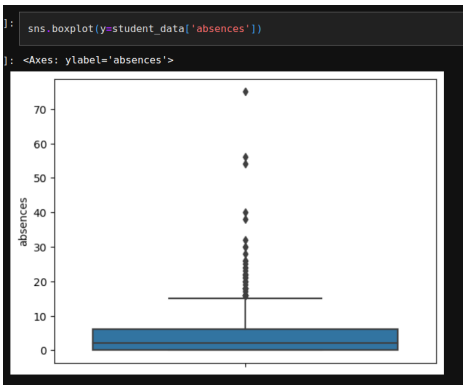


Figura 3: Outliers na feature 'absences'

### 2.1.3 Correlations

```
top_down_corr = corr_matrix.unstack()
top_down_15_corr = top_corr['G3'].nsmallest(15)

print(top_down_15_corr)
```

```
failures     -0.384046
Dalc         -0.129576
age          -0.124705
Malc         -0.121751
address      -0.117563
Walc         -0.115979
traveltime   -0.103030
goout        -0.099313
romantic     -0.096771
health       -0.081126
schoolsup    -0.079964
freetime     -0.066495
famsize      -0.062197
paid         -0.050081
guardian     -0.045188
dtype: float64
```

Figura 4: 15 melhores correlações da feature 'G3'

Top 15 correlations

```
top_corr = corr_matrix.unstack()
top_corr = top_corr[top_corr != 1]
top_15_corr = top_corr['G3'].nlargest(15)

print(top_15_corr)
```

```
G2            0.910720
G1            0.808999
higher        0.236317
Medu          0.202145
studytime     0.161770
Fedu          0.160244
internet      0.107930
reason        0.064427
famrel        0.055519
nursery       0.039350
activities    0.034418
Mjob          0.019427
famsup        0.013333
Fjob          0.006883
Pstatus      -0.032108
dtype: float64
```

Figura 5: 15 piores correlações da feature 'G3'

### 2.1.4 Distribuição dos valores

Top 15 correlations

```
top_corr = corr_matrix.unstack()
top_corr = top_corr[top_corr != 1]
top_15_corr = top_corr['G3'].nlargest(15)

print(top_15_corr)
```

```
G2            0.910720
G1            0.808999
higher        0.236317
Medu          0.202145
studytime     0.161770
Fedu          0.160244
internet      0.107930
reason        0.064427
famrel        0.055519
nursery       0.039350
activities    0.034418
Mjob          0.019427
famsup        0.013333
Fjob          0.006883
Pstatus      -0.032108
dtype: float64
```

Figura 6: Distribuição dos dados da target feature (G3)

## 2.2 Tratamento dos dados

Feita a análise que achamos necessária passamos ao tratamento de dados.

### 2.2.1 Label Encoding

Fizemos label encoding para converter os dados categóricos em valores númericos. Isto é útil para os algoritmos compreenderem e interpretarem os dados.

```
column_mapping = {
    'sex': {'F': 0, 'M': 1},
    'famsize': {'LE3': 0, 'GT3': 1},
    'Pstatus': {'A': 0, 'T': 1},
    'schoolsup': {'no': 0, 'yes': 1},
    'famsup': {'no': 0, 'yes': 1},
    'paid': {'no': 0, 'yes': 1},
    'activities': {'no': 0, 'yes': 1},
    'nursery': {'no': 0, 'yes': 1},
    'higher': {'no': 0, 'yes': 1},
    'internet': {'no': 0, 'yes': 1},
    'romantic': {'no': 0, 'yes': 1},
    'Mjob': {'at_home':0, 'services': 1, 'teacher': 2, 'health':3, 'other':4},
    'Fjob': {'at_home':0, 'services': 1, 'teacher': 2, 'health':3, 'other':4},
    'reason': {'course':0, 'home': 1, 'reputation': 2, 'other':3},
    'guardian': {'mother':0, 'father': 1, 'other': 2},
    'address': {'U':0, 'R':1}
}
for column, mapping in column_mapping.items():
    student_data[column] = student_data[column].map(mapping)
```

Figura 7: Label encoding das features categóricas

### 2.2.2 Remover os outliers

Decidimos unicamente remover os dados com valores acima de 45 da feature *absences* pois consideramos que os restantes valores considerados outliers poderiam ser interessantes para a previsão da nossa target. O único outlier da feature *age* também decidimos não remover pelo mesmo motivo, e após testar os modelos confirmamos que tomamos a melhor decisão. Em baixo, está a linha de código que permite a remoção desses outliers.

```
student_data = student_data[student_data['absences'] < 45]
```

## 2.3 Modelos desenvolvidos

Após a análise detalhada dos dados, bem como o seu tratamento, passamos à fase de desenvolvimento do modelo de previsão. Esta fase pretende obter um modelo que seja capaz de representar o dataset em questão, bem como fazer previsões corretas a partir de dados desconhecidos. Assim sendo, o grupo começou por efetuar a divisão dos dados em dados de treino e dados de teste, recorrendo-se ao *train_test_split*. Decidiu-se que 25% do dataset seria usado para teste e 75% para treino.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

Como isto é um problema de regressão, as métricas utilizadas são *mean squared error*, *mean absolute error* e *root squared error*, o que faz todo o sentido pois queremos saber quão perto estamos dos valores esperados.

### 2.3.1 Linear Regression

De seguida, testamos o modelo com *Linear Regression* para conseguirmos prevê-lo melhor. As nossas métricas melhoraram , e o seguinte gráfico mostra que os dados de treino se encontram dispersos pelo gráfico (cor azul), já os dados de teste concentram-se mais e não fogem muito ao resultados esperados (cor roxa).

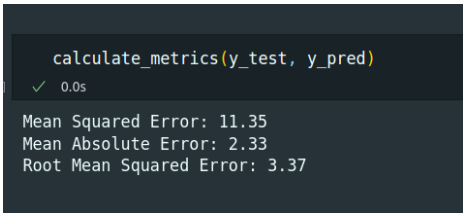


Figura 8: Resultados obtidos na Linear Regression

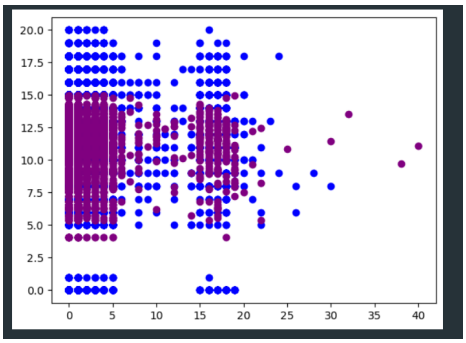


Figura 9: Gráfico da Linear Regression

Aplicamos também o Linear Regression com o GridSearch, pois assim torna-se possível avaliar os melhores parâmetros para o nosso modelo. Os parâmetros utilizados foram o *fit_intercept positive*. O primeiro indica se o modelo deve ter ou não um termo independente, o parâmetro *positive* , caso seja True, força os coeficientes a serem positivos. De seguida, apresentamos os melhores parâmetros e os resultados obtidos. Apesar de considerarmos esta uma melhor abordagem, os resultados não sofreram variação.

```
print("Best parameters: ", grid_search.best_params_)
✓ 0.0s
Best parameters:  {'fit_intercept': True, 'positive': False}

y_pred = grid_search.predict(X_test)
calculate_metrics(y_test, y_pred)
✓ 0.0s
Mean Squared Error: 11.35
Mean Absolute Error: 2.33
Root Mean Squared Error: 3.37
```

Figura 10: Resultados obtidos na Linear Regression com GridSearch

### 2.3.2 Random Forest

O random forest é um algortimo que tem por base usar vários resultados de árvores diferentes para chegar à conclusão do resultado final. O Random Forest usa *_bagging_*, sendo que este funciona da seguinte forma:

1. Seleção do subset: Escolhe um sample random do dataset.
2. Bootstrap sampling : Cada modelo é criado a partir deste samples, ou subsets.
3. Bootstrapping: É o passo de row sampling com replacement.

Fizemos também um gráfico com a importância de cada feature para o modelo e pode-se constatar que as features *failures* e *absences* tem um grande impacto na nota final do período.

```
calculate_metrics(test_labels, predictions)
✓ 0.0s

Mean Squared Error: 8.42
Mean Absolute Error: 2.13
Root Mean Squared Error: 2.90
```

Figura 11: Resultados obtidos no Random Forest

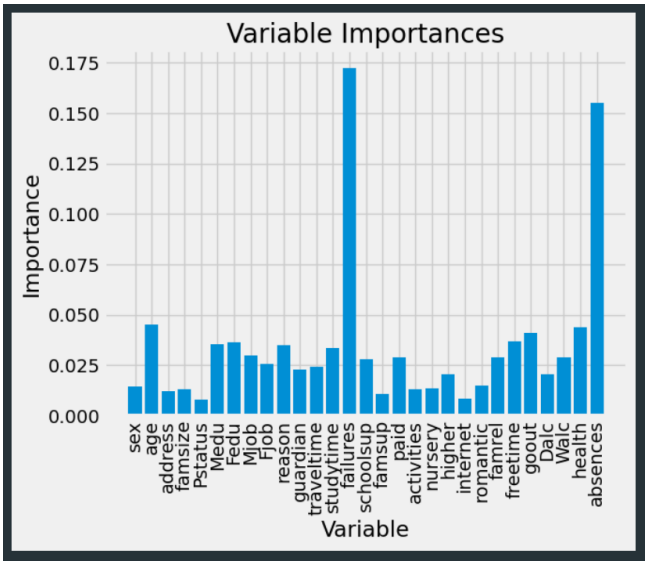


Figura 12: Importância das features no Random Forest

### 2.3.3 Neural Networks

Por último, testamos Neural networks, atrave Gerou-se um gráfico que mostra a performance, neste caso os valores perdidos dos dados de treino e de validação, ao longo de cada epoch. Ao longo do aumento das epochs os valores vão diminuindo, em ambas as curvas, o que mostra um modelo bem previsto, pois não há ocorrência de overfitting nem underfitting.

Por último, testamos o modelo com redes neurais utilizando as bibliotecas Keras e o GridSearch. Durante o processo, elaboramos um gráfico para visualizar o desempenho do modelo, especificamente observando os valores perdidos nos conjuntos de treino e validação em cada epoch. À medida que as epochs

aumentavam, notamos uma tendência consistente de redução nos valores perdidos em ambas as curvas. Este comportamento sugere que o modelo foi eficaz na aprendizagem dos padrões presentes nos dados, resultando em um bom desempenho. A ausência de overfitting ou underfitting é evidenciada pelo facto de que tanto os dados de treino quanto os de validação mostraram melhorias contínuas.

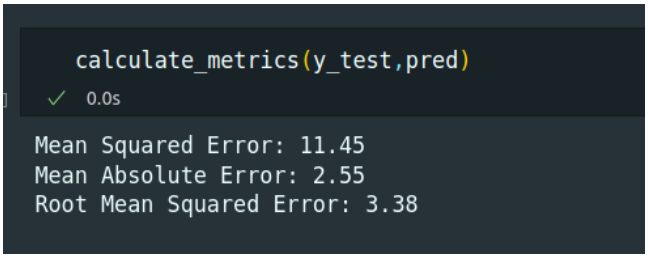


Figura 13: Resultados obtidos na Neural Network

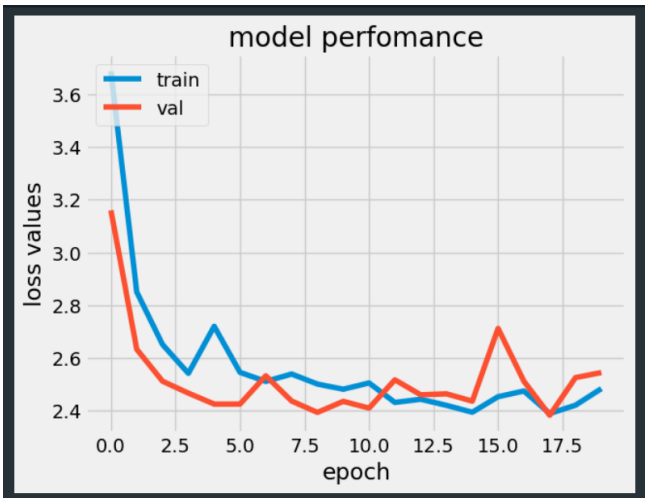


Figura 14: Model perfomance Neural Network

# 3 Dataset Competição

## 3.1 Análise e exploração de dados

O *dataset* de competição consistia em vários registos tanto de energia associados a um painel solar como a de meteorologia nessas mesmas horas. Num momento inicial havia uma divisão feita de datasets para teste e para treino do modelo, sendo que dois deles (energia e meteorologia) começava datado de setembro

de 2021 acabando o ano. E os outros dois começavam e completavam o ano de 2022. Os dados para teste datavam de inicio de 2023 ate ao mês 4 do mesmo ano. De reparar também algumas das colunas que à primeira vista pareceram mais chamativas antes de testar qualquer tipo de correlação, a coluna de **autoconsumo** e de **normal** presentes no dataset de energia, e as colunas de **temp** e de **humidity**. Chegou-se a esta conclusão depois de uma prévia busca sobre o funcionamento dos painéis solares que nos ajudou por todo o trabalho para tomarmos melhores decisões e escolhermos mais para a frente colunas a adicionar. O objetivo do modelo seria "adivinhar"a injeção na rede para cada hora do dataset de teste, por isso tratam-se os dados visando sempre a melhor decisão possível.

## 3.2 Tratamento e análise de dados

A primeira decisão tomada foi juntar todos os datasets para treinar o modelo, para isso, teve de se acrescentar algumas colunas para conseguir unir os dois datasets diferentes, estas colunas eram baseadas em pontos comuns entre eles (as datas e horas). Antes de se juntar os datasets decidiu-se tratar de alguns dados que fossem precisos, assim, fez-se o *label encoding* da coluna de injeção na rede.

```
column_mapping = {
    'Injeção na rede (kWh)': {'None': 0, 'Low': 1, 'Medium': 2, 'High': 3, 'Very High': 4},
}
for column, mapping in column_mapping.items():
    energiaDatasets[column] = energiaDatasets[column].map(mapping)

energiaDatasets.head()
```

Figura 15: Label Encoding da Injeção na rede

Ao unir os datasets de treino de meteorologia reparou-se em alguns missing values.

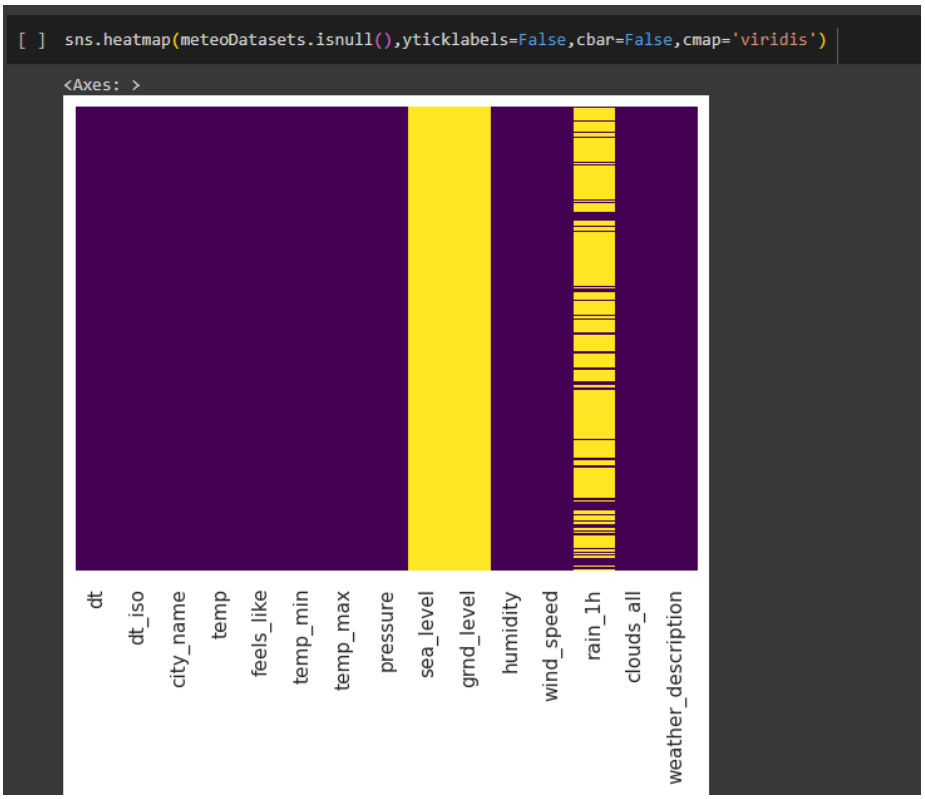


Figura 16: Missing Values meteorologia

Como as colunas do **sea_level** e **grnd_level** estavam sem valores nenhuns acabaram por ser *droped* logo no inicio, os missing values existentes no **rain_1h** foram passados para 0, a coluna acabou por ser eliminada mais a frente. Por fim o **city_name** como era sempre o mesmo também foi eliminada, sendo que não trazia nada de interessante ao modelo.

Como nos csv's da meteorologia se tinha valores que necessitavam de sofrer algum tipo de encoding, optou-se por fazer mais uma vez o *label encoding*. Esta coluna acabou por ser eliminada, desta vez porque mais a frente usamos uma API externa para completar dados e acrescentar colunas, e como este não constava nos dados, não podemos manter a coluna no nosso dataset.

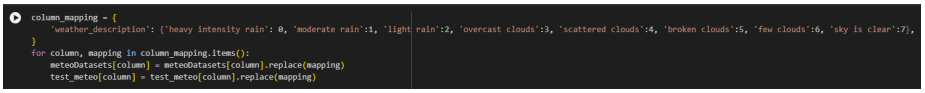


Figura 17: Label encoding weather description

Compararam-se as correlações entre as colunas do dataset da meteorologia e tiraram-se as que tinham maior correlação entre elas, isto fez-se para evitar o overfitting e para evitar usar mais do que uma coluna que traria resultados muito parecidos. Eliminaram-se então o temp_min e o temp_max.

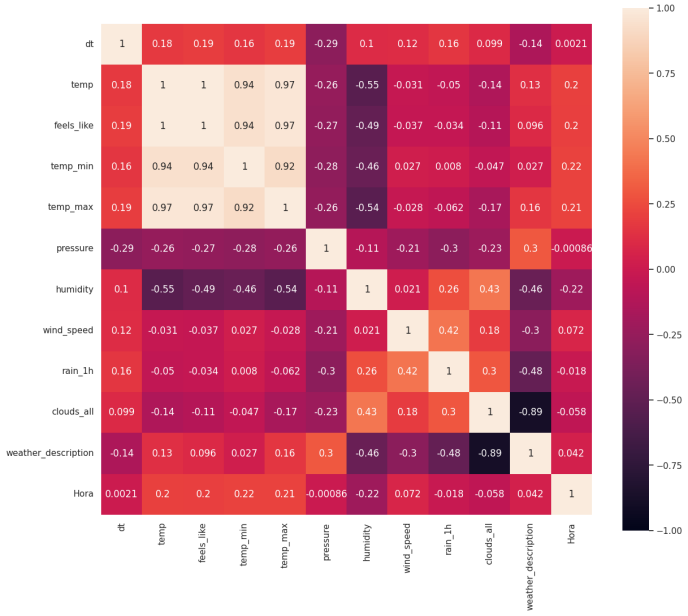


Figura 18: Matriz de correlação de meteorologia

Voltando à junção dos dois datasets diferentes, fez-se esta preparação de maneira a ser mais fácil treinar o modelo mais tarde.

```
meteoDatasets['dt_iso'] = meteoDatasets['dt_iso'].str.strip()
test_meteo['dt_iso'] =  test_meteo['dt_iso'].str.strip()

meteoDatasets['Data'] = meteoDatasets['dt_iso'].str.split().str[0]
test_meteo['Data'] = test_meteo['dt_iso'].str.split().str[0]

meteoDatasets['Hora'] = meteoDatasets['dt_iso'].str.split().str[1].str[:2].astype('int64')
test_meteo['Hora'] = test_meteo['dt_iso'].str.split().str[1].str[:2].astype('int64') # 2 primeiros digito
```

Figura 19: Junção de datasets

De maneira a vermos a correlação de todas as colunas dos dois datasets fez-se, uma vez mais, uma matriz de correlação. Nesta observou-se que a maior correlação com a coluna objetivo era com a coluna do **autoconsumo**, tendo também altas correlações com a **humidade**, a **temperatura** e a **feels_like**. Estas colunas foram todas mantidas, apesar de no inicio termos considerado retirar, esta coluna não trazia overfitting, e melhorava bastante a accuracy da solução, conclui-se então que se deveria manter a coluna.

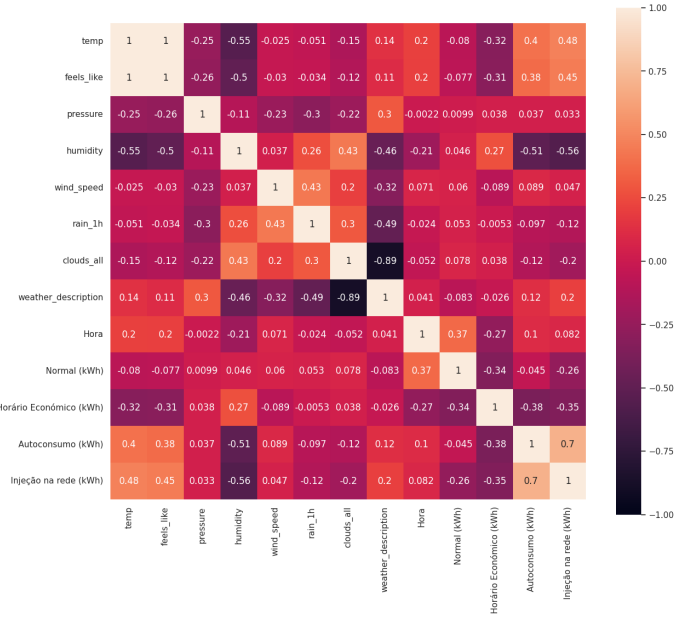


Figura 20: Correlação entre os dois datasets

Foi nesta altura que se reparou em alguns *missing values* por parte do *dataset* da meteorologia. Como neste caso não fazia sentido eliminar todas as colunas que constavam aqui, pois iriamos ter uma redução muito grande de dados de teste, o que não poderia acontecer sendo que a nossa submissão teria de ser feita com todas as linhas existentes. A primeira abordagem foi preencher os valores em falta com médias nos valores contínuos e modas nos valores discretos. Desta forma fomos conseguindo ter uma pontuação aceitável no **kaggle**, mas como não estava perto de ter uma pontuação agradável o suficiente para o grupo, decidiu-se ir mais longe e preencher os dados com valores reais, numa primeira tentativa, tentou-se contactar o **IPMA** sem sucesso, até encontrarmos, com a ajuda do grupo docente, uma aplicação que tinha uma API aberta à qual poderíamos fazer pedidos sobre os dados que precisássemos (**weatherBit**).

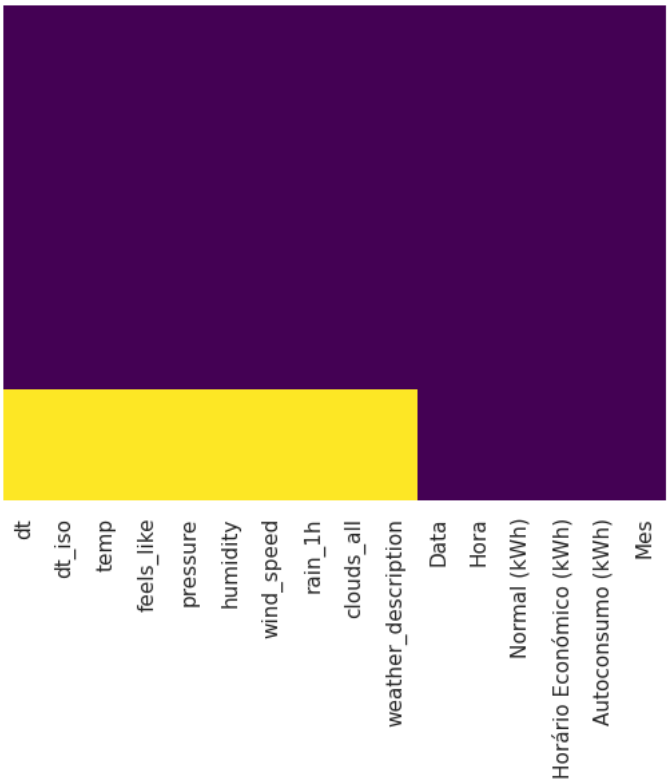


Figura 21: Missing Values - dataset de Teste

O uso do **weatherBit** fez com que se tivesse de abandonar algumas colunas na qual não continham informação nesta mesma aplicação, não recorremos a nenhum tipo de preenchimento alternativo pois estas não continham grande correlação com a coluna objetivo, tornando-se por isso dispensáveis nesta altura. Juntaram-se também novas colunas que se consideraram importantes para uma melhor performance do programa. Estas colunas foram:

- elev_angle - corresponde ao ângulo do sol
- solar_rad - radiação solar naquela hora do dia
- dni - *direct normal irradiance*, que corresponde à radiação solar que incide diretamente no painel

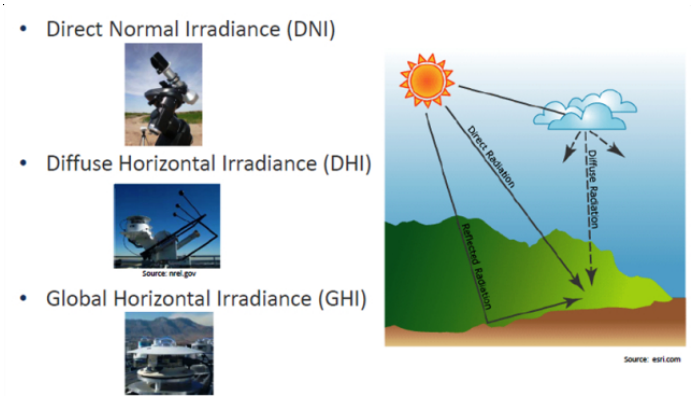


Figura 22:

Por fim, e após acrescentar algumas colunas novas ao modelo para termos uma maior correlação com a coluna objetivo e eliminamos outras que estavam bastante correlacionadas entre elas. As novas colunas foram criadas juntando colunas existentes usando métodos diferentes. Obteve-se então o seguinte matriz de correlação.

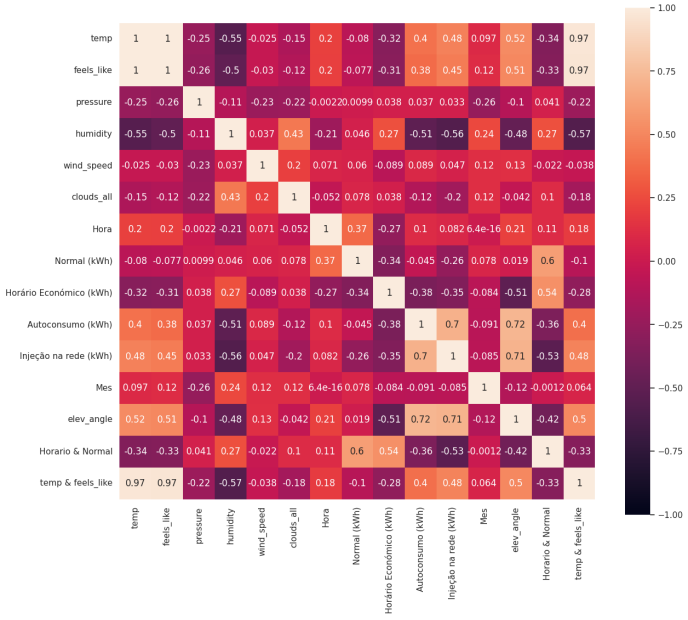


Figura 23: Matriz de correlação Final

É de notar que as colunas de **temp** e **feels_like** foram posteriormente eliminadas devido à grande correlação entre elas, as de **Normal** e **Horário**

**Económico** ficaram pois a correlação entre estas e a nova coluna não justificava a remoção das mesmas.

## 3.3 Modelos desenvolvidos

### 3.3.1 Support Vector Machine

O primeiro algoritmo de ML utilizado foi o Support Vector Machine, um algoritmo supervisionado utilizado para problemas de regressão e classificação, particularmente eficaz em problemas nos quais a separação entre classes é complexa e não linear. Utilizamos este algoritmo com o GridSearch , com os seguintes parametros:

```
    param_grid = {'C': [0.1,1,10,100,1000], 'gamma':[1,0.1,0.01, 0.001, 0.0001], 'kernel':[
```

Após as submissões no Kaggle, a accuracy mais elevada que se obteve foi de 83.43%.

### 3.3.2 Random Forest

De seguida, decidiu-se utilizar o algoritmo Random Forest pela sua eficiência. Este algoritmo também foi testado com o método GridSearch, onde os parâmetros passados eram:

```
param_grid = {
    'bootstrap': [True, False],
    'criterion': ['gini', 'entropy'],
    'max_depth': [80, 90, 100, 110],
    'max_features': [3, 4],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300, 1000]
}
rf = RandomForestClassifier()

grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                           cv = 3, n_jobs = -1, verbose = 2)
```

Após as submissões no Kaggle, a accuracy mais elevada que se obteve foi de 85.2%.

### 3.3.3 Xgboost

Por último, o algoritmo testado para as submissões finais foi o XbBoost com o método GridSearch. Este algoritmo é projetado para ser eficiente e escalável e , além disso, inclui a capacidade de lidar com dados ausentes sem a necessidade de pré-processamento e a incorporação de mecanismos para evitar overfitting. Os parametros utilizados foram os seguintes:

```
params = {
    'min_child_weight': [1, 5, 10],
    'gamma': [0.5, 1, 1.5, 5],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'max_depth': [1,3,5]
    }
```

Foi através deste algoritmo que obtivemos o valor de accuracy mais elevado, consequentemente, o valor final para a competição.

## 3.4 Submissões

A primeira submissão foi usando o SVM(support vector machine), nesta submissão obteve-se um score de 81%, apenas se fez as remoções básicas no tratamento de dados e o preenchimento do teste com média para os valores contínuos e moda para os discretos. Este valor ficou muito aquém do pretendido e serviu apenas de ponto de partida para o inicio do trabalho. Usaram-se também em algumas submissões algoritmos como o randomForest e o gradientBoosting de maneira a encontrar o algoritmo que nos iria permitir ter um resultado mais positivo, este acabou por ser o xgBoost, neste testamos a maior parte das nossas submissões fazendo sempre as alterações do tratamento de dados de maneira a ter um melhor resultado. Um dos pontos mais importantes a referir deverá ter sido a inclusão da coluna do autoconsumo que nos subiu a pontuação da ordem dos 85% para os 87%. Sendo que a maior subida que obtivemos a seguir a isso foi quando decidimos largar a ideia de "adivinhar"valores e recorrer mesmo a valores reais usando a aplicação WeatherBit, com o qual conseguimos a pontuação mais alta de 89%. A inclusão de novas colunas não trouxe nenhum aumento, assim, consideramos como melhor solução a que "apenas"se completaram os dados em falta recorrendo à API.

# 4 Conclusão

Durante o desenvolvimento do projeto, o grupo sentiu algumas dificuldades das quais se pode realçar o facto de encontrar o melhor processo de preparação dos dados de forma beneficiar os resultados dos modelos e encontrar os melhores modelos para fazer as previsões. Percebemos que a base de um bom trabalho neste âmbito reside mais na parte de preparação de dados do que propriamente no algoritmo escolhido para fazer a previsão. Consideramos que fizemos um trabalho bastante consistente, sendo que na competição ficamos apenas 1% abaixo do primeiro lugar, tendo feito sempre o que achávamos mais indicado para o problema apresentado tendo em conta todas as variáveis. Quanto ao dataset escolhido pelo grupo, achamos um tema bastante interessante que pode ajudar os alunos e os seus encarregados de educação a perceberem o que pode ajuda-los a obter melhor aproveitamento na escola.