Uma imagem com texto

Descrição gerada automaticamente

Introduction to Machine Learning — 2022/2023

Projeto final

André Guerra

Lucas Barrigó

Janeiro/2023

Conteúdo

[Introdução 3](#_Toc124085801)

[Classes e métodos 3](#_Toc124085802)

[Exercício 1 4](#_Toc124085803)

[Exercício 2 7](#_Toc124085804)

[Exercício 3 8](#_Toc124085805)

[Exercício 4 9](#_Toc124085806)

[Exercício 5 10](#_Toc124085807)

[Exercício 6 11](#_Toc124085808)

[Exercicio 7 12](#_Toc124085809)

[Exercício 8 13](#_Toc124085810)

[Exercício 9 13](#_Toc124085811)

[Conclusão 14](#_Toc124085812)

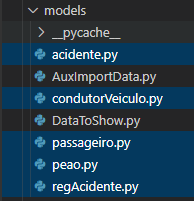
## Introdução

Neste relatório irá ser explicado resumidamente as classes criadas, funcionalidade dos métodos criados e por fim a execução e discussão dos resultados dos exercícios.

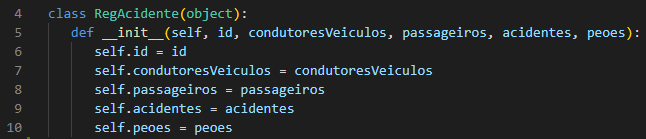
Este trabalho tem como objetivo analisar os dados sobre acidentes rodoviários em Portugal, desde o ano 2010 a 2019. Os dados foram fornecidos em formato Excel, 1 por cada ano, dos quais foram retirados dados de forma aleatória (2000 acidentes por cada ano) guardados em formato JSON que depois foram filtrados e normalizados para formato CSV, formato utilizado pelos algoritmos dos exercícios.

## Classes e métodos

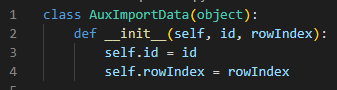
Começando pela extração de dados, foram criados objetos para cada folha dentro dos exeis, que contém os mesmos atributos presentes nas mesmas:



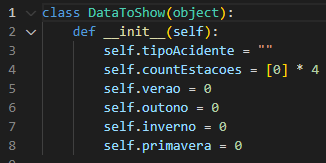
O objeto “RegAcidente”, presente na pasta dos models, representa um acidente que contem o id, condutor do veiculo (condutoresVeiculos), passageiros e peões, caso o acidente tenha esses atributos nas diferentes folhas do respetivo Excel:



O modelo “AuxImportData”, é um auxiliar para a extração dos dados, que irá ser explicada mais a frente:

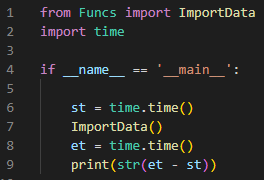


O modelo “DataToShow”, é um auxiliar para o exercício 3:



## Exercício 1

O exercício 1, presente no ficheiro exer1.py, consiste numa chamada do método que exporta aleatoriamente dados de todos os ficheiros para um formato de JSON, e mostra o tempo demorado da tarefa:

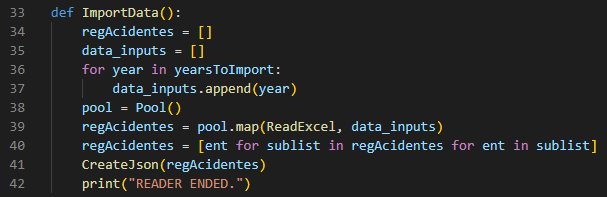


A exportação dos dados utiliza multiprocessamento para efetuar a extração mais rapidamente, tendo em conta que temos 10 exeis para exportar, são criados 10 processos que correm em núcleos separados do CPU, caso o CPU tenho menos de 10 núcleos, o próximo Excel apenas começa a exportar quando um núcleo fica disponível. A nossa extração contém 2000 acidentes de cada Excel, um total de 20.000 acidentes, escolhidos aleatoriamente, e tendo em conta o processador usado (i7-12700H), demora cerca de 40 minutos a extração.

O método que inicia a exportação de dados “ImportData” está no ficheiro “Funcs.py”, que também contem todos os outros métodos necessários para a exportação. Neste mesmo ficheiro começamos por declarar 2 variáveis que contem o número de acidentes e os anos a serem exportados:



O método “ImportData” começa por percorrer o array com os anos a serem exportados e cria o processo que realiza a exportação, os processos partilham uma variável global “regAcidentes”, que contem uma lista de objetos (“RegAcidente”) extraídos. Por cada processo é chamado o método “ReadExcel” que lê e extrai os dados:



O método “ReadExcel” começa por ler o ficheiro do ano que recebe, e as suas folhas. De seguida cria arrays com o objeto “AuxImportData” que contem o id e o índex da linha na folha, este array com o id e a posição da linha permite que a extração seja feita mais rapidamente (percorrer uma array de objetos pequenos ao invés das folhas no Excel):

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

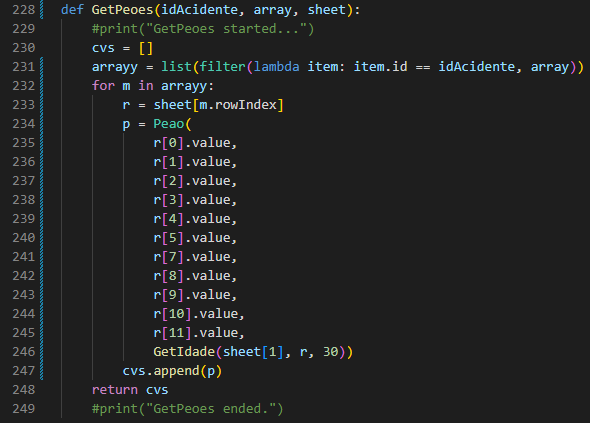
Após os arrays estarem construídos, percorremos o array 3 (ou a folha do Excel 3 “30 Dias \_ Acidentes”), até se atingir o número de dados pretendidos ou o array 3 chegar ao fim.

Esta extração pega numa posição aleatória do array, que escolhe o id do acidente a ser exportado, esse id é usado depois nos outros arrays, que depois de filtrados permite saber a posição do dado na folha de excel, essa exportação depois é feita nos métodos “GetCondutoresVeiculo”, “GetPassageiros”, “GetAcidentes” e “GetPeoes”, que cria o objeto “RegAcidente”. A cada acidente extraído, é removido dos arrays o acidente pelo id escolhido, impedindo a repetição do mesmo:

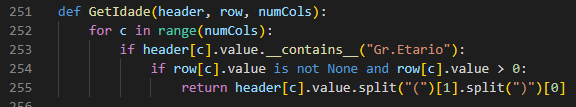
Uma imagem com texto

Descrição gerada automaticamente

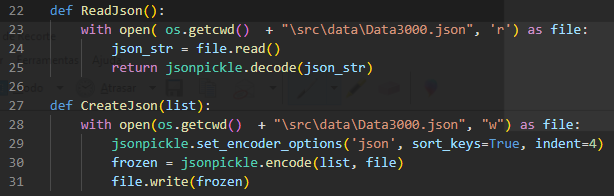
Os 4 métodos que extraem o objeto de cada folha do Excel são todos semelhantes. Explicando o “GetPeoes”, este recebe o id do acidente, o array da folha de Excel que contem os id’s e o index da posição dado presente na folha peões “30 Dias\_Peões”. O array depois de filtrado pelo id do acidente, é percorrido e extraída a informação da folha do Excel tendo em conta a posição do mesmo dado:



O método “GetIdade” retorna a idade de forma mais prática de se trabalhar mais a frente nos exercícios, retornando apenas a idade a que pertence o dado, por exemplo “40-44”:



Os métodos “ReadJson” e “CreateJson”, permitem ler e escrever em formato JSON os dados extraídos, estes dados estão presentes da pasta “data”:



## Exercício 2

Presente no ficheiro exer2.py, fazemos um tratamento dos dados exportados substituindo tudo por números e guardando tudo num ficheiro CSV, de forma a poder trabalhar com as bibliotecas:

Uma imagem com texto

Descrição gerada automaticamente

Este tratamento torna-se bastante eficiente na escolha dos dados, pois caso não seja possível converter para inteiro este é ignorado. A escolha dos dados a serem usados também é bastante simples e permite escolher apenas os que iram ser usados ou relevantes para o exercício, criando um ficheiro CSV com as colunas pretendidas:

Uma imagem com texto

Descrição gerada automaticamente

## Exercício 3

No exercício 3, não é utilizado o CSV tratado anteriormente, mas sim o JSON criado na extração. Neste exercício, para cada tipo de acidente, verificamos a estação do ano em que ocorreu:

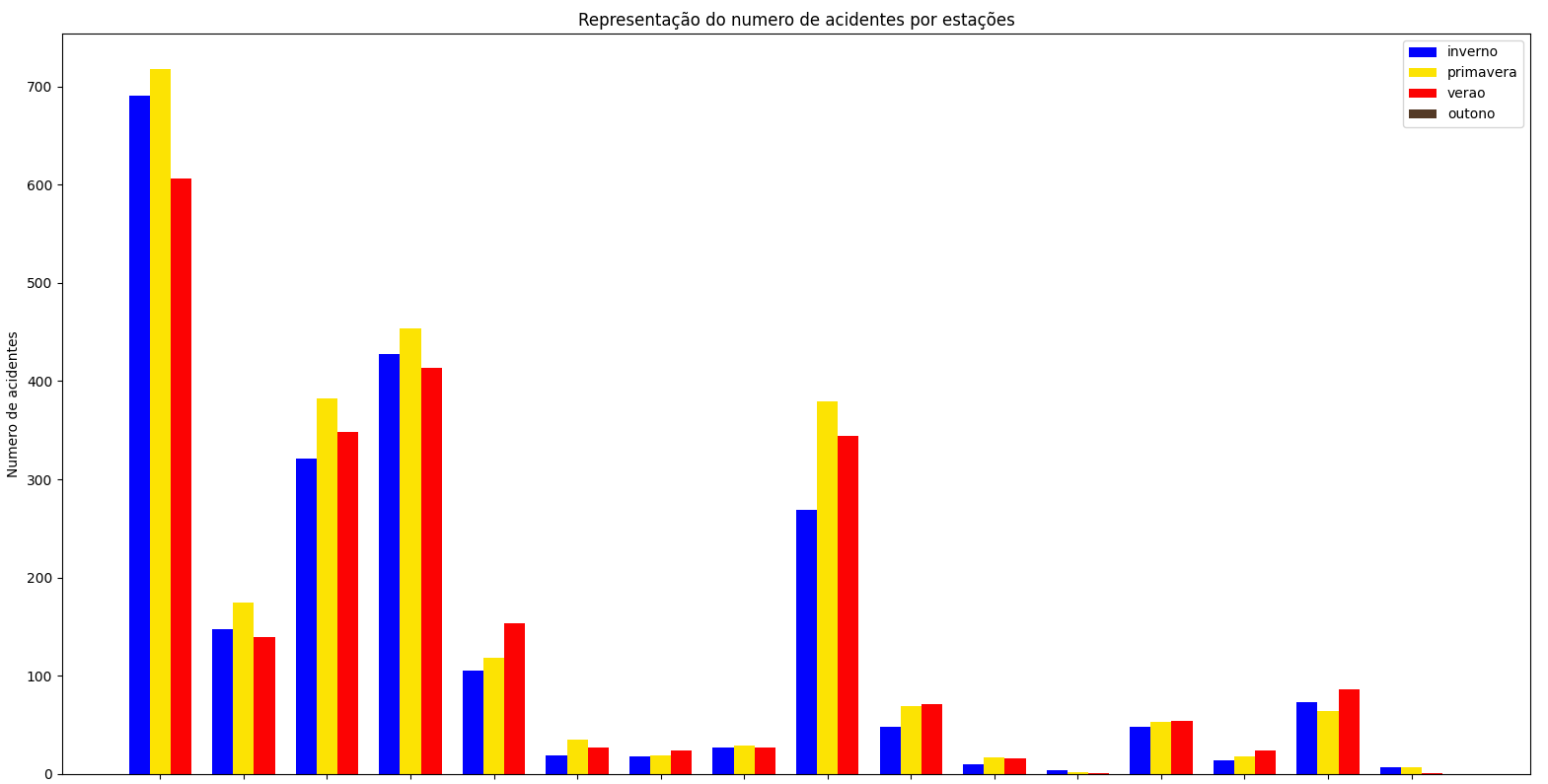
Uma imagem com texto

Descrição gerada automaticamente

E depois alteramos o formato dos dados contabilizados para o input do gráfico, o valor do tipo de acidente “não identificada” foi filtrado, pois tem um valor muito alto e irrelevante no exercício:

Uma imagem com texto

Descrição gerada automaticamente



Uma imagem com texto, utensílios de cozinha, captura de ecrã, linha

Descrição gerada automaticamente

Relativamente aos resultados obtidos no gráfico, podemos verificar que não existem acidentes no Outono, possivelmente devido á biblioteca usada na extração de dados ter uma limitação no tamanho da leitura da folha de Excel.

## Exercício 4

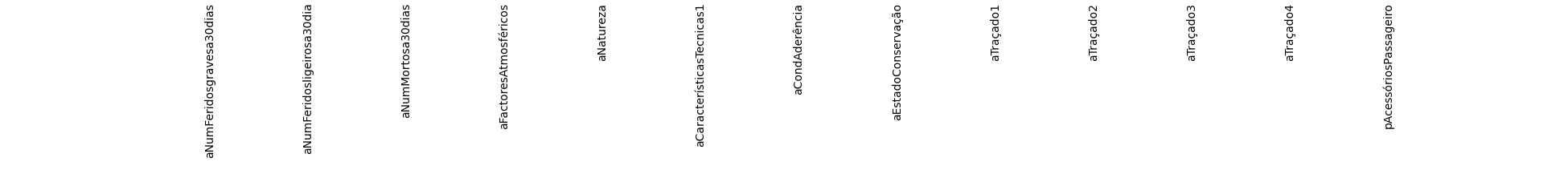
Tendo em conta o tratamento já feito no exercício 2, que filtra e trata os dados, este exercício 4 encontra-se incluído no anterior.

## Exercício 5

Neste exercício, utilizando o CSV criado no exercício 2, ao corrermos o algoritmo obtemos o seguinte gráfico:

Uma imagem com quadrado

Descrição gerada automaticamente



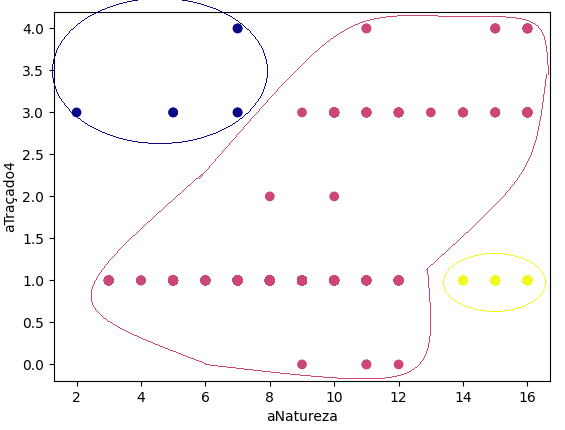
Podemos verificar que existe uma grande relação entre o número de mortos e feridos graves, pois esses feridos graves têm uma maior probabilidade de falecer que os feridos leves. Existe também uma alta relação entre as condições de aderência e os fatores atmosféricos, que podem ser explicados pela redução de aderência na estrada com piores condições atmosféricas. A relação entre o traçado 4 e a natureza, deve-se ao facto da maioria dos acidentes acontecer na via, fazendo com que exista uma alta correlação entre os dois fatores. A linha a branco no gráfico trata-se de um erro na biblioteca que produz o gráfico, visto que nos testes a alteração da ordem dos dados continua a mostrar essas linhas sempre nas mesmas posições, dando assim a entender que o bug não é devido aos parâmetros.

## Exercício 6

Neste exercício, utilizando o csv criado anteriormente, utilizamos uma biblioteca para ambos os algoritmos. O DBScan que com base no número de vizinhos cria um conjunto de clusters, avaliamos as colunas natureza e traçado do acidente, tendo em conta os dados que temos usamos os seguintes parâmetros, o número de vizinhos 5 por cada ponto, o epson 1.1 visto visto que os nossos dados se distanciam entre eles no mínimo 1 e mínimo de dados para criar um kluster 8.

Uma imagem com texto

Descrição gerada automaticamente

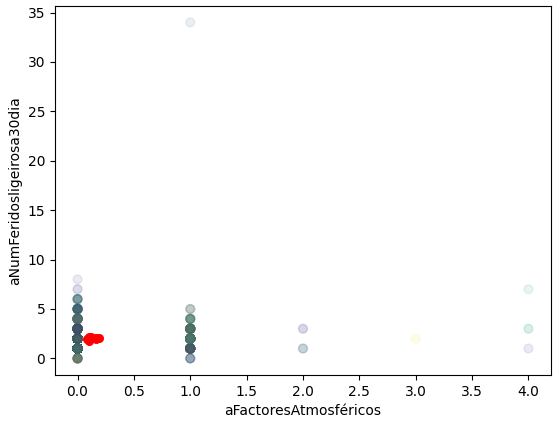


O algoritmo cria 3 kluster, a azul obtemos os acidentes “Na berma” e “No passeio”, nos quais a natureza é uma colisão ou atropelamento de peões, a vermelho obtemos todo o tipo de acidentes em todos os traçados, com exceção do tipo de acidente atropelamentos, a amarelo representa os acidentes com natureza de despiste “Em plena via”.

No algoritmo K-means, avaliamos os fatores atmosféricos e o número de feridos ligeiros, atribuímos o valor de 7 ao número de clusters, visto que são os diferentes tipos de fatores atmosféricos existentes.

Uma imagem com texto

Descrição gerada automaticamente



Tendo em conta que o maior número de dados existente ocorre quando está chuva e bom tempo (0 e 1), os centroides dirigem-se para esses pontos.

## Exercicio 7

No execicio 7, usamos um algoritmo chamado Adaboost, que calcula a percentagem de precisão, tendo em conta os dados escolhidos para chegar ao resultado de outro. Neste caso usamos as seguintes colunas: aNumFeridosgravesa30dias, aNumFeridosligeirosa30dia, aNumMortosa30dias, aFactoresAtmosféricos, aNatureza, aCaracterísticasTecnicas1, aCondAderência, aEstadoConservação, aTraçado1, aTraçado2, aTraçado3, aTraçado4, pAcessóriosPassageiro, cvSexo, cvIdade. Neste algoritmo podemos depois escolher qual a coluna que queremos prever.

Uma imagem com texto

Descrição gerada automaticamente

Escolhendo a coluna número de mortos, obtemos em média uma precisão de 99%, e conseguimos prever o sexo do condutor com uma precisão de 100%.

Já a natureza do acidente é mais difícil de prever, resultando numa precisão de 45%, como também a idade do condutor de 19% de acerto.

## Exercício 8

No exercício 8, testamos com o cross validation e 2 data sets diferentes, a precisão de acerto.

Uma imagem com texto

Descrição gerada automaticamente

Usando o data set “Data1102” obtemos os seguintes resultados:

Uma imagem com texto, rua

Descrição gerada automaticamente

No data set “Data2000Exer6” obtemos uma precisão média mais alta, pois o data set “Data1102” tem no seu total 10 mil dados extraídos, comparado com os 20 mil do “Data2000Exer6”.



Foi testado ainda outro data set “Data1101” que obteve resultados semelhantes ao “Data1102”:

Uma imagem com texto

Descrição gerada automaticamente

## Exercício 9

No exercício 9, implementamos 3 algoritmos, XGBoost, Multi Layer Perceptron (Neural Networks) e Decision Tree, para podermos averiguar a precisão média de cada um deles em 30 execuções. Também escolhemos várias colunas a serem prevista:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Multi Layer Perceptron | Decision Tree | XGBoost |
| Natureza do acidente | 87% | 100% | Fail |
| Número de mortos | 99% | 99% | 100% |
| Condições de aderência | 88% | 100% | 100% |
| Traçado 4 | 99% | 100% | 100% |
| Número de feridos ligeiros | 96% | 100% | 99% |

Neste exercício podemos observar que os algoritmos Decision Tree e XGBoost são mais precisos que o Multi Layer Perceptron, e ainda que em termos não conseguir chegar ao resultado o XGBoost quando os dados de treino ou os de teste, têm um valor não presente nos 2 tipos de dados este falha.

## Conclusão

Neste trabalho podemos concluir que a extração acabou por ser rápida, o que permitiu termos várias extrações para utilizarmos nos exercícios. O tratamento de dados é uma parte importante para que os exercícios corram de forma correta, o que pode influenciar os dados. Dos diferentes algoritmos implementados, o Decision Tree foi um dos mais precisos e eficazes.