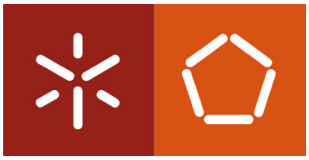
Universidade do Minho

Escola de Engenharia



Dados e Aprendizagem Automática

Mestrado em Engenharia Informática

Grupo 16

A93177 Cláudia Silva

PG50352 Eduardo Magalhães

PG50542 Laura Rodrigues

PG50628 Mariana Rodrigues

Janeiro 2023

Índice

1 Introdução

No âmbito da Unidade Curricular de Dados e Aprendizagem Automática foi-nos proposto pelos docentes a análise de dois *datasets*. O primeiro seria um escolhido pelo grupo e o segundo, apresentado pelos docentes, contém dados referentes à quantidade e características dos incidentes rodoviários que ocorreram numa cidade portuguesa em 2021.

No caso do dataset de grupo foi escolhido um sobre a esperança de vida em diferentes países num estudo realizado pela WHO e as Nações Unidas. Os dados apresentados foram obtidos a partir da plataforma Kaggle.

Com este relatório pretende-se fazer uma análise detalhada de todo o processo de análise, modelação e exploração dos datasets mencionados anteriormente.

Para ambos os datasets foi usada a metodologia CRISP-DM. Nesta abordagem realiza-se mineração de dados seguindo as seguintes etapas:

1. **Business understanding** – Avaliação do contexto do problema
2. **Data understanding** – Avaliação dos dados e do seu significado e contexto
3. **Data preparation** – Preparação dos dados para a modelação
4. **Modelling** – Desenvolvimento de modelos de Machine Learning
5. **Evaluation** – Análise dos resultados obtidos
6. **Deployment** – Acesso aos resultados

2 Tarefa Dataset Grupo

Dataset usado : Life Expectancy (WHO)

2.1 Business Understanding

A primeira tarefa deste trabalho prático consistia em escolher um dataset do agrado do grupo e proceder à sua análise, tratamento, treino e avaliação. Assim, tal como referido anteriormente, optámos pelo dataset *Life Expectancy (WHO)* disponível na plataforma kaggle. Este dataset agrega e apresenta dados de diversos países referentes à esperança média de vida das suas populações, às caracteristicas das diferentes faixas etárias destes e o impacto de algumas fatores nas populações.

O objetivo seria prever a esperança média de vida com base nos vários parâmetros apresentados.

2.2 Data Understanding

Numa primeira fase, começámos por realizar uma análise detalhada do contexto do problema de modo a melhor compreender a análise e o tratamento de dados que seriam necessários realizar.

O *Life Expectancy (WHO)* é um dataset que apresenta dados numéricos discretos, tipo inteiro e contínuos, float, e dados categóricos, como é o caso dos atributos Country e Status (objetos).

O dataset apresenta 22 atributos e contém 2938 instâncias referentes a 193 países diferentes. Os atributos incluem:

De modo a compreender melhor os mesmos fazemos uma pequena descrição deles :

```
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Country                          2938 non-null   object
 1   Year                             2938 non-null   int64
 2   Status                           2938 non-null   object
 3   Lifeexpectancy                   2928 non-null   float64
 4   AdultMortality                   2928 non-null   float64
 5   infantdeaths                     2938 non-null   int64
 6   Alcohol                          2744 non-null   float64
 7   percentageexpenditure            2938 non-null   float64
 8   HepatitisB                       2385 non-null   float64
 9   Measles                          2938 non-null   int64
 10  BMI                              2904 non-null   float64
 11  under-fivedeaths                 2938 non-null   int64
 12  Polio                            2919 non-null   float64
 13  Totalexpenditure                 2712 non-null   float64
 14  Diphtheria                       2919 non-null   float64
 15  HIV/AIDS                         2938 non-null   float64
 16  GDP                              2490 non-null   float64
 17  Population                       2286 non-null   float64
 18  thinness1-19years                2904 non-null   float64
 19  thinness5-9years                 2904 non-null   float64
 20  Incomecompositionofresources     2771 non-null   float64
 21  Schooling                        2775 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
```

Figure 1: Atributos do dataset de grupo

- *Country*: Nome do país cuja saúde está a ser avaliada;
- *Year*: Ano em que foi realizado o estudo;
- *Status*: Se o pais é desenvolvido ou está em desenvolvimento;
- *Lifeexpectancy*: Esperança média de vida em anos;
- *AdultMortality*: Mortalidade em adultos (probabilidade de morrer entre os 15 e 60 anos por 1000 pessoas);
- *infantdeaths*: número de mortes de crianças por 1000 pessoas;
- *Alcohol*: consumo de alcool per capita(15+), em litros de puro álcool
- *percentageexpenditure*: despesa na saúde como percentagem de produto interno bruto per capita(%);
- *HepatitisB*: Imunização contra a hepatite-B em crianças de um ano(%);
- *Measles*: número de casos reportados de sarampo;
- *BMI*: média do índice de massa corporal da população;
- *under-fivedeaths*: Mortalidade em crianças com menos de cinco anos por 1000 pessoas;
- *Polio*: Imunização contra o Polio(Pol3) entre crianças de 1 ano(%).
- *Totalexpendituree*: Gastos do governo com saúde como percentagem do gasto total do governo (%);
- *Diphtheria*: Imunização da populaçao em relação ao tétano, difteria e tosse convulsa (DT3) em crianças com um ano de idade (%);
- *HIV/AIDS*: nados-mortos por 1000 vivos com HIV/AIDS (0-4 anos);
- *GDP*: Produto interno bruto per capita(em USD);

- *Population*: População do país;
- *thinness119years*: prevalência de magreza entre crianças e adolescentes com 1 a 19 anos (%);
- *thinness59years*: prevalência de magreza entre crianças com 5 a 9 anos (%);
- *Incomecompositionofresources*: Índice de desenvolvimento humano em termos do valor dos recursos (index varia entre 0 e 1) ;
- *Schooling*: número de anos de escolaridade;

Como o nosso objetivo tendo em conta o dataset escolhido é prever a esperança média de vida(em anos) nos diferentes países com base nos vários atributos registados no dataset, foi escolhida para target a coluna *Lifeexpectancy* e aplicamos algoritmos de regressão para prever a mesma.

Após melhor compreender o dataset com que iríamos trabalhar procedemos à análise dos dados de modo a compreender que tipo de preprocessing seria necessário e interessante realizar.

A **distribuição dos dados** foi analisada recorrendo a histogramas. De entre os diferentes apresentados podemos destacar que houve valores mais baixos por *Measles* e poucas mortes por *HIV/AIDS* e de menores de 5 anos (under-fivedeaths) ou crianças (infantdeaths). A esperança média de vida geral ronda os 75 anos de idade.

O dataset apresenta vários **missing values**, contudo, os atributos que contêm mais dados em falta são *population*, *hepatitisB* e *GDP*, como podemos ver na imagem seguinte.

Country	0
Year	0
Status	0
Lifeexpectancy	10
AdultMortality	10
infantdeaths	0
Alcohol	194
percentageexpenditure	0
HepatitisB	553
Measles	0
BMI	34
under-fivedeaths	0
Polio	19
Totalexpenditure	226
Diphtheria	19
HIV/AIDS	0
GDP	448
Population	652
thinness1-19years	34
thinness5-9years	34
Incomecompositionofresources	167
Schooling	163

Figure 2: Missing values do dataset de grupo

Graças a uma matriz de correlação podemos observar que:

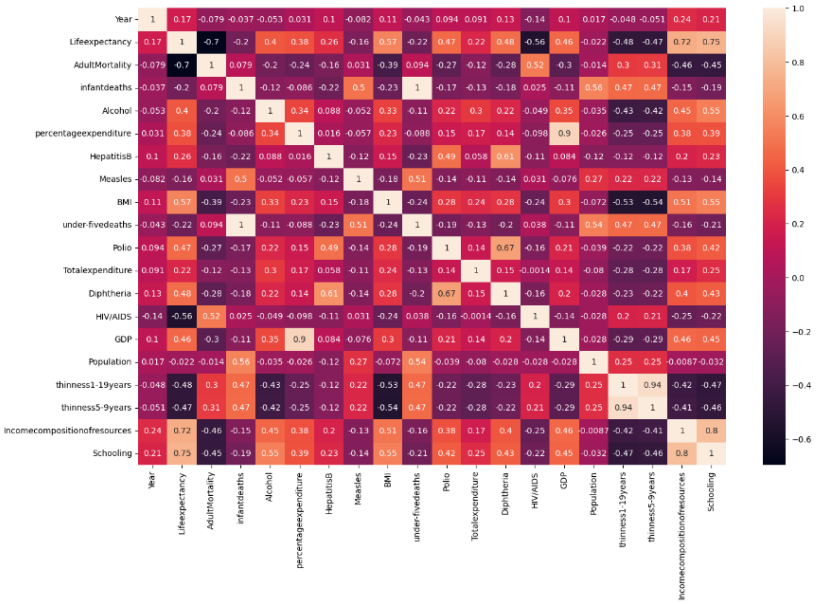


Figure 3: Correlação entre os atributos do dataset de grupo

· *under-fivedeaths* e *infantdeaths* estão completamente correlacionados. Podemos assim supor que ao retirar um destes parâmetros os resultados não serão afetados.
· *percentageexpenditure* e *GDP* estão fortemente correlacionados.
· *thinness1-19years* e *thinness5-9years* estão fortemente correlacionados.

Podemos também verificar que, de acordo com a matriz, os atributos que apresentam uma maior correlação com a coluna *Lifeexpectancy* são as colunas *AdultMortality*, *Incomecompositionofresources* e *Schooling*.

Adicionalmente, foi feita uma análise da relação entre os diferentes atributos e o atributo *target*. Através da análise desses gráficos, podemos verificar a existência de uma relação linear entre *Lifeexpectancy* e *Schooling* e *Incomecompositionofresources* e ainda uma relação inversa entre o atributo a prever e *AdultMortality*, o que apresenta uma justificação lógica, dado que, para uma população, quanto maior for o valor da mortalidade dos adultos, mais baixa será a esperança média de vida.

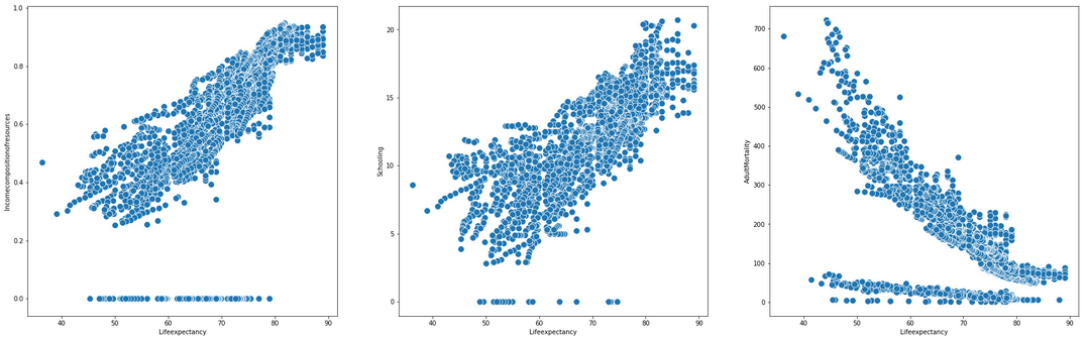


Figure 4: Correlação entre os atributos do dataset de grupo

Estes gráficos fundamentam as conclusões retiradas na análise da matriz de correlação.
Por outro lado, podemos observar que ao longo dos anos durante os quais foram recolhidos dados

para este dataset, a esperança média de vida aumentou de forma geral, com exceção de um dado outlier em 2010. Com uma análise do dataset verificamos que este valor estava relacionado com o sismo que ocorreu nesse ano no Haiti, o que realça que neste dataset não foram ignorados dados relacionados com desastre naturais ou situações atípicas.

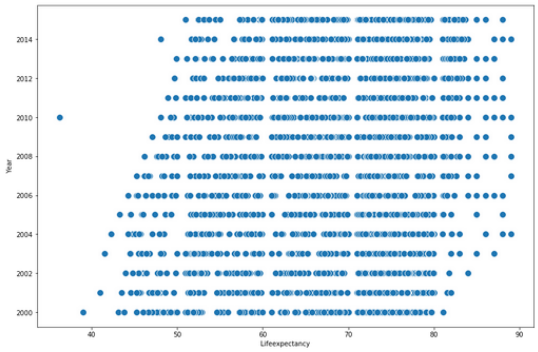


Figure 5: Correlação entre os atributos do dataset de grupo

Foi feita ainda uma análise dos outliers. Utilizando um diagrama de boxplot, podemos visualizar a distribuição dos dados de cada atributos. Neste gráfico, podemos também verificar a existência de outliers em diferentes atributos; posteriormente, serão aplicadas técnicas de tratamento desses outliers.

2.3 Data preparation

Missing Values

Houve várias abordagens que foram analisadas de forma a tratar da melhor forma os valores em falta: dar drop da linha, substituir o valor em falta pela média ou substituir o valor em falta de uma dada coluna pela média dos valores dessa coluna para o país correspondente. Esta última opção mencionada permitirá obter valores mais realista, isto é, dado que os valores variam bastante de país para país, utilizar a média do país permite obter valores mais "realistas". Contudo, em alguns casos, existem país que tem missing values em todos os dados de uma dada coluna. Portanto, houve necessidade de combinar este com outras estratégias.

Foram feitos vários testes de modo a descobrir qual seria a melhor abordagem a usar dependendo da situação de cada missing value. Estes resultados serão apresentados mais à frente.

Dados categóricos

De modo a poder tratar dos dados não numéricos e assim analisá-los houve a necessidade de os tratar. Para tal foram realizadas várias tentativas de modo a descobrir qual seria a melhor abordagem.

Foi testado:

- Dar drop dos atributos não categóricos (*country* e *status*)
- Substituir os valores com label encoding por valores numéricos
- Substituir os valores com one hot encoding, ou seja, separar os valores por colunas novas e substituir por valores binários (0's e 1's)

Feature Selection

De forma a poder analisar a melhor forma de fazer feature selection demos drop de 3 atributos com base na análise feita anteriormente: *Country* e *Year* que demonstraram não ser relevantes para o problema em causa (não tinham muita correlação com outros atributos) e *Population* por, para além de ter pouca correlação com os outros atributos, ter bastantes missing values.

Para além disso, foram definidas várias funções que nos permitissem dar drop de vários atributos menos relevantes, para uma eventual análise mais aprofundada.

Normalização de dados

Como forma de pré-processamento de dados definimos uma função que nos permitisse aplicar min-max feature scalling aos atributos que desejássemos. Para tal recorremos à biblioteca de preprocessing do sklearn e usámos o MinMaxScaler já definido nesta. Foi usado o intervalo default, ou seja, 0 e 1.

A normalização poderá ser relevante para algoritmos que dependam da *distância* entre os pontos, e se as colunas tiverem intervalos de distribuição de grandezas variadas, os *weights* de cada atributo podem não representar o impacto que esta terá no atributo target.

Tratamento de Outliers

Uma vez que foi notada a presença de outliers, tornou-se necessário analisar melhor o seu impacto no problema e perceber como mudariam os resultados caso estes fossem tratados. No entanto, após realizar vários testes percebemos que o *mean square error* dos modelos de previsão aumentou e portanto optámos por não alterar os outliers para a solução final.

2.4 Modelling and Evaluation

Modelos testados

Para este dataset usámos os algoritmos Ridge, Linear Regression, Random Forest Regression, Epsilon-Support Vector Regression (SVR), Stochastic Gradient Descent (SGD) e Lasso Regression. Para cada um destes analisámos as métricas *R2 Score, mean square error, mean absolute error e root mean square error*. Estes modelos foram testados com os parâmetros *default*.

Para obter estes resultados, foi aplicado o seguinte preprocessing ao dataset:

```
df = replace_mean_missing_values(df);  df = encoding_status(df);
df.drop(['Country'], axis=1, inplace=True); df.drop(['Year'], axis=1, inplace=True);
df.drop(['Population'], axis=1, inplace=True);
```

Model	R2	MSE	MAE	RMSE
Linear regression	1.00	2.540e-28	1.360e-14	1.594e-14
Ridge	1.000	1.324e-08	8.608e-05	0.0001
SVR	0.999	0.072	0.145	0.268
SGD Regressor	-127.19e20	1.113e+24	104.56e10	105.48e10
Lasso	1.000	0.010	0.083	0.102
Random Forest regression	0.992	0.657	0.497	0.811

Como podemos ver, obtivemos alguns modelos que nos deram resultados relativamente bons. Podemos também observar que, com os valores default, o algoritmo *SGD Regressor* não convergiu. Isto deve-se ao facto de o valor default para o hiper parâmetro *alpha* ser demasiado pequeno(0.0001) e o valor default do hiper parâmetro *max_iter* ser apenas 1000.

Como passo seguinte tentámos melhorar um pouco mais a eficiência destes algoritmos, otimizando os hiperparâmetros dos mesmos. Para tal, optámos por utilizar Grid Search disponibilizado pelo scikit-learn. Alguns destes algoritmos mostraram algumas melhorias mais significativas.

Model	**MSE**
Linear Regression	8.355e-29
Ridge	7.575e-10
SVR	0.002
SGD Regressor	0.001
Lasso	1.108e-8
Random Forest Regression	0.688

Salientamos ainda que relativamente ao algoritmo Lasso quanto menor o *alpha* mais semelhante é do linear regression deixando de fazer sentido o uso deste e usando-se o linear regression. Como se verificou uma melhoria no resultado consideramos que é mais vantajoso o uso de Linear Regression em vez de Lasso.

Em conclusão, dos algoritmos analisados, verificamos que o que nos deu os melhores resultados foi o Linear Regression, dado que foi o que obteve um menor *MSE*.

Analisando o peso de cada atributo, verificamos que as colunas que apresentam um maior peso são:

- under-fivedeaths: 3.41992070e-13
- AdultMortality: 2.10779577e-14
- HIV/AIDS: 1.96429638e-14

Deep Learning

Tentámos também implementar um modelo que utilizava Neural Networks. Para tal utilizámos a biblioteca TensorFlow e definimos uma rede neuronal com 3 layers (Multi Layer Perceptron) cuja função de ativação para todas as layers usadas foi a relu e com um learning rate de 0.01.

Na layer inicial, usamos 20 neurónios e a dimensão de *input=19* uma vez que o dataset depois de *tratado* possui 19 atributos. A layer intermédia foi definida com 10 neurónios e na layer de output definimos um neurónio, uma vez que pretendíamos prever apenas um valor. De modo a fazer tuning deste modelo, implementámos um grid search em que utilizamos diferentes funções de ativação e valores distintos para o learning rate. Usámos o método KFold disponibilizado pelo sklearn em conjunto com o KerasRegressor de modo a, com Grid Search, descobrir qual seria o melhor valor.

Para o processo de obtenção do modelo de Neural Networks, utilizámos como input o data set normalizado, *i.e.* após a aplicação do algoritmo MinMaxScaler. Posteriormente, para análise de resultados, foi necessário utilizar o scaler obtido do algoritmo referido anteriormente para desnormalizar

o atributo target. Se este atributo não tivesse sido desnormalizado, não seria possível estabelecer uma comparação com os restantes modelos.

Os resultados obtidos foram:

- **R2 Score:** 0.902
- **mean square error:** 8.725617052487936
- **mean absolute error:** 2.118575978415306
- **root mean square error:** 2.9539155459301702

No entanto, o modelo que apresentava melhores resultados não foi a Rede Neuronal, mas sim, a Linear Regression apresentada anteriormente.

2.5 Deployment

A última parte desta metodologia será o *Deployment*. Contudo, no trabalho que desenvolvemos, esta etapa não foi considerada. Dado que o propósito deste projeto era analisar o dataset e que resultados poderíamos obter de diferentes algoritmos e não utilizar os modelos obtidos para produção, esta etapa não foi concluída.

3 Tarefa Dataset Competição

3.1 Business Understanding

A segunda tarefa proposta para este trabalho prático consistia numa competição. A competicão proposta tinha como objetivo a previsão do número de acidentes que acontecerão em Guimarães a uma determinada hora. Foi-nos assim fornecido um dataset que continha dados recolhidos sobre os acidentes que ocorreram na cidade de Guimarães no decorrer do ano de 2021. Este dataset contém registos como *record_date*, *avg_rain*, etc... Como produto final, pretendemos obter um modelo de Machine Learning que consiga prever corretamente a gravidade de um acidente tendo em conta os atributos presentes no dataset fornecido.

3.2 Data Understanding

Numa fase inicial, começámos por analisar o dataset proposto.

O dataset tem 13 atributos e 5000 instâncias como podemos visualizar na imagem seguinte.

```
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   city_name           5000 non-null   object
 1   magnitude_of_delay  5000 non-null   object
 2   delay_in_seconds    5000 non-null   int64
 3   affected_roads      4915 non-null   object
 4   record_date         5000 non-null   object
 5   luminosity          5000 non-null   object
 6   avg_temperature     5000 non-null   float64
 7   avg_atm_pressure    5000 non-null   float64
 8   avg_humidity        5000 non-null   float64
 9   avg_wind_speed      5000 non-null   float64
 10  avg_precipitation   5000 non-null   float64
 11  avg_rain            5000 non-null   object
 12  incidents           5000 non-null   object
dtypes: float64(5), int64(1), object(7)
```

Figure 6: Atributos do dataset

De modo a compreender melhor o dataset, enumeramos os mesmos e complementando-os com uma pequena descrição.

- *city_name*: Nome da cidade onde ocorreu o acidente;
- *magnitude_of_delay*: Magnitude do atraso provocado pelos incidentes;
- *delay_in_seconds*: Atraso provocado pelos incidentes em segundos;
- *affected_roads*: Estradas afetas pelo incidente em questão;
- *record_date*: Data em que se registou o acidente;
- *luminosity*: Nível de luminosidade registado
- *avg_temperature*: Valor médio da temperatura;
- *avg_atm_pressure*: Valor médio da pressão atmosférica;
- *avg_humidity*: Valor médio da humidade;
- *avg_wind_speed*: Valor médio da velocidade do vento;

- *avg_precipitation*: Valor médio da precipitação;
- *avg_rain*: Avaliação qualitativa do nível de precipitação;
- *incidents*: Gravidade do acidente em questão.

O dataset em questão, como podemos ver na imagem anterior, apresenta uma coluna com dados numéricos discretos *delay_in_seconds*, numéricos contínuos, como *avg_temperature*, e categóricos como *record_date*, *affected_roads* e *magnitude_of_delay*.

O target deste dataset é o atributo *incidents*. Dado que este atributo tem valores categóricos iremos utilizar algoritmos de classificação. Com este dataset pretendemos determinar a gravidade de acidentes que possam ocorrer dependendo das caracteristicas de um dado dia.

Após conhecermos um pouco dos atributos deste dataset, podemos dar início a uma pequena análise do mesmo.

De facto, o atributo *affected_roads* deste dataset possui alguns missing values. No entanto, este dataset não apresenta linhas duplicadas.

```
** MISSING VALUES **
city_name              0
magnitude_of_delay     0
delay_in_seconds       0
affected_roads        85
record_date            0
luminosity             0
avg_temperature        0
avg_atm_pressure       0
avg_humidity           0
avg_wind_speed         0
avg_precipitation      0
avg_rain               0
incidents              0
```

Figure 7: Missing values do dataset

Como já mencionámos, este dataset possui alguns atributos cujos dados são categóricos. Dado que os modelos do sklearn só trabalham com dados numéricos, posteriormente será necessário o *encoding* dos dados categóricos.

De forma a verificar o balanceamento de dados, optamos por apresentar um histograma que mostrava o número de instâncias para cada classe.

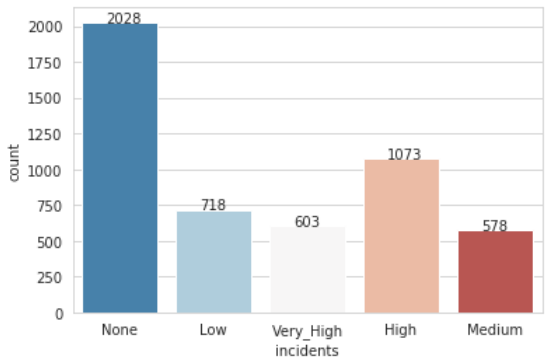


Figure 8: Instâncias por classe

Assim, podemos verificar que a classe *None* possui bastantes mais instâncias que as restantes classes. Posteriormente, poderá ser relevante colocar todas as classes com o mesmo número de instâncias.

Ademais, serão feitas novas visualizações dos dados após ter sido feito algum tratamento dos dados.

3.3 Data Preparation

Após analisar o dataset foi necessário entender qual seria a melhor abordagem e as melhores decisões a nível de preprocessing.

Dados categóricos

Como vimos anteriormente, este dataset possui dados categóricos. Para os podermos utilizar com os algoritmos do sklearn, foi necessário fazer um encoding dos mesmos. Para os atributos *city_name*, *magnitude_of_delay*, *luminosity*, *avg_rain* e *incidents* foi utilizado *label encoding*.

Para tratar os dados da *affected_roads* e *record_date* foram aplicadas estratégias diferentes.

Os valores do atributo *record_date* são as datas em que os acidentes ocorreram. Fazer encoding destes dados poderia não se revelar muito interessante dado seria muito provável que esta coluna apresentasse valores muito distintos e que poderiam ter pouca utilidade para prever casos futuros, dado que são muito especifico.

Assim, para tratar esta coluna, optamos por fazer Feature Engineering. Analisando novamente os valores da *record_date*, optamos por criar novas colunas para dados retirados desta coluna *record_date*. Estas novas colunas conterão as horas, dias, meses, ano. Assim, teremos 5 novas colunas.

Para tratar a coluna *affected_roads*, testamos diferentes estratégias. Esta coluna continha uma *string*, composta por diferentes códigos de estradas, como por exemplo, "N101,N101,N101,N101,N101". Para tratar esta coluna foram implementadas quatro opções diferentes. Contudo, apenas uma delas será seguida. A primeira opção seria remover a coluna. Contudo, com as diferentes versões que se seguiram verificámos que esta coluna teria algum interesse. A segunda opção permite identificar as diferentes estradas existentes numa dada instância. Uma terceira opção permite identificar o total de estradas mencionadas numa dada instância. Quando aplicamos os dois métodos anteriormente mencionados, o exemplo anteriormente referido resultaria nos valores: *1* e *5*, respetivamente. Implementámos ainda um versão que determinava todas as diferentes estradas que existiam no dataset, adicionavam uma coluna para cada uma dessas estradas, e para cada instância do dataset indicavam o número de vezes que cada estrada aparecia. Uma lógica semelhante à do *one-hot encoding*, contudo registávamos também a frequência com que aparecia cada estrada na feature *affected_roads*.

Missing values

Como foi visto anteriormente, o atributo *affected_roads* apresenta missing values. Assim, antes do encoding dos dados desta coluna, foi necessário tratar os missing values desta coluna.

Para o tratamento destes missing values duas soluções foram consideradas: realizar o drop dos missing values ou substitui-los. De facto, aquando a análise do dataset reparámos na existência de " " no ficheiro que eram considerados como missing values. Dessa maneira, os valores que eram considerados como missing values foram subsituido por " ". Assim, esta implementação não causa problemas na realização do encoding. Após testar as duas maneiras de tratamento de dados, averiguámos que a substituição dos missing values era a mais eficiente.

Feature Selection

Após ter sido feito o encoding e o tratamento de missing values, foi utilizado o método *describe*, para analisar alguns dados estatísticos e a distribuição dos mesmos no dataset.

	city_name	avg_precipitation	record_date_year
count	5000.0	5000.0	5000.0
mean	1.0	0.0	2021.0
std	0.0	0.0	0.0
min	1.0	0.0	2021.0
25%	1.0	0.0	2021.0
50%	1.0	0.0	2021.0
75%	1.0	0.0	2021.0
max	1.0	0.0	2021.0

Figure 9: Describe de *avg_precipitation*, *city_name* e *record_date_year*

Através da visualização do output deste método, foi possível verificar que atributos como *avg_precipitation*, *city_name*, *record_date_year* apresentava uma *standard deviation* igual a 0. Isto significa que todos os valores relativos a estas coluna são iguais, o que consequentemente introduz muita informação para o desenvolvimento de um modelo de Machine Learning. Assim , decidimos remover os atributos anteriormente mencionados.

Tratamento de Outliers

Outliers são dados que notoriamente diferem dos restantes e que, nalguns casos, podem contribuir negativamente para a modelação de algoritmos de Machine Learning. De modo a analisar de que maneira os outliers do nosso dataset se comportavam na modelação e se contribuíam negativamente ou positivamentem, decidimos implementar tratamento de outliers no processo de preparação de dados. Para identificarmos que atributos continham outliers, recorremos à utilização de boxplots dos quais realçamos os seguintes:

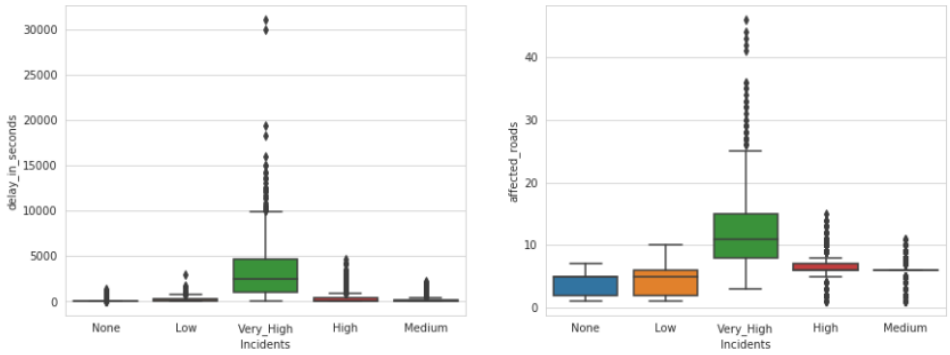


Figure 10: Boxplot

Através da análise do primeiro boxplot concluimos que *delay_in_seconds* apresenta uma distribuição diferente por classes e algumas classes apresentam bastantes outliers. Assim, esta coluna poderá ser utilizada para distinguir as diferentes classes. Esta apresenta tipicamente valores mais elevados para a classe *Very_High* e uma distribuição maior para esta classe. Relativamente ao segundo

boxplot, *affected_roads* apresenta valores superiores para a classe *Very_High* apresentando, também, uma distribuição bastante diversa entre as classes. Dado que apresenta distribuições distintas, podemos assumir que serão atributos pertinentes para a calibração dos algoritmos.

Após analisar a existência de outliers foi testada a remoção destes. Para tal, optámos por fazer uma subsituição dos outliers pelo valor do percentil mais próximo. No entanto, a accuracy dos modelos calibrados com este técnica de preprocessing diminui. Na nossa análise, concluímos que os outliers apresentam um interesse para os modelos calibrados, e que certos acontecimentos "atípicos" também apresentam um certo valor e contribuem positivamente para a previsão.

Balaceamento do atributo *incidents*

Como vimos anteriormente, as classes do atributo target não contêm o mesmo número de instâncias. Isto pode levar a que os algoritmos de Machine Learning aplicados aprendam a prever melhor as classes maioritárias, isto é, que contêm o maior número de registos. Para tal, foi realizado um balanceamento do número de instâncias. Implementámos duas técnicas de preparação de dados para realizar esta operação, ambas baseadas em oversampling.

A primeira técnica de preparação de dados utilizada foi o sampling dos registos classes minoritárias. Aumentámos assim o número de instâncias usando instâncias já existentes através do método *sample*. Esta operação tem que ser realizada com cuidado para evitar overfitting. Se aplicarmos esta operação a ambos train e test data, iremos fazer com que a accuracy do modelo não seja a verdadeira, dado que existe uma elevada probabilidade de o modelo prever um valor que foi usado para o treinar.

A segunda técnica utilizada foi a técnica de **Synthetic Minority Oversampling Technique (SMOTE)**. Esta técnica consiste na geração de novos dados a partir de casos minoritários existentes nos dados fornecidos. A diferença entre a técnica de **SMOTE** e a técnica anteriormente referida, é o facto de os dados adicionados não serem dados copiados e sim, dados que combinam características dos seus vizinhos mais próximos.

De seguida, poderemos ver o balanceamento dos dados de *Training* depois das operações anteriormente referidas.

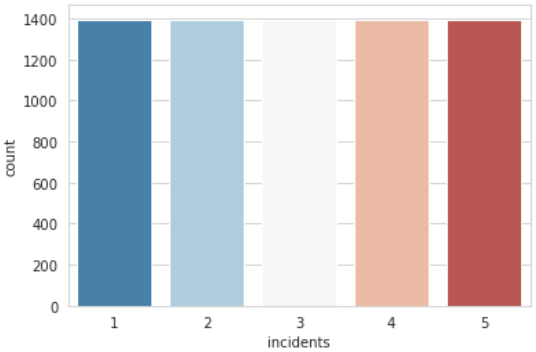


Figure 11: Instâncias por classe depois de balanceamento.

Feature Engineering

Feature Engineering é o processo de usar o conhecimento de domínio dos dados para transformar os recursos existentes ou criar novos atributos a partir dos existentes, com o intuito de enriquecer o modelo usado na aprendizagem.

Neste trabalho, tentámos utilizar Feature Engineering. Durante este trabalho, ponderámos diferentes maneiras em que poderíamos tratar a *record_date* e extrair conhecimento da mesma. Começámos por testar agrupar as horas em que ocorreram os acidentes em *early_morning*, *morning*, *afternoon*, *evening*, *night*. Adicionámos também a coluna *holiday_busy_day*, que assinala se um dado dia da semana é um feriado (sendo que os feriados teriam o valor 1 e, caso contrário, este atributo apresentaria o valor 0). De modo a que isto fosse possível utilizamos a biblioteca *holidays* do python.

Acrescentámos também informação relativa aos dias da semana em que os acidentes acontecem. Deste modo, foi acrecentado o atributo *record_week_day* que contém valores numéricos de 0 a 6, sendo o número 0 correspondente a segunda-feira e o 6 a domingo.

Acrescentámos também uma coluna que identifica se um dia seria um dia de fim de semana - atributo *weekend*. Para isso analisámos o atributo *record_week_day* e caso registasse um valor superior ou igual a 4 (sexta-feira), o atributo *weekend* registaria o valor 1.

Análise de Correlação

Depois de termos implementadas diferentes operações de processing, optámos por fazer uma análise da correlação. Optámos por deixar esta análise para o final, porque desta forma poderíamos analisar uma quantidade superior de dados.

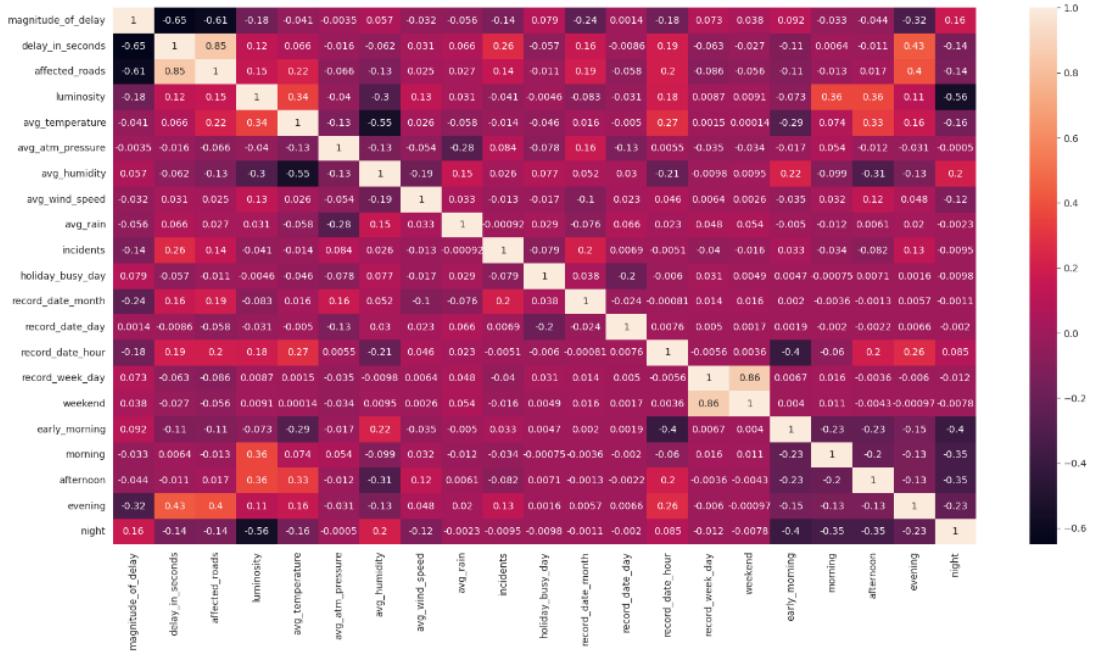


Figure 12: Matriz de correlação

Através da análise da matriz de correlação podemos verificar que a grande maioria dos atributos não apresenta uma correlação entre eles. No entanto, podemos verificar que os atributos *weekend* e *record_week_day* apresentam uma correlação mais próxima do 1. Podemos verificar que o mesmo se

verifica entre o atributo *affected_roads* e *delay_in_seconds*. Optámos por não fazer seleção de atributos para estas colunas, porque consideramos que estes valores não eram suficientemente próximos de 1.

Analisámos também a relação dos atributos com o *target attribute*. Verificámos que alguns atributos apresentavam uma correlação próxima de zero com o target. Alguns destes atributos foram posteriormente removidos numa feature selection realizada em função da importância destes atributos no modelo Random Forest.

3.4 Modelling and evaluation

Modelos testados

Para este dataset foram testados diferentes algoritmos. Estes incluem Decision Tree, Random Forest Classifier, AdaBoost Classifier, CatBoost Classifier, Gradient Boosting Classifier, Balanced Random Forest Classifier, Logistic Regression, SVC, Stochastic Gradient Descent Classifier (SGDC). Para cada um destes modelos foram recolhidas algumas métricas para determinar a eficácia dos mesmos. As métricas utilizadas foram a *confusion matrix*, *recall*, *precision* e *accuracy*.

Model	Accuracy	Precision	Recall
Decision Tree	0.91	0.89	0.88
Random Forest Classifier	0.92	0.90	0.90
AdaBoost Classifier	0.68	0.65	0.62
CatBoost Classifier	0.93	0.90	0.90
Gradient Boosting Classifier	0.91	0.87	0.88
Balanced Random Forest Classifier	0.92	0.89	0.90
Bagging	0.92	0.90	0.90
Logistic Regression	0.55	0.48	0.51
SVC	0.58	0.36	0.47
SGDC	0.23	0.44	0.34

Notas: Os parâmetros destes algoritmos possuíam os seus valores *default*. Como preprocessing foi usado:

```
encoding_categorical_data(df)
encoding_incidents(df)
drop_columns_zero_std(df)
binning_days(df)
replace_missing_affected_roads(df)
df = count_all_roads_per_line(df)
```

Os dados do train data foram balanceados. Não foram normalizados.

Como podemos ver, obtivemos alguns modelos que nos deram resultados relativamente bons. Como passo seguinte tentámos melhorar um pouco mais a eficiência destes algoritmos, otimizando os hiperparâmetros dos mesmos. Para tal, optámos por utilizar Grid Search disponibilizado pelo scikit-learn. Alguns destes algoritmos mostraram algumas melhorias mais significativas. Para alguns dos modelos, optámos também por alterar algumas funções de preprocessing, que foram previamente exploradas. Para os algoritmos SVC e SGDC, os dados do dataset foram *"standardized"*, dado que assim obtivemos melhores resultados.

Model	Accuracy	Precision	Recall
Decision Tree	0.91	0.89	0.88
Random Forest Classifier	0.93	0.91	0.91
AdaBoost Classifier	0.71	0.66	0.63
CatBoost Classifier	0.93	0.91	0.90
Gradient Boosting Classifier	0.93	0.90	0.90
Balanced Random Forest Classifier	0.92	0.90	0.90
Bagging	0.92	0.90	0.90
Logistic Regression	0.75	0.69	0.70
SVC	0.73	0.70	0.73
SGDC	0.66	0.59	0.58

Notas: Os parâmetros destes algoritmos foram escolhidos em função dos resultados do algoritmo do Grid Search. Como preprocessing foi usado:

```
encoding_categorical_data(df)
encoding_incidents(df)
drop_columns_zero_std(df)
binning_days(df)
replace_missing_affected_roads(df)
df = count_all_roads_per_line(df)
```

Os dados do train data foram balanceados.

Por último, optámos também por analisar o atributo *feature_importance* da Random Forest Classifier. Esta permitiu-nos conhecer os atributos que teriam uma maior importância para construir as Random Forest. Com esta informação optámos por reduzir o número de atributos que iríamos utilizar para calibrar a Random Forest. Assim, ao preprocessing da Random Forest foi adicionada a linha:

```
df = df.drop(['magnitude_of_delay', 'luminosity', 'avg_wind_speed', 'avg_rain',
'record_date_hour', 'weekend', 'early_morning', 'morning','afternoon', 'evening',
'night'],axis = 1)

model = RandomForestClassifier(criterion='entropy', max_depth=15,
max_features='log2', n_estimators=150, random_state=2001) #criterion = 'gini',
max_features = 'log2', max_depth=20, n_estimators = 175,random_state=2001)
```

Esta solução com os atributos selecionados foi aquela que foi selecionada como submissão final. Esta submissão deu-nos um bom resultado. Contudo, posteriormente, viemos a descobrir que existiam algumas submissões que nos deram resultados ligeiramente melhor.

Deep Learning

Tentámos também implementar um modelo que utilizava Neural Networks. Para tal utilizámos a biblioteca TensorFlow e definimos uma rede neuronal com 3 layers (Multi Layer Perceptron).

Para a layer de input definimos 11 neurónios/nodos e a função de ativação *relu*. A layer intermédia foi definida com 8 neurónios e com a mesma função de ativação que a layer de input. Na layer de output, definimos 5 neurónios, um para cada classe do atributo de target mas, ao contrário do que foi definido como função de ativação nas layers anteriores, usámos a função de ativação *softmax*.

De modo a fazer tuning deste modelo, implementámos um grid search em que utilizamos diferentes funções de ativação e valores distintos para o leaning rate do modelo.

No entanto, não conseguimos obter resultados razoáveis.

3.5 Deployment

A última parte da metodologia, após a *Evaluation*, seria o *Deployment*. Para este desafio, para podermos prever os valores da coluna *target*, foi necessário utilizar um modelo escolhido em função dos resultados dos testes para cada algoritmo vistos anteriormente.

Contudo, para este trabalho, não foi cumprida a fase de *Deployment*, uma vez que não foi construída um pipeline que permitisse que não foi desenvolvido nenhum pipeline que premitisse a produção desta solução.

4 Conclusão

Em modo de conclusão, consideramos que este trabalho nos permitiu consolidar a matéria lecionada, especialmente no que diz respeito ao tratamento de dados e desenvolvimento de modelos de Machine Learning. Realçamos ainda o uso de diferentes modelos de machine learning e técnicas de tratamento de dados.

Consideramos que fizemos uma análise pertinente e bem estruturada que seguia a metodologia que nos responsabilizamos a cumprir. Contudo, a implementação da metodologia não foi concluída, uma vez que a fase de Deployment não foi trabalhada.