Sistemas Autónomos

Sensorização Ambiente aplicada à Segurança Rodoviária

José Parpot Nº: PG41848, Raimundo Barros Nº: PG42814, Tiago Gonçalves Nº: PG42851, and André Soares Nº: A67654

Universidade do Minho, R. da Universidade, 4710-057 Braga, Portugal

Abstract. Com o aumento exponencial da densidade populacional nos grandes centros urbanos, tem-se aumentado a demanda por soluções inteligentes que permitem gerir e resolver, de forma eficiente e eficaz, todos os problemas subjacentes a estas cidades. Dessa forma, a segurança rodoviária tornou-se num crescente ponto de preocupação, sendo facilmente explicado pelo número substancial de acidentes e fatalidades nas rodovias. O presente trabalho tem como objectivo a construção de uma plataforma, cuja a base visa a conceção e implementação de técnicas de sensorização de ambientes tirando partido da integração de sensores virtuais (*Traffic API - TomTom*), focando domínios emergentes como a *Internet of Things* (IoT) ou as *Smart Cities*, sendo capaz de obter dados e gerar informação útil sobre o ambiente onde se encontra inserido. Portanto, a plataforma *Web* deve ser capaz, no âmbito do *Smart Cities*, ser capaz de responder a vários problemas, incluindo a segurança rodoviária, coletando dados através da API, aplicando técnicas de fusão sensorial, exibindo o resultado através do *Data Visualization* e fazendo uso de *Machine Learning* capazes de prever fenómenos futuros, com foco principalmente em incidentes na região de Braga e Porto..

Keywords: Sensorização Ambiente · Aplicação *Web* · *Machine Learning* · *Data Viz*

1 Introdução

Este projeto enquadra-se na área de sensorização, no sentido em que temos que recolher informação do ambiente e criar uma plataforma que, através de sensores virtuais para obtenção de dados, realize análises nestes dados e obtenha resultados, respondendo a algumas possíveis questões.

O relatório esta estruturado em três partes relacionados com o problema proposto. Na primeira parte está explicado as decisões tomadas pelo grupo em relação a recolha e tratamento dos dados para posteriormente ser usado para a previsão dos incidentes. Na segunda fase do projeto esta presente a construção das páginas web onde é apresentado os incidentes e as previsões em diferentes formatos. Finalmente na ultima parte do projeto é demonstrado os processos e métodos usados para a previsão realizada pelo grupo.

2 TOMTOM Traffic Incidents

Este trabalho prático incide sobre um sensor virtual, mais concretamente, na API da TOMTOM que com uma chave de API e definindo uma localização com latitudes e longitudes podemos fazer pedidos por HTTP que nos dá, em formato JSON, informação sobre incidentes rodoviários:

```
https://baseURL/traffic/services/versionNumber/incidentDetails
?key={API_KEY}&bbox={minLon},{minLat},{maxLon},{maxLat}
&fields=string&language=string&t=Traffic_Model_Id
```

Utilizando uma chamada à API recebemos muitos dados relativos aos incidentes na área definida como por exemplo o tipo de incidente (obras, trânsito, avaria automóvel, etc), a magnitude do atraso provocado pelo incidente (baixo, médio, alto), horas de início e fim do incidente, locais de início e fim, comprimento em metros, número de ruas afetadas e muitas mais informações.

2.1 Área de Estudo

Como dito anteriormente tivemos que definir uma área de estudo à qual as informações recolhidas vão pertencer. No nosso projeto recolhemos dados de dois sítios: o centro de Braga e Porto/Gaia. No entanto, para a página web que diz respeito ao tratamento de dados e visualização dos mesmos apenas usamos dados recolhidos no Porto/Gaia. Podemos ver em seguida as áreas definidas.

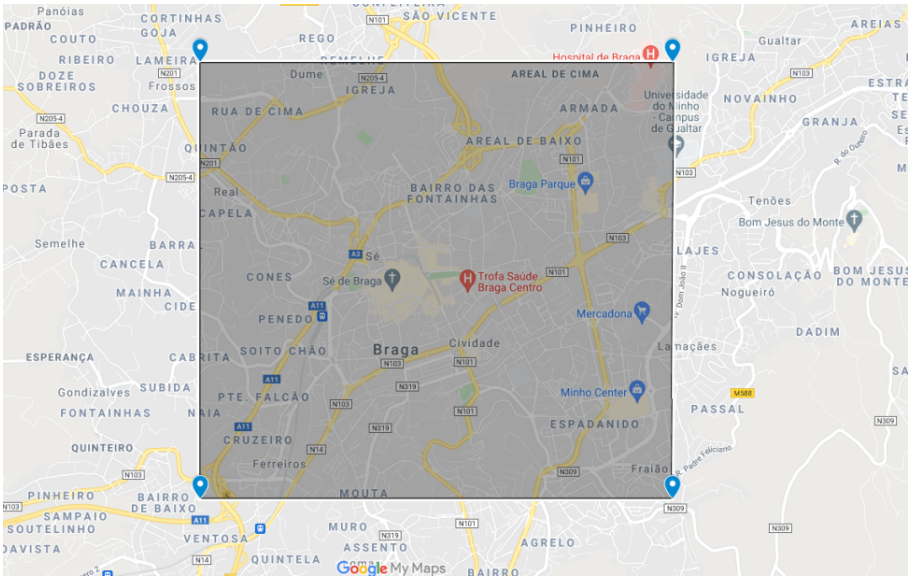

Fig. 1. Area Braga

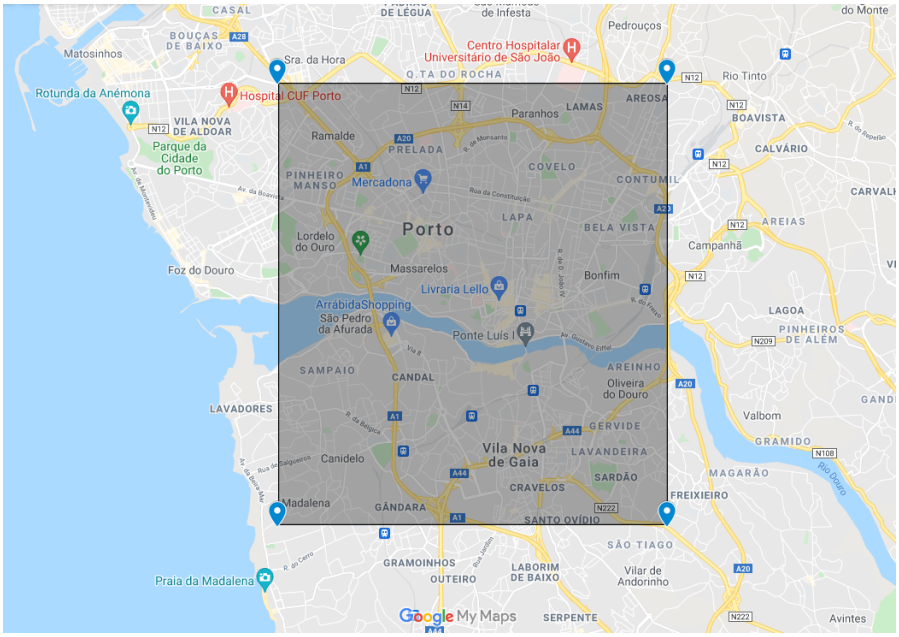

Fig. 2. Area Porto

2.2 Recolha de Dados

Inicialmente a forma de recolher dados não era automatizada. Da maneira que a nossa aplicação estava construída o utilizador tinha que pressionar um botão na página web para ser feita a recolha de informação. Como houve necessidade de automatizar este processo e aproveitando que estavamos a utilizar javascript foi utilizado o CRON que é um ferramenta que permite executar algo de X em X tempo. Para página web os dados foram coletados de 10 em 10 minutos durante 6 dias (10 a 16 de Maio) o que se traduziu em 867 leituras diferentes.

Quanto aos dados utilizados para o modelo de previsão, os dados eram recolhidos em um intervalo de 5 minutos através de um *script* feito em Python, em que este acedia o API *Rest* referente a cada sítio (Braga e Porto) e guardava seus valores em um ficheiro CSV. O *script* correu um pouco de 1 semana e em seguida seus dados foram tratados no Knime para remover os duplicados, em seguida foram pré-processados para serem utilizados no modelo de séries temporais e de classificação.

3 Página Web

Uma vez que a recolha de dados entrou em andamento, podemo-nos focar em como e onde queríamos tratá-los para mostrar ao utilizador. Neste aspecto, foi desenvolvida uma aplicação em JavaScript utilizando NodeJS. Tendo em conta que as informações recolhidas estão num ficheiro de texto, o objetivo é que seja feito um POST desses dados que depois possam ser recolhidos para tratar e mostrar.

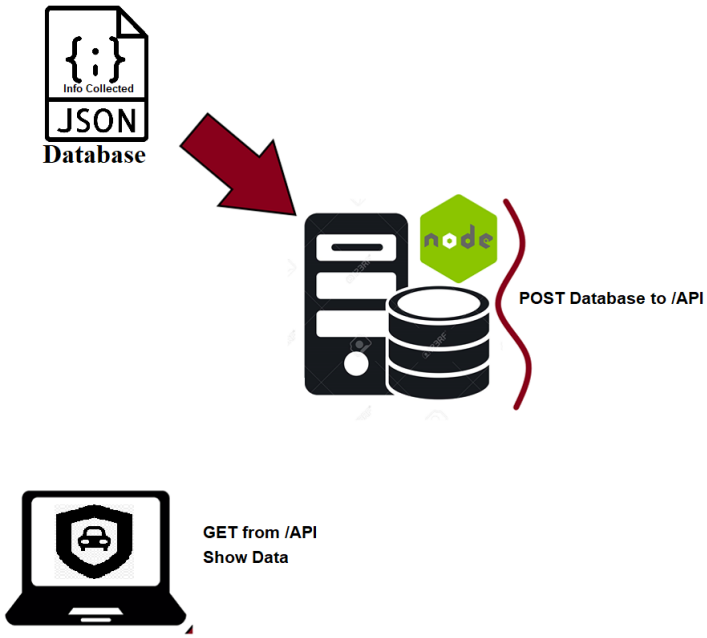

Fig. 3. Arquitetura

3.1 Métricas

Tendo acesso na nossa aplicação a todos os dados que foram recolhidos podemos iterá-los para chegar a informações relevantes de interesse ao utilizador. Ao entrar na página web, são imediatamente disponibilizadas métricas tais como: o número total de leituras, em quantos incidentes essas leituras se traduziram e quantos desses são únicos. É nos dada também informação sobre a hora em que foi detetada mais incidentes, o tempo médio de atraso provocado pelos incidentes e os dados dos incidentes com maior comprimento (em metros) e maior atraso provocado (em segundos). Podemos ver em baixo um screenshot destes parâmetros no resultado final do projeto:

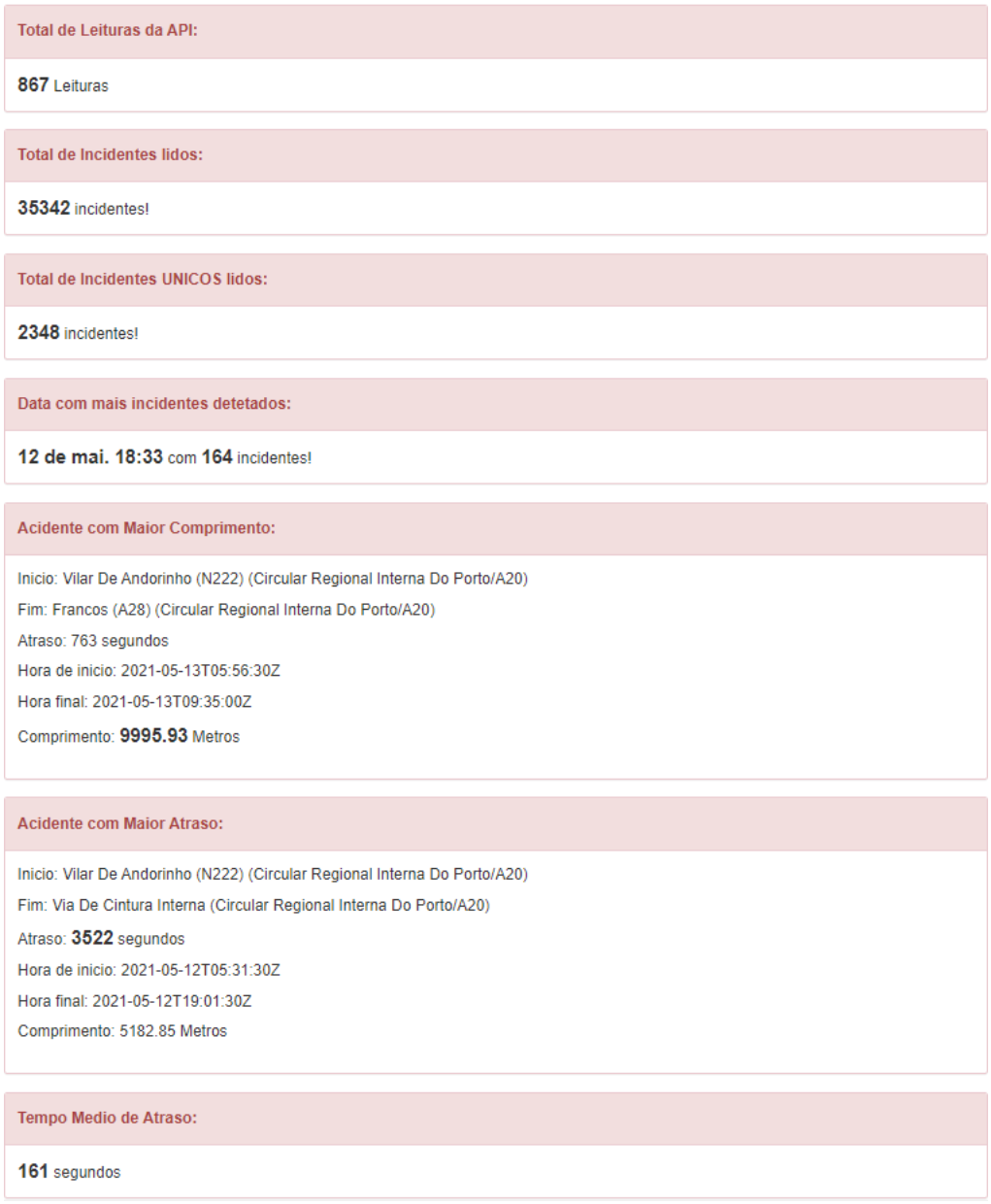

Fig. 4. Métricas estudadas

3.2 Mapas

Depois de termos feito a análise dos dados mostrando-os em métricas simples como observado no tópico anterior partimos então para o desenvolvimento e para a pesquisa de ferramentas que nos permitissem ter Data Visualization na página, seja através de gráficos, mapas, etc. Neste sentido, para representar mapas, utilizamos uma libraria open-source de javascript chamada LeafletJS que nos permitiu gerar mapas.

Criamos 3 mapas diferentes, o primeiro, diz-nos apenas a informação atual da área em estudo, no mapa são povoados um ponto por cada incidente detetado com simbolos diferentes (ruas fechadas tem um icon de barreira vermelha enquanto um incidente normal tem um icone que é um carro azul). Podemos clicar em cada ponto e é-nos mostrada informação sobre esse incidente.

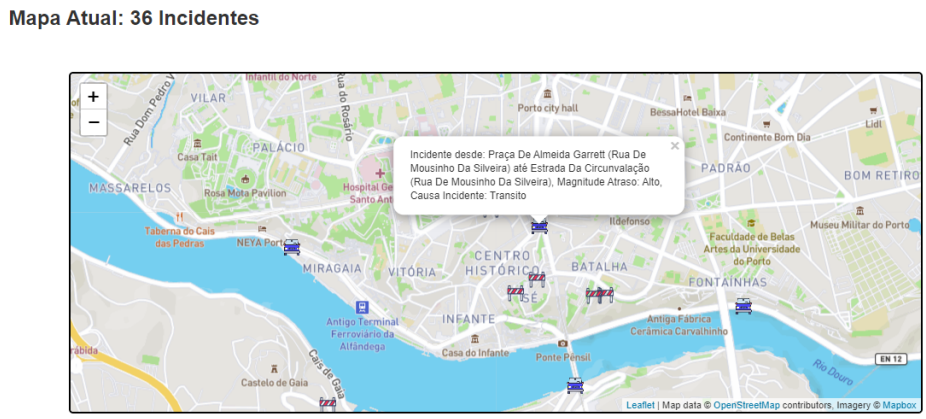

Fig. 5. Mapa atual

Em seguida, tendo consciência que o objetivo é fazer uma análise global dos dados recolhidos e não apenas do estado atual do sistema, desenvolvemos uma mapa que pinta todas as ruas afetadas durante este período de recolha de dados, mais uma vez pintamos os incidentes que dizem respeito a ruas/vias fechadas de vermelho e utilizamos azul para os outros incidentes:

Mapa Total

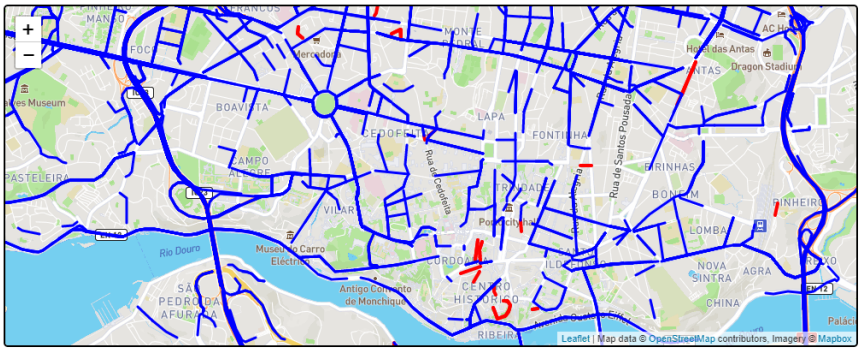

Fig. 6. Mapa total ruas

Visto que este mapa não nos diz o número de vezes que houve um incidente em rua X (uma rua onde houve um incidente é pintada da mesma maneira que uma rua onde houveram 100 incidentes), foi desenvolvido um mapa de calor. Fazendo zoom no mapa podemos observar bem onde foram os locais com mais incidência.

Mapa de Calor

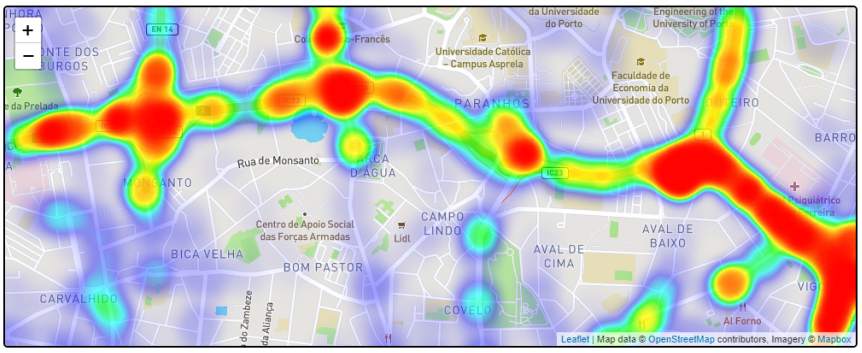

Fig. 7. Mapa total ruas

3.3 Gráficos

Após a construção dos mapas, partimos para o desenvolvimento de gráficos que permitissem visualizar dados de forma simples e intuitiva. Fizemos 3 tipos de

gráficos, um que establece a relação tempo-número de incidentes que tem 867 pontos que correspondem a todas as chamadas feitas à API. Fizemos também dois Pie Charts que nos dão informação sobre o tipo de incidente e a magnitude do atraso provocado pelos incidentes. Para isto, utilizamos a ferramenta ChartJS.

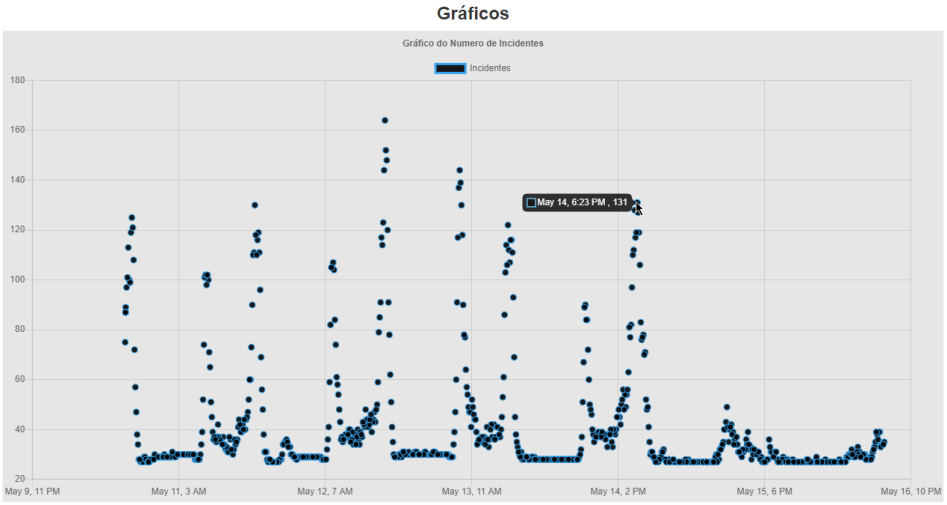

Fig. 8. Gráfico Geral

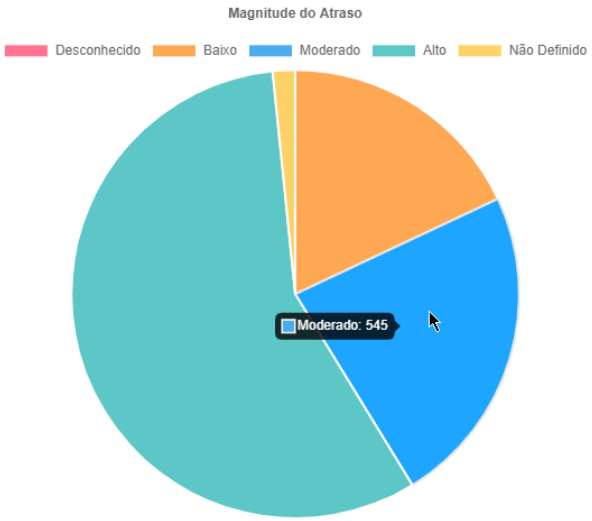

Fig. 9. Pie Chart Magnitude Atraso

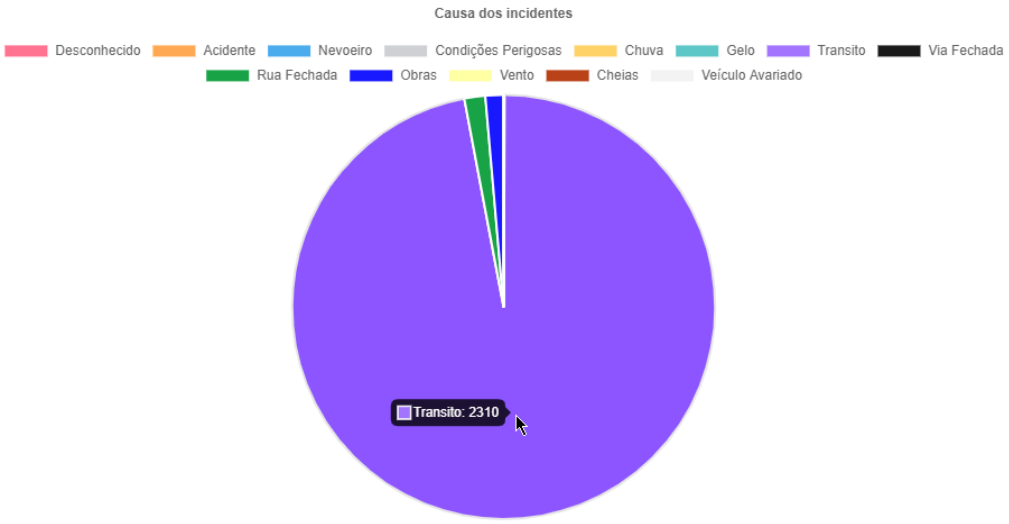

Fig. 10. Pie Chart Tipo Incidente

4 Previsão

A indústria de infraestrutura e de tráfego estão cada vez mais a desenvolver modelos e soluções para mitigar problemas relativamente a incidentes nas rodovias das grandes cidades. Os padrões envolvidos em incidentes podem ser detectados desenvolvendo modelos de previsão precisos, capazes de classificação automática informando o nível da magnitude dos incidentes ou prever a quantidade de incidentes em um determinado intervalo de tempo. Esses padrões comportamentais e de incidentes rodoviários podem ser úteis para desenvolver políticas de controle de segurança no trânsito. Acreditamos que, para obter os maiores efeitos possíveis no controle de incidentes com recursos orçamentários limitados, é importante que as medidas sejam baseadas em levantamentos científicos e objectivos das causas dos incidentes, e da gravidade destes. Esta secção resume o desempenho de dois paradigmas de aprendizado de máquina aplicados à modelagem dos incidentes informando a magnitude do *delay* associado com os incidentes e a número de incidentes durante um determinado período. Para tais abordagens consideramos redes neuronais treinadas para séries temporais, e XGBoost para classificação.

Os modelos preditivos de *Machine Learning* utilizam dados e algoritmos estatísticos para identificar a probabilidade de acontecimentos futuros a partir de dados históricos. A utilização dos modelos citados nas próximas secções permitem resolver problemas difíceis e descobrir novas oportunidades, sendo cada vez mais utilizados em diversas áreas.

Numa primeira fase da resolução do problema tivemos que recolher os dados do API *Rest* através do TomTom para armazená-los em um *dataset* e posteriormente remover os dados duplicados. Em seguida se fez necessário analisar e tratar o *dataset*. Desta forma, é importante visualizar a distribuição do *dataset* de modo a interpretar a sua relação com a variável *target*.

Ao longo do processo de tratamento de dados, os diferentes modelos preditivos foram treinados, para perceber, através de métricas de avaliação de performance, se o modelo se encontra validado.

4.1 Séries Temporais

De modo a implementar a previsão de eventos futuros, transformaram-se os *datasets* criados através da sensorização, em séries temporais.

Utilizando a ferramenta *Knime*, foram então criadas séries temporais em que se agrupou os dados em espaços de tempo de uma hora. Sendo séries temporais multi-variáveis foram utilizadas como colunas de *input*:

- média da magnitude do *delay*
- média do tamanho dos incidentes em metros
- média do tempo do *delay* em segundos
- moda do tipo de evento provocador dos incidentes
- moda das ruas afetadas pelos incidentes

– média dos valores numéricos de *label* atribuídos às duas primeiras coordenadas de cada incidente (os valores das labels são oriundos de um agrupamento dos dados feito com base num arredondamento dos valores de coordenadas a apenas três casas decimais, o que acaba por criar grupos em pequenas regiões geográficas)
– média da diferença temporal entre as datas de início e fim atribuídas aos incidentes
– dia da semana em que ocorreram os incidentes
– média da hora do dia em que ocorreram os incidentes (isto devido ao início da sensorização não corresponder com o início exato de uma hora e haverem casos em que a hora a que ocorrem os incidentes de um grupo não terem exatamente o mesmo valor de hora)

Como output das séries temporais, ou seja, como objetivo a ser previsto pelos modelos *Machine Learning*, é a coluna com o número total de incidentes que ocorreram a cada hora.

Relativamente ao modelo de *Machine Learning*, foi implementado um modelo bastante simples de *Multi-Layer Perceptron*, que consiste em três camadas com 200 unidades cada.

Com este modelo, os resultados obtidos foram satisfatórios. Para este exemplo o resultado esperado era 1 e o obtido foi 0,9916405.

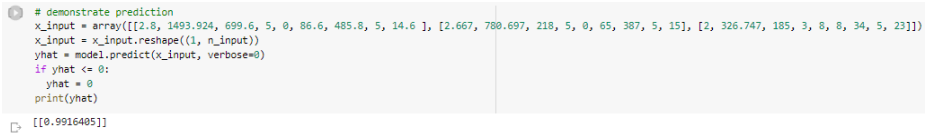

Fig. 11. Resultado do modelo de previsão de Séries Temporais

4.2 Classificação - Magnitude do *Delay*

Esta secção visa prever a intensidade do atraso associado com o incidente registado na cidade de Braga e Porto, podendo os valores serem representados conforme indicado abaixo:

– 0: Desconhecido
– 1: Pequeno
– 2: Moderado
– 3: Grande
– 4: Indefinido (usado para bloqueios de estradas e outros atrasos indefinidos.

Primeiramente, se fez necessário pré-processar os dados do *dataset* removendo os atributos que não são necessários para a fase de previsão, como *startTime*, *endTime*, pois foram utilizados a data e a hora em que os dados foram colectados da API *Rest*, e *coordinate* pois as coordenadas abrange uma região fixa do Porto e

Braga. Em seguida, se fez necessário transformar as variáveis em tipo categórico e numérico, e converter alguns atributos (*events* e *roadNumbers*) em inteiros ordinais para que estes possam ser processados mais a frente. Por fim, um novo atributo (*period*) foi criado, combinando a variável *date* e *time*, e em seguida transformando do tipo string para o tipo *datetime* permitindo manipulá-lo e utilizá-lo na criação e validação do modelo.

	iconCategory	magnitudeOfDelay	length	delay	roadNumbers	events	site	day	events_code	road_code	period	day_code	Hour	Minute
0	6	2	633	109	['N103']	Queuing traffic	Braga	Friday	2	3	2021-05-07 18:45:13	4	18	45
1	6	2	1754	172	[]	Queuing traffic	Braga	Friday	2	4	2021-05-07 20:25:25	4	20	25
2	6	3	252	233	[]	Stationary traffic	Braga	Friday	4	4	2021-05-07 20:30:26	4	20	30
3	8	4	136	0	[]	Closed, Roadworks	Braga	Friday	1	4	2021-05-07 20:35:27	4	20	35
4	8	4	128	0	[]	Closed, Roadworks	Braga	Saturday	1	4	2021-05-08 06:16:55	5	6	16

Fig. 12. *Dataframe* Pré-Processado - Braga

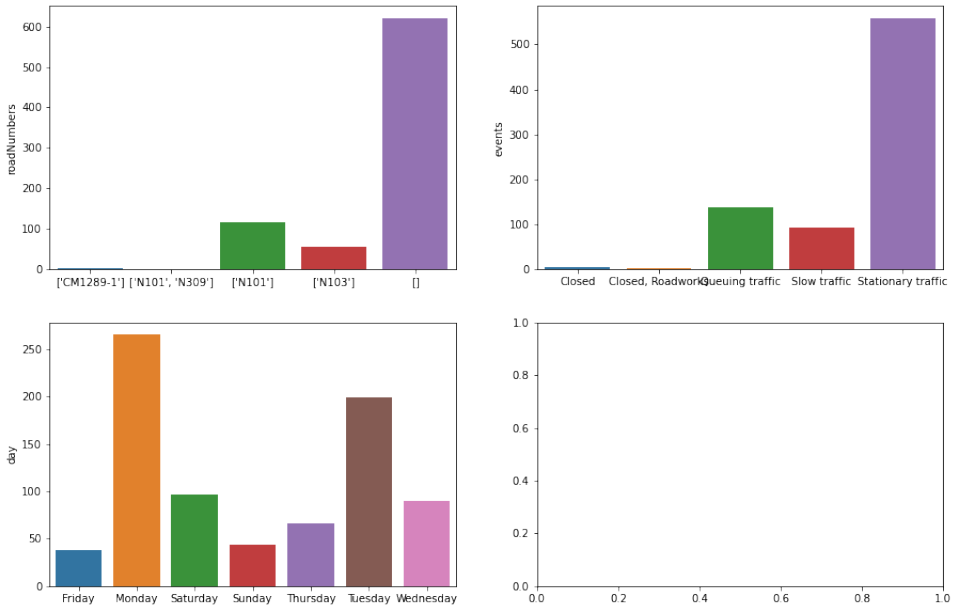

Fig. 13. Gráfico de Barras - Braga

Com base nos dados recolhidos do API *Rest* e apresentados nos gráficos de distribuição (Fig. 14) e no gráfico de barras (Fig. 13), podemos inferir que a maioria dos incidentes são engarrafamentos com intensidades de atraso alta, em que o *delay* é de 156 segundos (2 minutos e 36 segundos) com um comprimento médio de aproximadamente 300 metros. Em adição, podemos observar também que o dia que possui uma maior quantidade de incidentes é na segunda-feira e os horários que costumam ter grandes frequências de incidentes são as 8hr da manhã, 13hr da tarde e 18hr-19hr da noite, que são considerados os horários de maior movimento de tráfego.

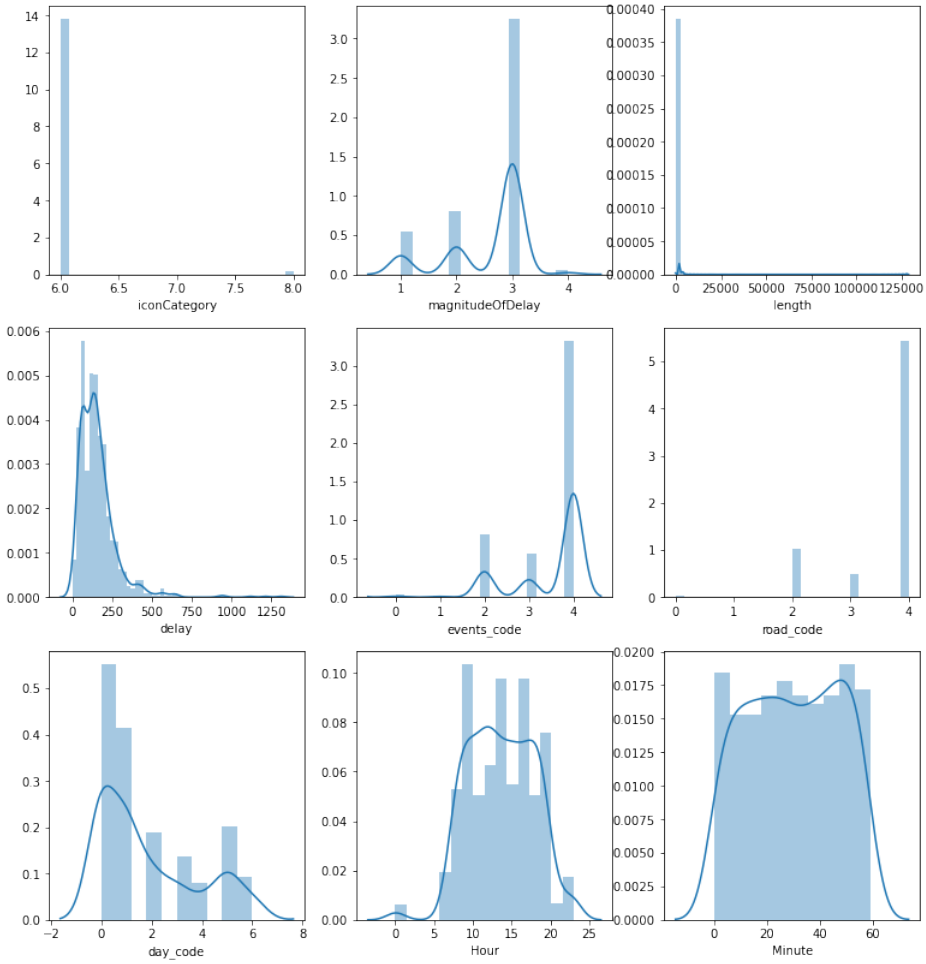

Fig. 14. Gráficos de Distribuição - Braga

Através do "mapa de calor" (*heatmap*) Fig. 15 podemos verificar que há uma grande correlação entre o *magnitudeOfDelay* e os eventos, isso faz grande sentido, pois, o valor do *magnitudeOfDelay* é um resumo dos detalhes dos eventos que ocorrem dentro de um determinado intervalo de tempo. Contudo observa-se que a categoria dos incidentes é inversamente proporcional aos eventos.

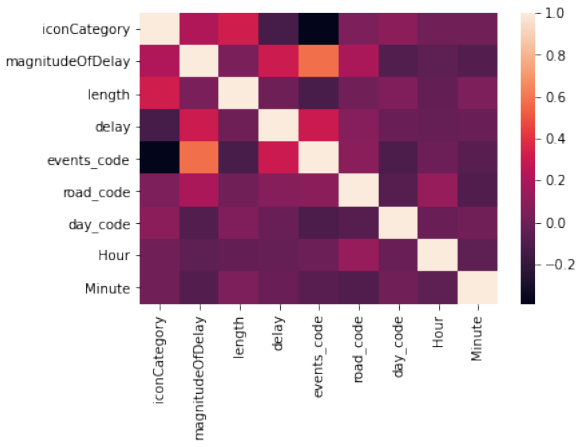

Fig. 15. Gráficos Mapa de Calor - Braga

Após correr a análise exploratória dos dados, foi criado um *dataset* com as variáveis preditoras (*length*, *delay*, *day_code*, *road_code*, *Hour* e *Minute*) e utilizado como variável alvo o *magnitudeOfDelay*. Para a aplicação dos modelos foi feita a separação do *dataset* em parte de treino e parte de teste, com uma divisão de 70% para o de treino e de 30% para o de teste.

Por fim, depois de realizados os testes utilizando *Gaussian Naive Bayes*, *Logistic Regression*, *Decision Tree*, *Random Forests* e *XGBoost*, e obtida a *accuracy* de cada modelo, decidimos comparar com o propósito de encontrar o modelo que apresente a melhor performance.

	Algoritmo	training_score	accuracy_score
0	LogisticRegression	0.849732	0.804167
1	Gaussian Naive Bayes	0.772809	0.720833
2	Decision Tree	1.000000	0.845833
3	Random Forests	1.000000	0.870833
4	XGBoost	1.000000	0.887500

Fig. 16. Gráficos Mapa de Calor - Braga

De acordo com a tabela contida na Fig. 16, podemos concluir que em termos de *accuracy* o melhor modelo é o de *XGBoost* com aproximadamente 88.75% (Braga) e 77,78% (Porto), essa diferença se dá pela quantidade de dados recolhidos em cada sítio. Contudo, o resultado obtido foi consistente entre a fase de treino e a fase de teste e obteve os melhores resultados gerados na matriz confusão.

Para exemplificar a aplicação do modelo, foi inserido novos valores das variáveis conforme consta na Fig. 17 e o resultado devolvido pelo modelo foi 2 (Moderado).

	length	delay	day_code	road_code	Hour	Minute
0	200	50	6	5	12	45

Fig. 17. Input dos dados para obter um resultado do modelo - Braga

5 Conclusão

O tratamento de dados vindos de sensores físicos ou virtuais é algo poderoso que pode ser usado em vários aspectos da nossa vida, principalmente com a integração de modelos de aprendizagem de máquinas.

Em conclusão, o grupo sente que conseguiu não só atingir os objetivos colocados pela equipa docente como também expandir o nosso conhecimento sobre esta área. Ao longo do projeto encontramos alguns problemas, mas depois de realizado pesquisa e testes conseguimos encontrar soluções para os problemas e concluir o projeto.

Em termos de melhoramento do projeto, num trabalho futuro, o grupo era capaz de adicionar mais funcionalidades e formas de visualização para as previsões criadas. Relativamente aos modelos de previsão o grupo poderia ter explorado mais o modelo de series temporais e classificação e ter integrado estes a aplicação Web, de forma que o usuário pudesse inserir os valores relativos ao sítio, data e hora, e obter como resultado o número de incidentes e a intensidade do atraso.

References

1. Leafletjs, https://leafletjs.com/. Último acesso 16 Mai 2021.
2. ChartJS, https://www.chartjs.org/. Último acesso 16 Mai 2021.
3. neDB, https://github.com/louischatriot/nedb. Último acesso 16 Mai 2021.
4. Forecasting of periodic events with ML, https://towardsdatascience.com/forecasting-of-periodic-events-with-ml-5081db493c46. Último acesso 15 Mai 2021.
5. TOMTOM - Incident Details, https://developer.tomtom.com/traffic-api/traffic-api-documentation-traffic-incidents/incident-details. Último acesso 16 Mai 2021.
6. How to Develop Multilayer Perceptron Models for Time Series Forecasting, https://machinelearningmastery.com/how-to-develop-multilayer-perceptron-models-for-time-series-forecasting/. Último acesso 16 Mai 2021.