



UNIVERSIDADE  
**LUSÓFONA**

# Otimização de recursos através de análise de chamadas de emergência

## **Trabalho Final de curso**

Relatório Final

Nome do Aluno: Tomé Roque

Equipa de Orientação: Daniel Fernandes, Gabriela Soares

Trabalho Final de Curso | LEI | 28 de junho de 2024

[www.ulusofona.pt](http://www.ulusofona.pt)

## **Direitos de cópia**

DEISI39, Copyright de Tomé Roque, Universidade Lusófona.

A Escola de Comunicação, Arquitectura, Artes e Tecnologias da Informação (ECATI) e a Universidade Lusófona (UL) têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objectivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

---

## Resumo

Com efeito, este relatório descreve todo o trabalho realizado durante o ano letivo de 2023/2024, sendo que este foi montado sobre os pilares que foram construídos por um primeiro trabalho final de curso, realizado no ano anterior (2022/2023) por Tiago van Krieken. Este primeiro trabalho foi intitulado de Estudo de Chamadas de Emergência do INEM.

Em suma, o trabalho antecessor recolheu dados sobre o número de chamadas diárias recebidas pelo Centro de Orientação de Doentes Urgentes do INEM, sendo este nome reduzido à sigla CODU. Feita esta recolha, investigaram-se várias fórmulas matemáticas estipuladas para prever quantas chamadas seriam feitas em meses futuros. Efetivamente, o trabalho deu frutos e foi possível prever o número de chamadas efetuadas para o CODU, durante os meses de janeiro, fevereiro e março de 2023 com margem de erro de apenas um por cento para dois desses meses e quatro por cento para o mês de fevereiro.

Consequentemente, o trabalho seguinte terá o objetivo de pegar nos desenvolvimentos estatísticos e matemáticos desenvolvidos pelo antecessor, avaliar os resultados obtidos, por exemplo avaliar o porquê de o mês de fevereiro ter uma margem de erro quatro vezes superior aos outros dois, e em caso de necessidade, afinar o algoritmo para que não haja discrepâncias desta natureza.

Assim, este projeto está dividido em três capítulos, sendo o primeiro descrito pelos parágrafos anteriores, onde existe todo um proveito retirado do trabalho que estipulou as fundações para este.

Feito isto, segue-se a criação do motor que será utilizado para tirar o projeto do papel. Decerto, este simbolismo é utilizado para resumir o processo que será composto por uma construção de uma API que irá recolher os números de chamadas em intervalos de 30 minutos. Posteriormente, serão estabelecidos algoritmos utilizando as equações recolhidas pelo trabalho anterior para realizar o tratamento dos mesmos para que esta mesma API forneça o número de chamadas esperadas para uma data que lhe será fornecida, e com estes valores apontar o número de operadores que deverão estar de serviço para que todas as chamadas recebidas possam ser respondidas.

Por fim, o último ponto deste projeto é a criação de uma aplicação, que fará a ponte entre utilizador e algoritmo, onde dará a possibilidade ao utilizador de introduzir uma determinada data, e através da API estabelecida, a aplicação irá formular todo um relatório sobre o intervalo de tempo introduzido.

## Abstract

In effect, this report describes the work carried out during the 2023/2024 academic year, which was built on the pillars of another final coursework, carried out in the previous year (2022/2023) by Tiago van Krieken, entitled “*Estudo de Chamadas de Emergência do INEM*”.

In short, the previous work collected data on the number of daily calls received by INEMs *Centro de Orientação de Doentes Urgentes*, reduced to the acronym CODU. After this collection, several mathematical formulas were investigated to predict how many calls would be made in future months. Effectively, the work bore fruit and it was possible to predict the number of calls made to CODU, during the months of January, February, and March 2023 with a margin of error of just one percent for two of those months and four percent for the month of February.

Consequently, the following work will have the objective of taking the statistical and mathematical developments developed by the predecessor, evaluating the results obtained, for example evaluating why the month of February has a margin of error four times higher than the other two, and if necessary, fine-tune the algorithm so that there are no discrepancies of this nature.

Therefore, this project is divided into three chapters, the first being described by the previous chapter, where there are benefits taken from the work that laid the foundations.

Next follows, the creation of the engine that will be used to get the project off the ground. In fact, this symbolism is used to summarize the process that will consist of building an API that will collect call numbers at 30-minute intervals. Subsequently, algorithms will be established using the equations collected by previous work to process them so that this same API provides the number of calls expected for a date that will be provided, and with these figures indicate the number of operators that must be working so that all incoming calls can be answered.

Finally, the last point of this project is the creation of an application, which will bridge the gap between user and algorithm, where it will give the user the possibility to enter a certain date, and through the established API, the application will formulate an entire report on the entered time interval.

---

# Índice

Resumo.....	iii
Abstract .....	iv
Índice .....	v
Lista de Figuras .....	vii
Lista de Tabelas .....	viii
1 Identificação do Problema .....	1
2 Benchmarking.....	3
3 Viabilidade e Pertinência.....	6
4 Solução Proposta.....	10
5 Calendário .....	13
6 Aperfeiçoamento do Algoritmo .....	14
6.1 Seleção do modelo.....	15
6.2 Engenharia de features.....	18
6.3 Implementação de lags.....	21
6.4 Aumento da granularidade.....	22
6.4.1 Previsões diárias .....	23
6.4.2 Previsões horárias .....	26
6.4.3 Previsões em turnos .....	28
6.5 Resultados desta fase .....	29
7 Estabelecimento de uma API .....	30
7.1 Criação de uma base de dados .....	31
7.1.1 Docker.....	31
7.1.2 Controladores.....	33
7.1.3 Adaptação do Código .....	34
7.1.4 Organização da BD.....	35
7.2 Instalação de uma API.....	37
7.2.1 API Pythonanywhere .....	37
7.2.2 API Docker .....	38
8 Aplicação INEM Insight.....	41
8.1 Identidade da aplicação .....	41

8.2	Log in.....	42
8.3	Previsões customizáveis .....	43
8.3.1	Relatório PDF.....	43
8.3.2	Relatório Excel .....	45
8.4	Dashboard.....	46
9	Resultados .....	48
	Bibliografia .....	49
	Glossário.....	51
	Anexo I – Ficheiro de valores previstos.....	52
	Anexo II – Código referente ao algoritmo.....	53
	Anexo III – Relatório por turnos criado pela aplicação .....	54
	Anexo IIII – Ligações relevantes .....	55

---

## Lista de Figuras

Figura 1 - Reta de Regressão Polinomial de fevereiro.....	7
Figura 2 - Reta de Regressão Polinomial de março .....	7
Figura 3 - Cronograma em formato Gantt.....	13
Figura 4 – Código a ser inserido na bash do <i>container Docker</i> .....	32
Figura 5 - Instalação do watchdog no <i>container Docker</i> .....	33
Figura 6 - Código para observar e notificar quando alguma alteração é feita .....	34
Figura 7 - <i>Container</i> do <i>Docker</i> utilizado pelo projeto.....	35
Figura 8 - Função recolhe_dados da API .....	38
Figura 9 - Permutações aplicadas aos valores recolhidos .....	39
Figura 10 - Utilização do software Postman para teste da API .....	40
Figura 11 - logotipo do site.....	42
Figura 12 - Página de criação de conta .....	42
Figura 13 - Página de previsões personalizadas .....	43
Figura 14 - Inicialização da biblioteca e do documento PDF .....	43
Figura 15 - Código para a inserção da tabela no PDF .....	44
Figura 16 - Código para gerar relatórios Excel.....	45
Figura 17 - Exemplo de um relatório em Excel .....	45
Figura 18 - código da função para cálculo do número de operadores necessários para um número de chamadas previstas .....	47
Figura 19 - Dashboard INEM Insight.....	47

## Lista de Tabelas

Tabela 1 - Resultados da previsão mensal do modelo de <i>Machine Learning</i> .....	6
Tabela 2 - Previsões de maio a setembro 2023 com dados previstos.....	14
Tabela 3 - Resultados obtidos com algoritmo mês/ano e <i>lag</i> de 7 .....	17
Tabela 4 - Algoritmo com várias <i>features</i> .....	19
Tabela 5 - Chamadas recebidas pelo CODU em agosto dos últimos 5 anos.....	19
Tabela 6 - previsões do algoritmo de <i>features</i> com a utilização de <i>lags</i> .....	21
Tabela 7 - Previsões diárias para o mês de maio de 2023.....	25
Tabela 8 - Algumas colunas retiradas do algoritmo de 30 minutos .....	27
Tabela 9 - Excerto de dados previstos agrupados em turnos.....	28



# 1 Identificação do Problema

De facto, o Homem é um ser muito frágil. Com efeito, esta afirmação torna-se cada vez mais óbvia a cada dia que nos deparamos com casos de problemas de saúde ou até de violência. Contudo, há sempre uma linha de apoio com que podemos contar para algum tipo de emergência desta natureza, sendo esta o Número Europeu de Emergência **[112]**, mais popularmente conhecido por 112.

Efetivamente, ao fazer uma chamada para a linha de apoio, ela será, numa primeira fase, atendida pela PSP ou GNR nas centrais de emergência, onde posteriormente serão avaliadas tendo em conta a natureza da emergência, podendo esta ser de uma emergência de saúde, de perigos variados. De seguida, sendo feita esta avaliação, caso se conclua que o tipo de emergência é de facto a nível de saúde, a chamada é então reencaminhada para os Centros de Orientação de Doentes Urgentes (CODU) do INEM.

Deste modo, podemos considerar que este serviço oferecido pelo INEM, é um bem essencial para a sociedade sendo que este permite uma melhor intervenção por parte dos profissionais de saúde. E mais importante que tudo, um apoio até à chegada de meios diferenciados. Com efeito, este apoio consiste na transmissão de indicações para minimizar possíveis sequelas ao acidentado. Estas indicações surgem após uma correta perceção do problema por parte do operador.

Efetivamente, toda a descrição feita demonstra o caso ideal onde a comunicação acontece, contudo, e tal como todos os sistemas onde o principal elemento é o Homem, este é limitado aos recursos que estão disponíveis. Isto é, se num determinado dia não estão de serviço pessoas suficientes para a quantidade de chamadas que são reencaminhadas para o CODU, as chamadas que sobrecarregarão os recursos não serão atendidas, e isto num pior caso poderá levar a que uma pessoa não seja socorrida, o que poderá levar mesmo à sua morte.

Da mesma forma, tal como ter um número de operadores em serviço num dado turno é um problema enorme, o mesmo pode ser dito do contrário. Na verdade, uma má gestão dos funcionários num sentido exagerado, irá obviamente reduzir o número de chamadas não atendidas, no entanto a um preço bastante elevado. Importa salientar que, atualmente, todo o atendimento de chamadas é feito por um ser humano, e apesar de muitos destes serem o mais próximo de heróis dentro da nossa sociedade, continuam a ser seres humanos, continuam a ter as necessidades básicas de um.

Evidentemente, isto é um fator crucial para um bom funcionamento de qualquer serviço, especialmente se este se tratar de uma central que gere o auxílio pré-hospitalar do país. Tomemos como exemplo, a reportagem que foi feita pelo canal televisivo TVI no dia 3 de novembro de 2023, onde é evidente o estado de exaustão em que os trabalhadores de todo este serviço de emergência. É de realçar que, esta reportagem transmite o estado de desgaste que pode ser notado sobre os trabalhadores desta área, passando a citar: *“Face à escassez do efetivo, face à sensibilidade e à complexidade deste serviço, os elementos (profissionais de atendimento do número 112) estão obviamente desgastados, e muitos deles acabam por ficar longos períodos de baixa”* **[REPTVI]**. Com efeito, esta reportagem não refere em específico os

operadores de chamadas de emergência que trabalham no CODU, no entanto refere-se aos seus parceiros, os agentes da PSP (Polícia de Segurança Pública) e GNR (Guarda Nacional Republicana) que trabalham na central que recebe e faz a filtragem de todas as chamadas recebidas pelo 112. Mesmo assim, isto é bastante relevante porque estamos a falar do mesmo tipo de trabalho, com as mesmas condições e os mesmos desafios, logo é uma comparação que pode ser feita.

Com certeza, uma forma fácil de resolver este problema seria a contratação de mais funcionários para aliviar a carga horária de todos, no entanto para além dos elementos financeiros que isso implicaria, é também perceptível que esse caminho não é tão viável quanto parece. Ainda sobre a reportagem anteriormente mencionada, nota-se que apesar de haver uma procura de funcionários, não existe resposta para a mesma, citando então esta reportagem: *“é que apesar de ter sido aberto um concurso para novas posições, a realidade atual exige outras respostas, além dos incentivos financeiros serem baixos”*. [REPTVI]

Então, é fulcral não sobrecarregar turnos que terão poucas chamadas, onde temos um número de operadores superior à necessidade face à quantidade de chamadas que irá chegar ao CODU. Com efeito, os turnos de trabalho são constituídos por 3 iguais partes, sendo assim cada um com 8 horas, e como é óbvio, este serviço está operacional todos os dias do mês, todos os meses do ano. Portanto, referimo-nos de pessoas que trabalham em turnos que variam de semana para semana, onde lidam com situações difíceis devido ao seu carácter de emergência, estão constantemente postos à prova devido a que, quando deparados com uma chamada, estes têm de ter a capacidade de avaliar a situação e desenvolver em um curto espaço de tempo, um plano para que o socorro seja feito da melhor forma e para garantir que esse é eficaz. Por isso, apesar de ser necessário ter um número suficiente de operadores, é também uma preocupação não ter um número maior do que aqueles que são necessários.

Assim, conclui-se que apesar de um sistema funcional, o 112 possui um fator muito frágil, o número de trabalhadores que se encontram de serviço a cada hora, de cada dia, de cada ano, no entanto não é possível prever um fator tão imprevisível quanto o número de emergências que irão acontecer no futuro. Ou será que é?

## 2 Benchmarking

De facto, temos então um problema ilustrado pela falta de funcionários em determinadas alturas devido a uma incapacidade de previsão de número de chamadas numa determinada altura.

Para além disso, temos também um primeiro trabalho onde foram utilizadas algumas fórmulas matemáticas que permitem, através de dados estatísticos de dias passados, calcular uma estimativa do número de chamadas que o CODU irá receber num determinado intervalo de tempo. De facto, este trabalho demonstrou que a partir de dados diários desde 2012, foi possível prever o número de chamadas em vários meses com uma margem de erro e um Coeficiente de determinação na casa dos 2% aos 4%. Com efeito, para alcançar estes valores foram utilizadas fórmulas e modelos matemáticos tais como a regressão linear, que utiliza elementos de *machine learning* para, através de dados de meses anteriores, consegue prever o número de chamadas para um outro determinado mês.

Contudo, apesar destes grandes avanços, este é ainda um projeto em construção, pois apesar de valores concretos, estes são ainda bastante ambíguos em termos de estudo para a solução pretendida. Isto é, apesar de sabermos quantas chamadas irão ocorrer num determinado mês, estes dados são ainda pouco úteis no apoio à gestão de turnos, estes algoritmos terão ainda de ser trabalhados para que isso possa acontecer, no sentido em que precisamos de aumentar a granularidade para obtermos os valores de intervalos mais pequenos, como dias ou idealmente em intervalos de 8 horas até intervalos de 30 em 30 minutos.

Efetivamente, este é um cenário muito mais favorável em termos práticos para o INEM pois através do número de chamadas e através de uma outra equação, esta chamada de Fórmula de Erlang [ERLANG]. De facto, este é um recurso indispensável não só para o CODU, como qualquer central de chamadas, pois ao utilizar esta fórmula, é possível estimar o número de chamadas que será realizada para essa mesma central, num determinado espaço de tempo. Efetivamente, isto ficará mais claro nos próximos parágrafos deste capítulo.

Com efeito, para este estudo, utilizaremos uma das fórmulas em específico que será a variação C, esta que se pode ver em seguida:

$$P_w(N, E) = \frac{\frac{E^N N}{N! (N - E)}}{\sum_{i=0}^{N-1} \frac{E^i}{i!} + \frac{E^N N}{N! (N - E)}}$$

Equação 1 - Fórmula de Erlang C

Decerto, apesar de complexa, esta equação pode ser explicada de forma simples pois todos os componentes são traduzidos por dados que, já são sabidos, ou que irão ser determinadas durante as melhorias que vão sendo feitas ao código base do projeto antecessor. Então, vamos dissecar esta equação e estudá-la por diferentes partes:

- Pw -> probabilidade que descreve se um utilizador terá de esperar para ser atendido;
- N -> Número de operadores em serviço;
- E -> Intensidade de tráfego em unidade de erlangs.

Assim, visto que o objetivo que este projeto almeja é que em qualquer que seja a hora ou local, que nenhuma pessoa com a necessidade de cuidados médicos não tenha o apoio que necessita devido à sua chamada não ser atendida, procuramos um valor de Pw bastante próxima de 1. Isto deve-se a que, se considerarmos o valor dessa variável como uma percentagem de um telefonema ser atendido, quanto mais próximo de 100%, ou 1 no caso da equação, determina que todos os contactos feitos para um determinado número serão atendidos.

Além disso, temos então E, que representa a intensidade do tráfego. Com efeito, é aqui que os dados de chamadas previsto por nós irá entrar em cena, onde após feita a conversão para erlangs teremos então essa intensidade, o que constituirá mais um valor para substituir na equação.

Por fim, teremos então uma última variável, que é a variável de interesse no estudo em questão, sendo esta o número de operadores em serviço, representada pela letra N. Deste modo, poderemos então indicar quantos operadores deverão estar em serviço num determinado espaço de tempo, seja ele de 30 minutos ou de 8 horas, sendo que este será apontado para um intervalo de tempo determinado pela estimativa que for fornecida.

Por outras palavras, tal como referido anteriormente, o trabalho antecessor conseguiu projetar quantas chamadas seriam efetuadas no espaço de um mês. Contudo, estes dados terão de ser ainda trabalhados para que a nossa equação possa ter utilidade para a solução que apresentamos para o problema de sobrecarga de chamadas no CODU. Neste sentido, ao aumentar a granularidade, e assim reduzir o espaço de tempo para o qual estamos a apontar uma previsão, conseguiremos idealmente definir quantos operadores serão necessários para um dia, um turno, ou até de meia em meia hora. De facto, este último representa o teto em termos de potencial deste projeto, pois este está dependente dos dados que são disponibilizados pelo INEM, e estes de facto são, na sua forma mais reduzida, realizados em intervalos de 30 minutos.

Assim, apesar de existir já o trabalho muito bem conseguido, realizado no ano letivo anterior, este representa ainda uma proposta embrionária daquilo que será a solução pretendida. De facto, para lá chegar, terá que ser ainda ajustado para garantir no mínimo, que intervalos de 8 horas serão estimados para que, assim, este algoritmo possa indicar ao INEM com a devida antecedência, quantos operadores deverão estar de serviço em cada turno, de cada dia. Deste modo, será fornecido um melhor apoio no momento do estabelecimento da escala mensal sendo que haverá melhores fundamentos nos quais os funcionários se poderão basear para fazer uma tomada de decisão, e assim alcançar o objetivo que este trabalho herda, a redução de chamadas de emergência que não conseguem ser atendidas.

Em termos de outras soluções já presentes ou a serem estudadas, para o problema que este trabalho propõe resolver, não há nenhuma em termos de conhecimento público. No entanto, sabe-se já que o 112 tem planos para integrar IA no apoio ao atendimento de

chamadas, contudo este não representa uma competição para o projeto proposto, estes até acabam por se complementar, iremos entender porquê.

De acordo com o Jornal de Notícias [JN], é expectado que em 2025 seja implementado uma conjugação humano-máquina no sistema que estamos a estudar. Isto, no sentido em que irá haver uma inteligência artificial que irá apoiar os centros operacionais do 112 no âmbito do atendimento de chamadas.

Contudo, porque é que esta IA pode ser considerada algo complementar? De facto, esta solução responde a um outro problema, sendo este a sobrecarga de chamadas recebidas nas centrais. Note-se que, e passando a citar: *“O objetivo do “voicebot 112” passa por ultrapassar alguns constrangimentos do serviço de emergência, como baixar o tempo de espera durante os picos de chamadas, reduzir a percentagem de chamadas abandonadas, que, em 2021, atingiu os 16%, e a eliminação das chamadas falsas”* (Rita Neves Costa, Jornal de Notícias 2023), ou seja, esta inteligência artificial irá apenas filtrar chamadas, não irá realizar escolhas ou atribuir equipas a utentes.

Assim, com a implementação da IA a gerir as chamadas que chegam ao 112, problemas como uma grande afluente de chamadas são resolvidos, acontecimentos estes que ocorrem quando há algum acidente, ou se dá um evento que envolva muitas pessoas, o que leva a que muita gente realize chamadas sobre o mesmo assunto, atuando assim como uma cortina de fumo para contactos de emergência que não estejam relacionados com o mesmo. Por outro lado, este trabalho final de curso almeja que todas as pessoas que necessitem de ser socorridas e que liguem para o 112, consigam receber a ajuda que precisam. Da mesma forma, podemos afirmar que este trabalho pretende que nenhuma chamada realizada para o 112 fique por ser atendida, contudo esta inteligência artificial não pretende substituir o fator humano, sendo que este é ainda único na sua capacidade de decisão. De facto, o “voicebot 112” pretende reduzir a afluência de chamadas sobre o mesmo filtrando todas as chamadas que forem feitas sobre um caso que já tenha sido respondido ou até mesmo chamadas falsas que infelizmente é um problema real.

### 3 Viabilidade e Pertinência

Com efeito, para avaliar a viabilidade deste projeto, teremos de notar todos os avanços que foram feitos pelo primeiro trabalho e como é que este vai continuá-lo e no fundo torná-lo num produto que possa ser utilizado para de facto haver um impacto nesta área que já vimos ser uma grande necessidade.

Efetivamente, os resultados herdados demonstram que esta proposta tem pernas para andar tendo em conta que estes provaram ser bastante positivos. Dito isto, faremos então uma investigação nos valores obtidos anteriormente:

Mês	Número Real de Chamadas Recebidas	Número Previsto de Chamadas Recebidas	Erro Médio Absoluto Percentual (EMAP)	Coefficiente de Determinação (R <sup>2</sup> )
Janeiro de 2023	129 474	126 600	2,22%	0.677
Fevereiro de 2023	118 027	126 991	7,60%	0.685
Março de 2023	127 016	126 953	0,50%	0.680
Abril de 2023	118 350	127 238	7,51%	0.686

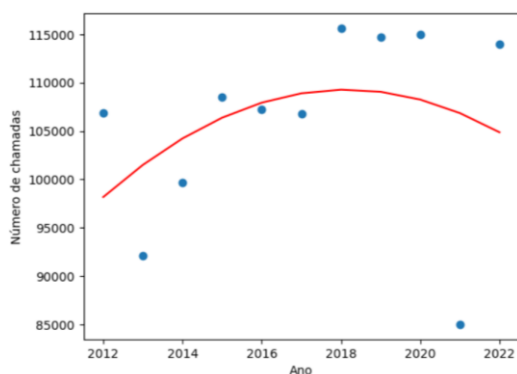
**Tabela 1 - Resultados da previsão mensal do modelo de *Machine Learning***

Acima, podemos ver a tabela obtida através da implementação de um modelo de regressão linear, alimentado por dados recolhidos na página de transparência do INEM [INEMTRA], página esta que publica o número de chamadas atendidas pelo CODU diariamente. Tal como podemos ver pela **Tabela 1**, com o modelo mencionado anteriormente, foi possível obter valores de EMAP de 0,50%, e valores de  $R^2$  de 0.686, o que demonstra que frutos foram colhidos na realização do projeto inicial.

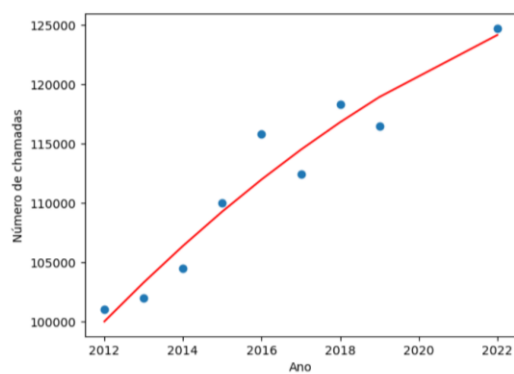
Dito isto, como o objetivo foi calcular o número de chamadas expectadas num mês, os resultados foram agrupados pelo mês correspondente e a cada agrupamento foi dado um valor inteiro para o caracterizar na lista. Posteriormente, foi então feita uma regressão linear, na qual temos como *feature* o número de chamadas recebidas que está em função do mês em que estas foram feitas, e utilizamos todos os dados que temos no *dataset* para obter uma expectativa.

Com efeito, tendo em conta a Tabela 1, repara-se que os valores reais e os previstos estão muito aproximados, o que indica que a ciência que explica que os valores futuros podem ser revistos nos passados não é de toda rebuscada. Tal como podemos ver, em janeiro foram

obtidos dados muito próximos, sendo que a diferença do número de chamadas previsto ditou apenas 2,22% de chamadas a menos que o valor real. Note-se também que o coeficiente de determinação **[R<sup>2</sup>]**, que representa, numa reta, a percentagem de variação dos valores de y que é explicado pelas variações dos valores de x. Isto significa que, na nuvem que é criada num diagrama de dispersão fica bastante próxima da reta de regressão. Então, pelas imagens seguintes, conseguiremos entender melhor esta afirmação:



**Figura 1 - Reta de Regressão Polinomial de fevereiro**



**Figura 2 - Reta de Regressão Polinomial de março**

De facto, estas figuras pertencem a dois meses diferentes para todos os anos para os quais temos dados registados, no entanto o propósito é apenas compreender a diferença entre valores de  $r^2$ . Neste caso, temos então dois diagramas de dispersão, com retas representadas pela cor vermelha, e vários pontos azuis que em conjunto representam a nuvem de dispersão. Então, na Figura 1, podemos reparar que temos pontos bastante dispersos pelo gráfico e mais afastados da linha que os pontos da Figura 2, o que nos indica que a última tem então um valor de  $r^2$  superior à primeira.

Decerto, esta métrica tem bastante importância para efeitos de comparação entre modelos e resultados, visto que vamos trabalhar os algoritmos e iremos entender que mesmo mantendo os valores de erro, o coeficiente de determinação **[R<sup>2</sup>]** irá flutuar bastante mais.

Dito isto, por mais que se queira já saltar para a conclusão de que este é de facto uma proposta viável, é necessário refletir em qual será a razão para que nem todas as centrais telefónicas tenham estes sistemas implementado. Para isto, ter-se-á que responder à questão: porque é que nem todas as centrais telefónicas optam por ter um apoio com intervenção de inteligência artificial, para fazerem uma melhor gestão de recursos?

Com efeito, podemos concluir que para além de um custo elevado que este recurso iria refletir e de algum trabalho acrescido para fazer os dados chegarem a essa nova entidade, há ainda uma grande dificuldade nos dias de hoje em implementar inteligência artificial em todo o tipo de projetos. De acordo com a empresa de consultoria Gartner, apenas 53% dos projetos que integram elementos de inteligência artificial vêm a luz do dia, e 47% nunca deixam a fase de prototipagem. Isto porque existe ainda uma grande dificuldade na criação e gerenciamento de um projeto deste calibre **[IAENG]**.

No entanto, de acordo com um artigo publicado pelo Forbes **[IAFORBES]**, há algumas estratégias que podem ser usadas para corrigir a abordagem feita numa resolução de projeto

desta natureza, sendo que este defende que será melhor pensar no que é necessário para chegar a uma solução, identificar os problemas, e depois partir o objetivo em vários semigrupos. Para além disso, é também indicado que deve ser feita uma análise em relação aos problemas encontrados, e priorizar aqueles que têm de ser resolvidos ao invés dos que são satisfatórios de corrigir. Desta forma, entende-se que criar um projeto que tenha complementos de *machine learning* não é “pera doce” o que leva a crer que o trabalho proposto será mais desafiante, contudo e tal como foi visto no último artigo, este é um problema que não deve ser contornado, deve ser dado o devido valor, tornando-o assim numa tarefa menos problemática, “navegando assim por águas mais brandas”.

Concluindo que este projeto é viável, passamos então para a questão da pertinência. Tal como já foi mencionado nos capítulos **Identificação do Problema** e **Benchmarking**, vimos que há dois grandes problemas no que toca à falta de informação na altura da delimitação do escalamento dos operadores do CODU, sendo estes dois problemas a falta de operadores em serviço em alturas em que o número de chamadas que chegam à central ser demais para os mesmos, ou por outro lado ter um número de operadores excessivo para a quantidade de chamadas de emergências médicas que chegam numa determinada altura.

Efetivamente, como demonstrado no capítulo **2**, o caso do trabalho excessivo por parte dos operadores foi já discutido, contudo temos também relatos do acontecimento do outro problema que estamos aqui a propor, a falta de trabalhadores.

De acordo com o artigo [ARTJORI], existem casos evidentes de escassez de operadores em serviço, onde o jornalista Carlos Diogo Santos escreveu: “*Além do tempo de espera no 112, há ainda pessoas a queixarem-se de atrasos no atendimento do INEM, quando a situação é direcionada para aquela linha*”. Evidentemente, este é um problema que já se verifica há alguns anos, sendo que o artigo foi escrito em 2018, mas que se mantém até aos dias atuais sendo que existem vários relatos de pessoas diferentes ao longo dos anos sobre esse mesmo problema.

Ainda nesse mesmo artigo, temos o depoimento de uma fonte não identificada que declarou que “*Cada elemento em cada turno de oito horas receberá em média cerca de 700/900 chamadas, o que é incomportável*” [ARTJORI], o que neste caso não refletem números que um funcionário do CODU recebe pois esta afirmação é feita em relação aos funcionários da GNR e PSP que representam a primeira linha do atendimento e que depois então fazem a filtração. Contudo, mesmo reduzindo o número tendo em conta as falsas chamadas e as que não são relacionadas com a saúde, estamos ainda a falar de números bastante elevados para chamadas que são mantidas durante alguns minutos para que o interlocutor possa receber a assistência que este necessita.

Além disso, temos ainda: “*sendo a gestão da escala de operadores ajustada de acordo com as necessidades e volume de chamadas a processar (...) em situações excecionais de maior complexidade podem existir momentos de um exponencial aumento de chamadas (incidentes de grande visibilidade e impacto, em que a mesma situação pode gerar centenas de chamadas de diferentes origens), resultando numa maior demora no atendimento, circunstância esta que não se resolveria com o aumento do número de efetivos/operadores por si só, dado o enorme volume de chamadas*” [ARTJORI]. Com efeito, esta citação foi publicada por um comunicado emitido pela PSP, que nos informa que a escalação já é feita tendo em conta as chamadas que poderão ser feitas num determinado dia, assim a proposta que fazemos faz todo o sentido sendo mais



um tipo de apoio para esse fim e ajustando então os números que já são usados para resolver o problema já mencionado. Para além disso, este teria também a capacidade de prever algumas situações excecionais tal como referido na citação, o que eliminaria por completo o tempo de espera especialmente acrescido nesse tipo de situações.

## 4 Solução Proposta

Visto então o problema em **Identificação do Problema** e depois de estudarmos os problemas que existem no âmbito da central do INEM em **Viabilidade e Pertinência**, conclui-se que apesar de ser já um sistema bastante desenvolvido e ter várias ferramentas para responder a muitas das necessidades da população portuguesa, há ainda um grande problema que se pode resumir com a falta de informação em relação ao número de trabalhadores necessários para cada dia.

Neste sentido, a solução que propomos para este mesmo problema é utilizar as bases de *machine learning* criados por um primeiro trabalho, que obteve valores estimados muito próximos dos números de chamadas reais em vários meses futuros, e tentar aumentar a granularidade dos espaços de tempo que são estimados. De facto, este será um passo importante para que possamos ao invés de indicar quantas chamadas irão ser recebidas num dia, indicá-lo em espaços mais úteis para as entidades envolvidas, para que assim se consiga no mínimo, uma estimativa com valores de erro e coeficiente de correlação baixos para espaços de 8 horas. Com efeito, este será um avanço importante pois, tal como já foi falado no capítulo 1, os turnos de trabalho dos operadores de chamadas no CODU são de 8 horas, havendo assim 3 turnos todos os dias.

No entanto, apesar de termos o caso mínimo necessário delimitado, pensamos que será possível aumentar ainda mais a granularidade de modo a termos respostas para idealmente intervalos de tempo de 30 minutos. Isto porque, da mesma forma que tentaremos aumentar para 8 horas, serão realizados estudos e testes para procurar qual o melhor modelo para conseguir um valor de erro que seja considerado insignificante. Decerto, estamos a falar em intervalos de 30 minutos e não qualquer outro tempo devido ao facto de que o INEM divulga as chamadas recebidas em intervalos de 30 minutos, sendo assim esse o teto possível deste trabalho.

Efetivamente, atribuímos estes processos a uma primeira fase deste projeto, à qual se dá uso às aprendizagens feitas nas unidades curriculares de Matemática I, Matemática II e em grande parte a Probabilidade e Estatística, que são lecionadas, entre outros, na licenciatura em Engenharia Informática, do Departamento de Engenharia Informática e Sistemas de Informação (DEISI) da Universidade Lusófona.

Feito isto, como é óbvio, ainda não será suficiente para considerar este projeto como completo, sendo que o CODU não irá realizar estes cálculos cada vez que quiser observar quantas chamadas serão esperadas para uma determinada hora. Assim, é necessário pegar nesses mesmos algoritmos criados na primeira parte, e implementá-los de uma forma que possam ser utilizados de forma simples e intuitiva. No entanto, ainda antes disso, será bastante útil criar um meio intermédio que faça esse mesmo tratamento de dados para que posteriormente tenha apenas de ser recolhido e apresentado. De facto, estamos a falar da construção de uma API, com o intuito de que esta realize a recolha dos dados emitidos pelo INEM no seu site [INEMTRA], e que introduza os mesmos na equação exposta em **Equação 1**, para devolver então o número de chamadas expectadas.

Ainda assim, sabemos que uma API não é de facto algo que possamos apresentar e esperar que o INEM considere um produto útil, e, portanto, a forma que propomos para apresentar esses dados de forma útil e simples é o desenvolvimento de uma aplicação móvel, onde os utilizadores poderão introduzir um intervalo entre duas datas para receber então um relatório em formato PDF com todos os resultados que forem pretendidos. Tal como nos pontos anteriores, existe também uma ligação deste ponto a uma unidade curricular onde este assunto foi trabalhado, sendo que neste caso foi em Computação Distribuída, onde será montado um servidor, neste caso sem *front end* sendo que este será falado já de seguida, onde este irá através de um método REST, receber uma *string* com o conteúdo da mensagem do utilizador da aplicação, e posteriormente irá então realizar todos os cálculos necessários. De facto, neste projeto será utilizado o método REST e não o SOAP devido à sua natureza mais leve e mais rápida, coisa que vamos descobrir ser importante, tendo em conta o que se pretende realizar para o *front-end*.

Então, a terceira parte deste projeto traduz-se na construção de uma aplicação, onde apesar de ter já a parte de resultados facilitada pois estes serão alimentados pela API que será criada na segunda fase do trabalho, contudo será ainda necessário realizar um login para que os dados possam ser restritos para quem é confiado pelo CODU. Para isso, será necessário também criar uma base de dados que guarde os utilizadores para que posteriormente, quando necessário, a aplicação a use para o seu sistema de login. Com efeito, para que esta base de dados possa ser montada, será tido em conta todos os elementos lecionados na unidade curricular de nome Bases de Dados, na Universidade Lusófona.

Para além disso, será também necessário que esta aplicação gere então relatórios em formato PDF, para que estes possam ser descarregados e conferidos. Desta forma, será feita uma investigação sobre a melhor forma de agrupar os dados que forem recolhidos na API num documento e como realizar toda essa escritura e até a própria criação de um PDF através de uma aplicação.

De facto, toda esta APP será criada em android sendo que esta é uma das plataformas que é lecionada na unidade curricular Computação Móvel, sendo esta a que ensina a programar aplicações móveis no curso de Engenharia Informática, do DEISI da Universidade Lusófona de Humanidades e Tecnologias, na linguagem Kotlin que é lecionada em Fundamentos de programação.

Em suma, propõe-se que se realize uma melhoria no algoritmo do trabalho anterior, ou em caso de melhor opção, adotar um outro modelo de *machine learning*, pegar no algoritmo finalizado e implementá-lo numa API de modo a que este possa funcionar de forma isolada e que possa servir para mais que um propósito caso necessário, e montar uma aplicação com o intuito de ser uma interface para os utilizadores recolherem dados das previsões nos formatos desejados, de forma a que seja mais comodo a fim de incentivar o seu uso. Feito isto, é enviado o pedido do utilizador para a API, que posteriormente irá fornecer as chamadas estimadas que serão recebidas no CODU em intervalos de tempo a definir, e assim a aplicação segue para a montagem de um relatório em formato PDF que demonstre todos os resultados obtidos para que, quando o INEM for delimitar a escala de um determinado mês, o possa fazer de forma mais fundada, com o objetivo de reduzir a falta de colaboradores em serviço e assim o tempo de

espera quando alguém realiza uma chamada, e o esgotamento dos trabalhadores deste serviço devido à possível redução de carga horária.

Com efeito, caso este projeto prove ser um sucesso, não será só o CODU que sairá beneficiado. Isto porque, sendo que todos os cálculos serão feitos à parte da aplicação, neste caso pela API, todos os processos poderão ser reutilizados para qualquer que seja a central que necessite de algo desta natureza. De facto, uma associação óbvia para referir será a central do 112, tendo em conta que muitos paralelismos foram feitos neste trabalho, para fins de estudo da situação atual dentro do CODU e a natureza entre estas duas centrais telefónicas são, no fundo, a mesma.

## 5 Calendário

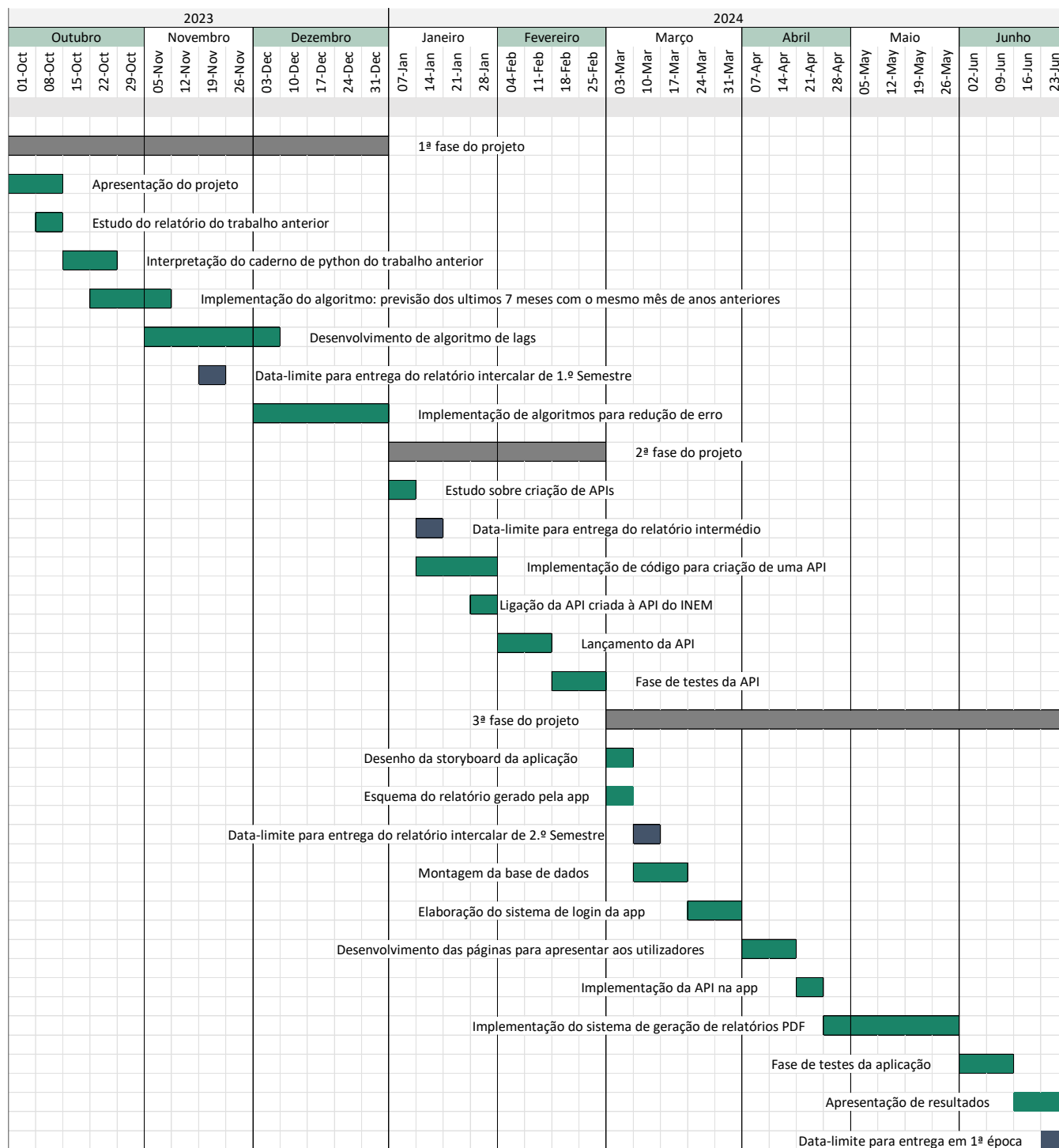


Figura 3 - Cronograma em formato Gantt

## 6 Aperfeiçoamento do Algoritmo

Tal como referido no capítulo 4, este trabalho começa por pegar no algoritmo de regressão linear desenvolvido pelo trabalho anterior, e posteriormente tenta melhorar a margem de erro de forma a que a percentagem seja menor e da mesma forma, relacionar melhor os resultados entre si e principalmente sobre a reta gerada por estes mesmos valores, ou seja, aumentar a taxa de  $R^2$  (coeficiente de determinação).

Assim, para que este modelo fosse entendido, foram realizados alguns testes inicialmente, sendo que o trabalho anterior realizou testes com os meses de janeiro, fevereiro, março e abril de 2023, foram então testados da mesma forma os 5 meses seguintes. Contudo, um grande fator a ter em conta desde as primeiras reflexões foi o alcance do algoritmo para gerar previsões de forma precisa. Isto porque, o algoritmo inicial utilizava a previsão de um único mês apenas, e ao fazê-lo adicionava ao banco de dados o valor real de chamadas nesse mesmo mês para prever o seguinte. Em suma, era apenas previsto um mês, nunca mais que isso.

No entanto, em termos práticos, isto pode ser um problema para o INEM sendo que as escalas mensais têm que ser desenvolvidas com antecedência, o que leva a que o algoritmo tenha que ser corrido antes de termos os valores mensais de um mês que precede o que está a ser estudado.

Deste modo, para além da testagem do modelo, foi também feita a testagem do alcance do mesmo. Efetivamente, isto foi realizado através da utilização do banco de dados reais do projeto, ao qual se iam adicionando os valores previstos de cada mês. Como exemplo, pensaremos no mês de junho, para prever o número de chamadas que chegariam ao CODU nesse mesmo mês serão utilizados todos os dados que existem na base de dados, sendo estes desde 2012 até abril de 2023, e ao invés de juntar o valor real de maio de 2023, tal como o trabalho anterior o fez, adicionamos então o valor previsto pelo algoritmo para maio de 2023.

Desta mesma forma, obteve-se a tabela seguinte:

Mês	Valor Real	Valor Previsto	Diferença	EMAP	$R^2$
Maio	127 719	126 904	815	0,637%	0,693
Junho	124 513	127 143	-2 630	2,113%	0,698
Julho	129 581	127 382	2 199	1,696%	0,703
Agosto	137 205	127 621	9 584	6,985%	0,707
Setembro	122 446	127 860	-5 414	4,422%	0,712

Tabela 2 - Previsões de maio a setembro 2023 com dados previstos

Tal como se observa pela tabela acima, os valores previstos são bastante próximos entre si, sendo que o algoritmo não prevê a subida espontânea do mês de agosto, nem a descida repentina do mês de setembro. Da mesma forma, é de notar que, apesar da diferença mínima no mês de maio entre valor real e valor previsto, sendo esta de apenas 815 chamadas, a oscilação de diferenças é ainda bastante grande, sendo então explicado pela linearidade dos valores previstos. Por sua vez, os valores de erro médio percentual demonstram que, numa primeira fase, está já bem encaminhado pois os valores continuam para já a ser bastante reduzidos, sendo apenas o mês de agosto fora do comum em termos de chamadas pois teve um aumento significativo em relação à norma. Por fim, em relação ao coeficiente de determinação, é de ressaltar que este está ainda longe de uma situação ideal. Isto porque, apesar de valores a rondar os 0,7 ser já um valor elevado, o projeto almeja ter valores que sejam mais elevados para que as dispersões dos pontos de previsão não estejam tão dispersas, como foi visto na **Figura 2**, tendo assim valores mais próximos da reta e desta mesma forma, valores mais próximos dos reais.

Neste sentido, o primeiro passo a dar será averiguar quais os dados que sejam relevantes incluir, tanto a mais do que a menos, para que os valores previstos possam aproximar-se mais dos valores reais.

## **6.1 Seleção do modelo**

Visto que, os valores obtidos com o primeiro algoritmo foram bastante próximos entre si, entende-se que existem ainda alguns passos a dar, sendo que isto não se traduz numa flutuação normal. Então, estudos foram feitos para adaptar o algoritmo a incorporar a variação inerente da natureza das chamadas.

Com efeito, para que isto seja possível, foram testadas as seguintes alterações no algoritmo:

- Alterar os meses que consideramos para a previsão de meses futuros:

De facto, o primeiro passo deste projeto foi avaliar a hipótese de, sendo então os dados próximos mais importantes que os distantes, desbastar o campo de dados de forma que as previsões sejam realizadas com dados mais recentes. Isto é, ao invés da utilização de dados mensais desde 2012, utilizar apenas os últimos 3 ou 4 meses anteriores, sendo que ambos foram testados.

- Utilização dos últimos 3 meses:

Efetivamente, ao utilizar apenas os últimos 3 meses, reparou-se que os valores previstos obtidos não fugiram muito aos valores reais, mantendo até a mesma margem de diferença, entre as 1 354 chamadas para o mês de setembro, e as 11 928 chamadas em agosto. Contudo, é agora possível notar uma oscilação entre os valores previstos, o que de facto era um dos objetivos desta fase de testes, no entanto a um preço elevado, sendo que esta redução de dados levou a que o coeficiente de determinação apresentasse valores incrivelmente baixos, o que indica que este não é de facto o caminho a seguir.

- Utilização dos últimos 4 meses:

Tal como no ponto anterior, este algoritmo demonstrou ser fiável em termos de valor previsto, apresentando diferenças maiores em meses como julho e setembro, com valores de 4 023 chamadas e 4 519 respetivamente, e apresenta um coeficiente de determinação demasiado baixo para os objetivos do projeto.

Desta forma, entende-se que estes últimos meses são de facto bastante importantes pois permitem que o algoritmo normalize de forma mais coerente, os dados de um mês, com o passado próximo e desta forma obter valores mais reais. No entanto, esta não será a forma mais viável de alcançar esta dita normalização devido à enorme discrepância entre coeficientes de determinação.

- Alterar a forma de prever valores:

Decerto, uma outra hipótese proposta foi a utilização de médias e medianas do mesmo mês, em anos anteriores, para prever então os valores de chamadas recebidas num mês futuro. Desta forma, sendo que o banco de dados possui valores desde 2012, e excluindo os dois anos afetados pela pandemia Covid-19, isto porque estes anos foram de facto exceções no que toca a chamadas recebidas pelo CODU, foram avaliados 9 valores diferentes para cada mês, e testado tanto a média desses valores, como a mediana.

Assim, à regressão linear que já realizámos anteriormente, adicionou-se também este valor obtido e assim originaram-se novos valores que, apesar de não oscilarem muito entre si, estando todos entre a casa das 122 mil e as 123 mil chamadas, continuam a ter um erro semelhante aos algoritmos anteriores, mas apresentaram valores de  $R^2$  elevados como 0,645 e até mesmo 0,92.

Contudo, este também não é ainda o algoritmo que procuramos sendo que apesar de um dos objetivos estar dentro dos parâmetros, sendo este o coeficiente de determinação, temos ainda o problema da falta de oscilação nos valores previstos.

- Mescla dos dois últimos testes:

De acordo com os estudos desenvolvidos, repara-se que ambos respondem aos dois objetivos diferentes, e que teoricamente se os juntarmos será então o algoritmo pretendido. Desta forma, às previsões com a utilização de apenas os últimos 3 meses, juntou-se a mediana dos valores do mesmo mês. No entanto, o que parecia ser a solução perfeita demonstrou ser um fracasso sendo que os valores previstos rondaram todos as 121 e 122 mil chamadas, no entanto o coeficiente de determinação demonstrou que um avanço foi feito, isto porque apontou valores bastante elevados, como por exemplo em agosto com 0,838 e setembro com 0,995.

- Utilização dos últimos 7 meses para realizar a normalização:

Para além destas tentativas, foi também recorrido a um outro tipo de algoritmo, este que utiliza o X número de valores anteriores ao que se pretende prever, para que este valor possa então estar mais enquadrado com os valores da realidade mais próxima do mesmo.



De facto, este recurso foi aplicado a vários dos algoritmos testados durante o desenvolvimento do projeto, sendo então assim um procedimento crucial para a obtenção de valores mais aproximados do real. Assim, este será mais aprofundado num ponto mais à frente, sendo este o capítulo **Implementação de lags**.

- Utilização de um coeficiente mês/ano:

Visto que a realidade de chamadas chegadas ao CODU não se mantém durante os anos, no entanto não podemos simplesmente ignorar valores para reduzir a dispersão de pontos na regressão linear, surgiu uma hipótese para colmatar esse problema. De facto, esta hipótese afirma que será bastante útil utilizar não os valores do mês que queremos prever, em anos anteriores, mas sim o que esse mês representou tendo em conta a realidade em que se encontrava, ou seja por exemplo, usar o valor de agosto de 2012 e dividir esse mesmo valor pela média de chamadas de emergência médica realizadas durante todo o ano de 2012, de forma a obter assim o valor a multiplicar com a média do ano em que o mês que se pretende prever se encontra. De facto, este coeficiente será posteriormente explicado em **Engenharia de features**.

Efetivamente, após implementar este algoritmo e acrescentar *lags* tal como em todos os outros algoritmos, obteve-se a seguinte tabela:

Mês	Valor Real	Valor Previsto	Diferença	EMAP	$R^2$
Maio	127 719	126 821	898	0,703%	0,682
Junho	124 513	121 933	2 580	2,072%	0,690
Julho	129 581	124 275	5 306	4,094%	0,698
Agosto	137 205	124 867	12 338	8,992%	0,706
Setembro	122 446	121 413	1 033	0,843%	0,713

**Tabela 3 - Resultados obtidos com algoritmo mês/ano e lag de 7**

De acordo com a **Tabela 3**, verifica-se que os valores previstos voltam a ter então uma oscilação mensal, coisa que se verifica também no valor real. Para além disso, vemos que os valores de diferença estão mais baixos também, sendo que maio, junho e setembro são valores bastante reduzidos, na casa dos mil aos dois mil. Da mesma forma, o coeficiente de determinação é também mais alto, o que indica que os valores obtidos estão bem mais relacionados com a reta de regressão linear.

Visto isto, temos então uma noção dos dados pertinentes a incluir no algoritmo de previsão daqui para a frente, sendo estes a utilização do *dataset* completo para que o algoritmo tenha mais informação para utilizar, contudo alocar maior importância a dados que traduzam da melhor forma algo que seja equivalente a números atuais, e claro juntar a média dos últimos meses para que os resultados fiquem normalizados.

## 6.2 Engenharia de features

Efetivamente, após verificar que dados são importantes para que o algoritmo possa prever um mês, segue-se a adição desses mesmos dados à tabela de dados. Isto porque, uma das características de um modelo de regressão linear é a possibilidade de fornecer várias variáveis independentes ao invés de uma apenas. Assim, dá-se o nome de *feature* a cada uma destas variáveis.

Evidentemente, no caso deste projeto em concreto, e tal como foi averiguado no ponto anterior, uma das variáveis que é bastante útil é a média do mês que está a ser previsto, nos últimos 4 anos. Com efeito, este foi um valor que foi pensado para iniciar a fase de testes, contudo como este apresentou um impacto positivo nos resultados acabou por ser então o escolhido. De facto, esta é relevante pois irá dar uma ideia geral de como costuma ser aquele mês em termos de chamadas que chegam ao INEM, para que o algoritmo tenha também em conta o historial daquele mês em anos passados.

Para além disso, um outro fator considerado muito importante é a noção de como aquele mês em específico é traduzido pela oscilação normal num ano. Isto é, se a uma primeira análise no ano de 2023, o mês de agosto mostra um valor muito acima da média de chamadas de todos os meses de 2023, é necessário ter em conta que isto pode ser indício de que o mês de agosto é à partida um mês que predispõe de mais chamadas que a média. Decerto, este coeficiente pode ser obtido através da seguinte expressão matemática:

$$\text{Coeficiente} = \frac{\text{valor mes}}{\text{media ano}}$$

**Equação 2 - Coeficiente mês/ano**

Com efeito, este coeficiente é relevante para que, quando aplicado aos últimos 4 anos, seja possível perceber como o valor médio daquele mês se relaciona com a totalidade dos anos em que ele está inserido. Desta forma, ao obter a média dos coeficientes, realiza-se uma simples multiplicação com a média anual do mês em questão. Isto porque, quanto mais antigos forem os valores, menor é o número de chamadas sendo que a realidade que existia há 10 anos não é mais a mesma que os dias de hoje. Assim, usamos valores que são adaptados à realidade do mês em questão, tendo em conta aquilo que o ano em que esse se insere dita.

Dito tudo isto, foi construída a seguinte tabela:

Mês	Valor Real	Valor Previsto	Diferença	EMAP	$R^2$
Maio	127 719	127 187	532	0,416%	0,943
Junho	124 513	120 612	3 901	3,133%	0,944
Julho	129 581	126 529	3 052	2,355%	0,945
Agosto	137 205	127 265	9 940	7,245%	0,946
Setembro	122 446	122 445	1	0,000%	0,946

**Tabela 4 - Algoritmo com várias features**

De facto, a tabela acima representa valores bem mais aproximados de algo que se assemelha com a realidade. Isto porque, como se observa pela mesma, maio apresentou valores muito positivos, sendo que a diferença entre o valor real e o previsto foi de apenas 532 chamadas, resultando assim num erro relativo com percentagem mínima, e é também de notar que o  $R^2$  alcançou um nível de 0,943 o que significa que a reta relaciona quase de forma perfeita, os pontos desenhados pela regressão.

No entanto, podemos ver que tanto junho e julho, obtiveram erros que podem ser ainda considerados como relevantes. Contudo, posto em perspetiva, se apontarmos este erro para os 30 ou 31 dias respetivamente de cada um desses meses, isto leva a que o por dia haja uma diferença de 130 chamadas, o que posteriormente quando forem utilizados para calcular o número de operadores necessários numa determinada altura, não seja suficiente para influenciar o objetivo principal deste projeto, ou seja não implicará se mais um operador estará a trabalhar ou não.

Pelo contrário, temos o caso do mês de agosto, que apesar desta alteração, continua ainda a ser um *outlier* no estudo realizado. Uma vez que, os algoritmos utilizados foram fortemente focados no espelhamento de anos anteriores, foi previsto um número tendo em conta esses moldes. No entanto, ao verificar os últimos meses de agosto obtemos os seguintes valores:

	Agosto 2018	Agosto 2019	Agosto 2020	Agosto2021	Agosto2022
<b>Chamadas Recebidas</b>	123 818	118 763			128 072
<b>Média Anual</b>	116 133	117 988			127 282
<b>Coefficiente mês/ano</b>	1,066	1,007			1,006

**Tabela 5 - Chamadas recebidas pelo CODU em agosto dos últimos 5 anos**

Com certeza, tal como se pode observar acima, exceto em 2018, agosto esteve sempre acima do valor mensal médio anual, mas sempre muito próximo deste, isto exceto em 2018 onde agosto descreve um número de chamadas de diferença de 7 685. Deste modo, como a média dos coeficientes acaba por ser igual a 1,026 o que leva a que o mês de agosto tenha sim uma subida em termos do número de chamadas recebidas, mas que não seja o aumento significativo que observamos pelo número de chamadas reais.

Evidentemente, uma das razões pela qual o mês de agosto têm uma discrepância tão grande para qualquer outro mês no ano de 2023 poderá ter sido o acontecimento das jornadas mundiais da juventude, que nesse ano ocorreu em Portugal, precisamente no mês de agosto. Isto é uma possibilidade tendo em conta que na concentração principal do evento, foi apontado um registo de 1,5 milhões de peregrinos, notícia destacada pelo observador [JMJOBS].

Para além disso, em 2023, agosto desfrutou de uma vaga de calor bastante acentuada tal como o instituto português do mar e da atmosfera, ou IPMA, noticiou [TEMPAGO], este mês foi o quinto agosto mais quente desde 1931. Ainda mais, na mesma notícia é referido que nesse mês houve duas ondas de calor, o que como referido anteriormente é um dos fatores que leva a que um mês possa ter uma subida acentuada no número de chamadas de emergência realizadas.

Efetivamente estes são dois dos fatores que podem ter influenciado o aumento bastante significativo do número de chamadas recebidas no CODU. Contudo, apesar de ser uma margem de erro que se considera grave, é algo que é impossível de prever ou controlar devido à espontaneidade de uma onda de calor, ou da imprevisibilidade de um evento da escala da JMJ visto que não existem dados numa situação equivalente registadas em Portugal. Dito isto, apesar da diferença entre o valor real e o previsto ser de 9 940 chamadas, repartido pelos dias daria uma diferença de aproximadamente 321 chamadas acima do previsto, o que numa escala de turnos descreve sensivelmente 107 chamadas a mais que o esperado. No entanto, esta discrepância será ainda estudada em relação à sua relevância quando for estudado o número de operadores necessários, e que poderá até ser posteriormente desenvolvido e aperfeiçoado por um trabalho final de curso futuro.

É de referir ainda que tanto 2020 como de 2021 foram excluídos de quaisquer algoritmos utilizados devido a terem sido os dois anos do covid-19 e os valores obtidos nesses anos foram realmente baixos, não traduzindo uma realidade fora de pandemia.

Por fim, tendo ainda em conta **Tabela 4**, foi obtido um erro percentual de 0% sendo que a diferença entre previsto e real foi de uma chamada, o que indica que este é de facto o caminho certo.

Em conclusão, tendo em conta todos os pontos discutidos neste capítulo, considera-se que o aumento das variáveis de interesse seja o caminho a seguir, sendo que estas não serão claro as mesmas dependendo do intervalo de tempo que se pretende prever. Assim, poderemos então ainda combinar este novo algoritmo com um outro que ajudará a normalizar os valores previstos tendo em conta o recente histórico mensal.

### 6.3 Implementação de lags

Com efeito, tendo já conseguido alcançar um algoritmo que se considere sólido, e tal como referido no fim do capítulo 6.2, é bastante importante utilizar o recente histórico mensal quando se realiza uma previsão para um mês futuro. Evidentemente, isto é importante para que se dê um ajuste nos valores tendo em conta a realidade mais próxima do mês que se pretende estudar.

De facto, isto é um fator importante devido a que o número de chamadas é, a cada mês que passa, cada vez mais superior, o que leva a que o algoritmo possa não acompanhar da mesma forma, ou até caso exista alguma discrepância de chamadas de forma negativa nos últimos meses, que esta mudança seja revista no valor de previsão.

Assim, um dos fatores que se considera fulcral é o fator atualidade. Com efeito, tendo feita a comparação mensal e anual no ponto anterior, resta ainda a comparação com os meses vizinhos para padronizar a previsão, de modo que esta fique dentro dos moldes dos anteriores.

Deste modo, a forma de alcançar este padrão em ciência de dados é a aplicação de *lags* num algoritmo, que nada mais é que uma previsão que utiliza um valor base e os últimos valores obtidos nesse mesmo *dataset*, sendo esse número definido previamente por quem está a estudar os valores.

Efetivamente, no caso das chamadas que chegam ao CODU, foram contabilizados os últimos 7 meses, sendo que nesta fase ainda estão a ser usados intervalos mensais, realizando assim um *lag* de 7. Decerto, a implementação deste número de *lags* deve-se ao conselho feito pelo trabalho pelo qual este se baseia, isto porque nesse mesmo trabalho, entendeu-se que existe uma dependência temporal num conjunto de dados que precede um dado mês, sendo que o conjunto é formado pelos últimos 7 meses.

Feito isto, obteve-se a seguinte tabela:

Mês	Valor Real	Valor Previsto	Diferença	EMAP	$R^2$
Maio	127 719	126 628	1 091	0,854%	0,943
Junho	124 513	123 154	1 359	1,091%	0,944
Julho	129 581	125 314	3 052	3,293%	0,944
Agosto	137 205	126 603	10 602	7,727%	0,945
Setembro	122 446	122 972	-526	0,430%	0,946

Tabela 6 - previsões do algoritmo de *features* com a utilização de *lags*

De acordo com a tabela acima, é possível observar que os valores ficaram mais alinhados entre os seus próprios moldes, o que leva a que meses como junho tenham um terço do erro que tinha sem a utilização de *lags*. Contudo, temos também exemplos de meses que apresentam erros um pouco maiores tal como os restantes.

No entanto, apesar de parecer que os *lags* são pouco relevantes, estamos a falar de espaços de tempo muito grandes, representando 30 a 31 valores diferentes agrupados num só. Neste sentido, quando a granularidade for aumentada, teremos dados diários ou até horários, o que irá permitir que uma grande quantidade de dados possa ser normalizada e desta mesma forma, tornar o algoritmo mais coeso e com menos discrepância de dados.

## 6.4 Aumento da granularidade

De facto, este é ponto é crucial para a relevância deste projeto. Como já foi dito, o objetivo principal deste projeto é fornecer ferramentas que melhor ajudem o INEM a realizar a escala mensal, e para isso serão necessários dados que permitam perceber quantos operadores serão necessários numa determinada hora ou turno.

Para isto, e tal como descrito no capítulo **Benchmarking**, a granularidade das previsões terá de ser aumentada. Neste sentido, o primeiro ponto a realizar será o estabelecimento de previsões diárias, visto que o banco de dados que temos, que é disponibilizado publicamente pelo INEM em **[INEMTRA]**, são dados de valores diários.

Quer isto dizer que, em primeira mão, apenas os dados diários serão alcançáveis. Contudo, e tal como ocorreu também no ano anterior, estabeleceu-se contacto com o INEM, não só para que fizesse parte do projeto, para haver uma melhor comunicação e entendimento sobre o projeto.

Nisto, conseguiram-se então dados com intervalos de 30 minutos, sendo este o intervalo obtido pois é neste mesmo intervalo que o INEM regista as chamadas, sendo então possível aumentar a granularidade da forma desejada e referida em **Solução Proposta**.

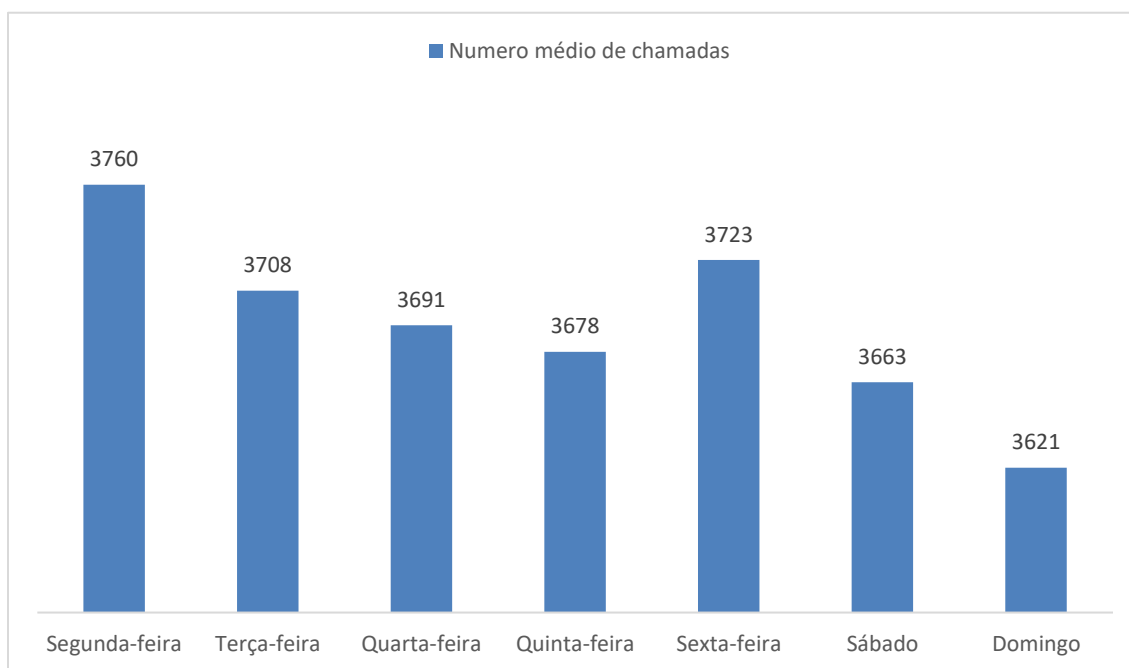
Assim, segue-se então o aumento da granularidade para os diferentes espaços de tempo.

#### 6.4.1 Previsões diárias

Com efeito, para chegar ao intervalo de tempo que se pretende prever, é inteligente não dar um passo maior que a perna, neste caso, passar de uma previsão mensal para uma previsão horária. Isto porque apesar de se saber já qual será o melhor algoritmo a utilizar, alguns ajustes terão de ser feitos ao mesmo para que as variáveis utilizadas possam ser utilizadas neste tipo de previsões.

Deste modo, nasceu uma nova hipótese, sendo que talvez exista um padrão dentro dos dias da semana que seja indicador de maior ou menor número de chamadas para um determinado dia. Dito isto, estudou-se esta possibilidade utilizando os dados disponíveis a que já se recorre para alimentar este projeto, e ao expor as médias tendo em conta o dia da semana, testar a veracidade desta aposta.

Decerto, o referido gráfico pode ser visível de seguida:



**Gráfico 1 - Número médio de chamadas por dia da semana**

Efetivamente, o **Gráfico 1**, construído com os valores de chamadas reais presentes no ficheiro que o INEM disponibilizou, com dados de 2015, e entre 2020 e 2022. De certo, o gráfico prova que a teoria lançada é de facto uma realidade. Como se pode verificar pelo gráfico de barras, existe uma flutuação de chamadas encaminhadas para o CODU tendo em conta o dia da semana em que as mesmas acontecem. De facto, as segundas-feiras demonstram ser os dias que mais chamadas de emergência são feitas em Portugal com um valor médio de 3760 chamadas, seguido pela sexta-feira com as suas 3723 chamadas, apontando assim uma diferença de 37 chamadas. Por outro lado, temos os domingos como os dias da semana onde menos chamadas de emergência são feitas, com um valor de 3621 chamadas realizadas, o que demonstra que, em média, a diferença de chamadas entre o dia em que mais chamadas são

realizadas e o dia em que menos são realizadas, ou seja, segunda-feira e domingo, é de 139 chamadas.

Importa salientar que, apesar da diferença parecer ser baixa, este fator tem uma influência drástica no algoritmo que será implementado. Isto porque, uma previsão precisa de um valor futuro já é por si só uma tarefa difícil, e em muitos casos quase impossível, então estamos já a contar com diferenças entre o valor previsto e o valor real. Deste modo, qualquer mitigação possível dessa diferença será um fator muito positivo e útil para esse mesmo algoritmo.

Deste modo, e visto que as *features* utilizadas para os algoritmos mensais não são transmissíveis para os diários, a alteração que será feita ao coeficiente mensal descrito em 6.2 será feita para a média dos últimos 4 valores que foram obtidos para os mesmos dias da semana, por exemplo, para obter a previsão de uma segunda-feira, serão recolhidos os valores das chamadas previstas ou reais das últimas 4 segundas-feiras.

Da mesma forma, e por falar em *features*, outra variável de interesse que é pertinente neste caso é o histórico recente, que neste caso será o fornecimento do número médio de chamadas, previstas ou reais, que foram recebidas no CODU nos 30 dias anteriores. Sem dúvida, estes valores serão bastante úteis de forma que os valores diários possam estar enquadrados na realidade em que estes se encontram, ou seja, possa estar mais assente com a altura do ano em que se encontram.

Para além das *features*, temos também os *lags* que também serão implementados, contudo também terão algumas diferenças na sua implementação. Tal como discutido em 6.3, foi implementado um sistema de lag de 7 para prever de melhor forma o valor pretendido e, desta forma, contabilizar os valores dos últimos 7 meses. Contudo, e tal como a correlação de dados mensais foram estudados pelo trabalho anterior, também os diários foram trabalhados. Ou seja, através de um estudo de autocorrelação, foi apontado também uma quantidade de *lags* ideal a utilizar para uma previsão diária, que foi neste caso de 250.

Enfim, foi construída a seguinte tabela:

Dia	Valor Real	Valor Previsto	Diferença	EMAP	$R^2$
1	3 922	4 167	-245	6,259%	0,802
2	4 507	4 054	453	10,042%	0,802
3	4 200	4 000	200	4,739%	0,802
4	3 926	4 112	-186	4,741%	0,802
5	4 058	4 099	-41	1,025%	0,802
6	3 850	3 916	-66	1,729%	0,802
7	3 852	3 875	-23	0,619%	0,802



8	4 280	4 205	75	1,749%	0,802
9	4 215	4 094	121	2,857%	0,802
10	3 944	3 994	50	1,271%	0,802
11	4 144	4 066	78	1,869%	0,802
12	4 083	4 085	-2	0,064%	0,802
13	3 825	3 881	-56	1,484%	0,802
14	3 941	3 886	55	1,383%	0,802
15	4 335	4 248	87	1,999%	0,802
16	4 172	4 121	51	1,217%	0,802
17	4 181	3 952	229	5,474%	0,802
18	4 192	4 139	53	1,260%	0,802
19	4 270	4 034	236	5,522%	0,802
20	3 904	3 912	-8	0,219%	0,802
21	3 870	3 856	14	0,356%	0,802
22	4 586	4 239	347	7,552%	0,802
23	4 322	4 073	249	5,748%	0,802
24	4 219	3 952	267	6,325%	0,802
25	4 336	4 055	281	6,458%	0,802
26	4 108	4 030	78	1,881%	0,802
27	4 017	3 886	131	3,240%	0,802
28	3 830	3 864	-34	0,900%	0,802
29	4 269	4 286	-17	0,412%	0,802
30	4 180	4 114	66	1,570%	0,802
31	4 181	3 931	250	5,978%	0,802

Tabela 7 - Previsões diárias para o mês de maio de 2023

De facto, a tabela a cima demonstra que apesar do aumento da granularidade, os valores previstos estão de facto muito aproximados com os valores reais sendo que a maior

diferença registada foi de 453 chamadas no dia 2, o que apesar de parecer um valor bastante grande para um dia em questão, repartido pelas 24 horas diárias daria aproximadamente 9 chamadas a mais por hora, o que poderá ser um número relevante ou insignificante, coisa que será mais facilmente estudada com uma granularidade ainda maior.

No entanto, a tabela demonstra também valores extremamente próximos do real, tal como no dia 12 com uma diferença de 2 chamadas ou no dia 20 com 8 chamadas. Ou seja, as previsões diárias continuam tão precisas quanto as previsões mensais, sendo que em termos de erro percentual, a grande maioria dos valores diários está abaixo dos 2% com 18 dos dias previstos, sendo que à parte do dia 2 com um erro de 10%, apenas 8 outros dias apresentaram um erro acima dos 5%. Além disso, o valor de  $R^2$  mantém-se constante nos 0,802 ao longo da tabela, e apesar de mais baixo que o valor obtido no melhor algoritmo mensal, este valor demonstra ainda que os valores estão bastante bem explicados na reta de regressão linear.

#### 6.4.2 Previsões horárias

Com efeito, foi por esta altura que o INEM se juntou pela primeira vez ao projeto, tendo sido este então apresentado e discutido nessa mesma reunião. De facto, esta primeira reunião contou com a presença da Doutora Isabel Rodrigues, que frisou o quão importante uma ferramenta da natureza deste projeto é. Para além disso, foram também explicadas alguns dos métodos que são usados para gerir os recursos dentro do CODU, sendo que o fator mais importante nesta fase de trabalho, a existência de operadores de reserva, ou seja, ficam em *stand by* e juntam-se à equipa caso seja necessário.

Efetivamente, este cargo é uma mais-valia para este tipo de previsão, pois qualquer que seja a margem de erro que for considerada insignificante, esta será ainda corrigida por mais um ou menos um operador, podendo sempre ter um operador para salvaguardar uma possível má previsão.

Desta forma, sendo que há a possibilidade de um operador ser chamado a uma certa hora de um determinado dia, será muito útil saber quais as horas que serão mais carregadas de chamadas. Visto que, o aumento da granularidade para as previsões diárias foi um sucesso, foi feita uma tentativa do aumento da granularidade para o máximo possível, sendo este em períodos de meia hora tal como já foi explicado em 2. Além disso, ao fazer já este algoritmo, será fácil obter também previsões de turnos, sendo que será apenas necessário agrupar a lista de valores de cada dia, e separá-los em três grupos diferentes, sendo estes então os três turnos que existem num dia de trabalho no CODU.

De facto, para que este processo fosse feito da melhor forma, alguns pequenos ajustes foram feitos ao algoritmo desenvolvido nas **Previsões diárias**, como por exemplo, na busca do valor de chamadas para os mesmos últimos 4 dias da semana, que neste caso foram consideradas as mesmas horas dentro do mesmo molde. Da mesma forma, para o coeficiente que temos também no último ponto, temos esta mesma filtragem para a mesma hora, contudo ao invés da média separada para cada ano para posteriormente ser feita a média dos 4 valores, foi realizada a média à partida devido à grande abundância de dados, que levaria a uma longa demora para o cálculo do algoritmo.

Visto que houve já um primeiro contacto com o INEM, esta primeira previsão seria com um mês mais recente. Contudo, os dados de 30 minutos não são divulgados publicamente, e sendo que estes dados não foram ainda enviados, decidiu-se utilizar os dados que o TFC do ano anterior recebeu, e testar para o último mês recebido que foi março de 2023.

Assim, a tabela seguinte apresentará apenas alguns resultados para fins de análise, o conjunto de previsões completo pode ser conferido em **Anexo I – Ficheiro de valores previstos**.

Dia	Hora	Chamadas recebidas	Chamadas previstas	Diferença	EMAP
<b>2023-03-03</b>	04:30:00	34	25	9	26.471%
<b>2023-03-03</b>	05:00:00	26	26	0	0.000%
2023-03-04	11:00:00	143	106	37	25.874%
2023-03-07	05:30:00	14	28	-14	100.000%
2023-03-31	18:30:00	91	92	-1	1.099%
<b>2023-03-31</b>	19:00:00	91	88	3	3.297%
<b>2023-03-31</b>	19:30:00	78	87	-9	11.538%
<b>2023-03-31</b>	20:00:00	75	85	-10	13.333%
<b>2023-03-31</b>	20:30:00	79	79	0	0.000%

**Tabela 8 - Algumas colunas retiradas do algoritmo de 30 minutos**

De acordo com o excerto retirado da tabela de previsões de 30 em 30 minutos, que se encontra acima, repara-se que existem entradas com diferenças entre valores previstos e valores reais nula ou muito próxima de nula, o que é de facto um ponto positivo. Contudo, na terceira entrada da tabela, temos o valor previsto com a maior diferença obtido pelo algoritmo, tendo obtido um valor de 37 chamadas de diferença para o valor real. Decerto, esta é ainda uma grande diferença visto que estamos a falar de 37 chamadas a mais numa meia hora, o que pode levar a uma sobrecarga dos recursos preparados.

Para além disto, temos ainda um exemplo de uma das entradas com maior erro, sendo esta a quarta entrada da **Tabela 8**, contudo esta não é uma preocupação visto que estamos a falar de valores baixos, e de uma diferença de apenas 14 chamadas, o que num espaço de 30 minutos não é problemático.

Por fim, a segunda metade da tabela são no fundo dados seguidos que se encontram no fim do mês previsto, isto para comprovar que é apesar de alimentar o algoritmo com os valores previstos do mês a calcular, os dados continuam muito aproximados do real sendo que até foi obtido um resultado exato como se pode verificar na última entrada da **Tabela 8**.

### 6.4.3 Previsões em turnos

Tal como referido no ponto anterior, a granularidade foi então aí aumentada até ao máximo, de modo a facilitar e no fundo saltar alguns processos necessários para chegar ao objetivo proposto. Para além disto, e tal como foi visto no ponto anterior, o INEM tem não só a necessidade de saber quantos operadores serão necessários para um turno, dia, hora, mês ou até ano, todos eles são de facto necessários.

Assim, este ponto pretende testar a hipótese de que, por mais que os dados obtidos com o algoritmo horário não tenham sido ideais, se ao agrupá-los por turno acabará por corrigir os erros obtidos.

Desta forma, agrupando os dados pelos turnos de trabalho do CODU, ou seja, agrupar os valores entre 00h e 07:30h, 08h e 15:30h e os valores entre 16h e 23:30, foi obtida uma tabela extensa que se encontra na secção **Anexo I – Ficheiro de valores previstos**. Contudo, um excerto dessa mesma tabela pode ser verificado em baixo:

Dia	Turno	Chamadas recebidas	Chamadas previstas	Diferença	EMAP
2023-03-14	Tarde	1256	1253	3	0.24%
2023-03-21	Noite	481	563	-82	17.048%
2023-03-28	Manhã	1792	1667	125	6.975%
2023-03-29	Manhã	1578	1728	-150	9.506%
2023-03-30	Manhã	1705	1743	-38	2.229%
2023-03-31	Manhã	1630	1720	-90	5.521%

**Tabela 9 - Excerto de dados previstos agrupados em turnos**

De acordo com a tabela acima, podemos ver um exemplo de um turno que obteve uma diferença entre o valor calculado e o real de apenas 3 chamadas, o que prova que o algoritmo está de facto a realizar estimativas próximas do objetivo. Para além disso, observa-se também pela segunda entrada desta tabela, que a estimativa com um maior erro foi no dia 21 de março onde este foi de 17%, que traduzindo para diferença foi de apenas 82 chamadas a menos. Para além disso, e tal como no ponto anterior, para averiguar o alcance deste agrupamento, as restantes entradas da tabela são os intervalos do turno da manhã no fim do mês previsto, que demonstram que tanto o erro percentual como a diferença obtida, apesar de altos, para um turno de 8 horas acabam por descrever um número reduzido de chamadas.

## 6.5 Resultados desta fase

Efetivamente, esta primeira fase do projeto foi de facto um grande sucesso, sendo que o objetivo inicial para a mesma foi alcançado.

Apesar de isto ser um facto, há ainda uma margem de erro, claro, tal como foi apresentado em **6.4.2**.

Contudo, tal como foi também discutido na **Solução Proposta**, um algoritmo que apresente um erro nulo é claro uma utopia que nunca será alcançada. De certo, o objetivo deste projeto é sim, chegar a um valor informativo que o INEM considere útil e aproximado o suficiente para que possam então gerir os recursos tendo em conta o que a aplicação sugere. Ou seja, o INEM é o único utilizador do produto final que será desenvolvido durante este trabalho, por isso, é esta a instituição que acaba por decidir se algoritmo já está ou não numa precisão adequada.

De facto, o INEM esteve presente durante o desenvolvimento desta primeira fase, no entanto, este acompanhamento foi um pouco mais esporádico, o que levou a que fossem então os restantes envolvidos no projeto a avaliar esta parte.

Deste modo, depois de alguns debates, chegou-se a um consenso para seguir com os algoritmos descritos ao longo destes pontos.

Para além disso, surgiu também a ideia de possivelmente deixar as várias partes deste projeto abertas a melhorias. Isto é, apesar de haver sim um desenvolvimento de um produto final e não várias pontas soltas que têm posteriormente de ser acabadas, dar a possibilidade a outros alunos em final de curso de pegarem numa das 3 partes deste projeto e aprimorá-la, seja licenciatura, mestrado, etc.

Ainda assim, gostaria só de ressaltar mais uma vez, os valores obtidos nesta primeira fase de delimitação de algoritmos, continuam a ser valores muito próximos do real e muito bons para os fins pretendidos.

Assim, seguimos para a próxima fase deste trabalho final de curso, para a construção de uma *API*.

## 7 Estabelecimento de uma API

Com efeito, de acordo com o plano de trabalho, a fase que se segue é então a construção de um servidor.

Efetivamente, este servidor recebe pedidos de utilizadores e realiza a sua função, voltando assim ao utilizador e devolvendo aquilo que foi trabalhado por ele. No fundo, é isto que é uma *Application Programming Interface*, mais popularmente conhecido como API.

De facto, este trabalho recorre a uma API, isolando os processos do mesmo, permitindo assim que os diferentes serviços que esta disponibiliza possam ser utilizados de forma autónoma, ou trabalhados de forma individual. Isto porque, este projeto engloba tecnologias variadas e distintas de várias áreas diferentes.

Assim, nesta fase do projeto, será criada uma API. Isto, com o intuito de o usar como intermediário entre a base de dados que irá conter um ficheiro *excel* com os valores de chamadas reais, e o utilizador por meio da aplicação.

Para isto, temos alguns pontos que precisam de ser tratados, inclusive por uma ordem específica, porque será necessário ter a base de dados estabelecida antes de trabalhar num servidor em si.

Em suma, esboçando assim de forma simples um caso prático com todas as interações correspondentes a este capítulo, temos a seguinte sequência de acontecimentos:

Primeiramente, o INEM começa por recolher a sua atualização mensal de chamadas de emergência que foram efetuadas e realiza o envio do mesmo para a base de dados utilizada pelo projeto. Feito isto, a receção do ficheiro é notada pela própria base de dados, e quando isto acontece o algoritmo de previsões arranca, criando um ficheiro *excel* com previsões de meia em meia hora, num total de 31 dias, pelo menos inicialmente. Com efeito, para motivos de exemplificação, vamos chamar a este ficheiro *previsoes.csv*. Feito isto, os dados alimentam então uma plataforma intermédia, sendo esta a API, que aguarda que um utilizador da aplicação faça um pedido de previsão, este pedido é recebido pela API que irá então recolher os valores do ficheiro *previsoes.csv*, manipulá-los de forma a ter uma resposta ao pedido do utilizador, e devolve então estes valores finais à aplicação.

Assim, o primeiro ponto neste capítulo será:

## 7.1 Criação de uma base de dados

Desta forma, a primeira etapa deste capítulo é a criação de uma base de dados. De facto, este conceito nada mais é que um armazenamento de informações úteis ao sistema que está a ser montado, sendo que este armazenamento é feito na nuvem, o que permite que o serviço esteja operacional a qualquer hora, em qualquer lugar, e assim, facilitar as condições de uso da aplicação.

Tal como todas as outras tecnologias, esta foi também trabalhada em unidade curricular lecionada na Universidade Lusófona. De facto, a unidade curricular onde esta tecnologia foi lecionada foi Bases de Dados.

Com efeito, existem várias formas diferentes de criar uma base de dados, sendo que para o caso prático deste trabalho, esta será estabelecida pelo INEM. Neste caso, será a organização a decidir em que formato a base de dados será estabelecida. No entanto, não foram obtidas grandes indicações do INEM, e assim, decidiu-se arranjar uma alternativa, que é a utilização do Docker, sendo que foi este que foi utilizado para estudar bases de dados na sua respetiva unidade curricular.

### 7.1.1 Docker

De facto, o que é o Docker? Com efeito, o Docker é um serviço de *Platform as a Service* (PaaS), ou seja, fornece uma plataforma em nuvem, com os mesmos recursos físicos que uma infraestrutura tecnológica possui.

Desta forma, esta ferramenta permite que os seus utilizadores criem pacotes isolados, denominados de *containers*, para armazenar, gerir ou até mesmo apresentar informações. Efetivamente, esta ligação pode ocorrer localmente, através do software do próprio serviço, ou através do *Docker Hub*, que no fundo acaba por fazer a ponte do *host* local para nuvem.

Com efeito, estes containers podem ser criados através de uma imagem base, sendo que a imagem utilizada tanto em aula, como no decorrer deste trabalho, foi a imagem de mysql da Microsoft.

De certo, é possível criar um novo *container* através do próprio software do *Docker*, no entanto, uma melhor prática será através de código na linha de comandos da máquina que está a ser utilizada. Isto porque, o *Docker* permite que os *containers* sejam criados já com algumas instalações, como por exemplo, o próprio *python*, no entanto este recurso não será usado para facilitar a explicação.

Assim, criado o container sem quaisquer instalações, iremos então precisar de aceder à linha de comandos do *container* para realizar as instalações necessárias. Para isso, basta aceder à linha de comandos da máquina que está a ser utilizada e introduzir o código:

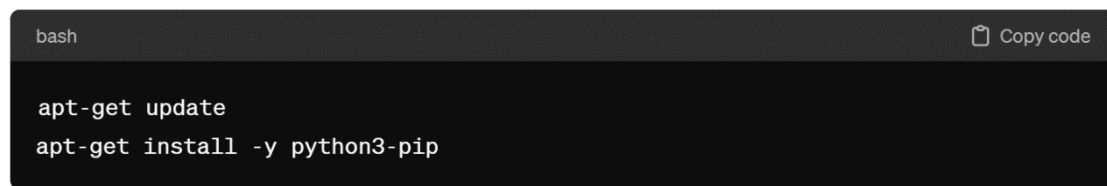
```
docker exec -it <nome do container> bash
```

Deste modo, temos já então o container criado e a sua *bash* aberta para poder dar instruções ao mesmo. Neste caso, o objetivo é guardar os valores de chamadas reais que o INEM possui nesta base de dados, por isso recorrer-se-á a esta mesma linha de comandos para copiar o ficheiro, que neste caso é em formato *.csv*, para dentro de uma diretoria no Docker. Para isto, basta abrir a linha de comandos da máquina que está a ser usada, dirigir-se à diretoria onde o ficheiro de chamadas que se pretende enviar para a base de dados se encontra, e recorre-se à seguinte linha de código:

```
docker cp <Nome do ficheiro .py> <nome do  
container>:/<diretoria>/<projeto>/<nome para o ficheiro .py>
```

Ora, corrida a instrução, temos já então o ficheiro de chamadas reais guardado dentro do container Docker. No entanto, para poder ainda utilizar comandos para correr ou manipular o algoritmo, será preciso ainda instalar algumas outras funcionalidades dentro do próprio container.

De facto, uma das ferramentas que será necessária é a linguagem *python*, tendo em conta que todo o código descrito no algoritmo está nesta linguagem de programação. Para isso, escreve-se a seguinte linha de código na linha de comandos do *container*:

A terminal window with a dark background. The title bar shows 'bash' on the left and a 'Copy code' button on the right. The terminal contains two lines of text: 'apt-get update' and 'apt-get install -y python3-pip'.

```
bash  
apt-get update  
apt-get install -y python3-pip
```

**Figura 4 – Código a ser inserido na bash do *container Docker***

Assim, temos já o container a ser utilizado pelo projeto, pronto para correr o algoritmo python. Deste modo, resta compreender a situação em que o código corre e como acionar o mesmo. Para isso, segue-se o seguinte sub-capítulo.



### 7.1.2 Controladores

De acordo com o último sub-capítulo, há já uma base de dados montada, com o ficheiro de chamadas reais guardado e algumas bibliotecas instaladas. Desta forma, está tudo a postos para que a *API* possa então recolher os valores reais e aplicar-lhes o algoritmo. No entanto, será este o melhor método no final de contas? Efetivamente, existem aqui dois cenários negativos nesta abordagem, a necessidade de execução do algoritmo de forma repetitiva, ou seja, cada vez que um utilizador realizasse algum pedido, ou a suspensão da *API*, devido a pedidos simultâneos.

Então, na primeira hipótese, o algoritmo de previsões atua sempre que um novo pedido é feito, para recolher então valores previstos para apresentar ao utilizador. Evidentemente, esta hipótese não é viável visto que o algoritmo é ainda bastante extenso o que leva a que o tempo de execução seja ainda bastante grande. Desta forma, sabe-se que será necessário correr o código do algoritmo uma única vez, claro que de x em x tempo, e guardar os valores previstos também no Docker.

Assim, seguimos para o segundo cenário, a *API* correr o algoritmo esporadicamente de acordo com uma condição. No entanto, este método levanta também alguns problemas, tais como a suspensão dos serviços de apresentação de previsões por parte da *API*, sendo que esta está então ocupada a correr o algoritmo. De facto, isto é um ponto negativo, contudo consegue ser mitigado através do agendamento destas atualizações para uma hora específica em que seja expectável que poucos utilizadores utilizem a app. Apesar disso, será que não há ainda uma opção melhor?

De certo, foi aqui que alguma investigação foi feita, afim de perceber se existe algum caminho melhor. Com efeito, foi aqui que nasceu uma hipótese, sendo esta a possibilidade de correr o algoritmo não pela *API* mas pelo próprio container do *Docker*, sendo que existem bibliotecas de *python* que suspendem instruções até que um ficheiro específico é alterado.

Neste caso, a biblioteca estudada e utilizada será a *watchdog* que no fundo aguarda que um ficheiro, dado o seu caminho, seja alterado, e quando isto acontece teremos então o respetivo código a ser acionado. Com efeito, para instalar esta biblioteca, introduzimos o seguinte código:

```
root@b879921c1a19:/# pip install watchdog
Collecting watchdog
  Downloading watchdog-4.0.0-py3-none-manylinux2014_x86_64.whl (82 kB)
    | 82 kB 787 kB/s
Installing collected packages: watchdog
Successfully installed watchdog-4.0.0
root@b879921c1a19:/#
```

Figura 5 - Instalação do watchdog no container Docker

Deste modo, o objetivo é levar a que o algoritmo corra e faça previsões para um determinado espaço de tempo (que será estudado tendo em conta a precisão de alcance do algoritmo), cada vez que um novo ficheiro de chamadas reais é carregado para o *container*.

Sendo assim, será agora necessário adaptar o código desenvolvido no caderno de *python*, para um caso prático que funcione dentro do próprio Docker.

### 7.1.3 Adaptação do Código

De facto, o código desenvolvido na primeira parte deste projeto foi exclusivamente desenhado para comparação com um mês específico, sendo este o último mês para o qual temos registo do número de chamadas reais, para fins de estudo e comparação.

Então, no fundo os afazeres neste sub-capítulo são a adaptação do código para que este preveja sempre os próximos X dias, sendo que estes dias são posteriormente determinados pela necessidade do INEM.

Para além disto, será também necessário implementar a biblioteca mencionada em 7.1.2, que não é uma implementação extensa por si, mas que não deixa de precisar da sua atenção. Então, para isto aplicamos o seguinte código:

```
def watch_directory():
    event_handler = FileModifiedHandler()
    observer = Observer()
    observer.schedule(event_handler, path='/home/inem', recursive=False)
    observer.start()
    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        observer.stop()
    observer.join()
```

**Figura 6 - Código para observar e notificar quando alguma alteração é feita**

De acordo com a **Figura 6**, entende-se que por este excerto, o observador é instalado dentro da diretoria `/home/inem` que no fundo foi o caminho gerado dentro do próprio *container* do *Docker*, para guardar todos os ficheiros do projeto. Neste caso, o que o código faz é instanciar um objeto da classe `Observer` para verificar, a cada segundo, se alterações foram feitas dentro da diretoria em questão. Para além disto, temos então uma classe que é também da biblioteca `watchdog`, que irá então verificar se a alteração na diretoria é de facto o ficheiro de chamadas reais, e se for aguarda que o ficheiro que está a ser alterado seja completamente carregado, e por fim corre o código do algoritmo.

Feito isto, têm-se já a condição estabelecida e codificada de modo a correr o algoritmo cada vez que um ficheiro com um determinado nome indicado é substituído. Assim, não se corre o risco de o observador correr de novo quando criamos um ficheiro dentro da base de dados.

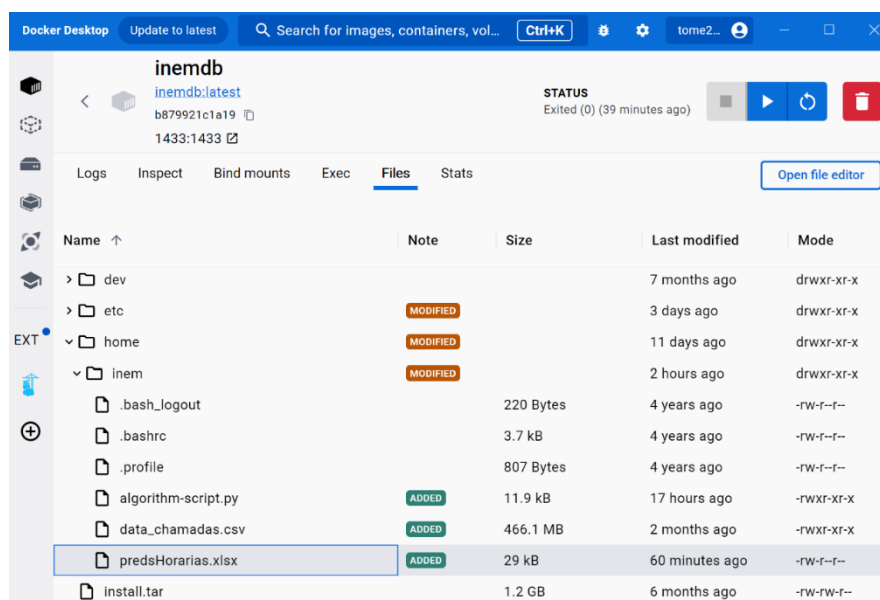
Deste modo, segue-se o algoritmo, que como referido anteriormente, terá que sofrer algumas alterações para funcionar sempre para os dias seguintes à última entrada dos valores reais guardados em Excel. Assim, em primeiro lugar, já não será necessário remover o último mês do grupo de teste do modelo de regressão linear, visto que isto foi apenas utilizado para a fase de testes. Da mesma forma, todo o código específico para o mês de março passa então a ser para os seguintes 31 dias, em primeira mão sendo este então um número pensado primeiramente para haver uma previsão mensal. Contudo, este intervalo pode, e muito provavelmente será, alterado para uma previsão em maior escala devido às necessidades que imaginemos existirem por parte do INEM, sendo que isto terá de ser debatido com a instituição. De facto, este aumento do intervalo de previsões será trabalhado idealmente ainda durante o desenvolvimento deste projeto, e/ou poderá também ser abordado por algum outro trabalho a ser desenvolvido na Universidade Lusófona.

Assim, com previsões calculadas, é criado um novo ficheiro Excel com esses mesmos valores, para ser lido e recolhido pela própria API.

Com efeito, todo o código descrito pelo algoritmo encontra-se em anexo em **Anexo II – Código referente ao algoritmo**.

#### 7.1.4 Organização da BD

Feito isto, temos então o sistema montado para receber um ficheiro, correr o algoritmo e por fim gerar um novo ficheiro Excel com os valores previstos. Com efeito, esta arrumação pode ser vista na seguinte figura:



**Figura 7 - Container do Docker utilizado pelo projeto**

Observe-se a **Figura 7**, à parte das diretorias e ficheiros raiz de qualquer *container*, temos então a diretoria `/home/inem`, que possui então todos os ficheiros referentes a este trabalho. Neste caso, temos então o ficheiro `data_chamadas.csv` que no fundo foi o nome básico

que optámos por atribuir aquele que é o ficheiro em posse do INEM com as informações sobre as chamadas reais, o ficheiro `predsHorarias.xlsx` que foi o nome atribuído ao ficheiro que contém todas as previsões calculadas pelo algoritmo contido em `algorithm-script.py`. Sendo assim, num caso prático o INEM envia para a base de dados um ficheiro `data_chamadas.csv` que por sua vez substitui o antigo, o *script* `algorithm-script.py` percebe que o ficheiro foi alterado e corre o código, e por sua vez o código realiza as previsões e guardando-as num ficheiro `predsHorarias.xlsx` que no fundo é substituído por aquele que já lá se encontra. No fundo, a ponte BD-API é feita apenas por este último ficheiro, sendo que são apenas os valores previstos que são transmitidos.

## 7.2 Instalação de uma API

No que diz respeito à API, temos então algumas opções que foram postas em cima da mesa e estudadas de forma sequencial até chegar aquele que será a melhor interface de comunicação entre base de dados e aplicação.

Com efeito, a primeira ideia foi recorrer ao *pythonanywhere*, que é uma plataforma online da Anaconda. Isto porque, na unidade curricular de Programação Web, também lecionada na Universidade Lusófona, foi utilizada a plataforma para fins de criação de uma aplicação web. Apesar disto, esta pode ainda ser utilizada para fins de comunicação e este será então a primeira opção para hospedar da nossa interface. Contudo, existe também uma alternativa que será a atribuição do host da API ao próprio Docker, coisa que é possível e bastante simples visto que temos já um container a postos e com todos os materiais necessários contidos no próprio. Efetivamente, isto é relevante até por motivos de segurança de dados, visto que pode haver menos transferência de dados entre plataformas, o que mitiga a possibilidade de fuga de dados.

Dito isto, a primeira hipótese estudada foi então a utilização do *pythonanywhere* como API, visto que o Docker é uma ferramenta que está a ser utilizada pura e simplesmente com fins demonstrativos, sendo que o INEM irá continuar a usar as suas próprias bases de dados, e, portanto, esta seria a parte que seria alterada utilizando assim a API em *pythonanywhere* para ser apenas necessário alterar o endereço onde esta iria recolher a informação.

Deste modo, desenvolveu-se um segundo script, para a própria API correr cada vez que um pedido do tipo GET fosse efetuado. Mas, o que é isto de um pedido GET? De facto, quando estamos a falar de uma API, temos imediatamente dois tipos de pedidos que podem ser feitos à mesma, um pedido de envio, chamado de POST, ou, o que será mais importante para este caso em específico, um pedido para receber que é então descrito por GET. Com efeito, este pedido é realizado pela própria aplicação de forma a receber os valores previstos vindos da API, que recolhe estes mesmos dados da base de dados.

De certo, isto é comum a qualquer que seja a forma como for decidido implementar a API, e portanto segue-se o estudo para verificar qual a melhor forma de o fazer.

### 7.2.1 API Pythonanywhere

Com efeito, a primeira hipótese estudada foi utilizar a plataforma *pythonanywhere*, uma hipótese que foi debatida em reunião e que fazia bastante sentido visto que esta é uma plataforma utilizada em aula. Para isto, foi criada a conta *inem.pythonanywhere.com* que de facto será utilizada ao longo de todo o projeto para qualquer efeito que seja adequado.

Portanto, criada a conta, temos já um sistema de coexistência entre a API e a BD, no entanto será necessário realizar a ligação entre as duas. Para isto, será necessário que o Docker faça o envio dos dados por si próprio para o endereço da API, visto que um ficheiro contido num *container* Docker não pode ser acedido diretamente pelo exterior. Efetivamente, para que isto seja possível, será criado um terceiro script bastante simples, que tudo o que fará será recolher os dados do ficheiro Excel, converter cada entrada num dicionário, dicionário este que nada

mais é que uma entidade singular em python, reunir todos os dicionários e preencher uma lista, e, por fim, enviar esta lista completa para a plataforma da API.

Assim, parece tudo bastante simples, no entanto, este script não poderá estar sempre a correr porque iria sobrecarregar o sistema, tal como não pode correr de x em x tempo visto que não se pretende que um tempo de inatividade exista. Por isso, o único caminho aqui será forçar este código a correr pelo Docker na própria API do *pythonanywhere*, o que pode ou não ser possível.

Para isto, recorreu-se a um subprocesso. Subprocesso este que, numa primeira instância, envia para o *container* aquele que será a linha de código a ser introduzida na sua *bash* pela própria API. No fundo, este subprocesso apenas irá indicar ao Docker que este deve correr então o terceiro script mencionado anteriormente.

No entanto, apesar de parecer uma ótima forma de resolver o problema de envio de dados, surgiu um outro problema. De facto, o problema não foi outro, porque foi sim um problema de envio de dados, mas não da forma esperada. Com efeito, entende-se que esta forma acaba por causar também alguns problemas, mais especificamente em termos de comunicação API-Docker. Foi então aqui, que o projeto acabou por ter o seu primeiro percalço.

Mesmo assim, este não apresentou um grande contratempo no final de contas, visto que outras hipóteses já tinham sido pensadas anteriormente.

### 7.2.2 API Docker

Com efeito, tal como visto em 7.2, para além de uma base de dados, o Docker permite ainda enviar os dados que guarda através de uma API. Tal como referido também anteriormente, isto acaba por ser bastante benéfico em termos de proteção de dados, visto que toda a parte de tratamento de dados é feita única e exclusivamente por uma plataforma, mitigando qualquer hipótese de fuga de dados.

Neste sentido, será escrita uma aplicação em *Flask*, para recolher os valores do ficheiro excel, e que os irá disponibilizar num servidor. Para isso, temos os seguintes pontos:

```
@app.route('/recolhe_dados', methods=['GET'])
def recolhe_dados():
    df = pd.read_excel('predsHorarias.xlsx')
```

Figura 8 - Função recolhe\_dados da API

Com efeito, esta será a função principal da API, que tal como a primeira linha indica, será a função que irá correr quando um pedido com o parâmetro “/recolhe\_dados” for feito. Quando de facto, um pedido destes é realizado, recolhem-se as entradas da tabela criada no capítulo 6, e posteriormente, aplicam-se as seguintes alterações:

```

# Verifica qual é o tipo de pedido e agrupa os dados da forma desejada
if group_by == "diários":
    df_grouped = df.groupby(df['Dia']).agg({'Valor Previsto': 'sum'}).reset_index()
elif group_by == "turnos":
    df['Hora'] = pd.to_datetime(df['Hora']).dt.time

    df['Turno'] = df['Hora'].apply(determina_turno)
    df['Hora'] = df['Turno'].apply(aplica_intervalo_temporal)

    df_grouped = df.groupby(['Dia', 'Turno', 'Hora']).agg({'Valor Previsto': 'sum'}).reset_index()

    turno_order = pd.Categorical(df_grouped['Turno'], categories=["Noite", "Dia", "Tarde"], ordered=True)
    df_grouped['Turno'] = turno_order

    df_grouped = df_grouped.sort_values(by=['Dia', 'Turno']).reset_index(drop=True)
elif group_by == "mensais":
    df_grouped = df.groupby(df['Dia'].dt.to_period('M')).agg({'Valor Previsto': 'sum'}).reset_index()
elif group_by == "anuais":
    df_grouped = df.groupby(df['Dia'].dt.to_period('Y')).agg({'Valor Previsto': 'sum'}).reset_index()
else:
    df_grouped = df

```

Figura 9 - Permutações aplicadas aos valores recolhidos

De facto, ao observar o código da figura anterior, compreende-se que as alterações aplicadas às entradas da tabela visam responder às necessidades específicas de cada pedido. Isto é, caso sejam pedidas previsões diárias, por turnos, mensais ou anuais, serão aplicadas as devidas alterações para que estes pedidos sejam realizados. Nomeadamente, para as previsões por turnos, são aplicadas duas funções para cada entrada da tabela:

- `determina_turno`: Neste caso, a função verifica a hora em que a entrada está registada, se for entre as 00h e as 08h, esta entrada terá o resultado “Noite” para a nova coluna “Turno”, se for entre as 08h e as 16h será “Dia”, e entre as 16h e as 00h será “Noite”,
- `aplica_intervalo_temporal`: Por sua vez, esta função verifica o valor da coluna “Turno” em cada entrada, e substitui o valor da coluna “Hora” para o intervalo de tempo a que esse turno equivale, tal como descrito no ponto anterior.

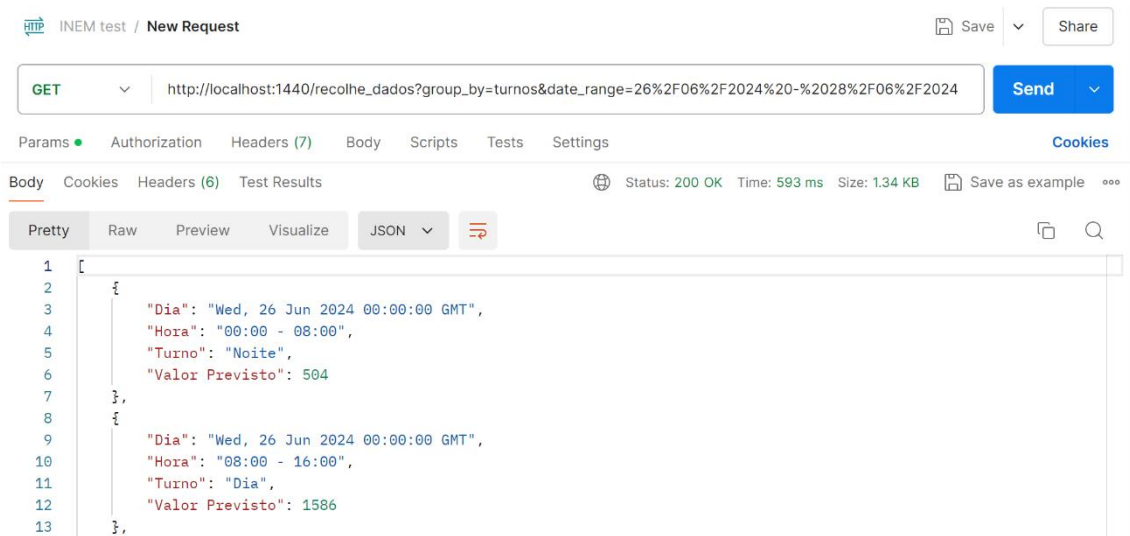
Para além disso, temos também um último caso, onde nenhum algoritmo é aplicado e a função passa o *dataset* sem que este sofra qualquer alteração, sendo que este contém já os valores em intervalos de 30 minutos.

Efetivamente, este ficheiro será colocado dentro do Docker, utilizando o comando `docker cp api_code.py inem:/home/inem` e posteriormente corrido através do código `docker exec -it inem python3 /home/inem/api_code.py` sendo que ambos são escritos na linha de comandos da máquina que está a ser utilizada.

Assim, entende-se que esta será a melhor opção porque, tal como referido anteriormente, mitigam-se as hipóteses de fuga de dados e facilita-se o processo da recolha em si, sendo que não haverá problemas de segurança por parte dos softwares usados.

Obviamente, este não será o processo ideal para o INEM, tal como o uso do Docker como base de dados, no entanto este trabalho não estando a ser alocado de raiz aos servidores do INEM, existe esta flexibilidade na alocação dos serviços, para fins de demonstração.

Neste sentido, resta apenas verificar se de facto os pedidos à base de dados são atendidos, e se são da forma correta. Para tal, a uma primeira fase, realiza-se essa verificação utilizando o software Postman.



**Figura 10 - Utilização do software Postman para teste da API**

Decerto, na imagem anterior, verifica-se que de facto temos a API operacional, sendo que temos dois turnos do dia 26 de junho a serem apresentados da forma correta. Assim, fechamos a segunda fase deste projeto, tendo já o algoritmo idealizado e a API completamente operacional, passando assim então para a criação de todo um front-end para esta ferramenta.



## 8 Aplicação INEM Insight

De facto, não há melhor forma de começar este capítulo como por aquilo que num contexto normal seria o ideal a fazer. Neste caso, aquilo a fazer em primeiro lugar quando o objetivo é desenvolver uma aplicação, é a maquete dessa própria aplicação. Efetivamente, a isto chamamos de storyboard, uma série de imagens montadas para cada ecrã, grupo este que constitui a aplicação em si.

No entanto, tal como aludido, este não foi de facto o caminho que foi seguido durante o desenvolvimento do trabalho. Mas por que motivo se decidiu saltar grande parte dos passos iniciais? Com efeito, e tal como referido em todo este trabalho, sendo este realizado em conjunto com o INEM, escassas foram as vezes em que a equipa reuniu com a instituição para debater qualquer que fosse o motivo. Efetivamente, foi também por esta altura que uma possível reunião presencial foi falada, e decidiu-se então realizar a parte funcional primeiramente para que esta pudesse ser logo discutida. Para além disso, isto fazia muito sentido também pois sendo os funcionários do INEM os únicos utilizadores da web app, eles melhor que ninguém poderiam sugerir aquilo que fizesse mais sentido.

Contudo, esta reunião acabou por não ser possível, o que levou a que esta decisão não fosse necessária no final das contas. No entanto, sendo que o desenvolvimento foi acontecendo em paralelo, no final de contas acabou por não se fazer uma storyboard e, portanto, seguimos para os pontos mais relevantes:

### 8.1 Identidade da aplicação

De facto, este foi um dos pontos que foi trabalhado mais à frente no trabalho, contudo faz sentido referi-lo já sendo que este estará presente em todos os próximos pontos. Certamente, este não é um ponto tão importante quanto outras aplicações, sendo que esta será utilizada à porta fechada, contudo uma aplicação tem de ter um nome, uma identidade.

Assim, refletindo um pouco sobre aquilo que é este projeto, e no fundo a palavra que foi considerada a que melhor simboliza a finalidade desta ferramenta foi *Insight*. Isto porque, no fundo o objetivo não é gerir os recursos pelo INEM, nem dar uma resposta definitiva daquilo que deve ser a sua gestão. Na verdade, o objetivo é sim dar uma opinião formada sobre aquilo que o INEM deve ter em conta, no fundo apoiando com uma informação extra.

Desta forma, foi também desenhado um logo simples para fazer parte desta identidade. Evidentemente, este logo foi criado com o auxílio da ferramenta da adobe, Photoshop.



Figura 11 - logotipo do site

## 8.2 Log in

Tal como referido no ponto anterior, o objetivo desta aplicação é que esta seja fechada apenas a funcionários do INEM, e desta forma, iremos controlar isso mesmo através de um sistema de criação de contas onde apenas emails com a extensão “@inem.pt” serão aceites. Com efeito, para que isto aconteça, pensou-se em inverter a sequência da criação de conta normal. Isto porque, numa criação de conta padrão, os utilizadores inserem o nome de utilizador, palavra-passe e email, e é lhes enviado um email para confirmação da conta. Neste caso, fazemos o inverso:

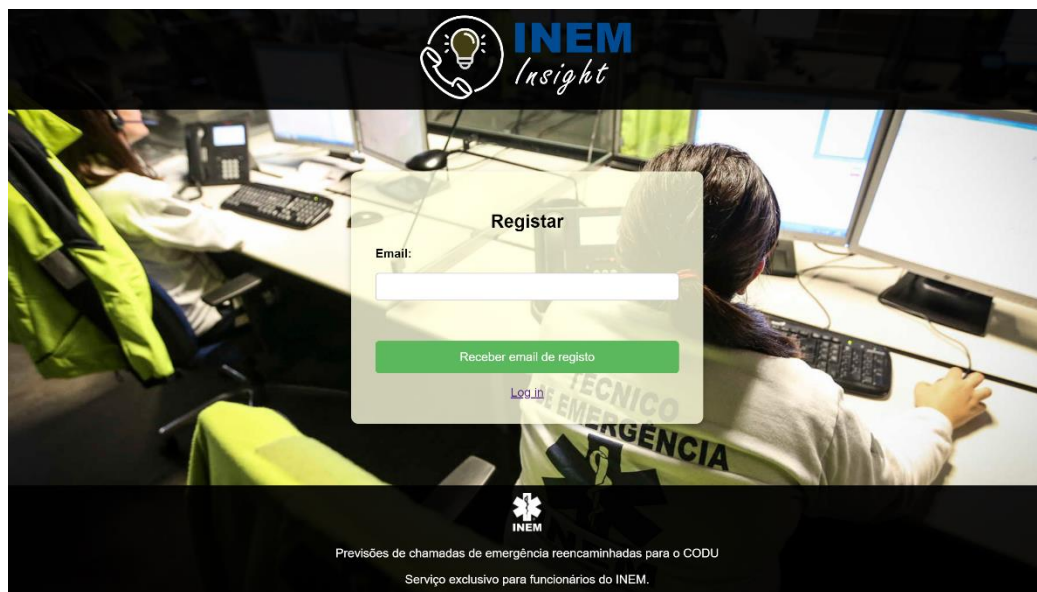


Figura 12 - Página de criação de conta

Os utilizadores introduzem um email, recebem nele um link para continuar o registo, onde inserem a palavra-passe e criam então assim a sua conta. O nome de utilizador é sempre a primeira parte do email, até ao “@”. Desta forma, os utilizadores criam então uma conta pessoal para poderem aceder o site.

### 8.3 Previsões customizáveis

Nesta página, teremos então a possibilidade de recolha de previsões tendo em conta aquilo que o utilizador procurar.

Figura 13 - Página de previsões personalizadas

Nesta página em questão, temos então dois *inputs*, um para o tipo de previsões: de 30 minutos, de turnos, diários... Para além deste, temos ainda o *input* para o intervalo de tempo que se quer verificar e um *slider* para escolher em que formato se quer gerar o relatório, PDF ou excel.

#### 8.3.1 Relatório PDF

Com estes *inputs* inseridos, há então a oportunidade de gerar um relatório de resultados, onde um pedido é enviado para a API com os parâmetros inseridos pelo utilizador. Feito isto, quando uma resposta é recebida, uma função “*generatePDF*” é inicializada, onde a resposta é enviada como parâmetro em forma de JSON. Esta função posteriormente utiliza a biblioteca jsPDF e autotable, para criar ficheiros PDF e para criar uma tabela nesse mesmo PDF respetivamente.

```
const { jsPDF } = window.jspdf;

const doc = new jsPDF({
  format: 'a4',
  unit: 'pt',
  orientation: 'portrait'
});
```

Figura 14 - Inicialização da biblioteca e do documento PDF

Na imagem anterior, temos então a recolha da biblioteca jsPDF através do comando `window.jspdf`, sendo que na página *HTML* em questão tem o link para o script desta biblioteca.

Feito isto, e aplicados os diferentes conteúdos ao PDF para o popular da forma pretendida, será necessário então usar a biblioteca *autotable* para construir a tabela que irá conter as previsões.

```
doc.autoTable({
  startY: line,
  head: headers,
  body: rows,
  alternateRowStyles: alternateRowStyles,
  theme: 'grid',
  styles: {
    halign: 'center',
    fontSize: 8,
    cellPadding: 5,
    lineWidth: 0,
    overflow: 'linebreak',
  },
  margin: { bottom: 40 },
  columnStyles: {
    0: { cellWidth: 180 },
    1: { cellWidth: 'auto' },
    2: { cellWidth: 'auto' },
    3: { cellWidth: 'auto' },
    4: { cellWidth: 'auto' },
    5: { cellWidth: 100 },
  },
  didDrawPage: function (data) {
    var pageHeight = doc.internal.pageSize.height;
    var marginBottom = 40;
    doc.setFillColor(255, 255, 255);
    doc.rect(0, pageHeight - marginBottom, doc.internal.pageSize.width, marginBottom, 'F');
  },
});
```

**Figura 15 - Código para a inserção da tabela no PDF**

Evidentemente, no código acima, inserimos uma tabela no documento criado anteriormente, onde atribuímos:

- uma altura na página do documento, através do *startY*;
- um *array* com os títulos das colunas, passado como *head*;
- no *body* da tabela, passamos as entradas da mesma, através de um *array* criado e manipulado tendo em conta o tipo de previsão que é pedido;
- é também passado um atributo *style* que é aplicado às linhas ímpares. Neste caso, as linhas ímpares ficam com um fundo amarelo-claro;
- atribuem-se os estilos base necessários para a tabela através do *styles*;
- em *columnStyles*, aplicamos a largura predefinida para cada coluna da tabela.
- por fim, temos ainda uma outra função, atribuída ao *didDrawPage*, que quando a tabela preenche uma página e passa para uma nova, aplica então estas mudanças na nova página. Estas mudanças em questão estão feitas para que exista um espaçamento entre a tabela e o rodapé de cada página.

Com efeito, para visualizar um exemplo de relatório, dirija-se a **Anexo III – Relatório por turnos criado pela aplicação**

### 8.3.2 Relatório Excel

Da mesma forma, que no relatório PDF, o pedido à base de dados acontece exatamente da mesma forma no relatório Excel. Deste modo, a única coisa a fazer quando os dados são recolhidos é aplicar a formatação na coluna da data.

Feito isto, será apenas necessário aplicar algumas funções da biblioteca XLSX, que foi importada para o próprio HTML, para poder gerar e descarregar o ficheiro.

```
async function jsonToExcel(data, filename) {
  const formattedData = formatDatesInData(data);

  const workbook = XLSX.utils.book_new();
  const sheet = XLSX.utils.json_to_sheet(formattedData);
  XLSX.utils.book_append_sheet(workbook, sheet, 'Sheet1');

  const wbout = XLSX.write(workbook, { bookType: 'xlsx', type: 'binary' });
  const blob = new Blob([s2ab(wbout)], { type: 'application/octet-stream' });

  const url = URL.createObjectURL(blob);
  const a = document.createElement('a');
  a.href = url;
  a.download = filename || 'data.xlsx';
  document.body.appendChild(a);
  a.click();
  setTimeout(() => {
    document.body.removeChild(a);
    window.URL.revokeObjectURL(url);
  }, 0);
}
```

**Figura 16 - Código para gerar relatórios Excel**

Com efeito, a ideia de permitir agrupar os dados em Excel surgiu pela facilidade de agrupamento e de busca de dados. Evidentemente, num relatório PDF temos dados numa forma mais limpa, e conseguimos fazer procuras através de palavras-chave. Contudo, com ficheiros Excel, facilita-se o manuseio desses mesmos resultados. De facto, a opção entre PDF ou Excel será uma mais-valia para o INEM, por essas mesmas razões.

Em baixo, segue um excerto de um ficheiro Excel gerado pelo INEM Insight:

	A	B	C	D	E
1	Dia	Hora	Valor Previsto		
2	segunda-feira, 24 de junho de 2024	00:00:00	41		
3	segunda-feira, 24 de junho de 2024	00:30:00	38		
4	segunda-feira, 24 de junho de 2024	01:00:00	35		
5	segunda-feira, 24 de junho de 2024	01:30:00	33		
6	segunda-feira, 24 de junho de 2024	02:00:00	31		
7	segunda-feira, 24 de junho de 2024	02:30:00	29		
8	segunda-feira, 24 de junho de 2024	03:00:00	27		
9	segunda-feira, 24 de junho de 2024	03:30:00	25		
10	segunda-feira, 24 de junho de 2024	04:00:00	25		
11	segunda-feira, 24 de junho de 2024	04:30:00	25		
12	segunda-feira, 24 de junho de 2024	05:00:00	24		

**Figura 17 - Exemplo de um relatório em Excel**

## 8.4 Dashboard

Efetivamente, esta será a *landing page* do site, e por isso mesmo há uma responsabilidade acrescida ligada à mesma. Decerto, será necessário incluir detalhes úteis para os utilizadores, e para isto será também necessário refletir um pouco sobre aquilo que é necessário saber para melhor gerir os recursos humanos do CODU.

Neste sentido, atendendo um pouco à investigação realizada no capítulo 6, entende-se que o INEM tem a possibilidade de gerir numa natureza de “última hora”. Isto que, foi uma forma de gestão implementada durante o COVID que permite até ter operadores que fazem turnos mais curtos e que podem até estar de serviço entre dos turnos, atendendo ao necessário.

Assim, será bastante útil passar informações imediatas nesta primeira página, sendo que identificámos como úteis as previsões para as seguintes seis meias horas. Aqui, é dada a hipótese do INEM identificar quais as necessidades momentâneas, o que auxilia até os próprios operadores a escolherem melhor as suas pausas durante o seu turno de trabalho.

Para além disso, pensou-se também ser importante perceber quantos operadores serão necessários nos próximos turnos. Efetivamente, pela mesma razão dada para os seguintes intervalos de meia hora, será também útil entender quantos operadores deverão estar escalados para os seguintes turnos. Isto para, caso seja necessário, possa haver correções de última hora.

Por último, será ainda útil fornecer as previsões para os dias seguintes, não só para dar uma perspetiva diária ao INEM, como para dar a entender a flutuação que acontece de acordo com os dias da semana.

De facto, para ter estes dados presentes na *dashboard*, realizamos três pedidos diferentes à *API*, utilizando as funcionalidades de script do *HTML*, neste caso em *JavaScript*. Assim, tendo já o número de chamadas previstas a chegar via *API*, será ainda necessário traduzir estas chamadas para o número de operadores necessários para atender às mesmas. Para isto, segue-se a equação já mencionada em 6, a fórmula de Erlang C que será traduzida para *JavaScript* [ECPYHTON]. Evidentemente, optou-se por fazer a conversão no front-end porque esta equação contém variáveis que influenciam o resultado que podem ser úteis para o INEM tendo em conta as circunstâncias. Neste momento, estas variáveis estão padronizadas em **80%** das chamadas que chegam ao CODU serem atendidas em menos de **20 segundos**, sendo que estimamos que cada chamada durará 5 minutos em média. Com efeito, estes valores rapidamente são alterados para aquilo que o INEM considerar necessário, e podem até ser criados inputs dentro do site para permitir que o cálculo use os valores padrão, ou permitir uma busca avançada com valores de variáveis inseridas.

Neste sentido temos o seguinte código:

```

function ErlangC(A, N) {
  if (N - A <= 0) {
    return 1;
  }
  const L = (Math.pow(A, N) / factorial(N)) * (N / (N - A));
  let sum_ = 0;
  for (let i = 0; i < N; i++) {
    sum_ += Math.pow(A, i) / factorial(i);
  }
  return L / (sum_ + L);
}

function calc_sl(calls, intervalo, avgCallDuration, targetServiceLevel, avgWaitTime) {
  if (avgCallDuration === 0) {
    return 0;
  }

  const T = avgWaitTime; // tempo esperado pretendido
  const A = (calls * avgCallDuration) / intervalo;
  const aht = avgCallDuration; // média de tempo de resposta

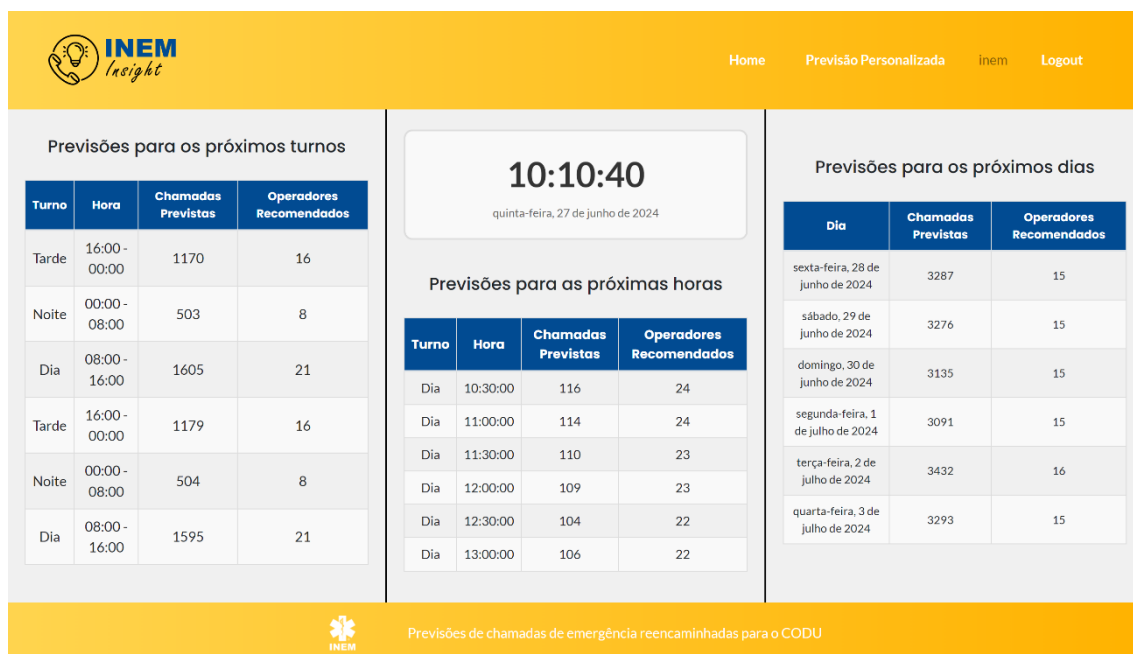
  for (let staff = 0; staff < 100; staff++) {
    const P = ErlangC(A, staff);
    const sl = (1.0 - (P * Math.exp(-(staff - A) * (T / aht)))) * 100.0;
    if (sl >= targetServiceLevel) { // percentagem de chamadas que são atendidas dentro do intervalo de tempo pretendido
      return staff;
    }
  }
  return null;
}

```

**Figura 18 - código da função para cálculo do número de operadores necessários para um número de chamadas previstas**

Com efeito, esta função é aplicada tanto às entradas das tabelas a ser apresentadas na *landing page*, tal como nos relatórios gerados pelo INEM Insight.

Neste sentido, construiu-se a página inicial seguinte:



**Figura 19 - Dashboard INEM Insight**

É de notar ainda que quaisquer valores apresentados neste relatório, apresentam dados de chamadas previstas forjadas, por questões de segurança de dados.



## 9 Resultados

Com efeito, este trabalho final de curso foi então separado em três núcleos. Como referido em 4, esta foi a melhor opção a seguir para que cada processo do projeto possa funcionar de forma autónoma.

Neste sentido, conseguimos ter um projeto que consideramos bastante forte em todas as vertentes. Contudo, é possível ainda melhorar, nomeadamente na parte do algoritmo. Apesar de termos já um algoritmo bastante forte, há alguns pontos que este ainda não consegue cobrir, tais como a espontaneidade.

Com isto, refiro-me a eventos ou vagas de calor, coisas que não acontecem numa data determinada. Isto é ainda um problema, visto que o algoritmo consegue apenas ver padrões normais dentro daquilo que é a realidade para certos dias. Neste sentido, diria que temos entre 70% a 80% das previsões muito aproximadas daquilo que é o real para todo o ano, faltando então esta última parte para poder tornar este algoritmo ótimo.

Por sua vez, a *API* e toda a base de dados terá de ser migrada para os servidores do INEM visto que neste momento está tudo alocado de forma local, visto que o objetivo era apenas testar e demonstrar as funcionalidades. Isto porque, qualquer que fossem os serviços escolhidos, seria sempre necessário fazer esta migração para que o INEM possa usufruir desta aplicação.

Em relação à aplicação INEM Insight, esta terá também de ser migrada para um servidor do INEM, visto que neste momento está alocada no *pythonanywhere*, que como serviço gratuito, possui uma data-limite para os sites que disponibiliza.

Em suma, os três núcleos estão prontos a ser utilizados e implementados na gestão de recursos Humanos do INEM, sendo apenas necessário realizar a migração para os seus servidores.

Para além disso, temos ainda algumas modificações que podem melhorar o algoritmo. E para isso, deixamos o desafio para um futuro trabalho realizado num contexto semelhante a este, para melhorar não só o sistema desenvolvido ao longo deste trabalho final de curso, como o sistema de emergência médica nacional, sendo que temos em mãos algo que tem potencial para salvar vidas.



## Bibliografia

- [112] Ligue 112, disponível em: <https://www.inem.pt/2017/05/30/ligue-112/> , acedido em: 15 de outubro de 2023
- [ERLANG] Fórmula de Erlang, disponível em: [https://en.wikipedia.org/wiki/Erlang\\_\(unit\)](https://en.wikipedia.org/wiki/Erlang_(unit)) , acedido em: 30 de outubro de 2023
- [JN] “Inteligência artificial vai gerir sobrecarga de chamadas no 112”. Artigo disponível em: <https://www.jn.pt/3237496914/inteligencia-artificial-vai-gerir-sobrecarga-de-chamadas-no-112/> , acedido em: 30 de outubro de 2023
- [REPTVI] “Pilotos do INEM em greve e profissionais que atendem o 112 à beira do esgotamento”. Reportagem feita pelo canal televisivo TVI a 3 de novembro de 2023 disponível em: <https://tviplayer.iol.pt/programa/tvi-jornal/63ef5eb50cf2665294d5f87a/video/6544edf80cf25f9953896dbb> , acedido em: 6 de novembro de 2023
- [INEMTRA] Site de transparência do INEM, disponível em: <https://transparencia.sns.gov.pt/explore/dataset/atividade-gripe-inem/table/?sort=periodo> , acedido em: 6 de novembro de 2023
- [R2] O que é o Coeficiente de determinação? disponível em: [https://wikiciencias.casadasciencias.org/wiki/index.php/Coeficiente\\_de\\_determinacao](https://wikiciencias.casadasciencias.org/wiki/index.php/Coeficiente_de_determinacao) , acedido em: 6 de novembro de 2023
- [ARTJORI] “Caos no 112. Em 8 horas mais de 170 chamadas ficaram sem resposta”, artigo sobre o sobrecarregamento de chamadas no CODU, disponível em: [https://ionline.sapo.pt/artigo/621590/caos-no-112-em-8-horas-mais-de-170-chamadas-ficaram-sem-resposta-?seccao=Portugal\\_i](https://ionline.sapo.pt/artigo/621590/caos-no-112-em-8-horas-mais-de-170-chamadas-ficaram-sem-resposta-?seccao=Portugal_i) , acedido em: 6 de novembro de 2023
- [IAENG] Artigo sobre a razão porque grande parte das aplicações com inteligência artificial falham, disponível em: <https://www.gartner.com/en/newsroom/press-releases/2020-10-19-gartner-identifies-the-top-strategic-technology-trends-for-2021#:~:text=Gartner%20research%20shows%20only%2053,a%20production%20grade%20AI%20pipeline> , acedido em: 12 de novembro de 2023
- [IAFORBES] Artigo sobre como não falhar num projeto com elementos de *machine learning*, disponível em: <https://www.forbes.com/sites/forbestechcouncil/2023/04/10/why-most-machine-learning-applications-fail-to-deploy/?sh=3b3a1ae0736d> , acedido em: 12 de novembro de 2023
- [JMJOBS] Artigo sobre número de pessoas que viajaram até Lisboa na JMJ, disponível em: <https://observador.pt/2023/08/06/vigilia-e-missa-no-parque-tejo-foram-os-setimos-maiores-eventos-de-sempre-numa-jmj/> , acedido em: 16 de dezembro de 2023
- [TEMPAGO] Artigo sobre o nível de calor que agosto atingiu, disponível em: [https://www.ipma.pt/pt/media/noticias/news.detail.jsp?f=/pt/media/noticias/textos/Boletim\\_climatologico\\_agosto\\_2023.html](https://www.ipma.pt/pt/media/noticias/news.detail.jsp?f=/pt/media/noticias/textos/Boletim_climatologico_agosto_2023.html) , acedido em: 16 de dezembro de 2023

- [DOCKER] Página da wikipédia sobre o Docker, disponível em: [https://pt.wikipedia.org/wiki/Docker\\_\(software\)](https://pt.wikipedia.org/wiki/Docker_(software)) , Acedido em 27 de março de 2024
- [ECPYHTON] Post no stackoverflow, sobre a fórmula de Erlang C em python, disponível em: <https://stackoverflow.com/questions/65398426/how-to-calculate-amount-of-call-centers-operators-erlang-calculator-on-python> , Acedido em 27 de junho de 2024

## Glossário

CODU	Centros de Orientação de Doentes Urgentes
INEM	Instituto Nacional de Emergência Médica
PDF	Portable Document Format
API	Application Programming Interface (interface de programação de aplicativos)
DEISI	Departamento de Engenharia Informática e Sistemas de Informação
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
PSP	Polícia de Segurança Pública
GNR	Guarda Nacional Republicana
IA	Inteligência Artificial
EMAP	Erro Médio Absoluto Percentual
IPMA	Instituto português do mar e da atmosfera
BD	Base de Dados

## Anexo I – Fichero de valores previstos

*Excell Sheet* com tabelas completas das previsões de **6.4.2** e **6.4.3**:

[https://docs.google.com/spreadsheets/d/1q7mcxN6yXWjApiGmnk3yB1YuRx3DKWwm6FDxZO9B\\_aQ/edit#gid=1678086769](https://docs.google.com/spreadsheets/d/1q7mcxN6yXWjApiGmnk3yB1YuRx3DKWwm6FDxZO9B_aQ/edit#gid=1678086769)


Neste caderno, estão duas páginas, a primeira é então a página de previsões horárias para o mês de março de 2023, e a segunda página é então a previsão por turnos que possui um filtro na coluna das horas para facilitar o estudo da tabela, podendo alternar entre uma hora do turno da manhã, tarde ou noite, para receber valores da soma dessas horas para esses dias.

## **Anexo II – Código referente ao algoritmo**

O código referente ao algoritmo está todo presente no repositório git, sendo que este é uma sub-secção de uma organização no github criada pelos docentes da Universidade Lusófona. O mesmo pode ser verificado em:

<https://github.com/DEISI-ULHT-TFC-2023-24/DEISI39>

## Anexo III – Relatório por turnos criado pela aplicação

		
<b>Previsões Diárias</b>		
Data inicial: 24/06/2024 Data final: 28/06/2024		
Data	Chamadas Previstas	Operadores Recomendados
segunda-feira, 24 de junho de 2024	3263	15
terça-feira, 25 de junho de 2024	3263	15
quarta-feira, 26 de junho de 2024	3263	15
quinta-feira, 27 de junho de 2024	3263	15
sexta-feira, 28 de junho de 2024	3263	15

Página 1

## Anexo III – Ligações relevantes

Vídeo de apresentação da aplicação INEM Insight:

[https://youtu.be/T\\_Zh3zR3OQ0](https://youtu.be/T_Zh3zR3OQ0)

Link para o repositório GitHub com ficheiros de código utilizados pelo trabalho:

<https://github.com/DEISI-ULHT-TFC-2023-24/DEISI39---Otimiza-o-de-recursos-atrav-s-de-an-lise-de-chamadas-de-emerg-ncia>

Para testar este projeto, basta dirigir ao github e descarregar os ficheiros `api_code.py` e `predsHorarias.xlsx`, e criar um container no Docker com as portas 1433, 1440 e 5000 ativas. Depois, colocar os dois ficheiros anteriores dentro do container.

Um exemplo de código na linha de comandos do computador seria: `docker cp api_code.py inem:/home/inem`.

Após a inserção dos dois ficheiros, mesma pasta, correr o ficheiro `api_code.py` através do código:

```
docker exec -it inem python3 /home/inem/api_code.py
```

Com os últimos passos completos, resta dirigir-se à aplicação e testá-la por si!

Link para o website:

<https://inem.pythonanywhere.com/inemapp/dashboard/>

Para entrar no site usar as credenciais criadas no vídeo:

Nome de utilizador: a21807126.ulht

Palavra-passe: INEM2324

Deve se referir que, esta representa apenas as partes 2 e 3 do projeto, sendo que a primeira parte, para gerar previsões de chamadas, utiliza dados sensíveis do INEM e daí não ser possível testar essa parte.

De ressaltar também, todos os dados presentes no ficheiro `predsHorarias.xlsx`, não representam valores previstos pela aplicação. São sim valores forjados para fins de teste.