Sensorização Ambiente aplicada à Segurança Rodoviária

André Figueiredo, Luís Ferreira, Pedro Machado, and Rafael Lourenço

Universidade do Minho, Departamento de Informática, 4710-057 Braga, Portugal

Perfil de Sistemas Inteligentes

Sistemas Autónomos, Trabalho Prático Nº2, Grupo 7

16 de Maio de 2021

e-mail: {a84807, a86265, a83719, a86266}@alunos.uminho.pt

Resumo Neste relatório será descrita e pormenorizada a arquitetura desenvolvida para o domínio da *Internet of Things* e *Smart Cities*, recorrendo a sensores ambiente físicos e virtuais.
Numa primeira parte, será efetuada uma pequena descrição do processo de recolha de dados da `API TomTom`, o processamento feito sobre eles e a forma como são armazenados e geridos, terminando com as duas modalidades desenvolvidas - monitorização em tempo real e histórica. De seguida, serão apresentados os *dashboards* desenvolvidos, de forma a expor a informação útil sobre o ambiente em estudo, explicando os gráficos pertencentes a cada um destes, assim como algumas personalizações efetuadas. Posteriormente, será descrito o modelo preditivo desenvolvido, cujo intuito é o de prever o número de incidentes que poderão ocorrer num dia futuro.
Por fim, será feita uma pequena apreciação dos resultados obtidos (com maior ênfase no modelo e nos seus resultados), de alguns dos problemas encontrados durante o desenvolvimento dos *dashboards* e, ainda, de algumas das decisões tomadas.

Keywords: *Data Visualization* · *Data Viz* · *Autonomous Systems* · *Intelligent Agents* · *Multi-Agent Systems* · *Ambient Intelligence* · *Internet of Things* · *Smart Cities* · Segurança Rodoviária · `TomTom`

1 Introdução

A sensorização ambiente refere-se a ambientes "sensíveis" e reativos à presença de indivíduos. Os aparelhos utilizados por estas arquiteturas têm vindo a tornar-se, progressivamente, mais "invisíveis" no ambiente em que estão colocados, devido ao desenvolvimento tecnológico cada vez mais rápido, aliado à diminuição do tamanho dos aparelhos em causa e ao facto de se estarem a tornar, gradualmente, mais conetados. Este "desaparecimento" permite que tal tecnologia se torne muito mais apelativa e essencial no quotidiano das grandes cidades.

Para tornar estes sistemas de sensorização ambiente ainda mais completos, é necessário integrar vários tipos de sensores (físicos e virtuais) e combinar os dados produzidos por eles. Estes passos têm o objetivo de potenciar a capacidade do sistema de gerar informação útil, tirando o máximo partido das ferramentas disponibilizadas no domínio da *Internet of Things* e de *Smart Cities*. No entanto, para além de

gerar informação relevante, é necessário desenvolver plataformas com ferramentas capazes de apresentar tal informação de forma simples, clara e interativa.

Tendo estes pontos em mente, a arquitetura aqui desenvolvida apresentará algumas etapas relevantes. A primeira será a recolha de dados, a segunda a apresentação da informação obtida com os dados e a terceira trata-se da previsão de eventos. No nosso caso em concreto, foi-nos solicitado que a nossa solução fosse desenvolvida para incidentes rodoviários.

Este tema é de grande relevância, tendo em conta o aumento da atividade rodoviária em todo o mundo, principalmente nos centros urbanos, sendo necessário desenvolver soluções fiáveis e inteligentes para gerir, monitorizar e, caso possível, mitigar os problemas que advêm desta atividade constante. Para além destes, é também importante ter em consideração todos os sub-problemas que derivam deles, como é o caso do tema deste trabalho prático - a segurança rodoviária.
Os acidentes rodoviários representam um problema atual[1], uma vez que, na última década, em Portugal, existiram cerca de 6.880 vítimas mortais resultantes (diretamente) da sinistralidade rodoviária. Apesar de estes valores terem vindo a diminuir[2], ainda existem várias medidas que podem ser tomadas por todas as ruas portuguesas, principalmente, nos centros mais movimentados, para diminuir, ainda mais, tais valores. Para tal, devem ser criadas soluções que permitam efetuar a monitorização constante e inteligente nas várias cidades do nosso país, mantendo a tendência atual da integração de *smart devices* e a adoção de projetos com o intuito de transformar as cidades portuguesas em *smart cities*.
Existem vários exemplos de projetos neste domínio[3] como, por exemplo, o projeto do município de Braga, em conjunto com a `TUB` (Transportes Urbanos de Braga), em que os autocarros estão equipados com sensores/leitores da qualidade do ar, temperatura e humidade (o que permite monitorizar os níveis desses componentes) e geram dados para permitir que os utilizadores possam consultar a aplicação móvel, sabendo, assim, quando e onde é que podem apanhar os autocarros.

Posto isto, inicialmente, devem ser recolhidos dados relevantes de cidades portuguesas, apresentando-os de forma clara e interativa e, posteriormente, deve ser desenvolvido um modelo de *Machine Learning* para a previsão de incidentes futuros, recorrendo a conceitos introduzidos em unidades curriculares anteriores, como Aprendizagem e Extração de Conhecimento e Computação Natural.

Em suma, neste relatório será descrita a fase de recolha de dados da `API TomTom`, o processamento aplicado a estes e, consequentemente, a produção de informação útil. De seguida, serão apresentados os *dashboard* construídos com recurso às ferramentas `Google Data Studio` e `Google Sites`. Posteriormente, será explicado o processamento feito sobre os dados para o modelo, assim como a arquitetura definida para este. Por fim, serão discutidos os resultados obtidos e expostas algumas limitações das ferramentas utilizadas, assim como os problemas encontrados durante o desenvolvimento deste projeto.

[1]Fonte: `https://www.publico.pt/2020/11/14/sociedade/noticia/portugal-regista-6880-mortos-acidentes-rodoviarios-ultima-decada-1939241`.

[2]Fonte: `https://www.pordata.pt/Portugal/Acidentes+de+via%c3%a7%c3%a3o+com+v%c3%adtimas++feridos+e+mortos+++Continente-326-3585`.

[3]Fonte: `https://insider.dn.pt/em-rede/22-projetos-smart-cities-portugal/3052/`.

2 Recolha de Dados

Tal como já foi referido, a recolha de dados é uma etapa crucial, pois são estes que permitem a execução das duas próximas etapas - a apresentação da informação e a previsão de eventos. Assim sendo, iremos começar por apresentar o processo seguido nesta etapa.

2.1 Pedido à `API`

Primeiramente, foi criada uma conta no *website* da `TomTom`. Esta tem associada uma *key* que é atribuída aquando do registo, sendo a sua utilização obrigatória para efetuar pedidos `GET` à `API`. Estes pedidos são efetuados através de *link*s, por exemplo, `https://api.tomtom.com/traffic/services/5/incidentDetails?bbox=-8.916%2C41.247%2C-7.829%2C42.1&fields=%7Bincidents%7Btype...`.

Visto que se tinha acesso a 2500 pedidos por dia, optou-se pela sua distribuição por 8 regiões de Portugal[4], resultando num pedido a cada 5 minutos[5].

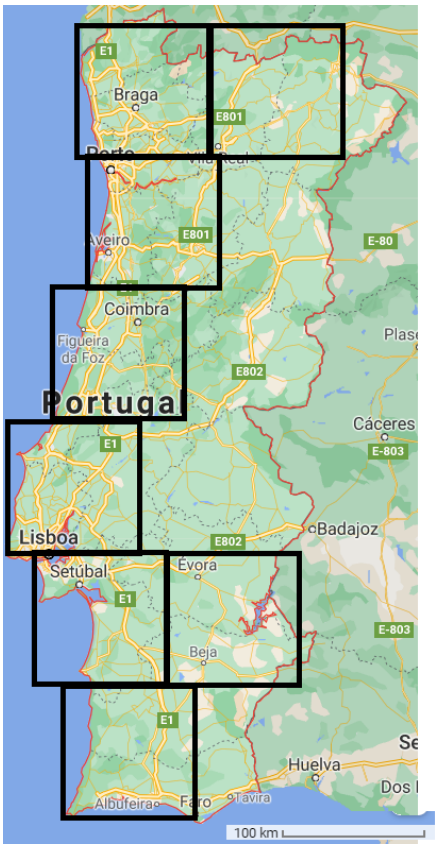

Figura 1: Regiões monitorizadas

[4]Estas regiões têm perto do limite máximo da área definido pela `API`, ou seja, 10000 km^2.

[5] $\frac{2500}{8*24} \approx 13$ pedidos por hora, permitindo, no máximo, um pedido a cada 4.6 minutos.

Depois de decidir o número de pedidos distintos e o período decorrido entre eles, foi necessário escolher a ferramenta correta para os efetuar, sendo que, neste caso, escolheu-se utilizar um `Raspberry Pi` para o efeito. Através de um *script* `Python` em conjunto com a ferramenta `Crontab`[6], o `Raspberry Pi` é capaz de efetuar pedidos de 5 em 5 minutos durante o intervalo de tempo pretendido, armazenando a informação em 8 ficheiros `JSON` diferentes, um para cada região.

2.2 Adição da versão aos dados

Ao fim de alguns dias de execução, constatou-se que certos incidentes continham informação diferente ao longo desse período. Por isso e para permitir que fossem armazenadas/analisadas várias versões do mesmo incidente, optou-se por manter todos os incidentes passados, adicionando apenas os novos incidentes e os incidentes passados, mas com informação nova. Desta forma, torna-se necessário avaliar se um incidente repetido contém informação nova e útil. Tal decisão é feita recorrendo aos campos *id* e *length*, visto que se considerou serem suficientes para diferenciar as várias versões dos mesmos incidentes.

Após esta alteração, foi necessário identificar cada uma das várias versões, sendo, para o efeito, criados os campos *timestamp*, que identifica o instante em que a informação foi recolhida, e *versão*, que permite obter a ordem pela qual a informação referente a um dado incidente foi colocada[7].

2.3 *Real-Time vs.* histórico

Para possibilitar a apresentação da informação recolhida, era necessário que os vários ficheiros estivessem agrupados num único ficheiro. Por isso, optou-se por passar todos os dados para o formato *CSV*, uma vez que, com este formato, é possível utilizar as *sheets* do `Google` no `Google Data Studio`. Assim sendo, foi também necessário filtrar os campos pretendidos e criar novos campos com informação obtida através dos dados recolhidos. A escolha sobre os campos escolhidos baseou-se na informação necessária para a representação gráfica e para o modelo preditivo, resultando nos seguintes campos:

- *id* do incidente;
- nome das estradas afetadas (obtido recorrendo a *Regex*);
- número de estradas afetadas;
- causas do incidente;
- comprimento do incidente;
- data e hora em que se deu o incidente e em que terminou;
- categoria do incidente (convertida de acordo com a `API`[8]);
- magnitude do atraso provocado (convertida de acordo com a `API`[8]);
- coordenadas geográficas;

[6] Ferramenta que permite agendar a execução de programas consoante o período/padrão definido pelo utilizador.

[7] Tal ordem poderia ser obtida através do *timestamp* adicionado, contudo seria um processo mais custoso, demorado e menos intuitivo.

[8] Disponível em: `https://developer.tomtom.com/traffic-api/traffic-api-documentation-traffic-incidents/incident-details`

- região (uma das 8 definidas);
- versão da informação;
- número de horas nas quais o incidente esteve ativo.

No entanto, após a análise dos dados obtidos, concluiu-se que era importante permitir a análise da situação atual (para além dos incidentes passados). Para tal, foram criadas duas modalidades distintas de dados - uma para a monitorização em tempo real e outra para a informação histórica. Todavia, para obter esta informação, era necessário separar a informação gerada em dois ficheiros diferentes. Para a monitorização dos dados históricos, a informação armazenada nunca é apagada, sendo, apenas, acrescentada a nova informação. Por outro lado, para a monitorização em tempo real, a informação passada teria de ser descartada, mantendo apenas a informação contida na última resposta da `API`, isto é, o ficheiro teria de ser reescrito apenas com a informação mais atual.
Ainda assim, surgiu um problema: para que a monitorização ocorresse em tempo real, era necessário que o ficheiro na *drive* fosse atualizado constantemente. Para o resolver, foi criado um *script* `Python` que permite fazer o *upload*/*update* de um dado ficheiro (mantendo o mesmo *url* e *id*, no caso do ficheiro já existir), através de uma conta de serviço (`IAM`) criada para o efeito. Como a informação era atualizada de 5 em 5 minutos, foi decidido efetuar o *upload* dos ficheiros de 6 em 6 minutos, para minimizar as hipóteses de fazer *upload* de um ficheiro vazio.

No que diz respeito à informação histórica, optou-se por utilizar apenas a informação mais recente de cada incidente. Esta escolha era realizada através do campo *versão* referido, utilizando, então, o incidente com o valor mais elevado deste campo. Por último, restava apenas decidir de que modo é que a informação seria agrupada e convertida num só ficheiro, pois, visto que cada incidente tem uma lista de coordenadas, foi decidido que cada incidente seria repetido várias vezes, alterando apenas as coordenadas, o que resulta em várias linhas com o mesmo *id*, mas coordenadas diferentes.

3 Apresentação da Informação

3.1 *Dashboard* 1 - Histórico de incidentes

O objetivo deste *dashboard* é analisar o histórico dos incidentes com base em vários indicadores, que utilizam a informação gerada a partir dos dados recolhidos previamente. Assim sendo, o *dashboard* desenvolvido foi o seguinte:

Figura 2: *Dashboard* referente ao histórico de incidentes

O primeiro gráfico (canto superior esquerdo) apresenta o número de incidentes e estradas afetadas por categoria de *delay*, ou seja, para cada categoria, quantos incidentes existiram e quantas estradas foram afetadas. No segundo gráfico, é possível verificar a distribuição dos incidentes por categoria de incidente. No último gráfico da primeira linha, podemos visualizar quantos incidentes existiram, diferenciados por grandeza de *delay*, ao longo de vários dias.
De seguida, ilustra-se um mapa onde são representadas as coordenadas dos incidentes, sendo que tamanho de cada bolha é definido pelo comprimento do incidente.
Para a última linha, é evidenciada uma tabela que contém o número de incidentes por descrição, uma vez que estas descrições podem ser variadas. De seguida, é apresentado um gráfico circular onde é representada a distribuição da duração de cada incidente em horas, sendo que esta foi dividida em intervalos de 30 minutos (e todos os incidentes com 12 ou mais horas foram agrupados numa só categoria).
Por fim, temos um gráfico que permite visualizar os incidentes por hora, podendo analisar os picos de incidentes.

É de realçar que todos os gráficos são interativos, ou seja, é possível aplicar vários filtros aos dados:

- Filtrar por região, estrada e período de tempo (definindo uma data inicial e final) a analisar. Para isso, devem ser utilizados os *drop-down's*. Caso nenhum valor seja selecionado, é utilizado o *CSV* na totalidade;
- Filtrar por magnitude (por exemplo, *Delay* moderado) no período escolhido e no dia(s) selecionado(s). Tal é realizado através de gráficos de barras;
- Através do gráfico circular da primeira linha, é possível realizar uma filtragem por categoria, nomeadamente, congestionamento, trabalhos na estrada, entre outros;
- Filtrar por descrição detalhada, através da única tabela apresentada;
- Filtrar por duração média de um incidente, ou seja, apresentar apenas os incidentes que demoraram, em média, por exemplo, 30 minutos ou 1 hora;
- Filtrar por hora, nomeadamente, visualizar as horas de pico/ponta onde ocorre o maior número de incidentes.

3.2 *Dashboard* 2 - *Real-time*

No segundo *dashboard* é possível analisar dados em *real-time*, que são atualizados de 6 em 6 minutos. Isto é realizado, tal como já foi referido, através de um `Raspberry Pi` que, durante esse intervalo de tempo, atualiza um ficheiro que se encontra na *drive*, ficheiro este que está diretamente ligado ao *dashboard*, através de um conetor do `Google Data Studio`, resultando, assim, no seguinte *dashboard*:

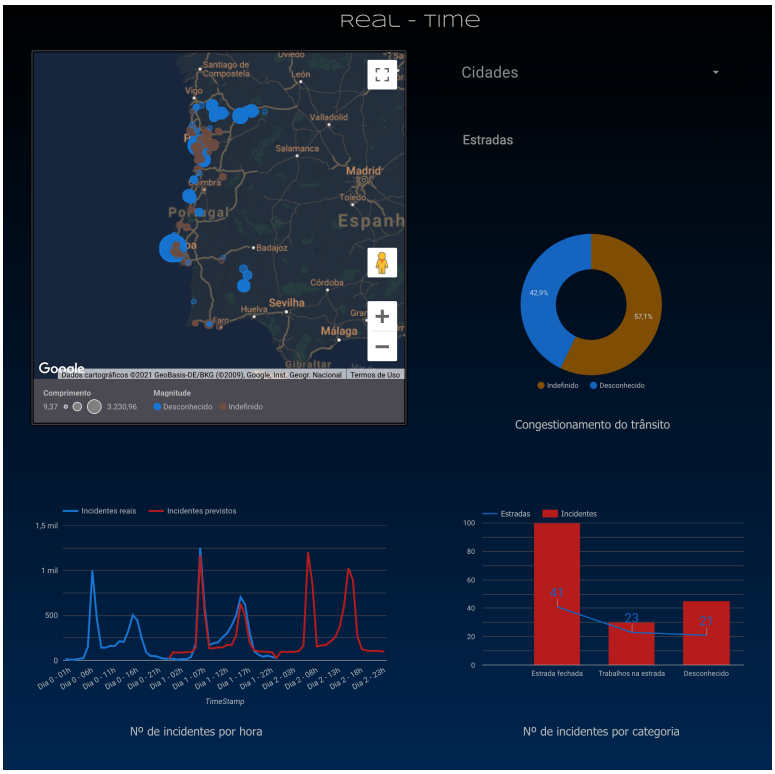

Figura 3: *Dashboard* referente ao *real-time*

Visto que são recolhidos dados de várias regiões do país e de forma a permitir a sua filtragem, foram colocados dois *drop-down's* que permitem selecionar a(s) região(ões) e a(s) estrada(s) que se pretende analisar. Ainda neste *dashboard*, é possível filtrar os dados por magnitude de *delay*, através do gráfico circular, e por categoria, através do gráfico de barras. Deste modo, torna-se possível a visualização, por exemplo, das estradas fechadas ou que se encontram em obras, através da combinação deste filtro com as coordenadas dos incidentes que serão apresentadas no mapa.
Por último, combinando os dados recolhidos com os dados resultantes do modelo preditivo (abordado na secção 4), foi possível a criação de um gráfico que permite a visualização dos incidentes esperados durante as próximas 24 horas.

3.3 *Google Sites*

O `Google Sites` é tipicamente usado para disponibilizar a informação destes *dashboards*, uma vez que torna a sua integração muito simples. No entanto, devido à `API` do `Google` ser paga, não foi possível representar o `Google Maps` lá, tendo, por isso, apenas sido colocados os gráficos permitidos. Dado que esta restrição limita bastante a capacidade de visualização dos dados, optou-se por não lhe conferir demasiada relevância.

Posto isto, o *website* criado contém duas páginas - uma correspondente ao histórico de incidentes e a outra correspondente ao *real-time* (muito similares aos *dashboards* apresentados anteriormente).

Figura 4: Página *Web* referente ao *histórico*

Nesta página, tem-se, praticamente, o primeiro *dashboard* completo (excetuando apenas o mapa da segunda linha), apresentando resultados similares.

Figura 5: Página *Web* referente ao *real-time*

Já nesta página, é possível aplicar todos os filtros que foram explicados anteriormente, exceto a visualização dos mapas. Assim sendo, o primeiro gráfico é a combinação do modelo preditivo com os dados *real-time*, o segundo é a apresentação do congestionamento devido aos incidentes e o último são os incidentes por categoria.

4 Modelo Preditivo

Com vista a conseguir analisar e dar um propósito aos dados, com bastante mais relevo, efetuou-se o desenvolvimento de um modelo preditivo, cujo objetivo é prever o número de incidentes nas futuras 24 horas, ou seja, para dados não recolhidos.

Desta forma, tornou-se necessário tratar os dados para a sua posterior utilização no modelo.

4.1 Tratamento de Dados

Para o tratamento de dados, mostrou-se relevante a filtragem dos incidentes pelo seu identificador, pois não era pretendido utilizar os mesmos incidentes para a previsão. Após essa remoção de dados desnecessários[9], era necessário agregar os restantes dados pelas datas, uma vez que se pretende analisar o número de incidentes por hora. Assim, procedeu-se à passagem da data para um inteiro que a representa da seguinte forma: *YearMonthDayHour* - `2021041022`, isto é, 22 horas do dia 10 de Abril de 2021.

Date	Number roads affected	Length	Number of incidents
2019081313	0	93.276295	1
2019082411	1	96.512976	1
2020033009	1	548.718780	1
2020060111	0	407.481342	2
2020061212	1	105.268755	1
...	...	...	...
2021050119	45	9106.822490	36
2021050120	53	17931.243716	37
2021050121	43	21401.828333	37
2021050122	31	14593.105304	23
2021050123	6	7054.055413	7

Figura 6: *Dataset* após a agregação dos incidentes pelas datas

Como é possível observar, apenas existem três colunas - número de ruas afetadas pelo incidente, comprimento em metros do incidente e número de incidentes naquela data e hora. No entanto, apenas se deu uso a duas variáveis na versão final, uma vez que o *length* (comprimento) foi removido, de modo a se possuir um *dataset* mais simples para, teoricamente, o modelo prever o número de incidentes com uma *performance* superior.
Esta seleção das *features* a serem usadas pelo modelo preditivo foi acompanhada da visualização da correlação dos atributos com a variável *target* (isto é, o número de incidentes).

[9] Das 630000 linhas do *dataset*, ficou-se, apenas, com 20000 incidentes únicos.

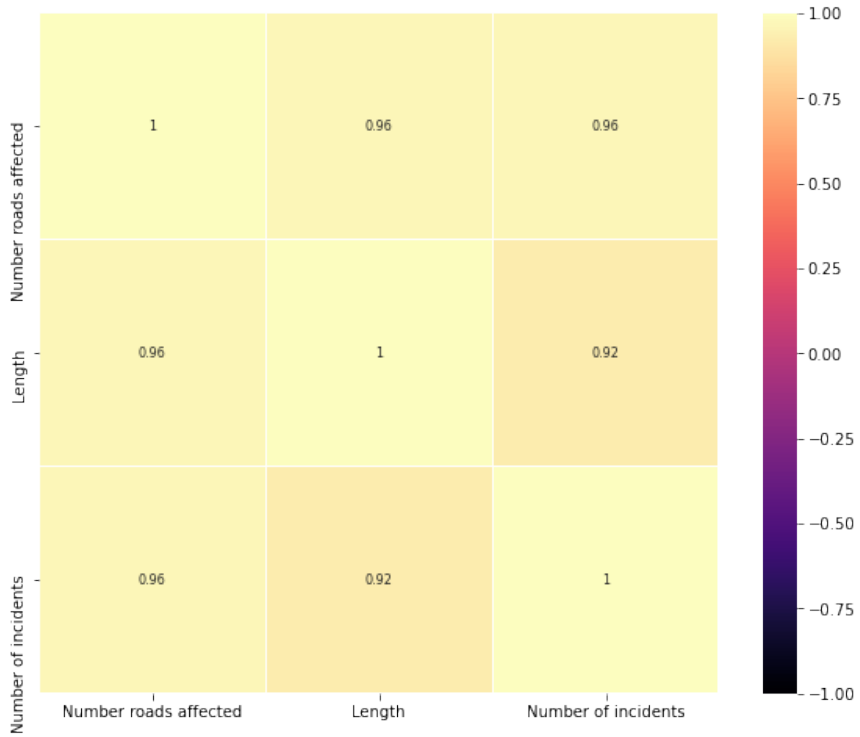

Figura 7: Correlação entre as *features* do *dataset* gerado

Após observação da figura prévia, conclui-se que a *feature* `Number roads affected` é a que consegue demonstrar melhor o comportamento do *target*, pelo que, por isso mesmo, foi mantida para a previsão.

Por fim, foi necessário remover todos os dias com menos de 24 horas de dados, uma vez que se pretendia prever o número de incidentes para um dia inteiro, não fazendo sentido conter dias com falta de dados.

4.2 *Time Series Forecasting*

Após todo tratamento de dados, é necessário executar uma normalização, para a sua utilização no modelo de séries temporais. Assim, executa-se uma normalização entre -1 e 1[10] da seguinte forma:

```
# [-1, 1] for LSTM due to internal use of tanh by the memory cell
scaler_f = MinMaxScaler(feature_range = (-1, 1))
scaler_target = MinMaxScaler(feature_range = (-1, 1))

f = 'Number roads affected'
t = 'Number of incidents'

incidents[[f]] = scaler_f.fit_transform(incidents[[f]])
incidents[[t]] = scaler_target.fit_transform(incidents[[t]])
```

[10]Esta normalização deve-se ao estado interno da célula de memória do `LSTM`.

Para proceder à execução do modelo para *forecast* do número de incidentes, é necessário definir valores fulcrais para as *time series* como o *timesteps*, *multisteps* e *batch size*. O *timesteps* define a sequência de dados da série temporal, isto é, neste caso, 24 horas. Já o *multisteps* corresponde ao número de horas que irão ser previstas, isto é, novamente, 24 horas. Por fim, o *batch size* dita de quanto em quanto tempo é que os pesos da rede são atualizados, isto é, novamente, nos modelos desenvolvidos, 24 horas.

No que diz respeito ao modelo preditivo, utilizou-se um modelo que dá uso a `LSTM`s como forma de aprendizagem para a previsão de dados futuros, relativos a incidentes no território português, onde os dados foram recolhidos.

O modelo abaixo foi obtido de forma iterativa, isto é, recorrendo a várias tentativas, com vista a otimizar os seus hiper-parâmetros e, consequentemente, a sua *performance*. Para tal otimização recorreu-se a *random search* de vários tipos de modelos (`LSTM`, `GRU`, `CNN`, `MLP` e outros mais complexos como `CNN-LSTM`), sendo, então, selecionado um dos modelos com melhor *performance* durante o processo de treino:

```
Model: "LSTM_Model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 24, 128)           66560
_________________________________________________________________
batch_normalization (BatchNo (None, 24, 128)           512
_________________________________________________________________
dropout (Dropout)            (None, 24, 128)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 24, 128)           131584
_________________________________________________________________
batch_normalization_1 (Batch (None, 24, 128)           512
_________________________________________________________________
dropout_1 (Dropout)          (None, 24, 128)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 24, 32)            20608
_________________________________________________________________
batch_normalization_2 (Batch (None, 24, 32)            128
_________________________________________________________________
dropout_2 (Dropout)          (None, 24, 32)            0
_________________________________________________________________
dense (Dense)                (None, 24, 64)            2112
_________________________________________________________________
batch_normalization_3 (Batch (None, 24, 64)            256
_________________________________________________________________
dropout_3 (Dropout)          (None, 24, 64)            0
_________________________________________________________________
dense_1 (Dense)              (None, 24, 1)             65
=================================================================
Total params: 222,337
Trainable params: 221,633
Non-trainable params: 704
```

Após a construção do modelo, procedeu-se ao seu treino, através do *Time Series Split*, de forma a potenciar e avaliar o modelo com diferentes dados, tanto de treino, como de validação. De seguida, realizou-se o treino do melhor modelo, já com os dados completos, para prever o número de incidentes do próximo dia.

É de realçar que, durante o desenvolvimento de vários modelos, se procedeu à análise da sua *performance*, utilizando uma ou mais variáveis, pelo que essa análise e os resultados obtidos se encontram detalhados na secção seguinte.

5 Análise de Resultados

A análise de *performance* e escolha do melhor modelo preditivo foi avaliada pela *loss* (através da métrica `RMSE`) e pela comparação do número de incidentes previstos com o número real de incidentes no último dia de dados recolhidos.

De acordo com a *loss* obtida nos vários modelos implementados, optou-se pela escolha de um problema univariável, descartando o número de ruas afetadas. A partir desse modelo e problema, já identificado anteriormente, conseguiu-se os seguintes resultados para as métricas de avaliação:

***Performance* do Modelo**	**RMSE**	**MAE**	**Tempo de execução**
`LSTM_Model`	0.27	0.20	299 segundos

Tabela 1: Comparação de dois indivíduos com os dados de teste

Estes valores, apesar de não serem ótimos, devido à diminuta quantidade de dados que foram recolhidos[11], correspondem a uma *performance* aceitável.
Assim e para analisar a *performance* do modelo, foi previsto o número de incidentes para o dia 30 de Abril (último dia com dados recolhidos) e comparou-se os valores resultantes com os valores reais obtidos. Tal comparação encontra-se demonstrada no seguinte gráfico:

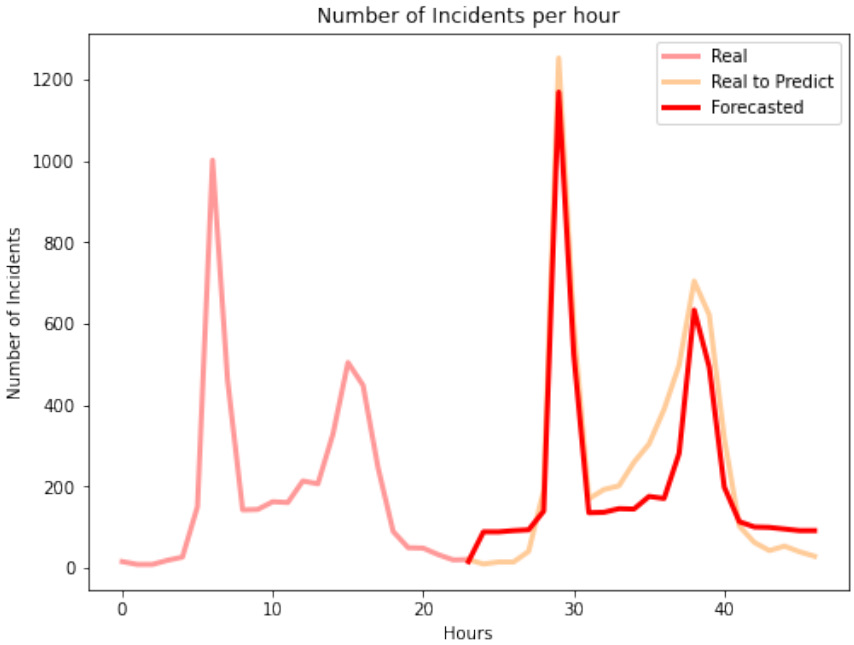

Figura 8: Comparação do número de incidentes previstos pelo modelo e reais

[11] A previsão do modelo baseou-se nos dados de 3 dias completos (27 a 29 de Abril) para o treino.

Como é percetível na figura, o modelo prevê o número de incidentes para as últimas 24 horas disponíveis com valores relativamente próximos. Para além disso, também se constata que a distribuição dos valores previstos segue, de maneira muito idêntica, a distribuição do número de incidentes reais. O modelo apontou o máximo de incidentes pelas 6 ou 7 horas da manhã com valores próximos dos 1200 incidentes, sendo que o valor real se encontra muito próximo deste (aproximadamente, 1250). Já em relação ao segundo pico de incidentes, este ocorreu pelas 14 horas com cerca de 650 incidentes.

De seguida, procedeu-se à previsão do número de incidentes para dia 1 de Maio, sendo que se obtiveram os seguintes resultados:

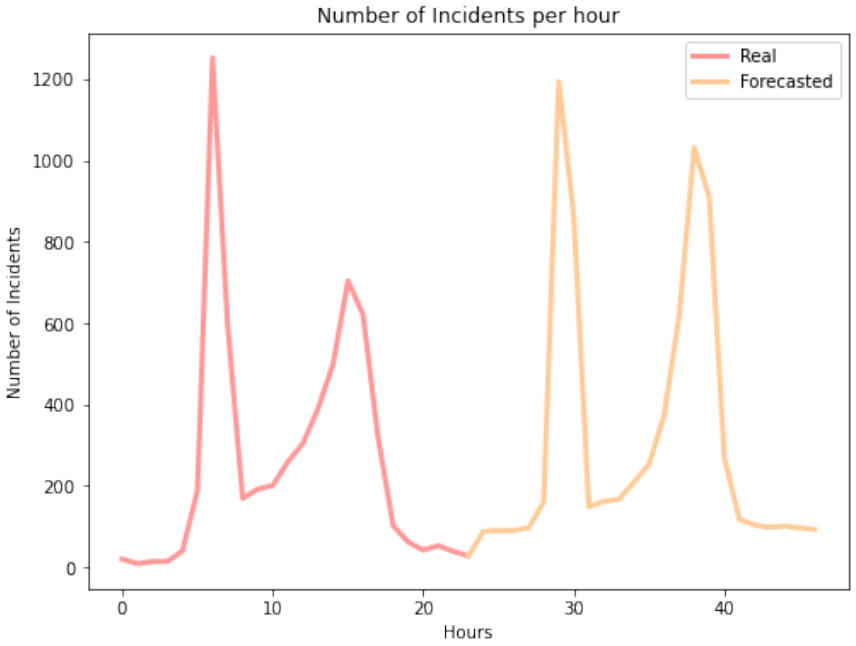

Figura 9: Previsão do número de incidentes do dia 1 de Maio

Relativamente a esse dia, o modelo previu um número de incidentes quase idêntico, com um pico de incidentes pelas 7 horas com pouco menos de 1200 incidentes, dentro das áreas recolhidas. Já o segundo pico do número de incidentes acontece por volta das 15 horas com um valor a rondar os 1000 incidentes. É de ressalvar que, segundo a previsão, este dia teve, pelas 14 ou 15 horas, um aumento de incidentes em relação ao dia anterior (cerca de 350 incidentes a mais).

5.1 Dados apresentados nos *dashboards*

Os *dashboards* permitem criar uma camada de interação mais simples e rápida e desenvolver sistemas que permitem tirar conclusões mais claras e com vários gráficos para análise.
Assim e em relação às regiões monitorizadas, puderam-se tirar as seguintes conclusões:

- Tal como seria de esperar, as cidades com mais incidentes são Lisboa, Porto e Braga, por essa mesma ordem;
- Cerca de 50% dos incidentes apenas demoraram 1 hora ou menos, visto que a sua maior causa é o trânsito parado/congestionamento. Isto deve-se, em grande parte, ao facto de esta ser a causa mais comum por grande margem, representando cerca de 90% do número total;
- Tal como constatado aquando da avaliação do modelo, os dias tinham, normalmente, dois picos - às 6 horas, com cerca de 1300 incidentes e às 15 horas com cerca de 800 incidentes. Por outro lado, as horas com menos incidentes ocorrem às 2 e 3 horas;
- Mais de metade dos incidentes são categorizados como *delay* elevado;
- Em relação à cidade de Braga, por exemplo, pôde constatar-se que a distribuição por horas segue um padrão muito semelhante ao nível nacional, contudo o segundo pico decorre às 16 horas, ao invés das 15 horas;
- Ao contrário da norma nacional, na região do Sul Litoral, o maior pico de incidentes ocorre às 15 horas;
- Na região de Bragança, existem dois picos antes do meio dia, mais precisamente, às 6 e às 9 horas;
- Os incidentes com um *delay* indefinido (denominação utilizada para estradas fechadas e qualquer outro tipo de "atrasos") demoraram, normalmente, 12 ou mais horas até se conseguir resolver tais situações e, tal como seria de esperar, as descrições mais comuns são "Encerrado" e "Trabalhos na estrada".

Já em relação aos dados em tempo real, é possível constatar que:

- O número de incidentes varia consoante a altura do dia, tendo mais incidentes durante a manhã e o início da tarde;
- Normalmente, é registado um número de incidentes mais elevado nas regiões do Norte Litoral, Porto e Lisboa.

5.2 Dificuldades

Tal como foi referido, a recolha de dados teve de ser alterada várias vezes, fazendo com que alguns dados fossem perdidos, restando apenas 4/5 dias contíguos para análise e previsão do modelo, tornando-o, inicialmente, algo inconsistente.

No entanto, se para o modelo os dados eram escassos, para o `Google Data Studio` estes excediam o limite máximo de linhas e tamanho dos ficheiros para esta plataforma, sendo, portanto, necessário aplicar algumas técnicas, de modo a mitigar possíveis problemas durante a apresentação dos mesmos. Por exemplo, para o ficheiro histórico, apenas eram colocadas um terço das coordenadas presentes nos dados.
Outra questão problemática relacionada com esta plataforma devia-se aos problemas de compatibilidade aquando da atualização do `Google Sheets` na *drive* e, por isso, foi necessário alterar a ligação para um *CSV* (também este na *drive*). Todavia, esta ligação era mais lenta e provocava perda de informação, passando a ter, normalmente, apenas 50% das linhas originais. Tal problema era facilmente visível quando se aplicava um filtro de magnitude, visto que começavam a aparecer novos dados em vários gráficos, como o aumento de incidentes causados por trabalhos na estrada ou o aumento de incidentes por região, entre outros.
Outro ponto a realçar, é a incapacidade do `Google Data Studio` de lidar com listas num ficheiro *CSV* e, devido a esta limitação, optou-se por recorrer a um filtro

externo que funcionava como um menu *dropdown*. Este problema afetou a representação das coordenadas que constituem um incidente, uma vez que foi necessário a replicação do mesmo incidente em várias linhas, alterando apenas essas coordenadas, pois, de outra forma, estas não eram representáveis no mapa.
Além disso, o facto de os gráficos que utilizavam a interface do `Google Maps` terem um limite muito baixo de coordenadas quando comparados com os dados armazenados, criou uma nova limitação, visto que das 240000 coordenadas armazenadas, apenas era capaz de apresentar 10000.

Por último, ao tentar incorporar os relatórios gerados no `Google Data Studio`, qualquer gráfico que utilizasse a interface do `Google Maps` tornava-se indisponível e, portanto, optou-se por manter ambas as soluções, isto é, desenvolver relatórios no `Google Data Studio`, utilizando os vários tipo de gráficos e a incorporação de relatórios no `Google Sites`, apresentando os gráficos desenvolvidos anteriormente, excluindo os gráficos que fizessem uso do `Google Maps`.

5.3 Trabalho Futuro

A arquitetura desenvolvida, no nosso entender, poderia sofrer algumas extensões, contudo, a principal melhoria a realizar seria a recolha de dados durante um período mais alargado. Deste modo, seria possível recolher uma quantidade maior e mais diversificada de dados que, consequentemente, iriam melhorar os resultados do modelo desenvolvido.

Considera-se, ainda, que poderiam ser explorados mais tipos de modelos preditivos e parâmetros. Outra possível melhoria seria a previsão de incidentes nas próximas 6 ou 24 horas em tempo real, todavia, para criar este tipo de arquitetura, era necessário que o `Raspberry Pi` estivesse a executar a previsão, por exemplo, de 6 em 6 horas ou 24 em 24 horas, algo que poderia gerar problemas de sobreaquecimento. Para além disso, seria também necessário que o resultado desta previsão fosse enviado para a *drive*, sem interação externa.

Relativamente à *performance* do modelo em si, seria possível efetuar uma filtragem dos dados, com vista a agrupá-los por dia da semana, possibilitando, ainda, o uso de um *batch size* maior. No entanto, tal melhoria não foi viável, pois a quantidade de dados era insuficiente.

6 Conclusão

A realização deste projeto permitiu aplicar os conhecimentos apreendidos nas aulas teóricas e práticas da unidade curricular de Sistemas Autónomos, bem como incentivar a exploração de ferramentas relacionadas com *Data Visualization* e *Time Series*, aprofundar conhecimentos na área de Sensorização Ambiente e *Machine Learning* e pesquisar as técnicas mais atuais nestas áreas.

Os principais desafios durante o desenvolvimento deste projeto prenderam-se com as dificuldades que as ferramentas utilizadas provocaram, assim como a quantidade de dados utilizáveis, principalmente, para o modelo preditivo.

A validação dos modelos gerados, bem como a análise dos seus resultados, sofreu bastante devido à reduzida quantidade de dias disponíveis, pelo que a previsão pode ser influenciada por algum *overfit* sobre as poucas horas disponíveis para treino. Assim, a aplicação deste modelo para previsão do número de incidentes com vários dias de diferença ou até fins de semana e feriados (que podem ter variações maiores) não nos garante resultados fidedignos. Desta forma, seria muito útil continuar a recolher dados para previsão destes.
Outro fator interessante seria a previsão do número de incidentes em tempo real, ou seja, com a aplicação do modelo na previsão do número de incidentes numa data selecionada pelo utilizador. Todavia, os resultados gerados, tendo em conta as restrições impostas, parecem-nos bastante positivos e garantem, com um certo grau de confiança, uma previsão útil para a utilização deste projeto num contexto real, por exemplo, um projeto para estudar a situação rodoviária de municípios para promover boas práticas de segurança.

Quanto à recolha de dados, considera-se que o período estabelecido (5 minutos) é apropriado, tendo em conta as regiões escolhidas. Aliado a esta, as alterações à arquitetura (como guardar várias versões do mesmo incidente), abrem portas no que diz respeito à análise de incidentes e da sua evolução. Para além disso, o modo de monitorização em tempo real é também muito interessante, pois permite suportar duas abordagens completamente diferentes em relação à forma como os dados são analisados.

Relativamente à representação de informação com *dashboards*, pode-se afirmar que a solução desenvolvida é bastante útil, de fácil uso e muito interativa, características estas muito importantes para a adoção de novas ferramentas. Quanto ao *dashboard* em tempo real, considera-se que o tempo de *refresh* é adequado (tendo em conta o período de atualização dos dados) e a informação apresentada é bastante útil.

Em suma, concluí-se que o trabalho desenvolvido se encontra bastante favorável e cumpre todos requisitos pretendidos pela equipa docente no contexto da problemática em questão e que foi desenvolvido, com sucesso, um sistema capaz de recolher e gerar informação, assim como criar vários *dashboards* que possiblitam a apresentação de informação de forma clara e concisa, podendo aplicar filtros sobre a informação armazenada e, deste modo, analisar a informação com um nível de granularidade superior. Outro ponto favorável foi a implementação de uma arquitetura que permitisse efetuar a monitorização em tempo real dos incidentes atuais, visto que é uma parte importante destes sistemas. Por último, o modelo desenvolvido apresentou resultados muito positivos, sendo capaz de prever, aproximadamente, os incidentes que iriam ocorrer no dia seguinte.