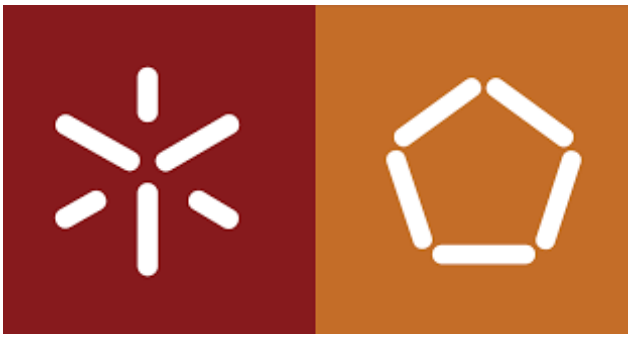

Universidade do Minho
Departamento de Informática

Dados e Aprendizagem Automática

Machine Learning

Grupo 20:
José Carvalho - PG53975
Délio Alves - A94557
José Barbosa - PG52689
Miguel Silva - PG54097

13 de janeiro de 2024

Conteúdo

Lista de Figuras

1 Introdução

Neste documento iremos relatar todas as etapas do desenvolvimento dos nossos modelos de decisão inteligente para o trabalho prático da unidade curricular de Dados e Aprendizagem Automática.

A estrutura deste relatório está dividida em duas secções predominantes nas quais falamos de como construimos modelos para responder aos problemas que surgem dos dados tanto para o *dataset* escolhido pelo grupo como para o proposto pelos docentes. Depois, temos mais duas secções uma para análise de resultados e outra na qual relatamos as nossas conclusões.

2 Dataset Grupo

2.1 Problema e Objetivos

Para o nosso *Dataset* Grupo selecionamos e trabalhamos sobre um *dataset* chamada *ks-projects-201801.csv*, que era sobre projetos no *KickStarter*, que é a maior plataforma online de *crowd funding*. O nosso objetivo e aquilo a que nos desafiamos a fazer, quando estivemos a tratar deste, foi em volta da label *state* sendo que queríamos descobrir se um projeto tinha atingido o seu objetivo ou se tinha falhado. Apesar da label poder ter outros valores (como, por exemplo, um projeto ser cancelado) esses não foram considerados.

2.2 Metodologia

Para este *dataset* seguimos uma metodologia semelhante à *CRISP-DM*. Os passos realizados foram:

- **Seleção e Objetivo** - O projeto envolveu a análise do *datasets "ks-projects-201801.csv"*, que contém informações sobre projetos de *crowdfunding* no *Kickstarter*. O foco principal foi identificar se um projeto alcançaria seu objetivo financeiro ou falharia, com base na variável '*state*'.
- **Preparação dos Dados** - Inicialmente, as colunas que não contribuíam para a análise, como '*name*' e '*ID*', foram removidas. Também tratamos valores ausentes, especialmente na coluna '*usd pledged*', removemos as entradas com dados em falta.
- **Filtragem e Redução de Dados** - Para simplificar a análise, filtramos os dados para manter apenas projetos com estados '*failed*' ou '*successful*'. Além disso, reduzimos significativamente o tamanho do *dataset* para facilitar o manuseio e a análise.
- **Enriquecimento dos Dados** - Criamos atributos, como '*country continent*', '*days launched*' e '*launched month*', para explorar diferentes variações dos projetos.
- **Análise Exploratória** - Realizamos uma análise exploratória para entender as categorias de projetos e sua relação com o sucesso. Observamos que certas categorias como *Theater* e *Comics* tinham maiores taxas de sucesso.
- **Preparação para Modelagem** - A análise preliminar indicou uma tendência maior de falha nos projetos, com uma taxa de sucesso pouco acima de 40%. Esta informação foi importante para as etapas seguintes de modelagem preditiva.

2.3 Descrição dos dados

O *dataset ks-projects-201801.csv* conta com 378660 entradas, o que representa 378660 projetos na plataforma *Kickstarter* cujos atributos são os seguintes:

- ID - identificador único do projeto
- name - nome do projeto
- category - categoria genérica do projeto, por exemplo: música, poesia, etc.
- main_category - categoria mais expecifica do projeto, por exemplo: publicação, albúm, produção, livro, etc.
- currency - moeda utilizada para o *crowdfunding*, pode ser: euro, dólar, iene, libras, entre outras.
- deadline - data limite para o fim do *crowdfunding*.
- goal - valor que o criador do projeto deseja obter.

- launched - data de começo do *crowdfunding*.
- pledged - valor monetário obtido até ao momento.
- state - estado do projeto, falhado, em pausa, bem sucedido, etc.
- backers - número de pessoas que doaram dinheiro para o projeto.
- country - pais de origem do projeto.
- usd_pledged - conversão do valor do dinheiro obtido pelo *crowdfunding* para dólares através da própria plataforma *kickstarter*.
- usd_pledged_real - conversão do valor do dinheiro obtido pelo *crowdfunding* para dólares através da API *Fixer.io*.
- usd_pledged_goal - conversão do valor do dinheiro desejado (goal) *crowdfunding* para dólares através da API *Fixer.io*.

2.4 Exploração dos dados

Após uma analise aos dados originais, inicializamos o tratamento e a exploração dos mesmos. Relembrando que a nossa *label* é o atributo *state* e que tencionamos prever se um projeto é bem-sucedido ou não.

Como o nosso *dataset* é enorme, iremos tentar remover entradas, sempre que possível.

Começamos por eliminar as colunas *name* e *ID*, devido ao facto de que cada entrada tem um valor único, não tem qualquer para a fase de modelação.

Em relação aos *missing values*, ao fazer uma análise, constatamos que a coluna onde isto era um problema era na *usd_pledged*, onde existiam 3797 valores em falta. Para os eliminar estes *missing values*, optamos por remover as entradas que continham valores nulos.

Como nós referimos anteriormente, a nossa *label* nos dados originais contém outros valores, que não ***failed*** ou ***sucessfull***, então começamos por remover essas entradas.

Linhas que possuíam outros erros ou dados que não nos diziam nada de útil, como, por exemplo, linhas onde *usd_pledged* era nula, linhas onde *launched* era igual a "1970-01-01 01:00:00", ou linhas onde o atributo *country* era vazio acabamos por remover também.

Tal como já foi referido, o *dataset* é enorme, contendo neste momento 378660 entradas e de forma a que nós o conseguirmos analisar e treinar modelos em tempo útil, achamos por bem apagar 326465 dessas entradas.

Para melhorar os modelos, criamos alguns atributos novos com base nos existentes. O atributo *country_continent* representa o continente em que se encontra o país de origem do projeto. Isto foi feito com o intuito de verificar que influencia tinha o continente em que o projeto começou no seu sucesso.

Além disso, criamos os atributos *days_launched* e *launched_month*. O *days_launched* representa o número de dias que um projeto já foi lançado. O *launched_month* é o mês em que o projeto foi lançado.

Uma aspeto que quisemos observar nos dados foi a no impacto que a *main_category* poderia ter no valor da *label*. A intenção é verificar se seria possível juntar algumas das categorias, de carácter semelhante, como, por exemplo, dança e teatro, uma vez que podia ser mais informação que nos podia ajudar nas nossas previsões, sendo que consideramos que algumas destas combinações poderiam dar-nos percentagens interessantes. Contudo, acabamos por verificar que não era algo que valia a pena. No entanto, constatamos que os projetos com maior percentagem de sucesso são os de **Theater (Teatro)**, **Comics (Banda-Desenhada)** e **Dance (Dança)**. Mas estas categorias principais representam uma percentagem muito pequena dos projetos do *Kickstarter*. *Film & Video (Filme e video)*, **Music (Musica)** e **Publishing (Publicações)**, são os tipos de projetos 3 mais comuns.

Depois deste tratamento dos dados, constamos que na nossa *label*, há uma tendência maior para os projetos falharem do que serem bem sucedidos, sendo que a taxa de sucesso é ligeiramente a cima dos 40%, ou seja, é algo a ter em conta aquando a modelação.

Após este processo analisamos a uma matriz de correlação final (fig 1).

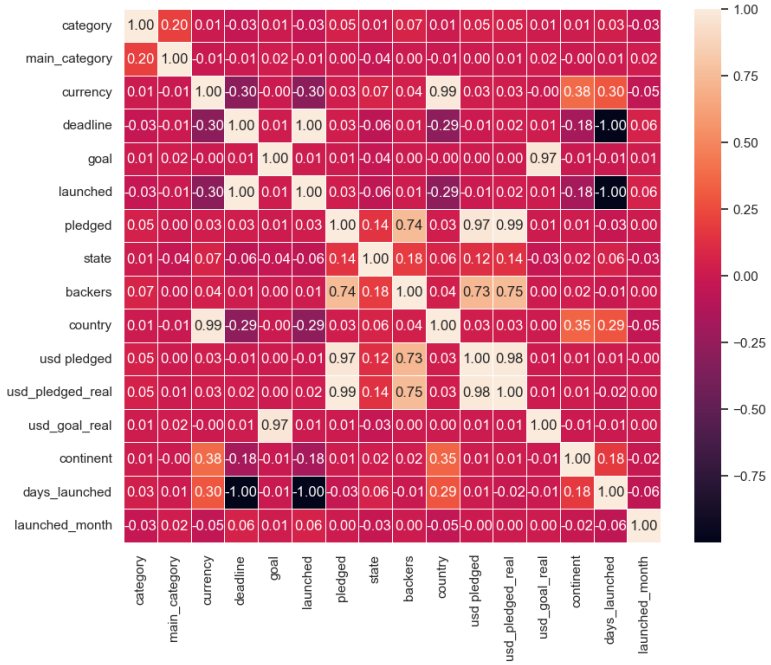

Figura 1: Matriz de Correlação

A partir da matriz de correlação, concluimos que os atributos mais importantes são:

- *pledged*
- *state* (*label*)
- *backers*
- *usd pledged*
- *usd_pledged_real*

Também analisamos alguns gráficos boxplot (fig 2), com o intuito de visualizar a relação da nossa label com alguns outros atributos (*goal, usd_goal_real, usd_pledged e usd_pledged_real*).

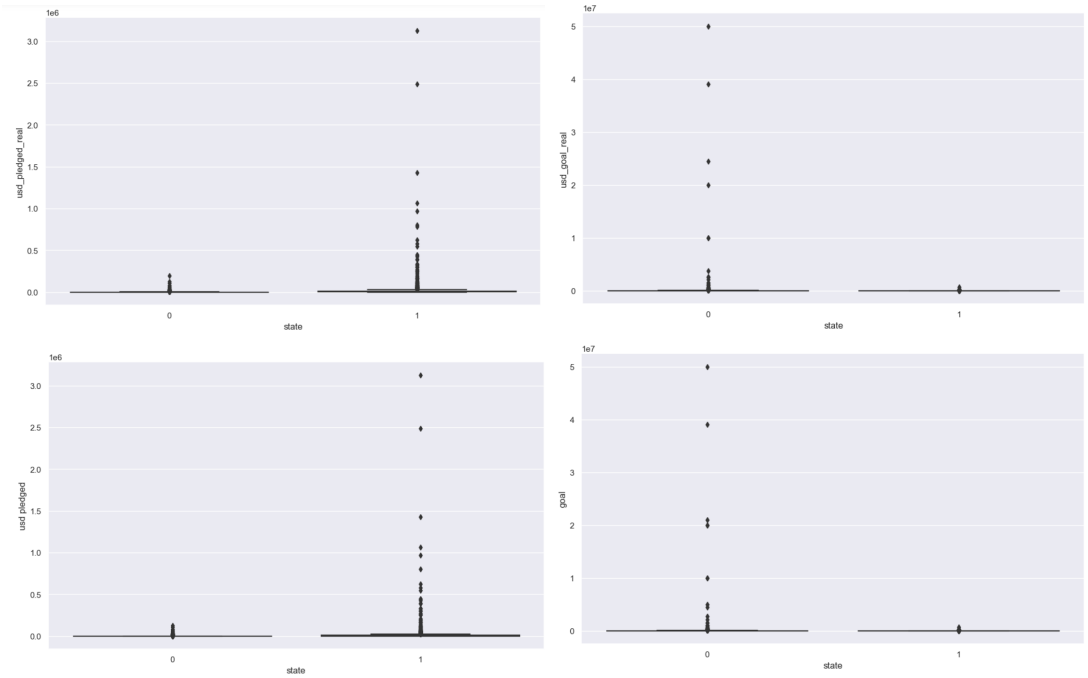

Figura 2: Boxplot Dataset Grupo

3 Dataset Competição

3.1 Problema e Objetivos

No que toca ao *Dataset* Competição, foram nos fornecidos 6 *datasets* diferentes que individualmente continham dados meteorológicos e energéticos dos anos de 2021, 2022 e 2023, na cidade de Braga. Nesta parte do trabalho prático, fomos desafiados a desenvolver modelos capazes de prever a quantidade energética, em KWh. Este problema, toca na previsão de energia, tendo repercussões notáveis não apenas na eficiência energética, mas também na diminuição das emissões de gases que causam o efeito estufa e na promoção da sustentabilidade.

3.2 Metodologia

Tal como no *dataset* Grupo, neste *dataset* seguimos uma metodologia semelhante à *CRISP-DM*. Os passos realizados foram:

- **Preparação e Limpeza dos Dados**
  - Concatenação dos *datasets* de treino (2021 e 2022) para meteorologia e energia.
  - Substituição dos valores ausentes na coluna 'Injeção na rede (kWh)' por 0.
  - Conversão de categorias em números utilizou-se *Label Encoding* para a *label*.

- **Transformação dos Dados**
  - Combinação das colunas 'Data' e 'Hora' em uma única coluna de data e hora, e conversão para o tipo *datetime*.
  - Remoção de colunas com muitos valores ausentes ('*sea level*', '*grnd level*') no *datasets* meteorológico e preenchimento de valores ausentes em '*rain 1h*' com 0.
  - Conversão de '*weather description*' para valores numéricos através de *Label Encoding*.

- **Preparação para Modelagem**
  - *Merge* dos datasets de energia e meteorologia com base na coluna 'Data'.
  - Criação do atributo 'Estacao' para analisar o impacto das estações do ano.
  - Análise de *boxplot* para visualizar a relação entre a *label* e outros atributos e identificar *outliers*.

- **Seleção de Atributos**
  - Análise da matriz de correlação para identificar e selecionar atributos relevantes para a modelagem.

- **Construção de Modelos de Machine Learning**
  - Aplicação de diversos modelos (Árvores de Decisão, *Random Forest Trees*, Regressão Logística, *SVM*, Redes Neuronais, *Bagging*, *Voting*, *XGBoost*) nos *datasets*.
  - Ajuste de hiperparâmetros e otimização de modelos usamos técnicas como *Grid Search*.
  - Uso de técnicas como *SMOTE* para tratar desequilíbrios nos dados.

3.3 Descrição dos dados

Os dados utilizados para este problema vêm de vários *datasets*, nomeadamente: *meteo_202109-202112.csv*, *meteo_202201-202212.csv*, *meteo_202301-202304.csv (test)*, *energia_202109-202112.csv*, *energia_202201-202212.csv*, *energia_202301-202304.csv (test)*

Ou seja temos dados para meteorologia e para consumos de energia, sendo portanto os dados dos *datasets* de meteorologia diferentes dos dados dos *datasets* de energia a nível de atributos.

Os *datasets* de energia contam com os seguintes atributos:

- Data - *timestamp* do registo;
- Hora - a hora do registo;
- Normal (kWh) - quantidade de energia elétrica consumida, em kWh e proveniente da rede elétrica, num período considerado normal em ciclos bi-horário diários (horas fora de vazio);
- Horário Económico (kWh) - quantidade de energia elétrica consumida, em kWh e proveniente da rede elétrica, num período considerado económico em ciclos bi-horário diários (horas de vazio);
- Autoconsumo (kWh) - quantidade de energia elétrica consumida, em kWh, proveniente dos painéis solares;
- Injeção na rede (kWh) - quantidade de energia elétrica injetada na rede elétrica, em kWh, proveniente dos painéis solares.

Em contra partida, os *datasets* de meteorologia aprentam como atributos:

- dt - *timestamp* do registo;
- dt_iso - a data associada ao registo medida até ao segundo;
- city_name - nome da cidade onde é feito o registo;
- temp - temperatura em ^{o}C;
- feels_like - sensação térmica em ^{o}C;
- temp_min - temperatura mínima sentida em ^{o}C;
- temp_max - temperatura máxima sentida em ^{o}C;
- 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.

3.4 Exploração dos dados

Nesta fase do trabalho prático recebemos, como já referimos em cima, seis *datasets*, três relacionados com meteorologia e três relacionados com energia.

Primeiramente começamos por concatenar os *datasets* de treino, ou seja, os *datasets* referentes aos anos de 2021 e 2022.

Em relação aos *missing values* no *dataset* de energia, constatamos que o atributo **Injeção na rede (kWh)**, que é a nossa *label*, tinha 7777 valores em falta. Para resolvermos este problema e como se tratava da nossa *label*, decidimos substituir os valores em falta, por 0.

Após isto, convertemos a *label* que é um atributo categórico, em um atributo numérico. Para isso utilizamos a técnica *Label encoding*. O objetivo era tornar os valores (*0, Low, Medium, High, Very High*) em um número, de 0 até 4.

A seguir, nos dois *datasets* de energia, procedemos à combinação das colunas *Data* e *Hora* numa única coluna de data e hora, acabando por guardar esse valor com o tipo *datetime* na coluna *Data* e removemos a coluna *Hora*.

Em relação aos *missing values* no *dataset* de meteorologia, identificamos que existiam nas colunas *sea_level*, *grnd_level*, *rain_1h* havia valores em falta. Optamos por remover as colunas *sea_level* e *grnd_level*, completamente, pois para além de ter um grande número de valores em falta (cerca de 11 mil), achamos que era algo cuja informação não teria um impacto significativo. No que toca ao *rain_1h*, preenchemos os valores em falta com 0, tal como o atributo **Injeção na rede (kWh)** do dataset da energia.

De seguida, decidimos por tornar o atributo *weather_description* num atributo numérico, em vez de ser categório. Para tal, usamos mais uma vez o *Label encoding*, utilizando valores de 1 a 8.

Convertemos a coluna *dt_iso* para um formato de data e hora, e fomos substituí-la por uma nova coluna *Data* com os valores da coluna *dt_iso*, com o objetivo de realizar um *merge* entre o *dataset* da energia e o *dataset* da meteorologia.

Após este processo fizemos um *merge*, tal como já referido, entre os *datasets data_energia* e *data_meteo*. Este *merge* foi feito com base na coluna *Data* de ambos os *datasets*. Assim, ficamos apenas com as linhas nos quais existiam informações sobre a mesma data nos dois *datasets* iniciais, para não ficarmos com dados incompletos nesta junção.

Todos estes passos foram também aplicados aos *datasets* de teste (*datasets* de 2023).

No entanto, nós verificamos que no *dataset* de teste de meteorologia, havia informação em falta. Mais precisamente, não tínhamos os dados meteorológicos entre 15/5/2023 e 5/4/2023. Ora bem, nós necessitamos dessa informação, de forma a prever a injeção das redes nessas datas. Primeiramente pensamos em simplesmente meter todos as colunas com um valor por defeito. Depois optamos por alterar e usar interpolação linear. Por último, alteramos novamente e decidimos usar os dados meteorológicos entre 15/5/2022 e 5/4/2022 e replica-los. Estas mudanças foram com o objetivo de obter melhores previsões nos nossos modelos, sendo que notamos uma melhoria sempre que efetuamos essas alterações.

No fim de este processo, ficamos com *data*, que contém os dados energéticos e meteorológicos 2021 e 2022 e idem para o *data_test*, só que para 2023.

Em relação a nova informação, criamos um atributo chamado *Estacao*, com o intuito de verificar o impacto da estação do ano nos dados em relação à nossa *label*, já que as condições meteorológicas variam considerando a estação.

Como falamos em cima, a nossa *label* possuía imensos valores em falta, algo que marcamos como 0. Devido a isso, concluímos que esse é o resultado predominante, entre os resultados possíveis, representando cerca de 70% dos dados.

Analisamos alguns gráficos boxplot (fig 3), com o intuito de visualizar a relação da nossa *label* com alguns atributos e obter informações sobre onde se encaixavam os valores em relação da *label* e informação adicional sobre *outliers* (*humidity , Autoconsumo (kWh) e temp*) representados.

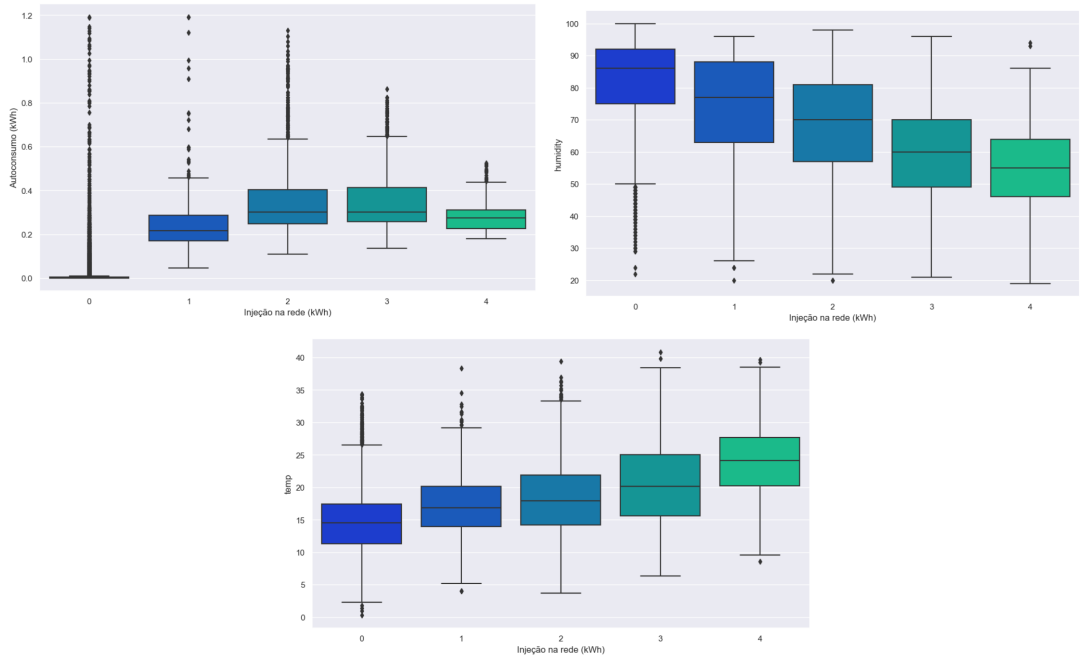

Figura 3: Boxplot Dataset Competição

Para completar esta secção, resta referir que, após análise de uma matriz de correlação (fig 4), definimos que os atributos que iríamos utilizar e manter durante a modelação eram:

- *Normal (kWh)*
- *Horário Económico (kWh)*
- *Autoconsumo (kWh)*
- *Injeção na rede (kWh) (label)*
- *temp*
- *feels_like*
- *temp_min*
- *temp_max*
- *humidity*
- *weather_description*
- *Estacao*

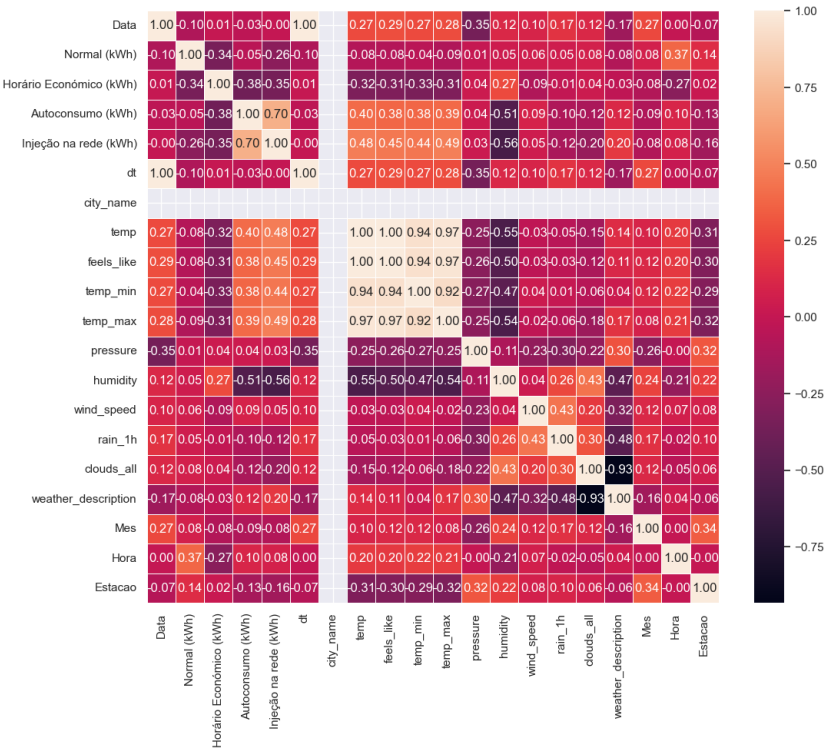

Figura 4: Matriz de Correlação

4 Modelos

Como forma de aprendizagem, aplicamos os mesmos modelos para ambos os *datasets*. Antes de partir para a explicação de cada um, conseguimos concluir ou presumir certos resultados. Como ambos os problemas são de classificação, técnicas como regressão linear poderão não ser muito ideias. A regressão logística poderá ser útil para o *dataset* grupo, pois a *label* só toma valores binários, mas para o *dataset* de competição poderá não ser muito indicado. Também é expectável que modelos como *XGBoost* tenha melhores resultados que o modelo de *Random Forest Trees* e este seja melhor que as árvores de decisão.

Houve uma etapa adicional aos dados nesta fase e que foi em comum em todos os modelos. Esta etapa foi Remover atributos com pouca correlação com a *label*.

4.1 Árvores de decisão + *Random Forest Trees*

Como as árvores de decisão e as *Random Forest Trees* são modelos com alguma relação, decidimos juntar ambas num único *notebook*. Estes modelos não precisam de nenhum tratamento adicional relativamente aos dados. Em ambos os *datasets* realizamos as seguintes etapas:

1. Partir os dados em **dados de teste** e **dados de treino**, sendo a proporção de 75% para 25% para o *dataset* competição e 80% para 20% para o *dataset* grupo respetivamente.
2. Analisar a proporção dos dados.
3. Construir uma Árvore de decisão.
4. Construir uma *Random Forest Trees*.
5. Usar o *Grid Search* para testar vários hiperparâmetros da *Random Forest Trees*.
6. Análise dos resultados dos 3 modelos (Árvore de decisão, *Random Forest Trees*, *Grid Search*).

Os resultados obtidos para o *dataset* competição estão na imagem 5 . A figura contém as matrizes de confusão para os 3 modelos, sendo a primeira relativa às Árvore de decisão, a segunda às *Random Forest Trees* e a terceira a *Grid Search*. As tabelas 4.1 contém valores das métricas de avaliação de modelos.

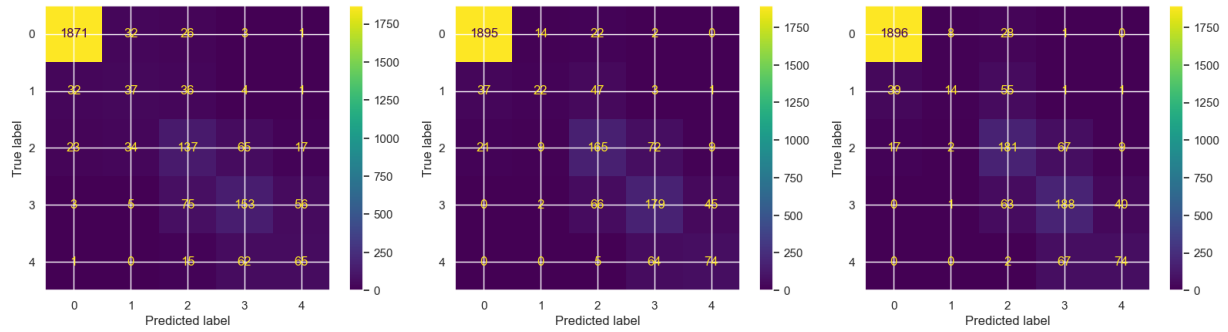

Figura 5: Resultados *Dataset* competição árvores

Modelo	*Accuracy*	Precisão	*Recall*	*F1-score*
Árvore de decisão	82%	82%	82%	82%
Random Forest Trees	85%	84%	85%	84%
Grid Search	85%	85%	85%	85%

Tabela 1: Resultados árvores

Para o *dataset* a imagem 6 contém as matrizes de confusão, sendo que a ordem dos modelos é a mesma que o *dataset* grupo, e os resultados das métricas estão na tabela 4.1,

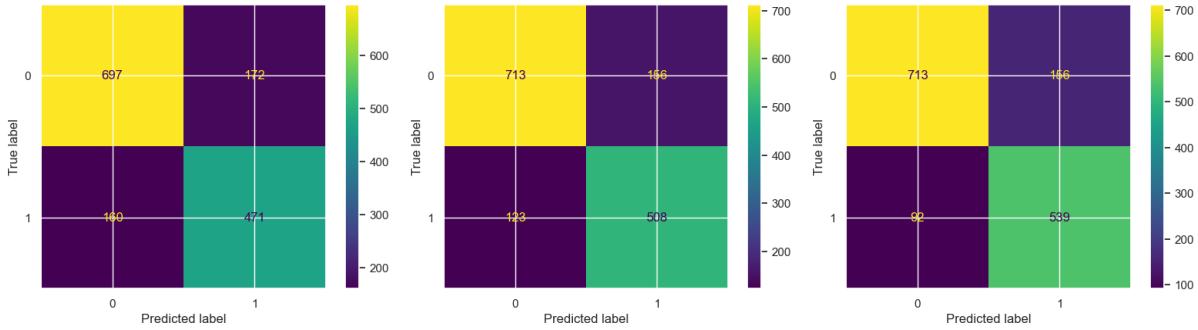

Figura 6: Resultados *Dataset* grupo árvores

Modelo	*Accuracy*	Precisão	*Recall*	*F1-score*
Árvore de decisão	78%	78%	78%	78%
Random Forest Trees	81%	82%	81%	81%
Grid Search	83%	84%	83%	84%

Tabela 2: Resultados árvores

4.2 Regressão linear

Semelhante aos modelos apresentados na secção 4.1, não foi necessário fazer um tratamento adicional aos dados em ambos os *datasets*. Como já foi referido na introdução da secção 4, este tipo de modelos é

presumível que não iremos ter bons resultados, pois ambos os problemas são de classificação. As etapas realizadas para esta modelação foram:

1. Partir os dados em **dados de teste** e **dados de treino**, sendo a proporção de 75% para 25% para o *dataset* competição e 80% para 20% para o *dataset* grupo respetivamente.
2. Analisar a proporção dos dados.
3. Analisar a relação dos atributos com a *label*.
4. Construir um modelo de Regressão linear.
5. Análise dos coeficientes.
6. Análise dos resultados.

A figura 7, demonstra a relação dos diferentes atributos com a *label* de ambos os *datasets*, utilizando gráficos de pontos. A parte de cima é em relação ao *dataset* competição e o segundo é referente ao *dataset* grupo. Com isto, conseguimos verificar que não é possível traçar uma reta que seja possível prever o resultado da *label* facilmente. Assim concluímos que as nossas previsões estavam corretas e este modelo não irá dar bons resultados.

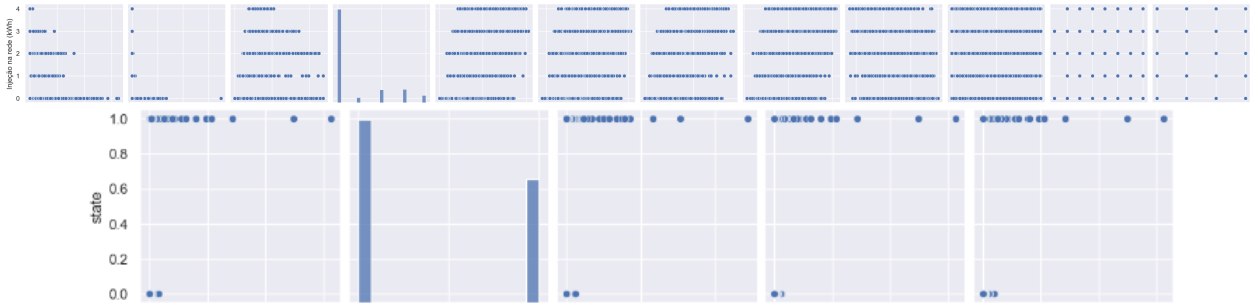

Figura 7: Correlação ambos *Datasets*

As figuras presentes em 8 contém os resultados de ambos os *datasets*, sendo a figura da esquerda relativa ao *dataset* competição e a da direita *dataset* grupo. Tal como já tinhamos concluído, os resultados são péssimos, mas esperados.

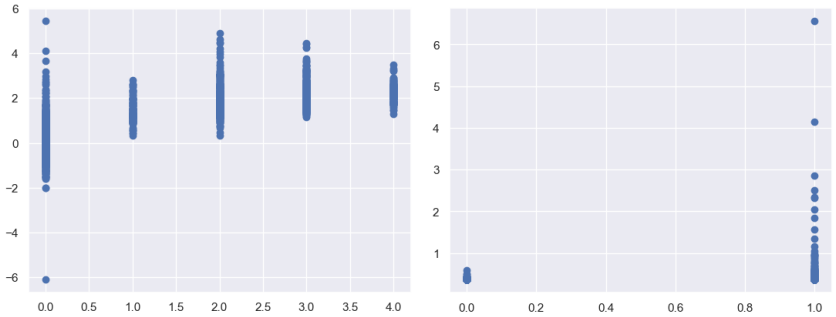

Figura 8: Resultados Regressão Linear

As métricas que utilizamos para analisar os modelos foram a *MAE*, a *MSE* e a *RMSE*. Os resultados encontram-se nas tabelas 4.2 e 4.2, sendo que a primeira é referente ao *dataset* competição e a segunda em relação ao *dataset* grupo.

Métrica	Valor
MAE	0.55
MSE	0.61
RMSE	0.78

Tabela 3: Resultados métricas competição Regressão Linear

Métrica	Valor
MAE	0.47
MSE	0.26
RMSE	0.51

Tabela 4: Resultados métricas grupo Regressão Linear

4.3 Regressão logística

Tal como já referido, o motivo pelo qual decidimos explorarmos a regressão logística foi sobretudo pelo *dataset* grupo, que é necessário prever valores binários.

Para ambos os *datasets* decidimos construir 3 modelos de regressão logística, em que a diferença é o hiperparâmetro *solver*, sendo que usamos os seguintes valores:

- *newton-cg*.
- *lbfgs*.
- *liblinear*.

O processo foi bastante similar aos modelos já apresentados, ou seja, particionamento dos dados, visualização da proporção dos dados, construir os 3 modelos e comparar os resultados.

A figuras presentes em imagem 9 ilustram os resultados obtidos para o *dataset* da competição. Cada uma das figuras representa um modelo, sendo que a figura mais à esquerda é em relação ao *newton-cg*, a central é *lbfgs* e a última é *liblinear*. Os resultados das métricas encontram-se na tabela 4.3.

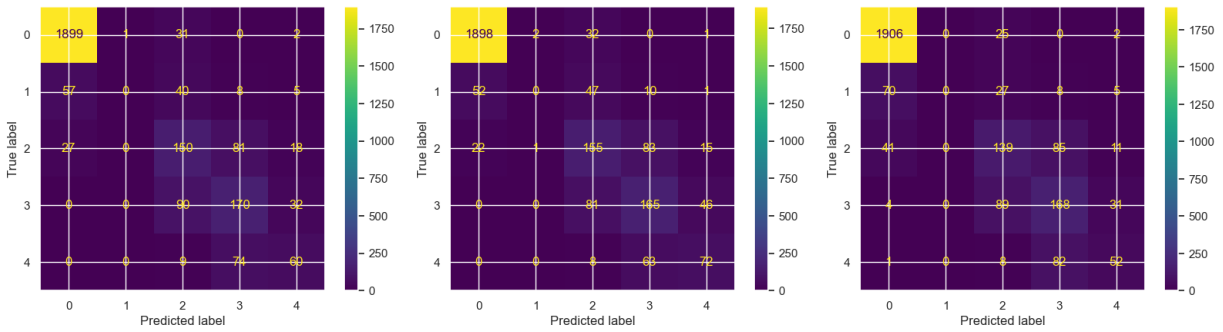

Figura 9: Resultados *Dataset* competição Regressão Logística

score	*Accuracy*	Precisão	*Recall*	*F1-score*
newton-cg	83%	80%	83%	81%
lbfgs	83%	81%	83%	82%
liblinear	82%	83%	82%	80%

Tabela 5: Métricas *Dataset* competição Regressão Logística

A figuras presentes em na imagem 10 ilustram os resultados obtidos para o *dataset* grupo. Tal como no caso anterior, cada uma das figuras representa um modelo, sendo que a figura mais à esquerda é em relação ao *newton-cg*, a central é *lbfgs* e a última é *liblinear*. Os resultados das métricas encontram-se na tabela 4.3.

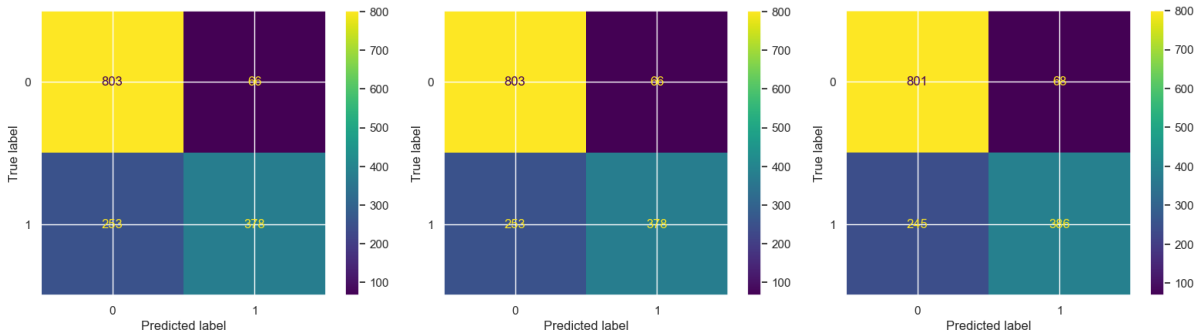

Figura 10: Resultados *Dataset* grupo Regressão Logística

score	*Accuracy*	Precisão	*Recall*	*F1-score*
newton-cg	79%	80%	79%	78%
lbfgs	79%	80%	79%	78%
liblinear	79%	80%	79%	78%

Tabela 6: Métricas *Dataset* grupo Regressão Logística

4.4 *Support Vector Machines*

As *Support Vector Machines*, ou as *SVMs*, foram outro modelo que decidimos abordar.

O motivo pelo qual optamos por utilizar este modelo foi pela diversa quantidade de parâmetros com os quais podemos experimentar. Com isto em mente utilizamos *GridSearch* para realizar *Hiperparameter Tuning*, obtendo então os melhores estimadores possíveis a partir dos fornecidos.

Acabamos por apenas ter selecionado *kernel rbf*, isto dá-se porque os restantes *kernels* davam resultados não muito distintos para os mesmos parâmetros e para outros parâmetros demoravam imenso tempo a correr. Preferimos portanto focar noutros modelos do que tentar otimizar ao máximo as *SVMs*.

Os resultados obtidos por este modelo para o *dataset* da competição podem ser consultados na figura 11, sendo que à esquerda é sem *GridSearch*, à direita com *Gridsearch*.

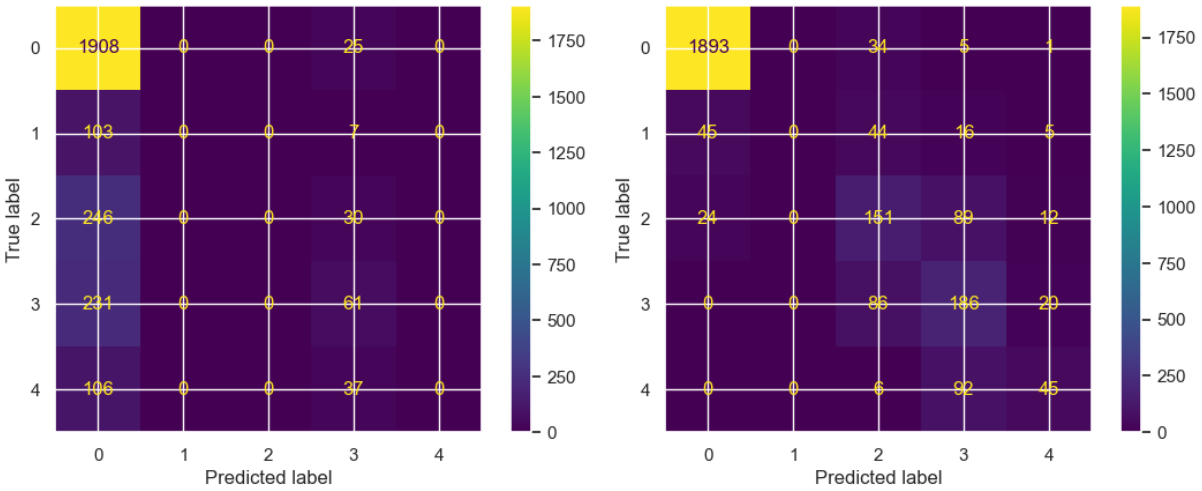

Figura 11: Resultados *Dataset* competição *SVMs*

Modelo	*Accuracy*	Precisão	*Recall*	*F1-score*
sem GridSearch	71%	56%	71%	62%
com GridSearch	83%	80%	83%	81%

Tabela 7: Métricas *Dataset* competição *SVMs*

Enquanto que no *dataset* escolhido pelo grupo obtivemos os resultados presentes na figura 12, mais uma vez à esquerda sem *GridSearch*, à direita com *Gridsearch*.

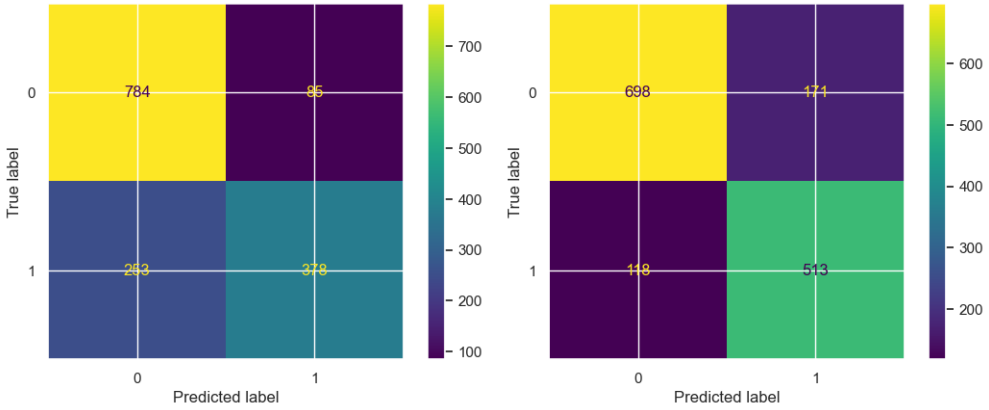

Figura 12: Resultados *Dataset* grupo *SVMs*

Modelo	*Accuracy*	Precisão	*Recall*	*F1-score*
sem GridSearch	77%	78%	77%	77%
com GridSearch	81%	81%	81%	81%

Tabela 8: Métricas *Dataset* grupo *SVMs*

4.5 Rede neuronal

Nas redes neuronais, inicialmente realizamos o particionamento dos dados e observamos essa mesma partição.

Para ambos os *datasets* decidimos normalizar os dados atributos que não são a *label*. Isto ocorre para eliminar diferenças nas grandezas.

Um aspeto importante de salientar, é quando já tinhamos os modelos feitos, a nossa rede neuronal para o *dataset* da competição estava sempre a dar como previsão *None*. Isto acontece por causa que os dados originais estavam muito desproporcionais entre o resultado *None* e os restantes. De forma a equilibrar, usamos a técnica *SMOOTE*. A biblioteca usada para tal efeito foi a *imbalanced-learn*. Para mais informações sobre o *SMOTE* e sobre esta biblioteca, temos a secção 7.1.

Após isso, para ambos os *datasets* começamos a construir as redes. Para os 2 casos, usamos uma rede neuronal com 4 camadas e toda a informação das redes neuronais para ambos os *datasets* encontra-se nas tabelas 4.5, 4.5 e 4.5.

Dataset	Nodos 1	Nodos 2	Nodos 3	Nodos 4	Ativação
dataset competição	32	16	8	5	*relu* + *softmax*
dataset grupo	16	12	8	1	*sigmoid*

Tabela 9: Rede neuronal construção 1

Dataset	Perda	*batch_size*	*epochs*
dataset competição	*sparse_categorical_crossentropy*	40	20
dataset grupo	*binary_crossentropy*	40	60

Tabela 10: Rede neuronal construção 2

Dataset	*learning_rate*	Métricas
dataset competição	0.01	*accuracy*
dataset grupo	0.01	*accuracy + precision + recalls*

Tabela 11: Rede neuronal construção 3

O motivo pelo qual usamos mais nodos na rede neuronal no *dataset* competição em comparação ao *dataset* grupo tem a ver com o facto que a nossa rede neuronal estava *underfitted* para um número pequeno de nodos, mas também em compensação usamos um número menor de *epochs*.

Mais uma vez também optamos por usar o *Grid Search* para encontrar qual o melhor otimizador das redes, sendo que para o *dataset* grupo foi o *SGD* e para o *dataset* competição foi o *RMSprop*.

Após isso, optamos por usar o melhor estimador fornecido pelo *Grid Search* para as previsões.

A figura 13 ilustra os gráficos obtidos da rede neuronal para o *dataset* da competição e a figura 14 ilustra os mesmos gráficos, mas desta vez para o *dataset* grupo.

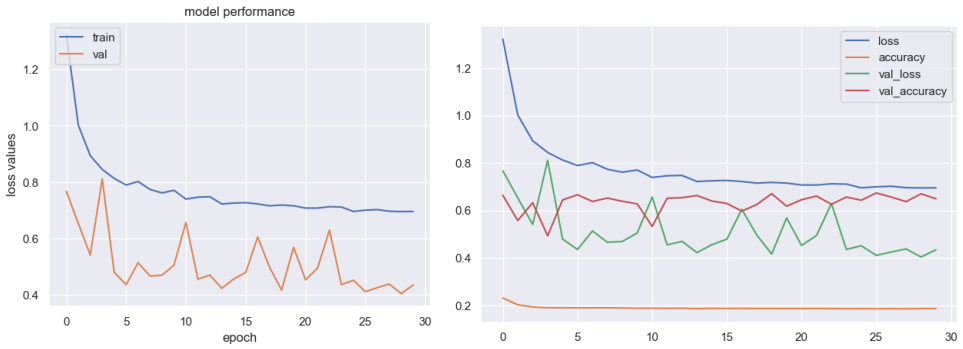

Figura 13: Avaliação modelo *Dataset* competição Rede Neuronal

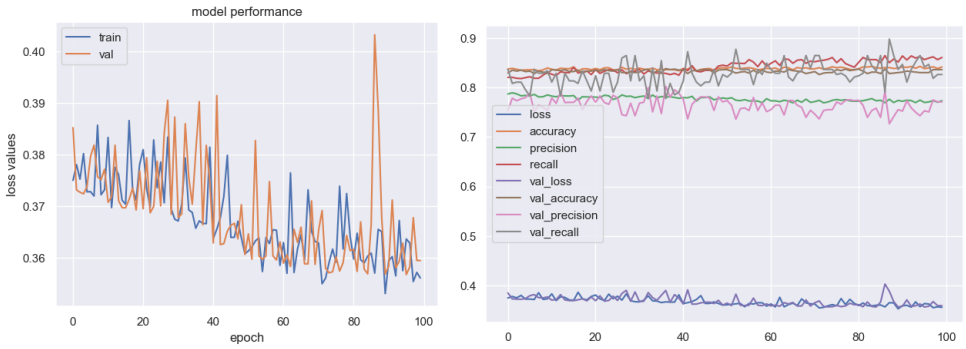

Figura 14: Avaliação modelo *Dataset* grupo Rede Neuronal

Com estes gráficos concluimos que os nossos modelos não estão nem *underfitted* nem *overfitted*, então seguimos em frente. É importante salientar, que inicialmente tivemos modelos que estavam *underfitted* e foi a partir destes gráficos que melhorarmos os nossos modelos.

Em relação aos resultados, a figura 15 ilustra os resultados obtidos sob a forma de matriz de confusão, para ambos os *datasets*, e a tabela 4.5 demonstra os resultados das métricas para ambos os *datasets*.

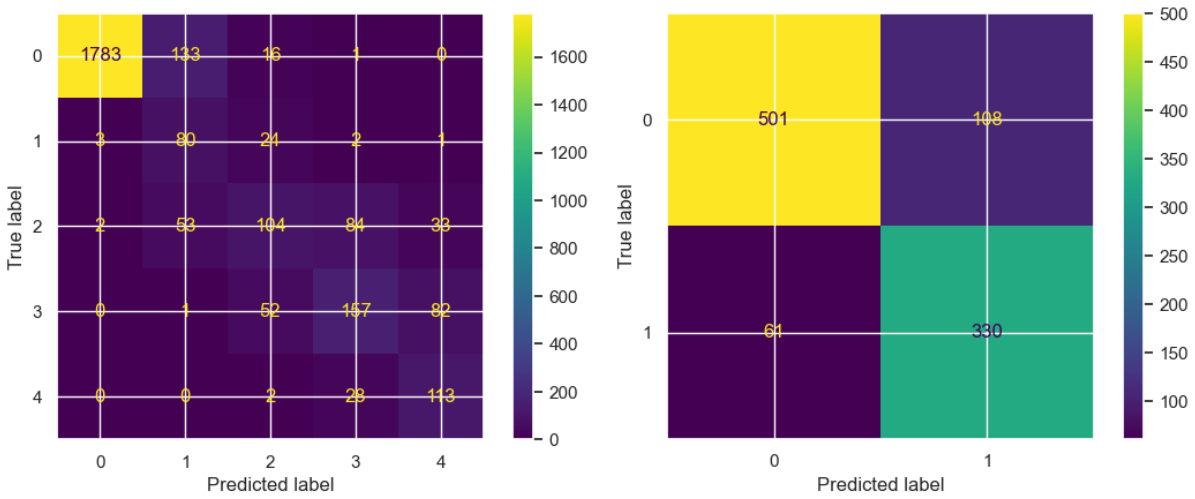

Figura 15: Resultados Rede Neuronal

Dataset	*Accuracy*	Precisão	*Recall*	*F1-score*
dataset competição	81%	85%	81%	82%
dataset grupo	83%	84%	83%	83%

Tabela 12: Métricas Rede Neuronal

4.6 *Voting*

O próximo modelo que abordamos foi a votação, ou *voting*. Nesta fase já tínhamos uma boa quantidade de modelos feitos e achamos por bem juntar vários e através de um sistema votação chegar a uma previsão final.

Os passos iniciais são bastante aos modelos anteriores, sendo que talvez a única diferença significativa é que precisamos dos modelos já construidos para a construção deste modelo.

Para o *dataset* da competição usamos como base os seguintes modelos, estando dentre de parênteses o peso de cada modelo na votação:

- *Random Forest Tree* (1.3).
- Árvores de decisão (0.4).
- *SVMs* (1).

Para o *dataset* grupo usamos os seguintes modelos e os seus respetivos pesos:

- *Random Forest Tree* (1.3).
- *Random Forest Tree* com *GridSearch* (1.5).
- Árvores de decisão (0.8).
- *SVMs* (0.8).
- Regressão logística (1).

A atribuição dos pesos em ambos os *datasets* foi feita com base no desempenho obtido de cada modelo.

Os resultados obtidos estão presentes na imagem 16, sendo que à esquerda temos a matriz de confusão do *dataset* competição e à direita a mesma mas para o *dataset* grupo.

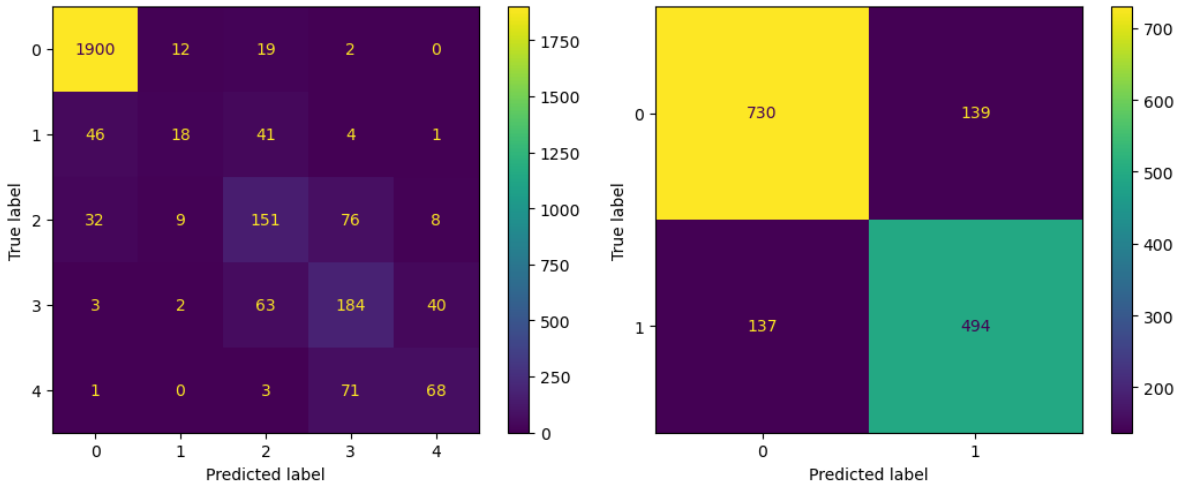

Figura 16: Resultados obtidos para ambos os *datasets Voting*

A tabelas 4.6 apresenta os resultados das diferentes métricas para ambos os *datasets*.

Dataset	*Accuracy*	Precisão	*Recall*	*F1-score*
dataset competição	84%	83%	84%	83%
dataset grupo	82%	82%	82%	82%

Tabela 13: Métricas *Voting*

4.7 *Bagging*

O *Bagging* foi uma das técnicas de *essemble* que decidimos aprofundar.

Inicialmente para o *Bagging* não acrescentamos nada de relevante em relação aos outros modelos.

Como estimador do modelo *bagging*, usamos as árvores de decisão já construidas, sendo que depois usamos o *GridSearch* para nos ajudar a encontrar os hiperparâmetros ótimos.

Para este modelo, optamos por fazer uma etapa adicional no *dataset* competição. Decidimos aplicar o *SMOTE* aos dados e ver se o comportamento do modelo, o que ao fim ao cabo, não houve grande diferença, pois o *Bagging* já estava a generalizar direito.

A figura 17 contém os resultados obtidos para ambos os *datasets*. As duas primeiras figuras são relativas ao *dataset* competição, sendo que a primeira foi sem *SMOTE* e a segunda com *SMOTE*. A última é em relação *dataset* grupo.

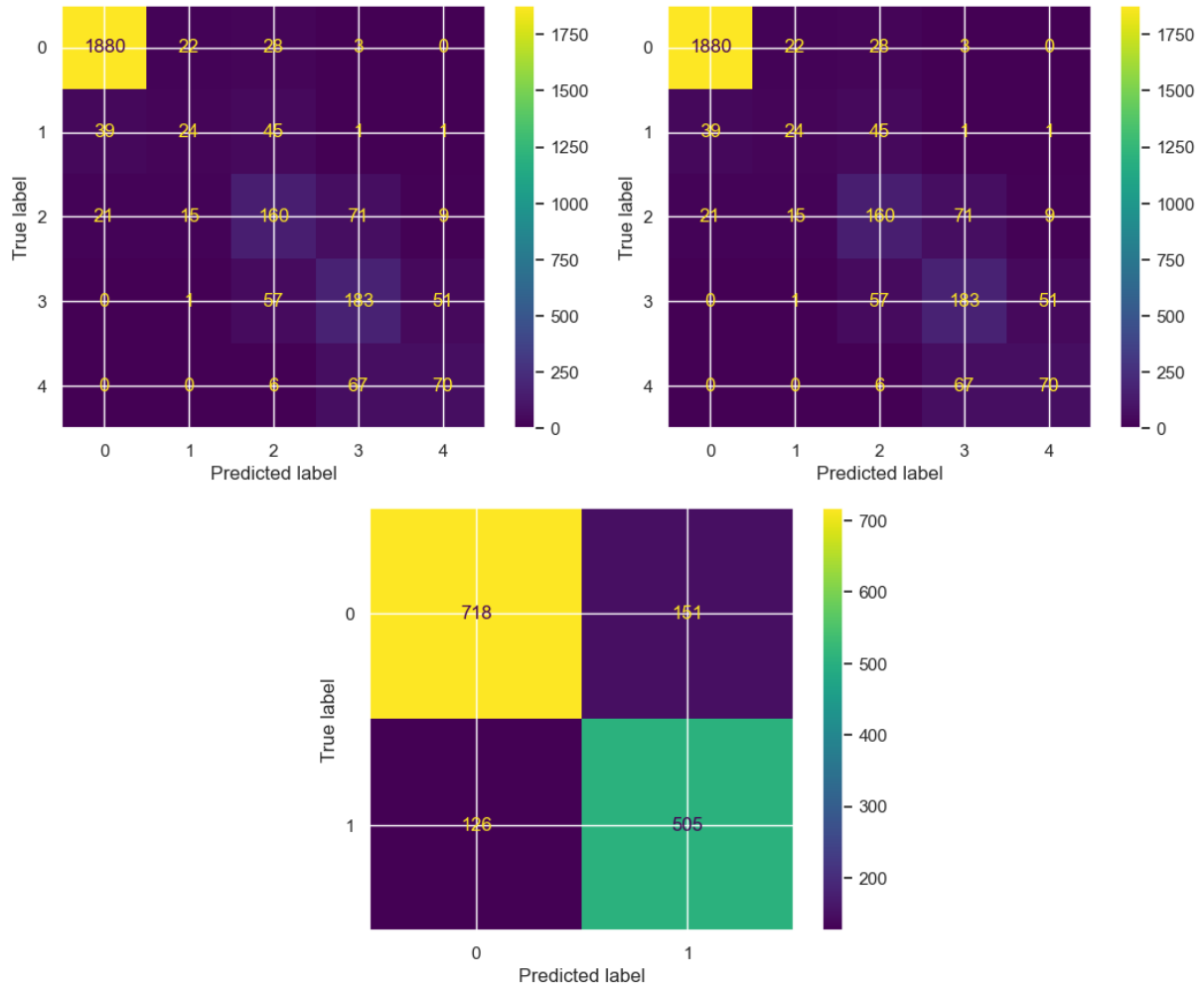

Figura 17: Resultados obtidos para ambos os *datasets Bagging*

Os resultados das métricas obtidos para ambos os *datasets* para este modelo estão presentes na tabela 4.7

Dataset	*Accuracy*	Precisão	*Recall*	*F1-score*
dataset competição s/ *SMOTE*	83%	85%	83%	83%
dataset competição c/ *SMOTE*	83%	85%	83%	84%
dataset grupo	82%	82%	82%	82%

Tabela 14: Métricas *Bagging*

4.8 *XG-Boost*

Para terminar, decidimos fazer mais um modelo de *essemble*, neste caso o *XG-Boost*.

De forma a testar vários hiperparâmetros, usamos em ambos os *datasets* o *GridSearch*. Mais uma vez, não houve nenhuma etapa adicional das já mencionadas para os outros modelos, ou seja, partimos os dados em dados de treino e dados de teste, observamos graficamente a divisão por classe, construimos o modelo, passamos ao *GridSearch* e analisamos os resultados.

O XGBoost acabou por ser o nosso melhor modelo no que diz respeito à competição em si, alcançando a melhor *accuracy* dentro dos modelos que criamos. Assim, numa tentativa de aumentar ainda mais a performance do modelo acabamos por implementar num novo *notebook SMOTE*. Este modelo correu durante mais de 1000 minutos (quase 17 horas).

A figura 18 ilustra as matrizes de confusão resultantes de ambos os *datasets*, sendo a figura da esquerda relativa ao *dataset* competição, a da direita relativa ao *dataset* grupo e a de baixo referente ao modelo que usa *SMOTE*.

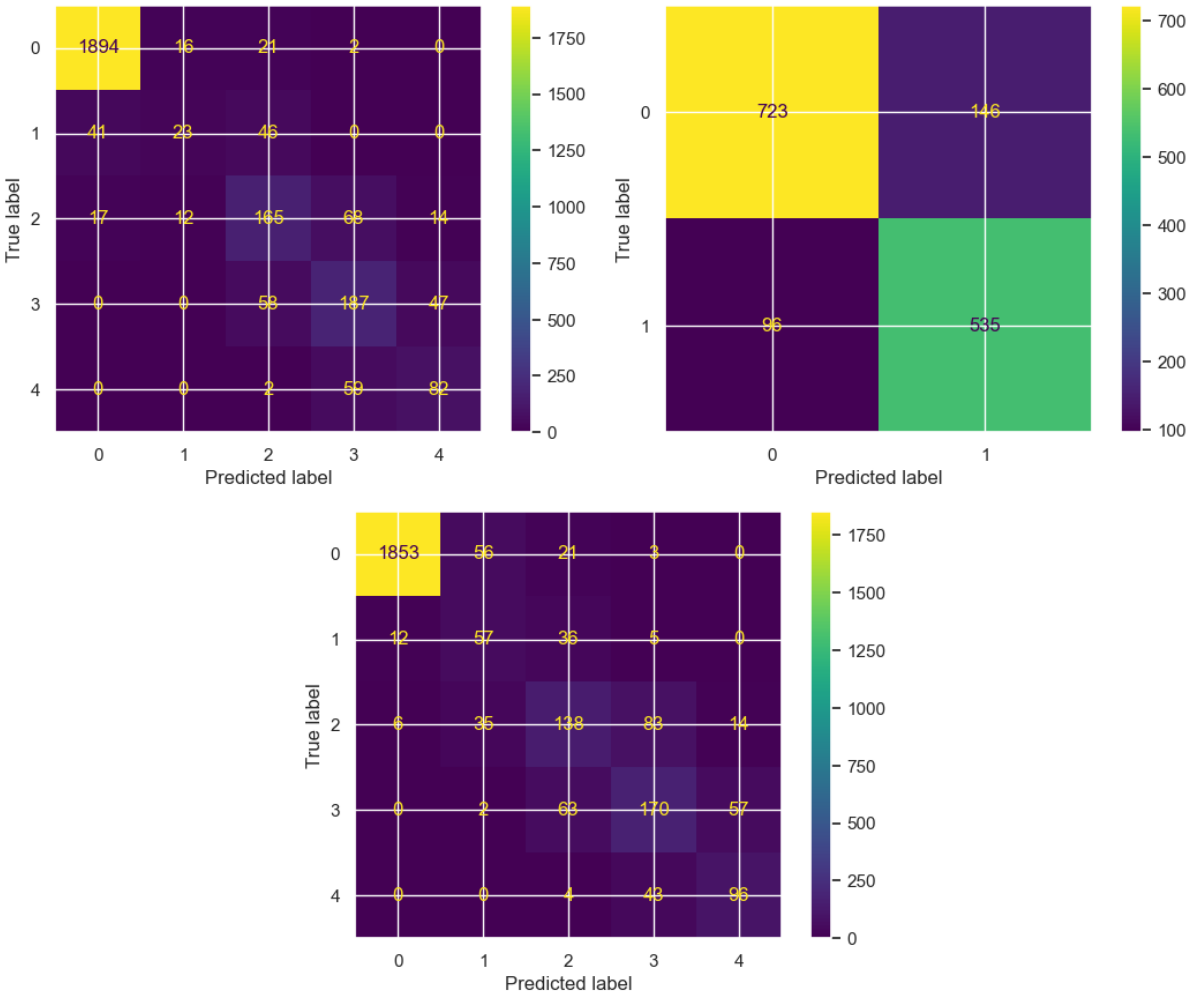

Figura 18: Resultados obtidos para ambos os *datasets XG-Boost*

A tabela 4.8 demonstra os resultados obtidos para as métricas em ambos os *datasets*.

Dataset	*Accuracy*	Precisão	*Recall*	*F1-score*
dataset competição s/ *SMOTE*	85%	85%	85%	85%
dataset competição c/ *SMOTE*	84%	85%	84%	85%
dataset grupo	84%	84%	84%	84%

Tabela 15: Métricas via *XG-Boost*

5 Análise de resultados

Relativamente a análise, desenvolvemos dois gráficos que comparam o desempenho de todos os modelos construidos para ambos os *datasets*. Esse desempenho é feito com base na *accuracy* de cada um dos modelos. Esses gráficos estão demonstrados na figura 19. O gráfico da esquerda é relativo ao *dataset* competição e o da direita ao *dataset* grupo.

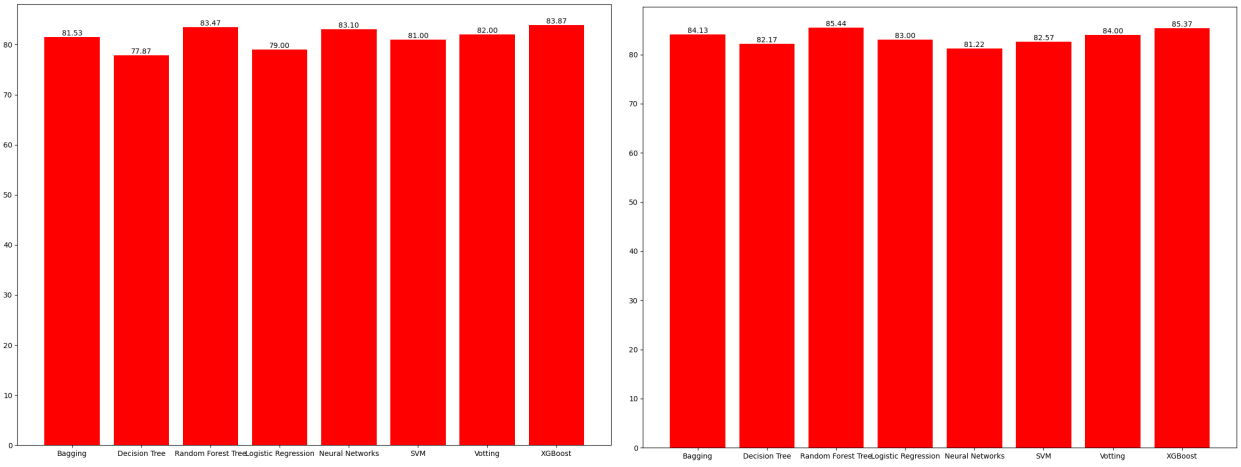

Figura 19: Desempenho dos modelos para ambos os *datasets*

A partir dos gráficos mencionados, podemos concluir os modelos *Random Forest Tree* e o *XG-Boost* foram os nossos melhores modelos.

Relativamente à *accuracy* na competição, a nossa melhor prestação foi aproximadamente 86% e foi através do modelo *XG-Boost* com o uso de *Grid-Search*. No entanto, podemos salientar outros modelos que tiveram uma boa classificação, sendo eles:

- *Bagging*: 84%.
- *Voting*: 85%.
- *Random Forest Tree*: 84%.

O uso da técnica de *SMOTE* só nos contribuiu para um melhor desempenho nas redes neuronais, pois tal como já foi apontado, estava a ficar *overfitted* e só previa valores nulos para a *label*.

Concluindo, achamos que os resultados que obtivemos são os supostos, pois os modelos com melhor classificação, são modelos que tem uma melhor capacidade de generalizar, dando destaque aos modelos de *essemble*. No entanto, achamos que as redes neuronais ficaram um pouco aquém do expectável. Mas tal como já foi referido, a razão para isto acontecer é do facto de que os dados não estão balanceados, o que contribuiu para que a rede neuronal desse *overfitted*. Outro ponto a destacar é que o modelo regressão logística teve um melhor desempenho no *dataset* grupo comparado ao *dataset* competição. Isto deve-se ao facto que a *label* no primeiro caso é a previsão de valores binários e a segunda não.

6 Conclusão

Achamos que no geral fizemos um trabalho bom em ambos os *datasets*. Não tivemos grandes problemas em desenvolver os modelos, exceto a Rede Neuronal no *dataset* competição. Esta facilidade adveio-se a uma boa análise e tratamento dos dados. Os problemas da Rede Neuronal no *dataset* competição, já foram apontados na secção 4.5. Todos os elementos do grupo trabalharam de igual forma, sendo que as partes inicias, isto é, análise do problema, análise e exploração dos dados foi feita em conjunto e depois na parte dos modelos, cada elemento tratou de 2 de modelos diferentes para ambos os *datasets*.

No entanto, embora achemos que o nosso desempenho na competição tenha sido bom (84.9%, posição 42^{o}), comparativamente com outros grupos, ficamos mal classificados. Achamos que talvez a melhor forma de conseguir um melhor resultado seria arranjar outros dados que nos ajudassem. Achamos que os dados originais, embora sejam importantes, não sejam completos e achamos que há certos fatores que poderiam ser adicionados, tal como os dados meteorológicos em falta, atributos como radiação, a inclinação dos raios solares, etc.

7 Anexos

7.1 *SMOTE*

Nos nossos modelos, tivemos situações em que optamos por criar dados artificiais. Estes dados servem para balancear os *datasets* pois são criados de forma a garantir uma incidência igual para todos os casos possíveis. No entanto, há casos em que a tendência que surge no *dataset* é importante pois poderá ser relevante para tirar conclusões sobre os dados. Um dos casos em que usamos foi nas redes neuronais artificiais pois previam corretamente apenas um dos possíveis valores. Para utilizar *SMOTE*, basta instalar a biblioteca *Imbalanced-learn* com o comando:

$$pip install - U imbalanced - learn$$

Esta biblioteca conta com uma implementação de *SMOTE* que é facilmente instanciada no código.