Instituto Politécnico de Viseu
Escola Superior de Tecnologia e Gestão de Viseu

Francisco António Alves Sério

Modelos de Machine Learning na gestão de consumos de energia

Outubro de 2020

IPV - ESTGV | 2020
Modelos de Machine Learning na gestão de consumos de energia
Francisco Sério

Instituto Politécnico de Viseu

Escola Superior de Tecnologia e Gestão de Viseu

Francisco António Alves Sério

Modelos de Machine Learning na gestão de consumos de energia

Tese de Mestrado

Sistemas e Tecnologias de Informação para as Organizações

Professora Doutora Ana Cristina Wanzeller Guedes de Lacerda

Professor Doutor Filipe Cabral Pinto

Outubro de 2020

Aos meus pais

RESUMO

Ao longo destes últimos anos, a quantidade de sensores espalhados pelas cidades tem aumentado significativamente, o que, por consequência, leva a um incremento no volume de dados, originando o *Big Data*. Muitos desses sensores foram colocados em candeeiros e em contadores de energia elétrica, permitindo, tanto ao utilizador, como ao município, verificar os respetivos consumos, muitas vezes em tempo real, fazendo da cidade uma cidade inteligente.

Com essa grande quantidade de dados gerada, seria possível aplicar técnicas de *machine learning*, com o objetivo de fazer previsões de dados no tempo, encontrar anomalias, efetuar algumas estatísticas e retirar informações úteis. Isto permite que o município consiga ir em conta aos seus objetivos, tornando a cidade numa cidade sustentável, melhorando, assim, a qualidade de vida dos seus cidadãos.

Em suma, com este trabalho pretende-se criar modelos de *machine learning*, utilizando bibliotecas de código aberto (e. g. TensorFlow, Keras) sobre dados reais de energia elétrica de uma cidade, com o objetivo de prever os consumos para os próximos tempos, de forma a que o município tenha uma melhor tomada de decisão. Aliada a esta previsão, pretende-se, também, criar uma REST API que disponibilize essas previsões numa ferramenta de *business intelligence*, para que o município possa ter uma melhor visão das mesmas.

No geral, com a previsão dos consumos, será possível resolver o problema, que é mútuo não só a municípios, mas também a outras entidades, da verificação da gestão do orçamento em relação à energia e indo em conta às suas expectativas.

ABSTRACT

Over the past few years, the number of sensors spread across cities has significantly increased, which, consequently, leads to an increase in data volume, giving rise to Big Data. Many of these sensors were placed in street lamps and electricity meters, allowing both the user and municipal entities to check their consumptions, often in real time, making the city a smart city.

With this large amount of generated data, it would be possible to apply machine learning techniques, with the objective of making data predictions over time, finding anomalies, performing some statistics and finding useful information. This allows the municipal entity to reach its objectives, making the city a sustainable city, improving the quality of its citizens' life.

In short, this work intends to create machine learning models, using open source libraries (e. g. TensorFlow, Keras) on real electric energy data from a city, in order to predict consumption data for the next times, so that the municipality has better decision-making. Allied to this forecast, it is also intended to create a REST API that makes these forecasts available in a business intelligence tool, so that the municipality can have a better view of them.

In general, with the consumption forecasts, it will be possible to solve the problem, which is mutual not only for municipal entities, but also for other entities, of verifying the management of the budget in relation to energy and over-reaching their expectations.

PALAVRAS CHAVE

Cidades Inteligentes
Consumo de energia
Internet das coisas
Big Data
Business Intelligence
Machine Learning
LSTM

KEY WORDS

Smart Cities
Energy consumption
Internet of Things
Big Data
Business Intelligence
Machine Learning
LSTM

AGRADECIMENTOS

Em primeiro lugar, gostaria de agradecer à minha família, em especial aos meus pais que, não só me apoiaram durante o desenvolvimento desta tese, mas também em todo o meu percurso académico.

Um forte agradecimento para a professora Doutora Ana Cristina Wanzeller Guedes de Lacerda e para o professor Doutor Filipe Cabral Pinto que me ajudaram desde o início da tese e, no geral, durante o meu percurso académico de mestrado.

Por último, gostaria de agradecer à Altice Labs, por me ter dado a oportunidade e o tempo necessário para o desenvolvimento desta tese e a todos os meus colegas de trabalho e amigos que tanto me apoiaram.

Muito obrigado a todos.

ÍNDICE GERAL

ÍNDICE DE FIGURAS

ÍNDICE DE QUADROS

ABREVIATURAS E SIGLAS

ML	Machine Learning
LSTM	Long Short-Term Memory
API	Application Programming Interface
ARIMA	Autoregressive Integrated Moving Average
RF	Random Forest
BI	Business Intelligence

1. Introdução

Este capítulo está dividido em quatro secções: enquadramento, motivação, objetivo e estrutura do documento.

1.1 Enquadramento

Atualmente, a energia elétrica é um bem essencial para todos, sendo utilizada não só nas nossas casas, através de equipamentos eletrónicos, mas também na agricultura, indústria, iluminação de vias públicas e municípios.

Com a rápida urbanização das cidades, aliada ao crescimento da população, os municípios tendem a enfrentar, cada vez mais, o problema da gestão dos consumos de energia (Ejaz, Naeem, Shahid, Anpalagan e Jo, 2017). Com a resolução deste problema, os municípios teriam maior facilidade de alcançar metas de sustentabilidade propostas, que alguns são obrigados a cumprir, melhorando, assim, a qualidade de vida dos seus habitantes. Com isto, os municípios teriam, também, uma maior visão da sua economia, podendo reduzir custos em algumas áreas, o que, por outro lado, resultaria num possível aumento de desempenho noutras mais importantes, como a de saúde e educação (Pérez-Chacón, Luna-Romera, Troncoso, Martínez-Álvarez & Riquelme, 2018).

Uma possível resolução para o problema anterior passa pela utilização de algoritmos de *machine learning* em dados de energia para a previsão dos respetivos consumos. Isto permite ao município ter uma melhor visão dos consumos, ajudando-o a baixar custos e a ter uma melhor tomada de decisão, indo, assim, em conta às suas expectativas.

Este é um tema muito pouco investigado, uma vez que os estudos têm-se centrado no consumo de energia elétrica de edifícios ou de determinados componentes isolados e não em cidades onde existem múltiplos fatores que podem influenciar as respetivas previsões.

1.2 Motivação

Devido à importância da previsão dos consumos de energia, muitos investigadores estão a explorar a utilização de diferentes tipos de algoritmos de *machine learning* com o objetivo de melhorar essa previsão. Alguns destes exemplos são: Árvores de decisão e *Random Forest*, Regressão linear e não linear, Redes neuronais e Redes Bayesianas (Chatterjee, Sk, Singh and Sanyal, 2019).

A previsão de consumos de energia de cidades não era tão fácil de realizar como nos dias de hoje, devido à não disponibilização de dados por parte dos parceiros (Lima, 2015). Atualmente, já existem equipamentos como, por exemplo, contadores inteligentes, que apresentam informações relativa aos consumos numa página web, onde praticamente cada utilizador desses equipamentos, através da sua conta pessoal, consegue aceder.

O aumento desses dados disponíveis, que origina o *Big Data*, está diretamente relacionado com o aumento dos dispositivos de IoT. Estima-se que, nos próximos anos, o número de dispositivos crescerá exponencialmente para 50 bilhões (Albertin e de Moura Albertin, 2017), atingindo, um crescimento de, aproximadamente, 140%.

Assim como a IoT e o *Big Data* estão diretamente relacionados, também estão a análise de dados e o *Business Intelligence*. Estas áreas poderão ser bastante úteis, não só para empresas, pois são capazes de oferecer indicadores e estatísticas para uma melhor tomada de decisão das mesmas, levando a uma possível vantagem competitiva no mercado, mas também para municípios, podendo estes verificar anomalias nos relatórios em relação à energia e outras métricas, melhorando, assim, a gestão da cidade no que respeita ao consumo energético (Mohammadi, Al-Fuqaha, Sorour and Guizani, 2018).

1.3 Objetivos

Assim sendo, com a realização deste projeto pretende-se criar modelos de *machine learning* sobre dados reais do consumo da energia de uma cidade, para que o município consiga visualizar a previsão dos consumos nos próximos tempos. Com isto, espera-se que o município consiga obter uma melhor tomada de decisão com base nas previsões, bem como gerir melhor os recursos, evitando desperdícios e indo ao encontro das suas expectativas.

No fim da realização do projeto, é esperada a obtenção de conhecimentos aprofundados na área de *Machine Learning*, podendo, estes, serem aplicados não só a este cenário, mas também em outros, como previsão da qualidade da água dos rios ou do ar.

Também, com este projeto, pretende-se dar um contributo na comunidade científica no que diz respeito à previsão de consumos de energia, em cenários de *smart cities*.

1.4 Metodologia

Para levar a cabo a concretização do trabalho, foram seguidos os seguintes passos:

1 – Estudo e escolha dos algoritmos de *machine learning* mais apropriados para o problema em questão, para fazer previsões de consumos de energia elétrica com dados baseados no tempo

2 – Análise do conjunto de dados a utilizar no projeto

3 – Processamento dos dados

4 – Construção de vários modelos com base nos algoritmos escolhidos no estado da arte

5 – Ajuste do modelo que melhores resultados obteve, como objetivo de melhorar as previsões de novos dados

6 – Previsão de novos dados para além do conjunto de dados

7 – Implementação de uma REST API para treinar o modelo escolhido e disponibilizar as previsões de novos dados com base nesse modelo

8 – Apresentação das previsões numa ferramenta de *business intelligence*

A Figura 1.1 apresenta a abordagem graficamente.

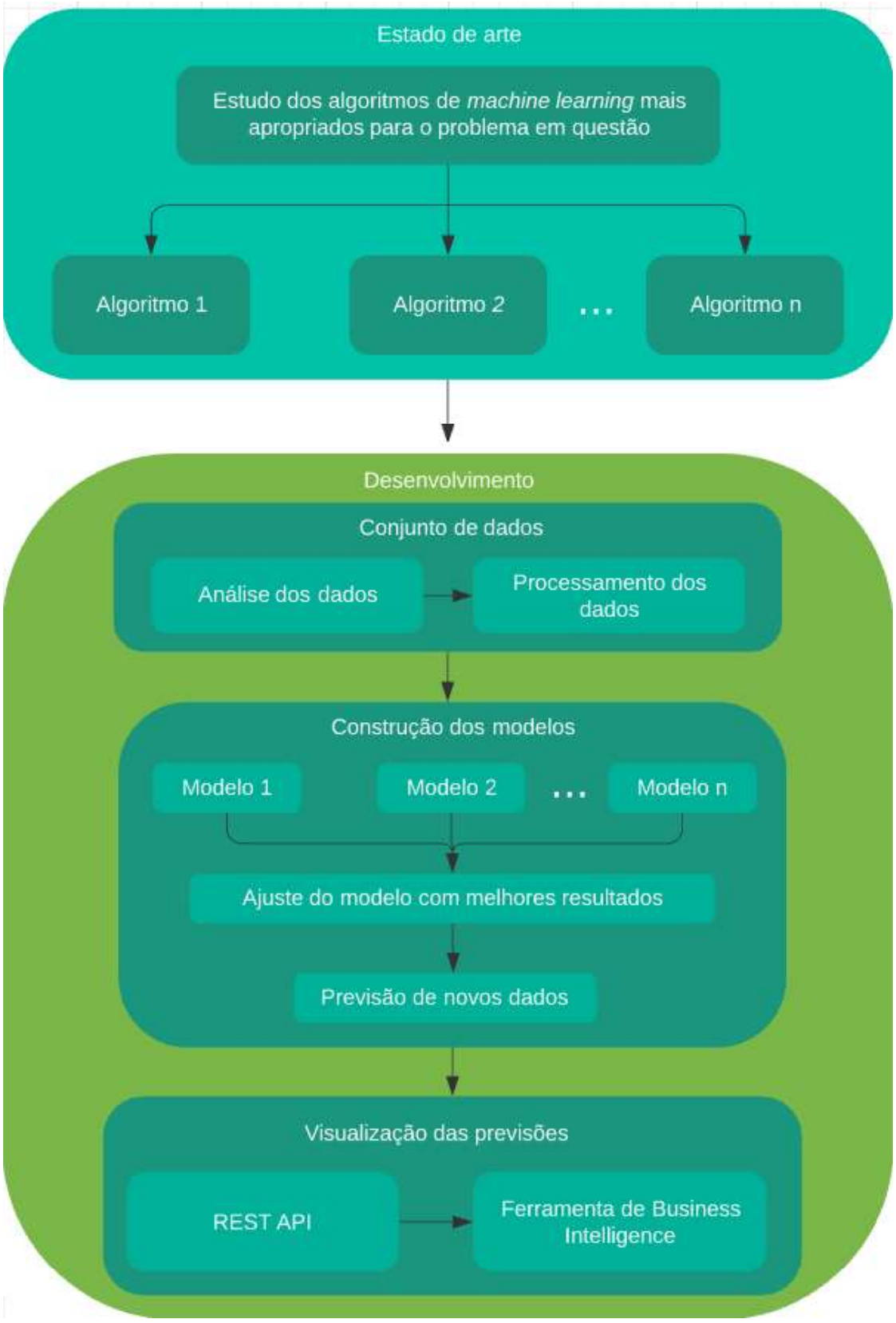


Figura 1.1: Representação gráfica da metodologia a utilizar no projeto

Para ajudar no desenvolvimento de projetos de *Data Mining* e *Machine Learning,* é utilizada a metodologia SEMMA (*Sample Explore Modify Model Assess*). Esta é constituída por uma série de etapas sequenciais e iterativas, nomeadamente Amostra, Explorar, Modificar, Modelar e Avaliar (Figura 1.2).

Esta metodologia foi a escolhida para o desenvolvimento deste projeto, não só por assentar no objetivo do mesmo, mas também por oferecer um processo bastante fácil de perceber, permitindo uma gestão rápida e organizada do desenvolvimento deste (Dåderman e Rosander, 2018).

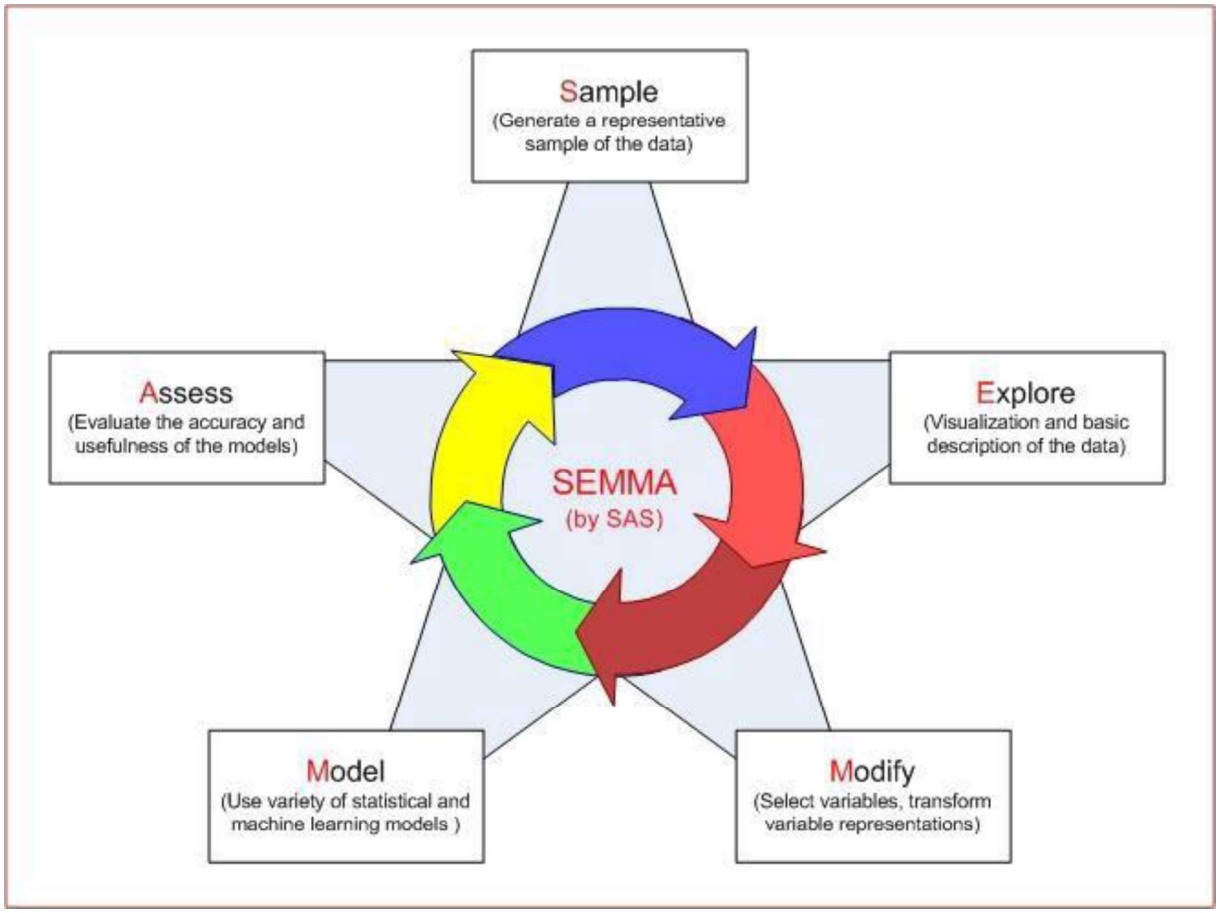


Figura 1.2: Visão geral da metodologia SEMMA[1]

Esta metodologia é composta por 5 fases que são as suas iniciais.

A primeira fase, amostra, concentra-se na recolha de amostras de dados que devem ter uma quantidade significativa para que contenha informação suficiente para realizar, no caso deste projeto, os algoritmos de *machine learning* de forma eficiente.

A segunda fase, explorar, é a fase que abrange a compreensão e visualização dos dados com o objetivo de, não só encontrar relações entre eles, mas também anomalias.

[1] Fonte: https://paulovasconcellos.com.br/crisp-dm-semma-e-kdd-conheça-as-melhores-técnicas-para-exploração-de-dados-560d294547d2

A terceira fase, modificar, é a fase de preparação da modelação dos dados, onde se criam e transformam as variáveis.

A quarta fase, modelo, é a fase onde se foca na modelação dos dados (no caso deste projeto, criam-se os modelos de *machine learning*) com as variáveis e dados preparados nas etapas anteriores.

A última fase, avaliar, concentra-se em avaliar a utilidade e a confiabilidade dos resultados obtidos.

1.5 Cronograma

O Quadro 1.1 e a Figura 1.3 apresentam o cronograma do projeto.

Quadro 1.1: Cronograma do projeto

Nome da tarefa	Data início	Data fim
Elaboração da dissertação	04-11-2019	01-10-2020
Elaboração da pré-proposta	04-11-2019	25-11-2019
Entrega da pré-proposta	25-11-2019	25-11-2019
Continuação da pesquisa do estado de arte	25-11-2019	05-03-2020
Apresentação da proposta	14-12-2019	14-12-2019
Recolha/estudo dos dados	14-12-2019	10-01-2020
Estudo dos algoritmos de *machine learning*	04-01-2020	12-03-2020
Processamento e análise dos dados	12-03-2020	02-04-2020
Estudo das bibliotecas de *machine learning*	16-03-2020	14-04-2020
Criação dos modelos de *machine learning*	14-04-2020	13-08-2020
Análise de resultados	13-08-2020	22-08-2020
Previsão de novos dados e análise dos resultados	22-08-2020	10-09-2020
Elaboração da API para visualização dos dados	10-09-2020	26-09-2020
Preparação da apresentação do projeto	26-09-2020	02-10-2020
Entrega do projeto	02-10-2020	02-10-2020
Apresentação do projeto	10-2020	10-2020

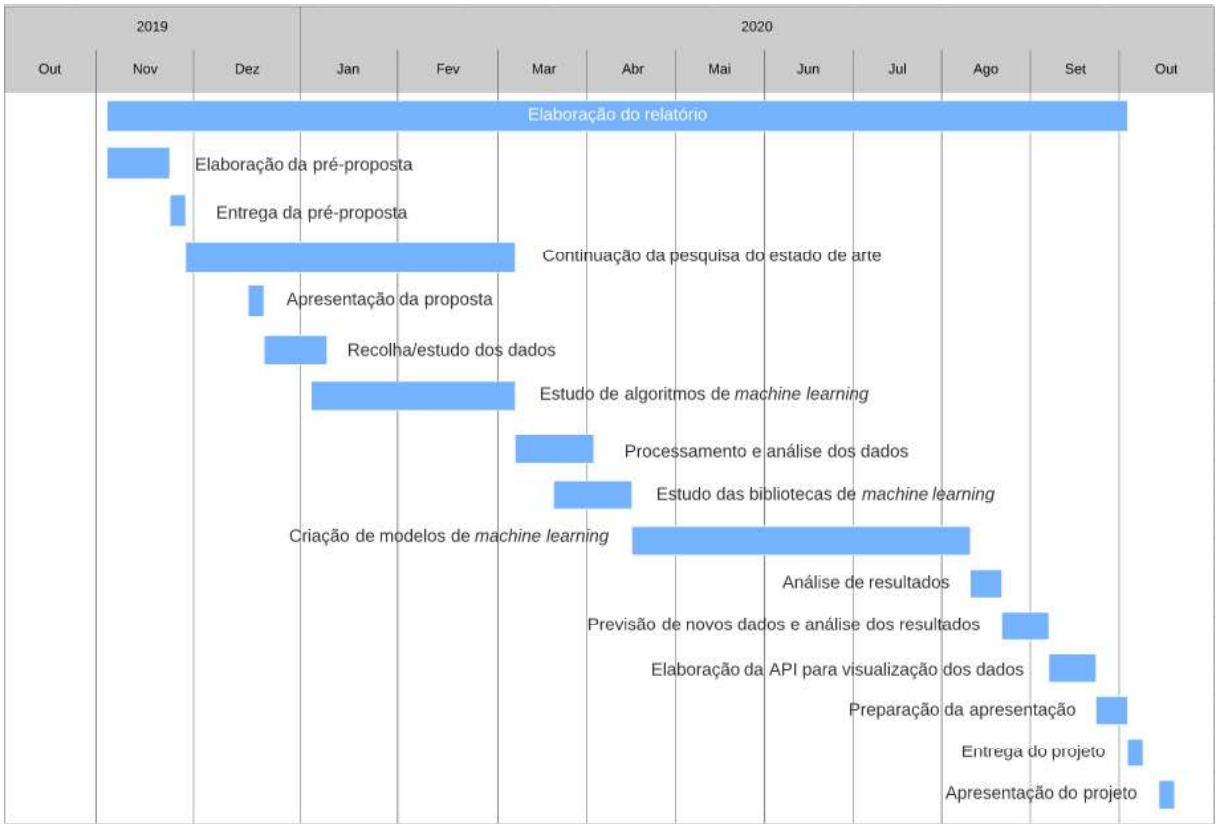


Figura 1.3: Cronograma da tese

1.6 Estrutura do documento

Este documento está estruturado em quatro capítulos, alguns deles com secções, com o objetivo de estes ficarem mais bem organizados e compreensíveis. Os capítulos são: Introdução, Estado da arte, Cenário, Desenvolvimento e Conclusão.

No capítulo da introdução é apresentado um enquadramento geral sobre o propósito deste trabalho, bem como a motivação que levou a ser realizado, os objetivos, a metodologia e o cronograma.

No estado da arte é apresentada a revisão da literatura, abordando várias áreas em que se enquadra o projeto: *Machine Learning*, *Smart Cities*, *IoT*, *Big Data*, previsão dos consumos de energia e *Business Intelligence*.

No capítulo do cenário é apresentado a história de um possível cenário onde é aplicado o projeto.

No capítulo do desenvolvimento são apresentadas as etapas que conduziram à implementação prática do projeto, como análise do conjunto de dados, aplicação da *feature engineering*, comparação dos resultados dos diferentes algoritmos de *machine learning*, ajuste do modelo

escolhido, previsão de novos dados e elaboração de uma REST API para a visualização dos dados.

O último capítulo refere-se à conclusão, onde são expressas as conclusões retiradas no final do desenvolvimento do projeto, bem como os desafios e os trabalhos futuros.

2. Estado da Arte

2.1 *Smart Cities*

O conceito de *Smart Cities,* ao longo dos últimos anos, tem aparecido com mais frequência, sendo que a tendência é para aumentar. Tal como diz Bolívar (2015), nos últimos anos, as cidades estão cada vez mais conscientes do conceito de *smart city* e desenvolvem diversos métodos com o objetivo de se tornarem inteligentes por si só e oferecerem um controlo com maior eficiência dos recursos que a cidade proporciona.

Atualmente existem diversas definições acerca deste conceito que surgiu, pela primeira vez, em 1998 (Anthopoulos, 2015). Segundo Su, Li e Fu, (2011), uma cidade inteligente é a abordagem real do planeta inteligente aplicada a uma região específica, alcançando a gestão informativa e integrada das cidades. O mesmo diz que também se pode dizer que é uma inteligente integração eficaz de ideias de planeamento, modos de construção, métodos de gestão e abordagens de desenvolvimento.

Com uma ideia semelhante, Rana, Luthra, Mangla, Islam, Roderick, e Dwivedi, (2019) dizem que uma cidade inteligente pode ser definida como um território tecnologicamente avançado e moderno que lida com vários aspetos sociais, técnicos e económicos do crescimento, com base em técnicas de computação inteligente, para desenvolver componentes de infraestrutura superiores e serviços.

Por outro lado, com a chegada do conceito de "*Internet of Things*", Arasteh et al (2016) dizem que uma cidade inteligente é equipada com diferentes dispositivos eletrónicos constituídos por diversas aplicações, como, por exemplo, câmaras de vigilância colocadas nas ruas e sensores para sistemas de transporte, visando a melhoria da qualidade de vida dos cidadãos.

Com o rápido crescimento da população nestes últimos anos e a constante migração da população da zona rural para a zona urbana, muitas cidades vão tendo alguns problemas no

que diz respeito à sustentabilidade, afetando a vida dos seus habitantes. Alguns dos problemas estão na gestão dos resíduos, controlo da poluição do ar, congestionamento do trânsito, deterioramento de edifícios, entre outros (Chourabi et al, 2012).

A fim de resolver este tipo de problemas, muitas cidades têm criado iniciativas e projetos, não só para melhorar a qualidade de vida dos seus habitantes, mas também para cumprirem metas de sustentabilidade impostas a estas (Arroub, Zahi, Sabir e Sadik, 2016).

2.2 *Internet of Things*

Atualmente, a internet das coisas (*Internet of Things* - IoT) tem vindo a crescer exponencialmente. A Figura 2.1 mostra uma previsão do número de dispositivos IoT conectados, em bilhões, até 2025.

De acordo com De Francisci Morales, Bifet, Khan, Gama e Fan, (2016), a Internet das coisas é uma rede de dispositivos físicos que originará uma grande quantidade de dados no futuro próximo.

Segundo Patel e Patel (2016), a IoT é uma infraestrutura global para a sociedade da informação, com o objetivo de permitir que as coisas sejam conectadas a qualquer hora, em qualquer lugar, com qualquer coisa e com qualquer pessoa, usando qualquer rede e qualquer serviço.

Com uma outra ideia, Madakam, Ramaswamy e Tripathi, (2015) diz que a IoT é uma revolução tecnológica que representa o futuro da computação e das comunicações, sendo que o seu desenvolvimento depende de inovações técnicas em vários campos importantes, desde sensores sem fios até à nanotecnologia, com o objetivo de os automatizar, monitorar e controlar.

Com uma definição um pouco diferente, Dorsemaine, Gaulier, Wary, Kheir e Urien, (2015) diz que a IoT é um grupo de infraestruturas de dispositivos conectados, dispositivos estes que podem ser sensores, que produzem dados para futuras aplicações de mineração de dados, ou atuadores, onde é possível enviar informações para o dispositivo efetuar uma certa ação.

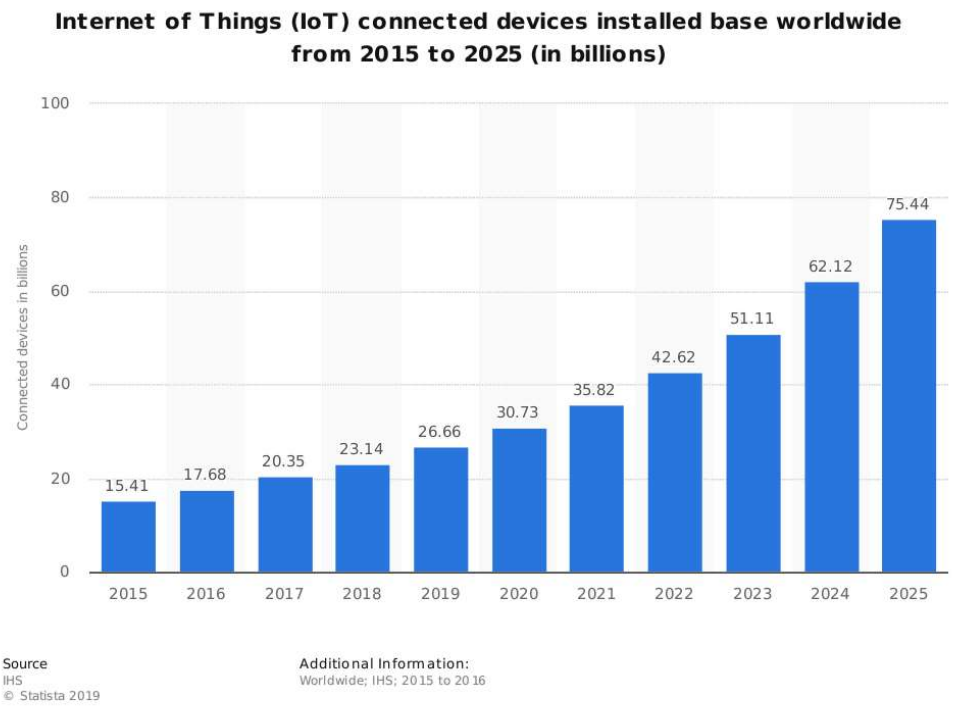


Figura 2.1: Número de dispositivos IoT conectados, em bilhões, de 2015 até 2025[2]

A IoT tem-se destacado, nestes últimos tempos, em cenários de cidades inteligentes e casas inteligentes. Isto tem acontecido devido ao declínio do preço dos dispositivos de IoT, o que faz com que muitos municípios tenham colocado sensores em quase todos os tipos de locais, como, por exemplo, na energia, nos contentores do lixo, nas águas e no transporte, com o objetivo de conhecer o estado dessas métricas (Kim, Ramos e Mohammed, 2017).

Um dos principais problemas das cidades está, precisamente, no uso dessas métricas para gerir a mesma, principalmente nas grandes cidades onde, diariamente, emitem grandes quantidades de emissões poluentes para a atmosfera e gastam enormes quantidades de energia. Com isto, a IoT poderá ter um grande impacto, podendo resolver grande parte dos problemas existentes, contribuindo para que uma cidade, ao mesmo tempo, fique cada vez mais inteligente e sustentável, ajudando os municípios a melhorarem a qualidade de vida dos seus cidadãos (Ahlgren, Hidell e Ngai, 2016).

2.3 *Big Data*

O *Big Data*, à semelhança da IoT, tem tido diversos significados, descritos por investigadores e escritores. Porém, todas essas definições levam a um ponto em comum, que é a grande quantidade de dados.

[2] Fonte: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ [consultado a 18/02/2020]

Segundo De Mauro, Greco e Grimaldi, (2016), Big *Data* é o conjunto de informação caracterizado pelo seu grande volume, velocidade e variedade, que requer tecnologia e métodos analíticos para transformar essa informação em valor.

Com uma ideia semelhante, Youssra e Sara, (2018) diz que *Big Data* refere-se a grandes conjuntos de dados diversificados, constituídos por dados estruturados, semiestruturados e não estruturados, onde estão, cada vez mais, a chegar dados com uma maior rapidez, mencionando, mais uma vez, os três "V's": Volume, Velocidade e Variedade.

Estes dois termos, IoT e *Big Data*, estão diretamente relacionados. De Mauro et al (2016) referem que a internet das coisas gera o *Big Data* por diversas razões, entre elas a velocidade associada aos dispositivos IoT e a grande variedade de dados que diferentes sensores produzem no dia-a-dia, fazendo com que, cada vez mais, haja mais dados.

Para que se possa treinar um dado modelo de *machine learning*, é necessário ter inúmeros dados, cenário esse que é favorável a uma cidade, visto que esta pode produzir dados de diversas fontes, através de dispositivos IoT.

Com este grande fluxo de dados, os algoritmos de *machine learning* podem ser aplicados sem quaisquer problemas. Zhou, Pan, Wang, & Vasilakos, (2017) referem que "O *big data* permite que os algoritmos de machine learning descubram padrões mais detalhados e façam previsões mais precisas do que nunca".

Porém, o uso de grandes quantidades de dados gera alguns problemas, alguns até impossíveis de lidar. Hasan, Shamsuddin, e Lopes, (2014) mencionam que os sistemas computacionais, ao correrem algoritmos de *machine learning*, enfrentam grandes dificuldades no que diz respeito ao hardware, devido ao enorme volume de dados. O mesmo diz que, em alguns casos, até é impossível em algumas arquiteturas de CPU, pelo que é necessário escolher um sistema computacional adequado para o problema e quantidade de dados em questão.

2.4 *Business Intelligence*

Outra área de interesse relevante e que não pode ser deixada de ser mencionada é o *business intelligence*.

Segundo Stackowiak, Rayman e Greenwald, (2007), *Business Intelligence* (BI) é o processo de recolher grandes quantidades de dados, com o objetivo de analisá-los e apresentar um conjunto de relatórios de alto nível que compreendem a essência desses dados na base das ações de negócios, permitindo que a equipa de gestão tome decisões comerciais diárias fundamentais.

O BI pode ter um papel preponderante tendo em conta os dados provenientes das cidades inteligentes, nomeadamente, oferecer uma boa decisão final para, no final, ter um aumento do retorno. Esta ideia pode ser confirmada através de Santos, Sério, Abrantes, Sá, Loureiro e Wanzeller, (2019), dizendo que os dados das empresas podem indicar a viabilidade de um determinado produto e determinar indicadores-chave para uma possível expansão e/ou

crescimento futuro. Desta forma, os dados podem ajudar a maximizar as receitas e reduzir custos.

No que diz respeito a consumos de energia, o uso de ferramentas de BI pode ser determinante para se efetuar uma boa gestão, sendo possível visualizar dados em tempo real, podendo obter, até, informações úteis para melhorar a tomada de decisão do município, a fim de melhorar a economia (Al-Ali, Zualkernan, Rashid, Gupta e Alikarar, 2017).

2.5 *Machine Learning*

O conceito de *Machine Learning* tem sido bastante mencionado nos últimos anos. Segundo El Naqa e Murphy (2015), é um ramo dos algoritmos computacionais com o objetivo de simular a inteligência humana que aprende com o ambiente que o rodeia. Com uma ideia semelhante, Surden (2014) diz que *machine learning* é um ramo da inteligência artificial onde o sistema computacional pode aprender com a experiência, melhorando o seu desempenho em algumas tarefas ao longo do tempo.

Machine Learning e *Data Science* são duas áreas que estão, cada vez mais, associadas uma com a outra. Porém, muitas pessoas pensam que os cientistas de dados constroem e treinam modelos de *machine learning*, aplicando os respetivos algoritmos. Mas, na verdade, a ciência dos dados está mais focada nos dados, no recolher, entender e processar os mesmos (Grus, 2019).

Porém, não quer dizer que os cientistas de dados não tenham de adquirir conhecimentos aplicados em *machine learning*, até pelo contrário. Muitas das tarefas de *machine learning* são feitas, também, na ciência de dados, como, por exemplo, o processamento dos dados, que requer conhecimentos ao nível de bibliotecas existentes (Mueller e Massaron, 2019).

Outra área semelhante, que é cada vez mais popular nos dias de hoje, é a Data Mining. Segundo Witten, Frank, Hall e Pal, (2016), a mineração de dados é o processo de descoberta de padrões consistentes nos dados, tipicamente já armazenados em bases de dados, com o objetivo de retirar informações úteis para o utilizador.

A mineração de dados é uma etapa que faz parte de um processo denominado Extração de Conhecimento em Base de Dados (*knowledge discovery in databases* - KDD). Este processo, que é iterativo, tem várias etapas, que podem ser visualizadas na Figura 2.2, no qual o objetivo é extrair conhecimento dos dados relacionados sem intervenção humana (Barazandeh e Gholamian, 2016).

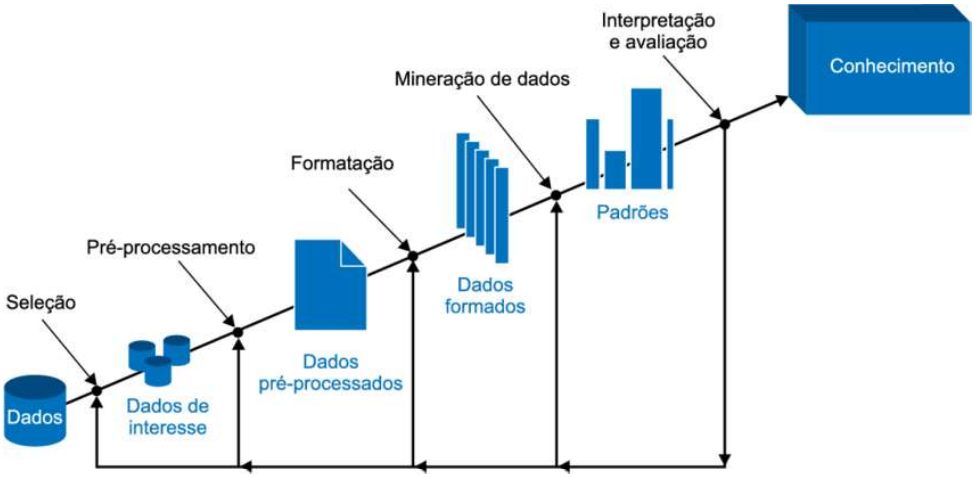


Figura 2.2: Ciclo das etapas da extração de conhecimento em base de dados[3]

A primeira etapa, seleção dos dados de interesse, tem como objetivo criar uma base de dados somente com os dados de interesse específico para o utilizador. Na segunda (pré-processamento dos dados) é feita a limpeza dos dados pois, na maior parte dos casos, ou há valores em falta ou estes têm dados que não correspondem aos verdadeiros valores (e. g. um pico de valor de temperatura no meio de dois valores muito inferiores, num curto espaço de tempo). Esta é uma das mais importantes etapas, pois se há dados incorretos entre os corretos, o resultado poderá ser bastante diferente. Na terceira etapa (transformação dos dados) é definido o processo de transformação dos dados no formato apropriado para a terceira etapa, a mineração de dados, pois, geralmente, os dados não vêm todos da mesma fonte, pelo que é necessário uniformizá-los para ser mais fácil de os manusear. A quarta etapa é a principal no processo do KDD. Nesta é feita a mineração de dados para retirar padrões dos dados recolhidos e preparados nas etapas anteriores, com o objetivo de retirar informações úteis para o utilizador. Aqui poderão entrar algoritmos de *machine learning* para ajudar no enriquecimento das informações retiradas, nomeadamente se os dados estiverem continuamente a serem inseridos. Na última etapa (interpretação e avaliação) verifica-se se os padrões são os esperados, caso contrário descartam-se e começa-se o processo novamente. Nesta etapa, geralmente, é incluída a visualização dos resultados, utilizando, por exemplo, ferramentas de visualização de dados (Tan, Zhang, Ma e Mao, 2015).

O *machine learning* pode ser dividida em três tipos: aprendizagem supervisionada (*supervised learning*), aprendizagem não supervisionada (*unsupervised learning*), e aprendizagem por reforço (*reinforcement larning*) em que, cada um destes tipos tem propósitos diferentes *(*Lison, 2015*)*, explicados a seguir.

Atualmente existem diversos algoritmos de *machine learning* que, de acordo com o seu propósito, são categorizadas pelos tipos de aprendizagem referidos anteriormente (Portugal, Alencar e Cowan, 2018). Alguns exemplos de algoritmos são: Árvores de decisão, *Nearest Neighbor*, Redes Neurais e Regressão Linear (Shalev-Shwartz e Ben-David, 2014).

[3] Fonte: https://www.researchgate.net/figure/Figura-1-Diagrama-de-formalizacao-do-processo-de-extracao-do-conhecimento-do-ingles_fig1_315671494 [consultado a 10/03/2020]

A aprendizagem supervisionada é utilizada quando é fornecido um conjunto de exemplos de dados com as respostas corretas, incluindo dados de entrada e saída. Com base nesse conjunto de dados, o algoritmo utilizado, posteriormente, irá tentar responder a todas as entradas possíveis (Marsland, 2014).

Este tipo de aprendizagem pode ser dividido em duas subcategorias: regressão e classificação. A regressão é utilizada para prever valores contínuos com base nos dados de entrada e saída. Uma possível aplicação deste tipo de aprendizagem pode ser na previsão dos preços de um automóvel tendo em conta o número de quilómetros, o ano, a potência do motor, etc., utilizando o algoritmo de regressão linear (Noor e Jan, 2017).

A classificação é utilizada quando se pretende retornar um valor categórico (e. g. Sim ou Não, Benigno ou Maligno). Um exemplo de onde se pode utilizar este tipo de categoria é na descoberta de um dado correio eletrónico ser *spam* ou não, utilizando o algoritmo K-Nearest Neighbour (Awad e ELseuofi, 2011).

A aprendizagem não supervisionada é utilizada para encontrar padrões num conjunto de dados onde não são implícitas algumas informações sobre esses dados (Sathya e Abraham, 2013).

O algoritmo mais conhecido e utilizado deste tipo de aprendizagem, que é também conhecida por *clustering*, é o *K-means*. Dada a sua simplicidade e eficiência na sua utilização, este algoritmo tem sido usado em diversos cenários, como reconhecimento de padrões, mineração de dados e na área da bioinformática (Peng, Wang, Pérez-Jiménez e Riscos-Núñez, 2015).

Este tipo de aprendizagem tem sido cada vez mais utilizado nestes últimos anos. Um exemplo onde este pode ser aplicado é na deteção de fraudes nas redes de transações de moedas virtuais. Segundo o estudo prático de Monamo, Marivate e Twala, (2016), o algoritmo *K-means* pode oferecer resultados bastante mais promissores, no que diz respeito a deteção de fraudes nas transações de moedas virtuais (no caso do estudo foi o Bitcoin), que outros algoritmos podem não oferecer.

A aprendizagem por reforço é diferente dos outros dois tipos de aprendizagens. Esta tem, como principal objetivo, aprender consoante o meio que o envolve e não através de conjuntos de dados inicialmente inseridos. Um agente que é colocado em campo, como, por exemplo, um carro autónomo, com o passar do tempo, terá de ser capaz de adquirir conhecimento ao longo do tempo, com o objetivo de optar sempre pela melhor ação. Em resumo, o agente efetua uma ação com base no ambiente em que está. Essa ação é analisada e, quanto melhor for a escolha, melhor recompensa o agente recebe (Sutton e Barto, 2018). Este ciclo das etapas descritas pode ser visualizado na Figura 2.3.

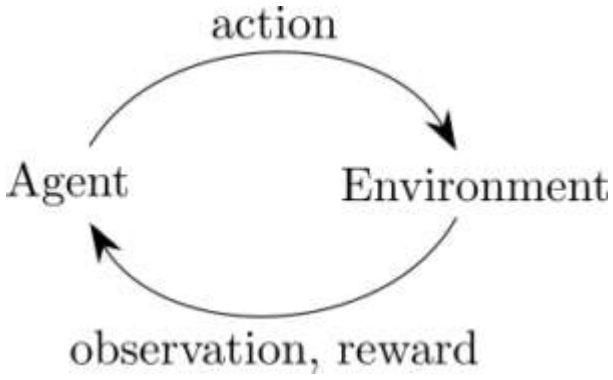


Figura 2.3: Ciclo das etapas de um agente numa aprendizagem por reforço[4]

Um dos principais problemas deste tipo de aprendizagem é que nem sempre se tem um conjunto de dados com todos os dados de entrada e saída devidamente identificados, à semelhança do que acontece com a aprendizagem supervisionada, para se prever com maior precisão os resultados pretendidos, o que torna este tipo de aprendizagem mais desafiante que os dois tipos anteriormente apresentados (Sutton et al, 2018).

Um cenário concreto de utilização deste tipo de aprendizagem passa pela previsão de consumos de energia em edifícios. Tal como o Mocanu, Nguyen, Kling, e Gibescu, (2016) experienciaram, não é necessário um conjunto de dados com o histórico dos consumos para auxiliar na aprendizagem do modelo. Esta, ao longo do tempo, vai adquirindo o próprio conhecimento. Para esta experiência, o autor utilizou um dos mais conhecidos algoritmos em aprendizagem por reforço, o *Q-learing*, por ser um dos algoritmos mais robustos e com uma percentagem de erro relativamente baixa.

Este tipo de aprendizagem, que tem sido utilizado há alguns anos, pode ser aplicado, não só na previsão de consumos de energia, mas também em inúmeros cenários, como em jogos, condução autónoma de veículos, robótica, gestão do controlo de preços de produtos em lojas, controlo de elevadores de prédios, entre outros (Szepesvári, 2010).

A Figura 2.4 representa uma síntese dos tipos de aprendizagens, com algumas das suas aplicações, referidos nas secções anteriores.

[4] Fonte: https://devblogs.nvidia.com/train-reinforcement-learning-agents-openai-gym/ [consultado a 20/03/2020]

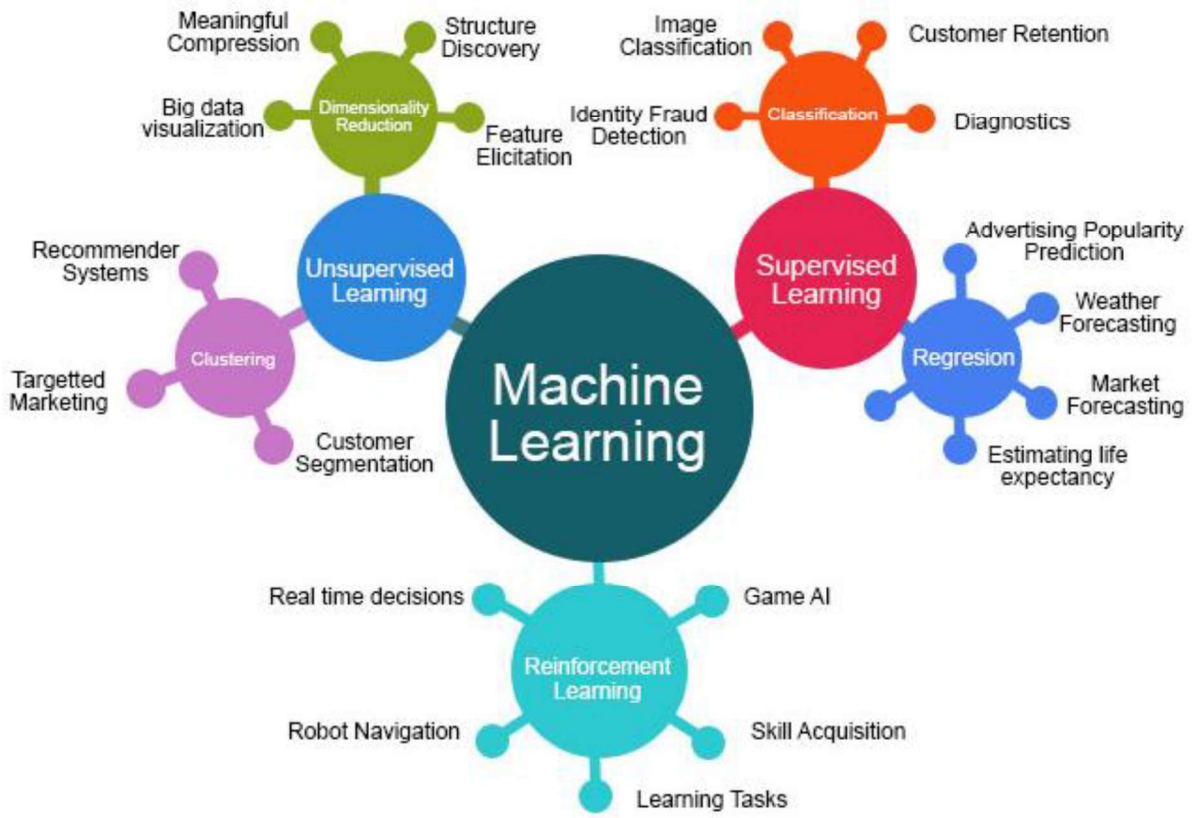


Figura 2.4: Tipos de aprendizagens em *machine learning*[5]

Atualmente existem diversos modelos de *machine learning*, cada um para o seu propósito. No caso da previsão de consumo de energia, os conjuntos de dados, na maior parte dos casos, são completos, isto é, com dados de entrada e saída e, esses dados, vêm associados ao tempo em que foram recolhidos, o que significa que o conjunto de dados é uma série temporal (time series). Assim sendo, a previsão de consumos de energia está inserido no tipo de aprendizagem supervisionada de regressão e, portanto, os modelos escolhidos para essa previsão foram o *Long Short-Term Memory* (LSTM) e *Autoregressive Integrated Moving Average* (ARIMA), pois não só são os mais utilizados neste tipo de dados, mas também são os que melhores resultados obtêm (Siami-Namini, Tavakoli e Namin, 2018).

2.5.1 *Long Short-Term Memory* (LSTM)

As *artificial neural networks* (ANNs), nos últimos anos, têm sido bastante utilizadas, devido à sua capacidade de conseguir resolver problemas complexos de classificação e séries temporais.

Este modelo é baseado no cérebro e sistema nervoso Humano, conforme se pode verificar na Figura 2.5. Numa rede neuronal, o neurónio de um cérbero humano pode ser representado por um *perceptron*, que não é nada mais nada menos que uma expressão matemática. Estes *perceptrons* juntos formam a rede neuronal, onde cada um recebe valores de entrada. Estes

[5] Fonte: https://www.educba.com/machine-learning-algorithms/ [consultado a 12/04/2020]

valores são multiplicados pelos respetivos pesos, passando, por fim, numa função de ativação, produzindo um valor final. Este valor passa para a próxima camada, repetindo o processo em todos os *perceptrons* até chegar à camada final, denominada *Output layer*, que originará o conjunto de valores de saída (Walczak e Cerpa, 2003).

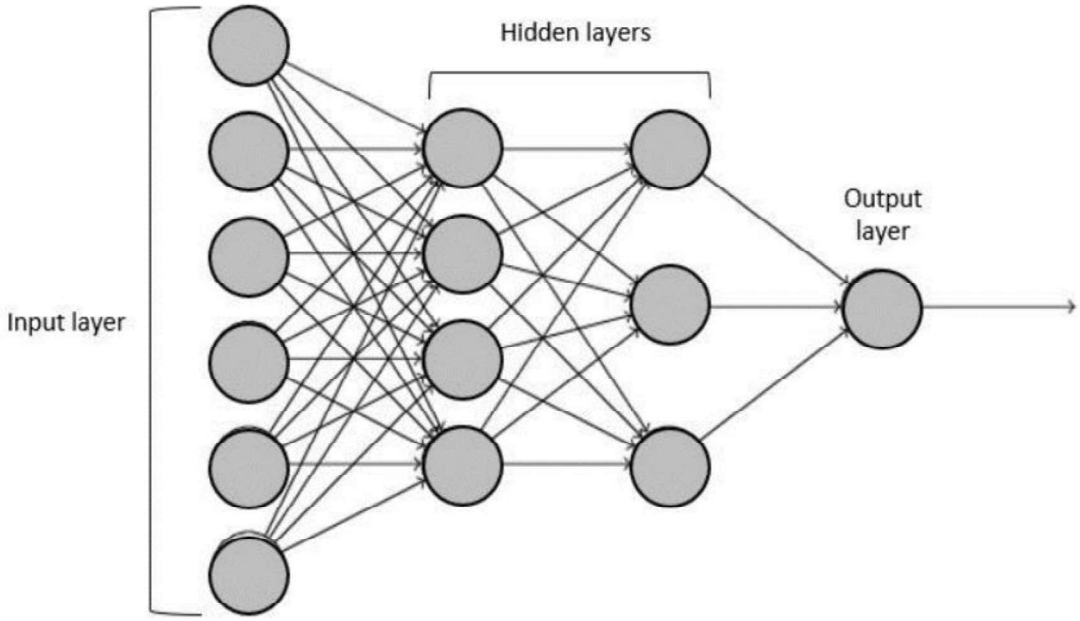


Figura 2.5: Exemplo de uma rede neural[6]

Numa típica ANN, dois inputs sucessivos são independentes um do outro. Porém, existem cenários onde os inputs dependem dos anteriores. Por exemplo, para a previsão do preço de um stock com base no tempo ou prever a próxima palavra numa dada sequência, as ANNs não são a mais adequadas. Para esse tipo de problemas, existem as *Recurrent Neural Networks* (RNN), onde cada camada é dependente da anterior, isto é, os outputs realimentam-se a si próprios, permitindo que a célula tenha uma espécie de memória. Porém, este tipo de rede neuronal tem o problema do *vanishing gradient* e do *exploding gradient*, o que significa que, na prática, são difíceis de treinar quando é necessário memorizar uma grande quantidade de dados sequenciais anteriores. Para resolver esse problema, foi criada uma variante da RNN, denominada *Long Short-Term Memory* (Yue, Fu, e Liang, 2018).

As redes LSTM são um típico específico das *Recurrent Neural Networks*, criadas para resolverem o problema referido na secção anterior. Estas redes são uma das mais utilizadas nestes últimos anos, especialmente para resolver problemas onde os dados são sequenciais, como reconhecimento de voz, previsão de stock, entre outros (Fischer e Krauss, 2018).

A arquitetura de uma célula LSTM, que é representada pela Figura 2.6, é constituída por 3 *gates*, *Forget gate*, *Input gate* e *Output* gate e 2 células, uma de memória (*Memory*) e outra de estado (*Hidden state*)

[6] Fonte: https://www.mdpi.com/2624-6511/2/2/9/htm [consultado a 08/05/2020]

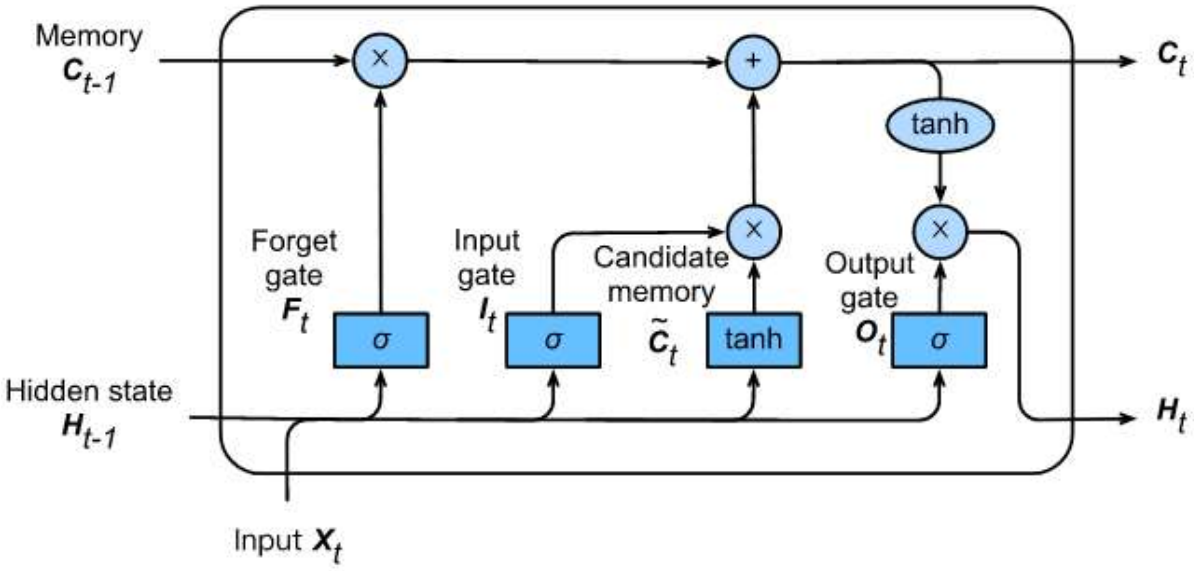


Figura 2.6: Arquitetura da LSTM[7]

A *Forget gate* é responsável por remover informação proveniente da célula *State*, que é o output da unidade anterior. Esta é uma operação importante para otimizar o desempenho da rede neuronal, pois elimina informação que já não é mais necessária ou que já não tem uma grande importância. A *Forget gate* (F_t) é definida pela seguinte expressão (2-1), onde σ é a função de ativação sigmoid, X_t é o *input* a um determinado tempo, o W_{xf} e W_{hf} os pesos das matrizes, o H_{t-1} o *Hidden State* e o b_f o *bias*:

$$F_t = \sigma\,(X_t W_{xf} + H_{t-1} W_{hf} + b_f) \tag{2-1}$$

A *Input gate* (I_t) é responsável por adicionar informação à célula *State*. Esta gate tem duas partes: a primeira (2-2) é semelhante à Forget gate, onde filtra toda a informação proveniente da célula *State* e do *Input* e a segunda (2-3), onde transforma essa informação num vetor e adiciona-o à célula *State* (Hu e Balasubramaniam, 2008).

$$I_t = \sigma\,(X_t W_{xi} + H_{t-1} W_{hi} + b_i) \tag{2-2}$$

$$C_t = tanh\,(X_t Wxc + H_{t-1} W_{hc} + b_c) \tag{2-3}$$

A *Output gate* (O_t) seleciona a informação útil da célula *State* atual (2-4). De seguida, é aplicada a função de ativação tangent hyperbolic (*tanh*) para que este crie um vetor com valores compreendidos entre -1 e +1 (2-5) (Kim, Kim, Thu e Kim, 2016).

[7] Fonte: Hu, X., & Balasubramaniam, P. (2008)

$$O_t = \sigma\,(X_t W_{xo} + H_{t-1} W_{ho} + b_o) \tag{2-4}$$

$$H_t = O_t * tanh\,(C_t) \tag{2-5}$$

2.5.2 *Autoregressive Integrated Moving Average* (ARIMA)

Autoregressive Integrated Moving Average (ARIMA) é um dos algoritmos mais populares para previsões em conjuntos de dados que têm como base o tempo, devido à sua grande flexibilidade e propriedades estatísticas (Liu, Hoi, Zhao e Sun, 2016).

Este algoritmo, que é um subconjunto do modelo de regressão linear, utiliza valores anteriores com o objetivo de prever futuros valores. É constituído pela junção de dois modelos, que são nada mais nada menos do que expressões matemáticas, o *Autoregressive* (AR) e *Moving Average* (MA). O primeiro, também conhecido como AR(*p*) e definido pela expressão (2-6), calcula os valores futuros com base nos *p* valores anteriores. O segundo, conhecido também por MA(*q*) e definido pela expressão (2-7), prevê dados futuros de uma forma semelhante ao AR(p). Porém, em vez de considerar os valores passados, considera *q* número de erros passados.

$$y_t = \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \cdots + \varphi_p y_{t-p} + \varepsilon_t \tag{2-6}$$

$$y_t = \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-1} \tag{2-7}$$

Juntado os dois modelos ARMA(*p*,*q*), pode ser definida pela seguinte expressão (2-8).

$$y_t = \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \cdots + \varphi_p y_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q} \tag{2-8}$$

O y_t significa o valor previsto, φ são os coeficientes associados a cada valor observado, y_{t-i} são os valores passados e ε_t o chamado "ruído branco", o que significa que a série temporal não tem dados passados suficientes para fazer previsões, normalmente possui um valor de média igual a zero e variância constante.

Normalmente, o modelo ARMA(*p*,*q*) é aplicado em séries temporais estacionárias, quando a sua média, variância e autovariância permanecem iguais em diferentes períodos temporais. Porém, por vezes, os conjuntos de dados são não-estacionários. Para transformar esses dados em dados estacionários, é utilizado um processo comum onde se aplicam diferenças sucessivas, utilizando o operador ∇, até que $\nabla^d y_t$ seja uma série temporal estacionária, formando o modelo denominado ARIMA(*p*,*d*,*q*). Este modelo pode ser representado pela seguinte expressão (2-9).

$$w_t = \varphi_1 w_{t-1} + \varphi_2 w_{t-2} + \cdots + \varphi_p w_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q} \quad (2\text{-}9)$$

O w_t é representado por $w_t = \nabla^d y_t$, onde o *d* pode ser definido como 0, 1 ou 2. Se, por exemplo, se substituir o y_t da expressão (8) por w_t onde $d = 0$, o modelo passa a ser o ARMA(*p*,*q*), o que quer dizer que a série temporal já é uma série estacionária e não necessita de transformações (Wang, Chau, Xu, e Chen, 2015).

2.5.3 *Random Forest* (RF)

O *Random Forest* é um algoritmo, do tipo de aprendizagem supervisionada, bastante popular, não só na área do *machine learning*, mas também noutras como processamento de imagem, devido à sua grande eficiência em problemas de regressão e classificação. Este pertence a um grupo dos métodos *ensamble*, isto é, utiliza outros algoritmos para que, no final, seja um algoritmo mais robusto e complexo, podendo oferecer melhores resultados. A Figura 2.7 representa a estrutura do RF.

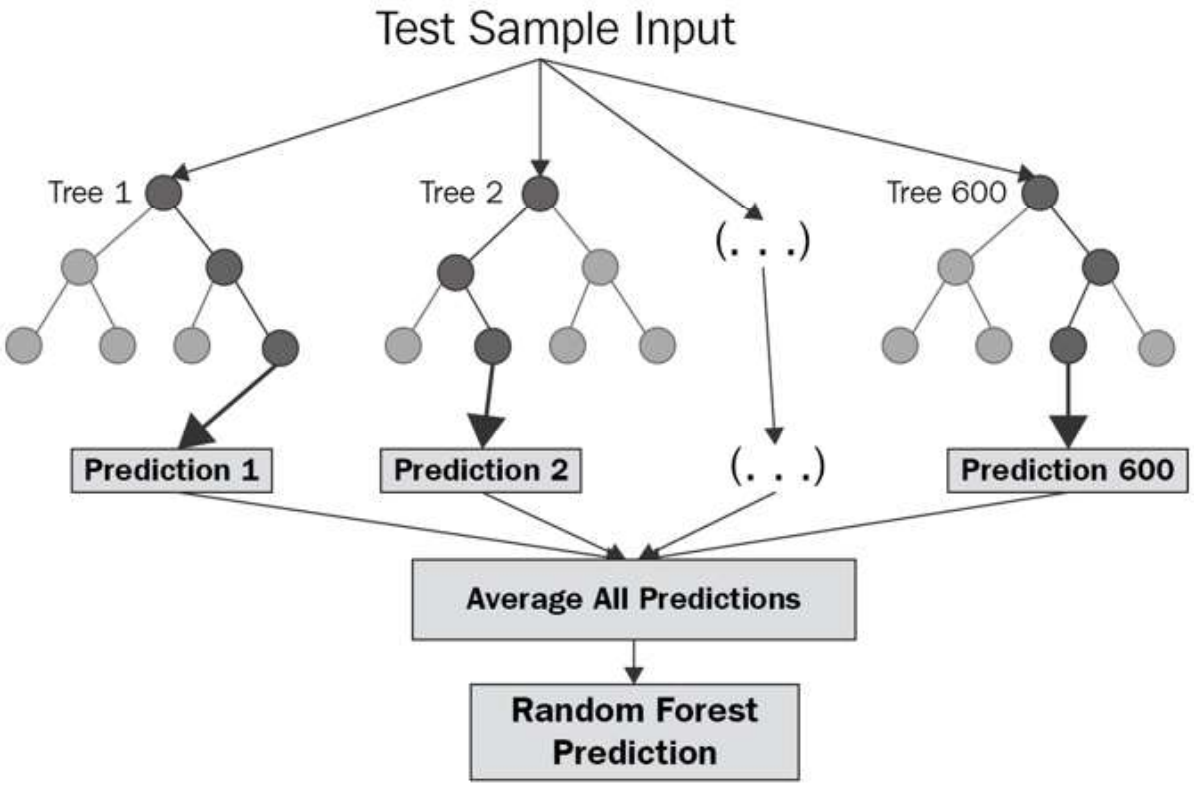


Figura 2.7: Estrutura de uma *Random Forest*[8]

O RF é a combinação de diversas árvores de decisão binárias. Cada nó "pai" terá 2 nós "filhos". Para a escolha do atributo que o nó inicial da árvore irá ter, recorre-se ao cálculo da entropia de cada atributo, isto é, o grau de impuridade de um dado conjunto de valores, com o

[8] Fonte: https://corporatefinanceinstitute.com/resources/knowledge/other/random-forest/ [consultado a 24/07/2020]

objetivo dos nós seguintes serem os mais puros possíveis. Dada uma coleção de dados *S* de *c* dados, o cálculo da entropia é definido pela expressão (2-10)

$$Entropy(S) = \sum - p(I)log2p(I) \quad (2\text{-}10)$$

onde *p(I)* é a probabilidade de *S* pertencer à coleção de dados *I.* No final, é calculado o ganho, que é a diferença entre o valor da entropia antes da separação e o valor da entropia depois da separação do conjunto de dados. O cálculo do ganho é definido pela expressão seguinte (2-11).

$$Gain(S, A) = Entropy(S) - \sum\frac{|Sv|}{|S|}Entropy(S_v) \quad (2\text{-}11)$$

onde S_v é o subconjunto de *S* e *A* o atributo, que tem o valor *v*. O atributo que tiver maior ganho é o escolhido para inicializar o nó da árvore (Yang, Li e Song, (2007). No caso do RF, a seleção do atributo do nó inicial não é calculada com base em todos os atributos do conjunto de dados, mas sim de maneira aleatória Genuer, Poggi, e Tuleau-Malot, (2010).

2.6 Previsão de consumos de energia

Juntando os dois principais termos abordados neste capítulo, ou seja, aplicar modelos de *Machine Learning* em *Smart Cities*, poderemos dizer que é um ramo que ainda é muito recente nestes últimos tempos, mas que terá resultados bastante importantes e úteis. Tal como dizem Wang e Sng, (2015), muitos sensores são instalados numa cidade inteligente para recolher um enorme volume de dados, como vídeos provenientes de câmaras de vigilância, de ambiente e dados de transporte. O mesmo completa que os algoritmos de ML são frequentemente usados e alcançam resultados muito promissores em uma ampla gama de aplicações, quando utilizados em uma grande quantidade de dados, concluindo que o *machine learning* pode facilitar o desenvolvimento de cidades inteligentes.

Nas cidades, um dos problemas mais comuns está na gestão dos consumos de energia. Segundo Robinson et al (2017), este problema pode ser amenizado utilizando algoritmos de *machine learning* para prever consumos, ajudando os municípios a cumprirem os requisitos relativamente a metas de sustentabilidade. A mesma ideia pode ser confirmada por Melzi, Same, Zayani e Oukhellou, (2017), dizendo que os algoritmos de ML podem ser utilizados para extrair padrões dos consumos com o objetivo de os otimizar.

Esta ideia da previsão de consumos de energia já não é propriamente nova. Muitos investigadores têm tentado encontrar o algoritmo com melhor acerto de previsão no que toca a este tipo de dados.

Seguem-se alguns dos trabalhos relacionados mais significativos na escolha dos algoritmos de ML a utilizar neste trabalho.

No caso de Edwards, New e Parker, (2012), com dados de sensores em três edifícios residenciais, conseguiram aplicar algoritmos de ML para prever o consumo de energia dos edifícios para a próxima hora, com resultados bastante positivos, principalmente para o algoritmo LS-SVM (*Least-squares Support-vector Machine*).

Com um projeto semelhante, Ahmad, M.W. et al (2017) utilizaram dois algoritmos de *machine learning*, *Artificial Neural Network* (ANN) e *Random Forest* (RF), para prever consumos de energia de edifícios. Neste trabalho foi utilizado um conjunto de dados de um hotel para prever os consumos para a próxima hora. O algoritmo com melhores resultados foi a ANN.

Por outro lado, Seyedzadeh, Rahimian, Glesk e Roper, (2018) compararam quatro dos mais conhecidos modelos de machine learning para previsão de consumos de energia elétrica de edifícios. Os modelos são: *Artificial Neural Network*, *Support Vector Machine*, *Gaussian-based Regression* e *Clustering*. Com esta pesquisa, foi concluído que todos estes modelos têm as suas vantagens e desvantagens, consoante o tipo de dados/variáveis de entrada.

Num outro projeto, Amasyali e El-Gohary, (2016) utilizaram o algoritmo SVM (*Support-vector Machine*) para previsão de consumos da iluminação de edifícios. Nestas previsões, as variáveis utilizadas foram o tipo de dia e a cobertura do céu. Com este trabalho concluiu-se que este algoritmo pode ser aplicado, com sucesso, na previsão de consumos da energia de iluminação.

Numa outra experiência, Vinagre Pinto, Ramos, Vale e Corchado, (2016) usaram, mais uma vez, o algoritmo SVM para preverem o consumo de energia de um edifício. Os dados eram recolhidos de 10 em 10 segundos e utilizados para prever o consumo de energia do último dia útil de cada mês. Esta experiência mostrou que é possível utilizar este algoritmo para prever este tipo de dados, mesmo em cenários complexos, superando, até, o algoritmo ANN.

Num outro projeto, Camara, Feixing e Xiuqin (2016) utilizaram os algoritmos Autoregressive Integrated Moving Average (ARIMA) e, mais uma vez, ANN em dados de consumos de energia de casas residências dos Estados Unidos. Este projeto tinha como objetivo verificar se estes dois algoritmos conseguiam obter bons resultados nas previsões para este tipo de dados. O algoritmo que obteve melhores resultados foi o ANN.

3. Desenvolvimento

Nesta secção será descrito todo o processo de desenvolvimento do projeto, começando pelo conjunto de dados, apresentando as suas características, o seu processamento e as várias etapas do processo de aplicação dos modelos de *Machine Learning* na previsão de dados de energia elétrica para os próximos tempos. Por último, será mostrado o desenvolvimento de uma API REST que faz a previsão dos dados, envia-os para a base de dados, mostrando os resultados numa ferramenta de *business intelligence*.

3.1 Conjunto de dados

O conjunto de dados escolhido para a previsão de dados de energia elétrica foi o PJM *Hourly Energy Consumption Data*, da cidade de *Dayton* (Ohio, Estados Unidos da América), disponibilizados, abertamente, pela organização PJM *Interconnection*. Esta é uma organização regional de transmissão que coordena o movimento de eletricidade em toda ou parte dos estados Delaware, Illinois, Indiana, Kentucky, Maryland, Michigan, Nova Jersey, Carolina do Norte, Ohio, Pensilvânia, Tennessee, Virgínia, Virgínia Ocidental e Columbia (PJM, 1999). A Figura 3.1 mostra os 10 primeiros dados do conjunto.

O conjunto de dados tem 2 colunas, sendo a primeira referente à data de recolha de cada dado e a segunda à energia, na unidade de medida Megatwatt, com dados recolhidos entre o dia 1 de outubro de 2004 e o dia 3 de agosto de 2018, de hora a hora, o que dá um total de 121275 amostras.

Pode-se afirmar que este conjunto é uma série temporal univariada, pois apresenta dois atributos que, neste caso, o primeiro é a coluna da data de recolha de cada amostra e a segunda é referente ao consumo de energia.

	Datetime Date	DAYTON_MW Megawatt Energy Consumption
1	2004-10-01 01:00:00	1621.0
2	2004-10-01 02:00:00	1536.0
3	2004-10-01 03:00:00	1500.0
4	2004-10-01 04:00:00	1434.0
5	2004-10-01 05:00:00	1489.0
6	2004-10-01 06:00:00	1620.0
7	2004-10-01 07:00:00	1859.0
8	2004-10-01 08:00:00	2007.0
9	2004-10-01 09:00:00	2025.0
10	2004-10-01 10:00:00	2067.0

Figura 3.1: Primeiras 10 leituras do conjunto de dados

Para uma melhor visualização do conjunto de dados, apresenta-se, na Figura 3.2, um gráfico com o total de dados em que o eixo do X representa a data de recolha dos dados e o do Y as respetivas medidas, em *Megawatt*.

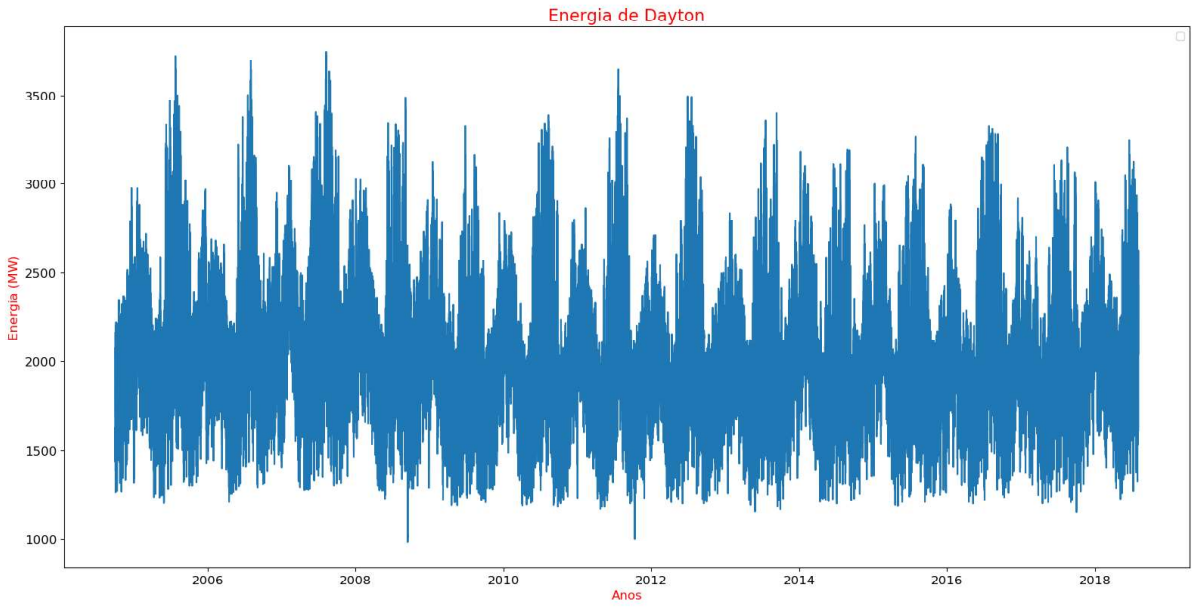


Figura 3.2: Visualização gráfica do conjunto de dados

Para um melhor conhecimento do conjunto de dados, nomeadamente, de forma a obter informações adicionais que poderão ser relevantes, não só para o estudo do mesmo, como também para a aplicação de algoritmos de *machine learning*, adicionou-se a este uma nova coluna com a temperatura de cada dia[9].

Com o conjunto de dados completo, efetuou-se o agrupamento, por mês, da média, tanto da energia como da respetiva temperatura (Quadro 3.1), com o objetivo de perceber quais dos meses havia um maior ou menor consumo de energia, bem como se a temperatura tinha de alguma forma influência no mesmo.

Quadro 3.1: Consumo de energia e temperatura agrupados por mês

Mês	Energia (Megawatt)	Temperatura
janeiro	2217.72	5.79
fevereiro	2177.28	5.40
março	1974.69	6.36
abril	1791.57	11.77
maio	1848.22	17.45
junho	2129.02	22.04
julho	2238.72	23.61
agosto	2263.79	23.10
setembro	1962.59	19.30
outubro	1834.12	12.65
novembro	1922.73	6.96
dezembro	2103.92	4.68

Para uma melhor visualização dos dados da tabela anterior, efetuou-se uma normalização dos mesmos, com valores entre 0 e 1, com o objetivo de ficarem no mesmo intervalo para se obter uma melhor comparação. O resultado apresenta-se na Figura 3.3.

[9] Dados retirados da API do website https://www.wunderground.com/

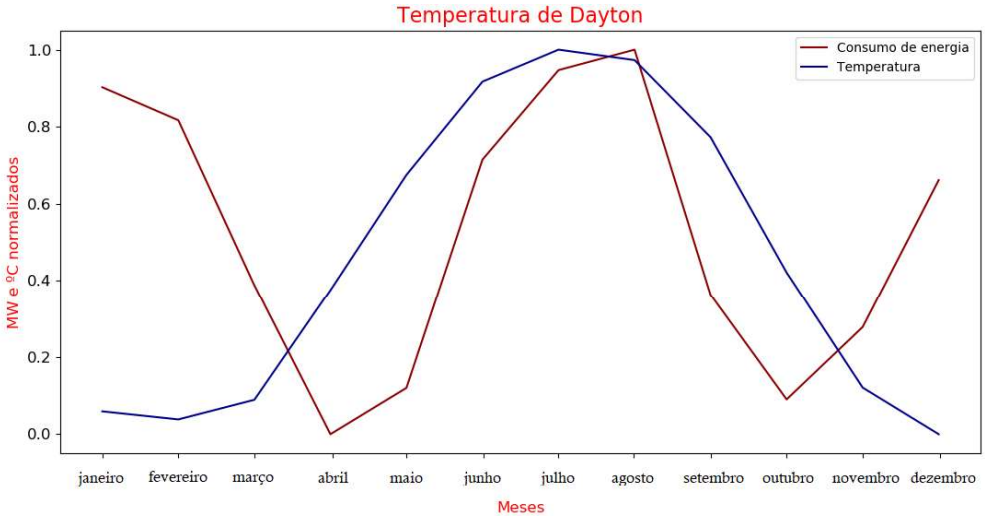


Figura 3.3: Visualização gráfica do consumo de energia e temperatura agrupados por mês

Com base nos resultados apresentados, pode-se tirar diversas conclusões. A primeira é que, nos meses onde a temperatura é mais baixa (dezembro, janeiro e fevereiro), o consumo de energia é mais alto (média acima de 2000 MW). Nos meses onde a temperatura é mais alta (junho, julho e agosto), o consumo é, também, mais alto. Por outro lado, nos meses onde a temperatura média está entre os 6ºC e os 22ºC (março, abril, maio, setembro, outubro e novembro), o consumo é menor.

Com isto, pode-se dizer que nas épocas do verão e do inverno, onde as temperaturas são mais altas e baixas, respetivamente, há um maior consumo de energia devido, provavelmente, no primeiro caso, à utilização excessiva do ar condicionado e no segundo à utilização excessiva de aquecedores elétricos.

Nas épocas da primavera e do outono, onde a temperatura média nem é muito alta nem muito baixa, o consumo é mais baixo devido, provavelmente, à não utilização excessiva tanto do ar condicionado, como de aquecedores elétricos.

Também se pode concluir que, pelo Quadro anterior e, também, pela Figura 3.3, o conjunto de dados tem um comportamento sazonal, pois apresenta comportamento semelhante num determinado período de tempo que, neste caso, é de um ano.

Com todas estas conclusões, pode-se afirmar que, de facto, o consumo de energia está diretamente relacionado com a temperatura, nomeadamente no verão e inverno, devido, maioritariamente, às razoes mencionadas nos parágrafos anteriores (De Cian, Lanzi e Roson, 2007). Porém, ao utilizar o atributo da temperatura, os conjuntos de dados ficariam com diferentes números de dados, pois os dados da temperatura estão agregados por dia e os dados do consumo de energia estão agregados por hora. Seria possível converter o conjunto de dados original para um novo com dados agregados por dia. Todavia, o número total do mesmo seria muito menor que o original e, como foi dito no estado da arte, um modelo com um conjunto de dados maior, no geral, tem melhores previsões. Devido a isso, o atributo da temperatura não foi utilizado no modelo final.

3.2 Preparação dos dados

Na preparação dos dados, mais conhecida como pré-processamento, não foram necessárias quaisquer técnicas de limpeza, como remoção de dados com valores nulos e alteração de dados que estejam fora da gama normal do conjunto, pois o conjunto de dados tinha os dados todos corretos e dentro da norma. A única alteração foi o ordenamento dos dados, por ordem ascendente de data de recolha e o recurso a *Feature Engineering* e a Normalização, que serão explicadas seguidamente.

3.2.1 Feature Engineering

Como foi dito na secção anterior, o conjunto de dados é uma série temporal univariada, o que quer dizer que só tem um atributo (*feature*). Para se obter uma maior quantidade de atributos foi feita a *feature engineering*, que é o processo de usar o domínio do conhecimento dos dados para criar atributos, que serão utilizados no ML como dados de entrada. Com isto, não só conseguimos obter um maior conhecimento do conjunto de dados, mas também ajuda na validação do modelo construído (Zaidi, 2015).

Através da coluna do tempo (Datetime), foram criadas as colunas "year", "dayofweek", "month", "day" e "hour", que representam o ano, dia da semana, o ano, o mês, o dia e a hora em que foram recolhidos os dados relativos ao consumo de energia, respetivamente. O quadro 3.2 apresenta os primeiros 5 dados do conjunto de dados com os novos atributos.

Quadro 3.2: Primeiros 5 dados do conjunto de dados após a *feature engineering*

Datetime	DAYTON_MW	year	dayofweek	month	day	hour
2004-10-01 01:00:00	1621.00	2004	4	10	1	1
2004-10-01 02:00:00	2146.55	2004	4	10	1	2
2004-10-01 03:00:00	2146.07	2004	4	10	1	3
2004-10-01 04:00:00	2136.36	2004	4	10	1	4
2004-10-01 05:00:00	2085.28	2004	4	10	1	5

Por vezes, uma grande quantidade de atributos não significa que os modelos tenham uma melhor previsão. Tendo em conta o exemplo do conjunto de dados apresentado no quadro anterior, se a coluna do mês não estiver diretamente relacionada com o consumo de energia, muito provavelmente seria melhor retirá-la para evitar previsões erradas. Devido a isso, foi necessário escolher os atributos que realmente influenciam na previsão final.

Para efetuar a seleção dos atributos, foi utilizado o método "*Mutual Info Regression*" proveniente da biblioteca *open source* "Scikit-learn". Este método é um método não paramétrico que avalia a correspondência mútua de uma variável alvo contínua com base na

estimativa da entropia, a partir de distâncias do algoritmo *k-nearest neighbors* (KNN). Este método devolve um valor para cada atributo. O atributo que tiver um maior valor significa que tem uma dependência mais alta (Nahuis, Guyeux, Arcolezi, Couturier, Royer e Lotufo, 2019).

A Figura 3.4 apresenta o resultado do método "*Mutual Info Regression*" nos 5 atributos provenientes da *feature engineering*.

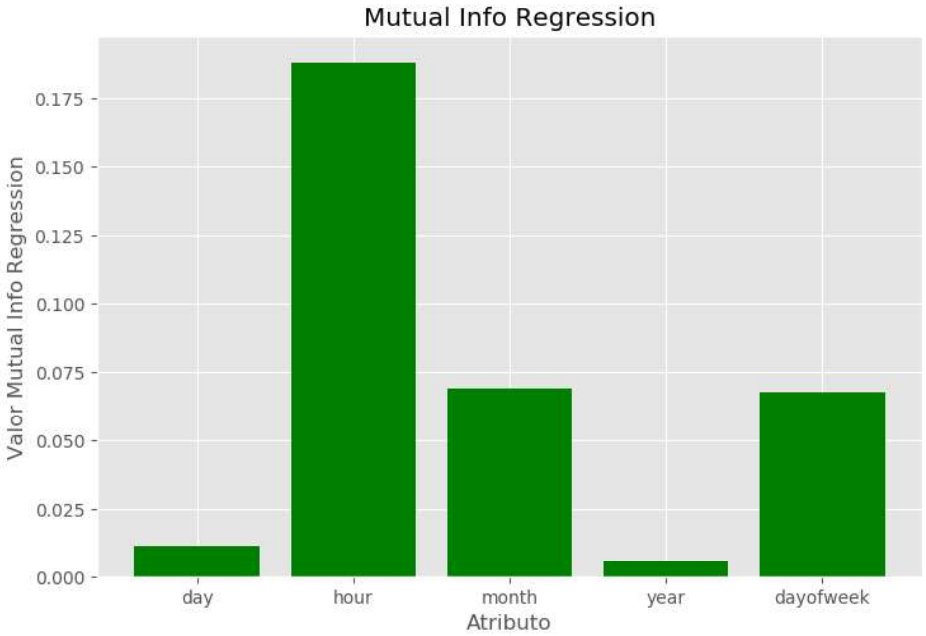


Figura 3.4: Classificação dos atributos utilizando o método "Mutual Info Regression"

Como se pode ver na figura anterior, os atributos "day" e "year" têm valores relativamente mais baixos que os restantes. Isto quer dizer que têm pouca influência quando utilizados no treino dos modelos de regressão, pelo que, se retirados do conjunto de dados final, poderão oferecer um melhor resultado.

Para uma melhor visualização destes dois atributos, foram feitos agrupamentos, no conjunto de dados, da média dos dois atributos, isto é, por dia e por ano. As Figuras 3.5 e 3.6 apresentam o resultado desses agrupamentos, graficamente.

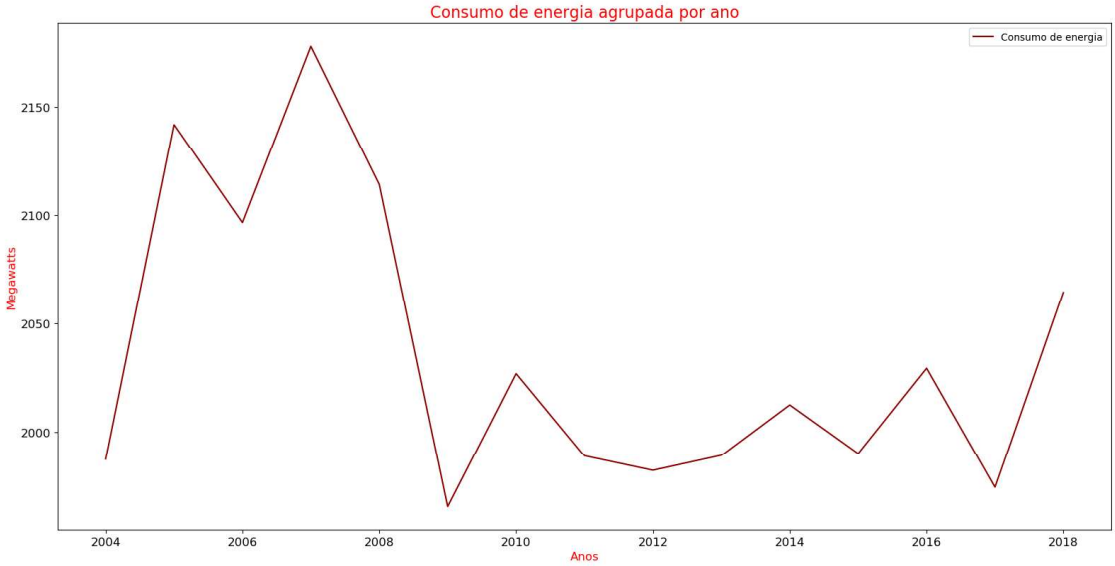


Figura 3.5: Consumo de energia elétrica agrupada por ano

Como se pode verificar na Figura 3.5, os valores do consumo de energia em função dos vários anos são bastante inconsistentes, não havendo um padrão em específico, dificultando, assim, a previsão do algoritmo.

Uma possível resposta à discrepância dos dados, poderá ser que, em anos que o consumo de energia foi mais baixo, poderia ter havido uma crise ao nível do país. Se assim for, é muito complicado o algoritmo prever, com base neste conjunto de dados, quais anos é que serão afetados, no futuro, por uma eventual crise ou por outra razão que desencadeasse estes valores. Daí, este atributo ser uma má escolha para este conjunto de dados.

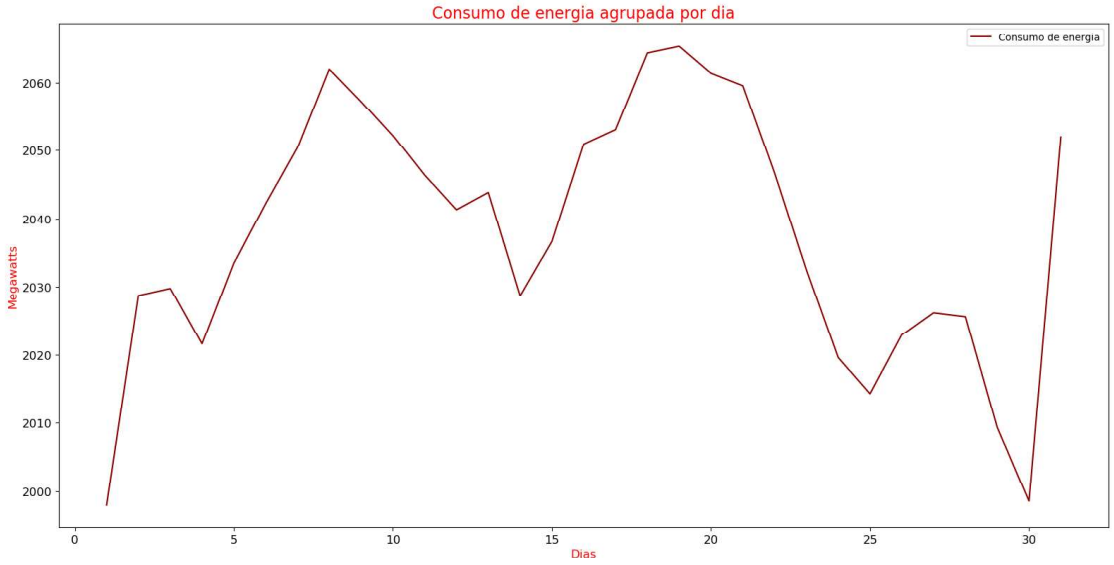


Figura 3.6: Consumo de energia elétrica agrupada por dia

Tendo em conta a Figura 3.6, à semelhança do que acontece com o consumo de energia agrupado por ano, os valores agrupados por dia são bastante inconsistentes, não havendo nenhum padrão, tornando a previsão do algoritmo bastante difícil. Daí, este atributo ser, também, uma má escolha para este conjunto de dados.

Por outro lado, ao se verificar os gráficos com os agrupamentos dos restantes atributos, apresentados nas figuras seguintes, verifica-se o oposto ao indicado com os dois atributos anteriores.

No caso do atributo "hour", ao se agrupar os dados por hora, como se apresenta na Figura 4.7, consegue-se verificar que há um determinado padrão que poderá fazer sentido. Durante o período da noite, isto é, das 00:00h até por volta das 05:00h, o consumo de energia é mais baixo. Pode-se considerar que, neste período de tempo, estes valores são normais devido a ser o período quando a maior parte da população está a dormir e, por isso, o consumo é mais baixo. Por outro lado, a partir das 05:00h, o consumo de energia começa a aumentar gradualmente, pois a população começa por volta dessa hora a ir para os seus habituais trabalhos.

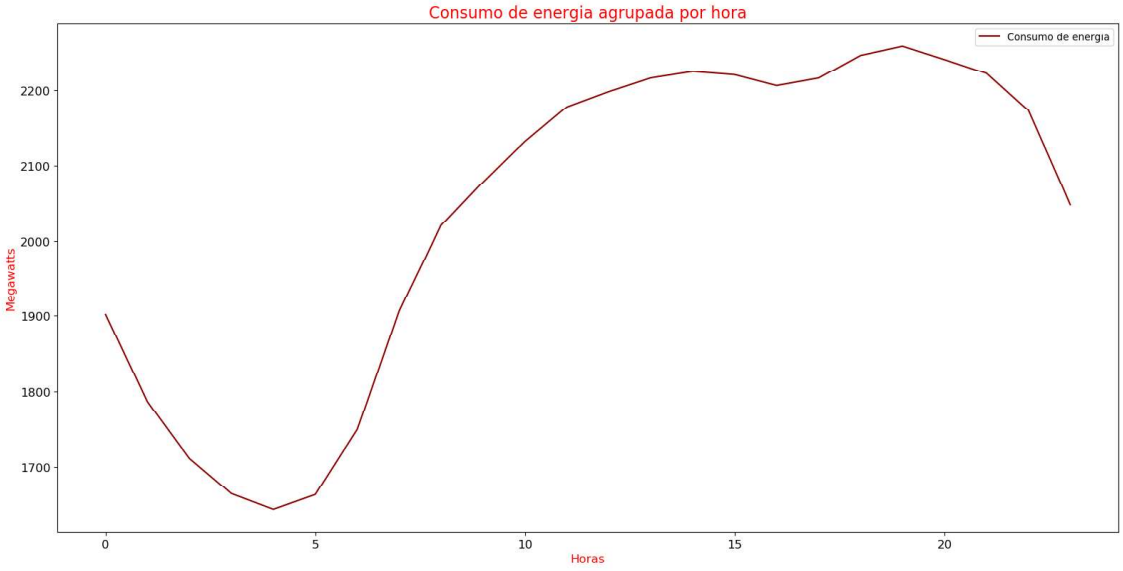


Figura 3.7: Consumo de energia elétrica agrupada por hora

Relativamente ao atributo do mês, como se pode verificar na Figura 3.8, apresenta um certo padrão. Nos meses relativamente às estações do verão e do inverno, os consumos de energia são maiores que da primavera e do outono. Isto poderá ter em conta a temperatura: no verão e no inverno, devido ao uso intensivo do ar-condicionado e do aquecedor nos edifícios nas respetivas estações, o consumo é maior, enquanto na primavera e outono a temperatura é mais amena, não sendo necessário o uso desses dois aparelhos com tanta intensidade.

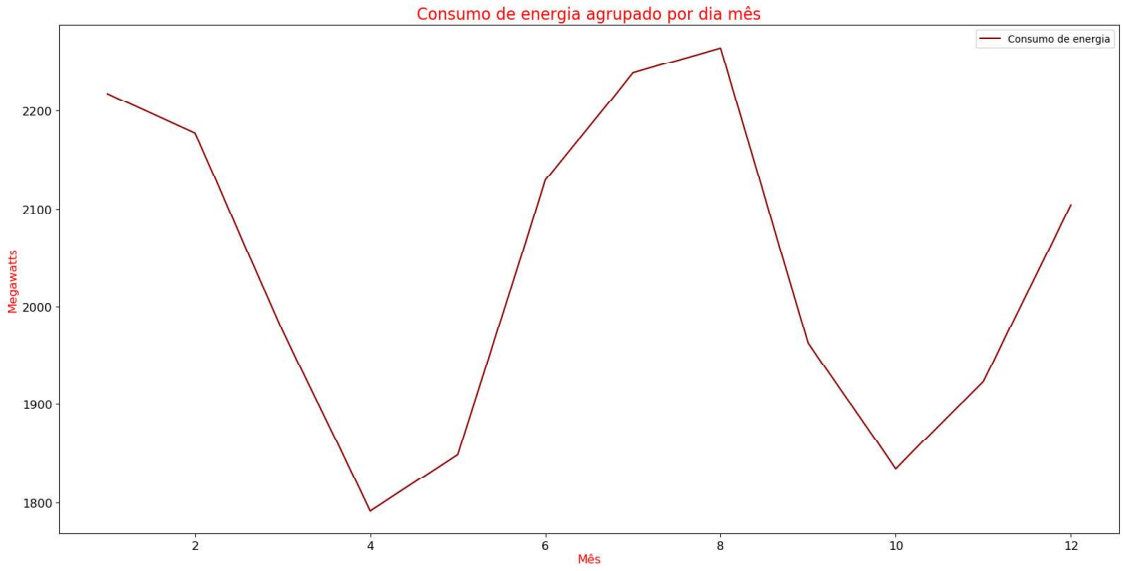


Figura 3.8: Consumo de energia elétrica agrupada por mês

Em relação ao atributo "dayofweek", verificando o gráfico da Figura 3.9, pode-se concluir que também apresenta um padrão. De segunda-feira a sexta-feira, a média do consumo de energia é mais alta que no sábado e domingo. Estes números podem-se considerar dentro dos normais, pois, durante a semana normal de trabalho, por exemplo, no caso das empresas de tecnologias de informação, tendem a gastar mais energia devido ao uso intensivo de computadores, monitores, impressoras, entre outros equipamentos. Já aos fins-de-semana, normalmente essas empresas estão fechadas, daí a diminuição do consumo médio de energia.

Figura 3.9: Consumo de energia elétrica agrupada por dia da semana

A escolha dos melhores atributos nem sempre é benéfica para todos os algoritmos de ML, poderá haver casos onde existem algoritmos que têm melhores resultados com todos os atributos do que com os escolhidos. Para se verificar se realmente há influência na escolha dos atributos, na secção seguinte serão testados diferentes modelos com e sem a escolha dos mesmos.

3.2.2 Normalização

A normalização do conjunto de dados, que é uma tarefa bastante popular na área de *machine learning* e *data mining*, é o processo no qual se transformam os valores dos vários atributos do conjunto de dados num determinado intervalo, como por exemplo entre o valor 0 e 1, diminuindo, assim, a distância entre os valores.

Por norma, os algoritmos de *machine learning* obtêm melhores resultado quando os dados estão normalizados, porém, noutros algoritmos este processo é insignificante. A normalização dos dados é, não só importante para se obterem melhores resultados nas previsões, mas também para melhorar a velocidade de treino dos modelos e obter informações úteis sobre os dados (Wan, 2019).

Tendo em conta o conjunto de dados da temperatura utilizado anteriormente e o conjunto de dados original, juntaram-se ambos os conjuntos e fez-se o agrupamento, por dia da semana, do conjunto final, com o objetivo de retirar algumas informações desse conjunto. Com o conjunto de dados final, fez-se, também, outro conjunto de dados exatamente igual, mas com os dados normalizados, de valores entre 0 e 1, com o objetivo de verificar qual destes conjuntos de dados daria melhores informações.

Seguem-se, nas Figuras 3.10 e 3.11, uma visualização gráfica dos dois conjuntos de dados.

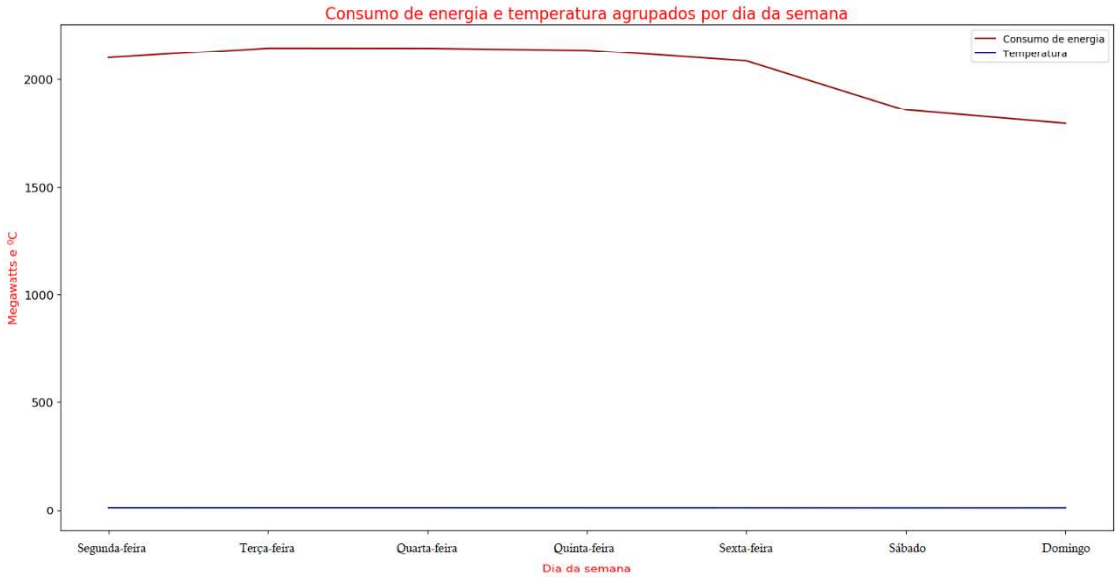


Figura 3.10: Consumo de energia elétrica e temperatura agrupados por dia da semana sem normalização

Como se pode verificar na Figura 3.10, é muito complicado retirar qualquer informação útil no que diz respeito ao consumo de energia e à temperatura. Isto porque os dados da temperatura variam entre os 12 e os 14 ºC e os dados do consumo de energia entre variam entre os 1700 MW e 2200 MW, o que torna a visualização bastante complicada.

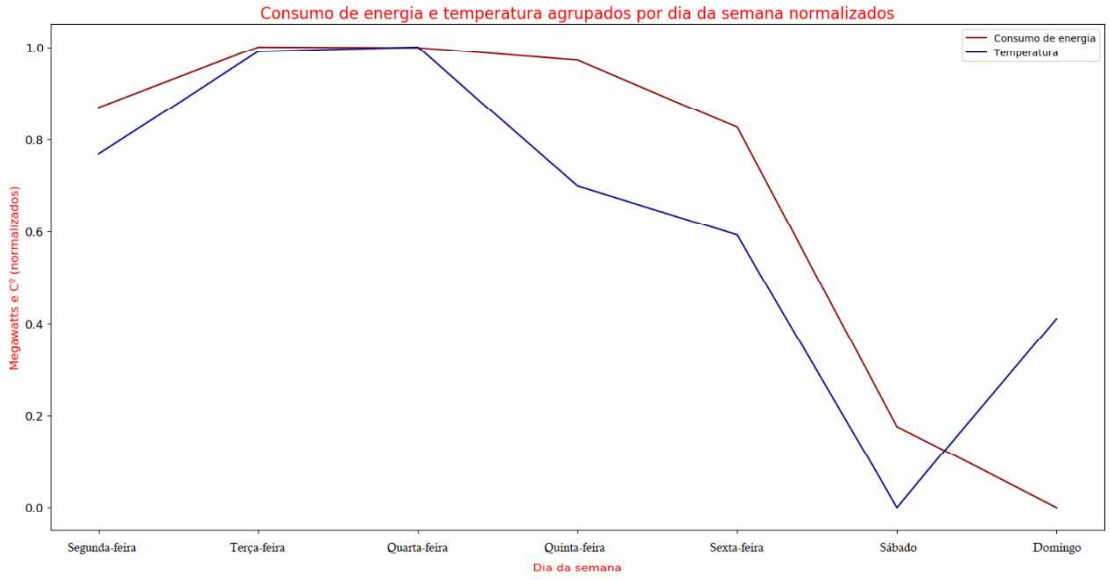


Figura 3.11: Consumo de energia elétrica e temperatura agrupados por dia da semana normalizados

Por outro lado, tendo em conta a Figura 3.11 referente ao mesmo conjunto de dados, mas com a normalização, podemos retirar algumas conclusões devido a todos os valores estarem no mesmo intervalo. Podemos concluir que, por exemplo, a temperatura varia com o aumento/diminuição do consumo de energia.

Com este exemplo, pode-se concluir, então, que a normalização dos dados é bastante importante para retirar informações dos conjuntos de dados com mais do que um atributo no mesmo gráfico.

A fim de verificar se a normalização dos dados tem algum impacto nos resultados dos modelos de *machine learning* utilizados, no capítulo seguinte serão efetuados modelos com três tipos de normalizações mais utilizados: MinMaxScaler, RobustScaler e StandardScaler.

3.3 Construção dos modelos e análise dos resultados

Neste capítulo serão apresentados vários modelos de *machine learning* desenvolvidos, utilizando os três algoritmos mencionados no estado da arte: o *Long Short-Term Memory*, o *Autoregressive Integrated Moving Average* e o *Random Forest*.

Após a construção dos modelos, estes serão analisados e comparados, utilizando o RMSE (*Root Mean Square Error*), com o objetivo de escolher o melhor algoritmo para este tipo de

problema e dados. Esse algoritmo escolhido será, no capítulo seguinte, melhorado, para ser utilizado na REST API.

A avaliação através do RMSE foi escolhida devido ao facto de ser bastante utilizada em avaliações de modelos de ML de regressão e valores numéricos (Roondiwala, Patel e Varma, 2017).

Dado um conjunto de n dados e e como sendo erros do modelo calculados como (e_i, i = 1,2, …, n), o RMSE pode ser calculado pela seguinte expressão (3-1) (Chai e Draxler, 2014).

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \tag{3-1}$$

Os modelos serão desenvolvidos tendo em conta duas vertentes da secção anterior: *feature engineering* e normalização. Na primeira, foram utilizados conjuntos de dados com e sem *feature engineering*, incluindo com os atributos selecionados. Na segunda vertente, foram utilizados conjuntos de dados com os três tipos de normalização mencionados anteriormente: MinMaxScaler, RobustScaler e StandardScaler.

A comparação de modelos nunca é uma tarefa fácil, especialmente entre modelos de *machine learning* e *deep learning*, devido aos vários parâmetros que uns têm e outros não como, por exemplo, o tamanho de *batch* que dentro destes três algoritmos, só o LSTM é que tem. De modo a que a comparação dos modelos seja o mais justa possível, isto é, sem qualquer ajuste do modelo com o objetivo de melhorar o resultado de um em específico, foram utilizados os parâmetros por omissão que as bibliotecas utilizadas na construção destes oferecem.

Para a construção dos modelos, a linguagem de programação utilizada foi o Python, devido à experiência de utilização e por ter uma grande comunidade na área do ML. Para o LSTM, foi utilizada a *framework* Tensorflow 2.0, como *back-end* e a biblioteca Keras, que oferece uma fácil API para os vários modelos, incluindo o LSTM. Para o ARIMA foi utilizada a biblioteca Statsmodels e para o RF foi utilizada a biblioteca Scikit-Learn.

Para futura validação dos modelos, o conjunto de dados foi dividido em 75% para treino e 25% para teste. Devido a ser um conjunto relativo ao tempo, as datas e a sua ordenação importam e, por isso, os 75% de teste pertencem aos primeiros dados do conjunto e os restantes 25% aos últimos (Reitermanova, 2010).

Os Quadros 3.3, 3.4 e 3.5 apresentam os parâmetros para a construção dos modelos LSTM, ARIMA e RF, respetivamente.

Quadro 3.3: Parâmetros utilizados na construção dos modelos LSTM

LSTM	
Parâmetro	Valor
Units	1
Camadas	1 de entrada e 1 de saída
Épocas	1
Tipo de Normalização	MinMaxScaler entre os valores 0 e 1
Tamanho do *batch*	128
Função de ativação	Tangente hiperbólica (*Tanh*)
Dropout	0.2
Dados anteriores	20

Quadro 3.4: Parâmetros utilizados na construção dos modelos ARIMA

ARIMA	
Parâmetro	Valor
p	1
q	1
d	1

Quadro 3.5: Parâmetros utilizados na construção dos modelos RF

RF	
Parâmetro	Valor
Árvores	1

Como se pode verificar nos três quadros anteriores, os três modelos requerem diversos parâmetros na construção dos mesmos, pelo que, como foi dito anteriormente, dificulta na comparação dos diferentes modelos.

Segue-se uma descrição dos vários modelos testados.

3.3.1 Sem Feature Engineering

Para o primeiro modelo, foram testados os três algoritmos com o conjunto de dados original, isto é, sem a *feature engineering*, somente com as colunas relativamente à data da recolha dos dados e os respetivos valores. Segue-se a tabela com os resultados.

Quadro 3.6: Resultados dos modelos sem *feature engineering*

	LSTM	ARIMA	RF
RMSE	389	626	498
Tempo de treino	4.8s	0.1s	0.02s

Para uma melhor visualização dos resultados, construiu-se um gráfico para cada modelo com a comparação do conjunto atual e previsto. As Figuras 3.12, 3.13 e 3.14 apresentam um excerto dos resultados dos respetivos modelos.

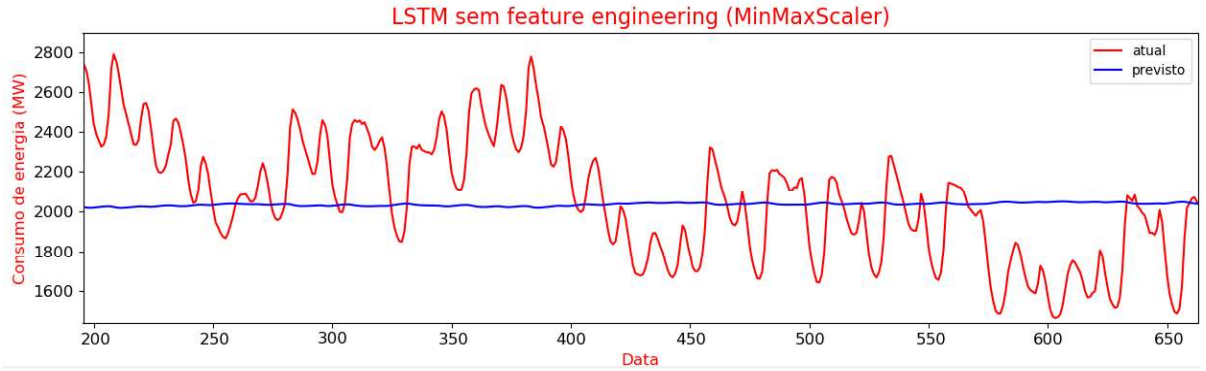


Figura 3.12: Excerto da comparação do conjunto de dados atual e previsto LSTM sem *feature engineering*

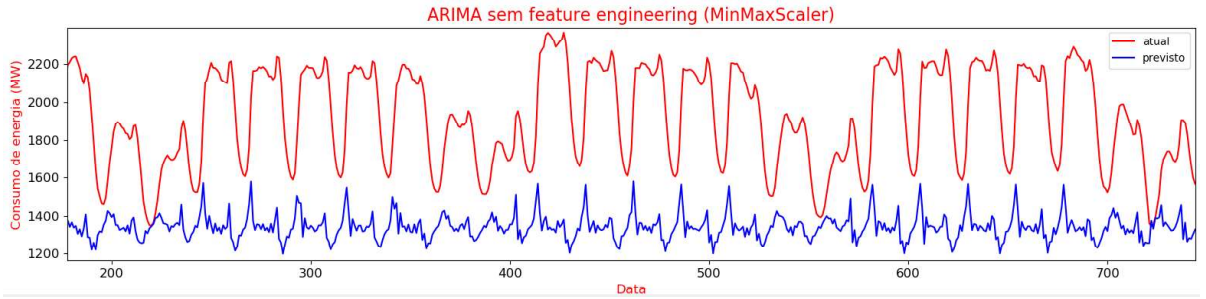


Figura 3.13: Excerto da comparação do conjunto de dados atual e previsto ARIMA sem *feature engineering*

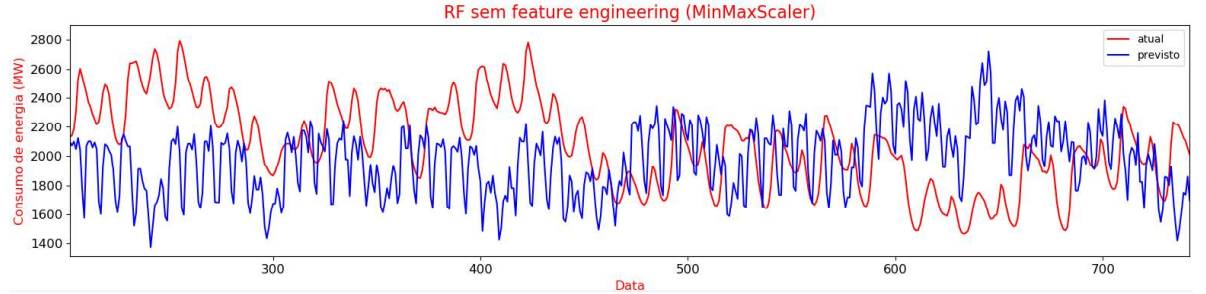


Figura 3.14: Excerto da comparação do conjunto de dados atual e previsto RF sem *feature engineering*

Tendo em conta os resultados do Quadro 3.6, o algoritmo que melhores resultados obteve, isto é, que obteve menor valor de RMSE, foi o LSTM e o pior resultado foi o ARIMA. Ao se verificar as Figuras 3.12, 3.13 e 3.14, à primeira vista pode-se dizer que o modelo RF foi o que melhores resultados teve. Contudo, em alguns momentos da previsão, os valores estão completamente opostos, isto é, por vezes existe um pico alto no conjunto atual e a previsão é um ponto baixo, ou vice-versa, o que faz com que o RMSE tenha um valor alto e, por conseguinte, haja uma má previsão em valores futuros.

3.3.2 Com *Feature Engineering* e todos os atributos

Para o segundo modelo, onde o conjunto de dados a testar foi o conjunto com todos os atributos resultantes da *Feature Engineering*, foram testados somente o LSTM e o RF. O ARIMA não foi testado, pois é um algoritmo só utilizado para conjuntos univariados (somente com um atributo). Segue-se o Quadro com os resultados.

Quadro 3.7: Resultados dos modelos com todos os atributos da *feature engineering*

	LSTM	ARIMA	RF
RMSE	143	-	260
Tempo de treino	4.94s	-	0.2s

Para uma melhor visualização dos resultados, construiu-se um gráfico para cada modelo com a comparação do conjunto atual e previsto. As seguintes figuras apresentam um excerto dos resultados dos respetivos modelos.

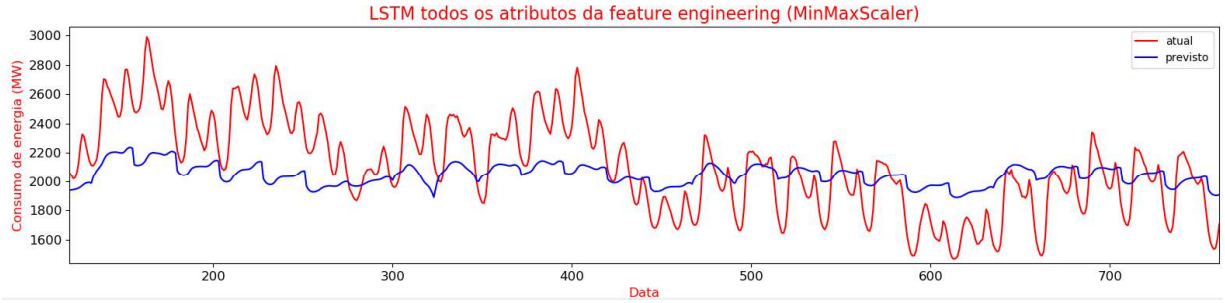


Figura 3.15: Excerto da comparação do conjunto de dados atual e previsto LSTM com *feature engineering* (todos os atributos)

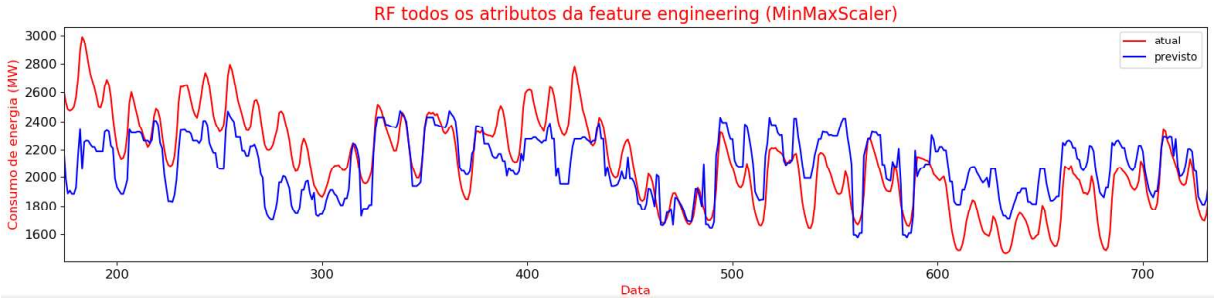


Figura 3.16: Excerto da comparação do conjunto de dados atual e previsto RF com *feature engineering* (todos os atributos)

Relativamente ao Quadro 3.7, pode-se concluir que, em ambos os modelos, houve uma grande diferença nos resultados em comparação com a outra tabela. Efetivamente, os resultados foram substancialmente melhores com este conjunto de dados do que o conjunto original.

O modelo que melhor resultados obteve foi, novamente, o LSTM. Tendo em conta os excertos de conjuntos das Figuras 3.15 e 3.16, consegue-se notar algumas diferenças para melhor, em ambos os modelos em comparação às anteriores. Ainda assim, ambos os modelos ainda longe do esperado.

3.3.3 Com *Feature Engineering* e atributos escolhidos

No que respeita a *feature engineering* com os atributos escolhidos, nos seguintes modelos, foram utilizados conjuntos de dados após a escolha dos atributos da *Feature Engineering*. À semelhança do teste anterior, para este conjunto de dados serão testados somente os modelos LSTM e RF, devido ao conjunto de dados ser um conjunto multivariado. Segue-se a tabela com os resultados.

Quadro 3.8: Resultados dos modelos com os atributos da *feature engineering* escolhidos

	LSTM	ARIMA	RF
RMSE	152	-	223
Tempo de treino	4.93s	-	0.1s

Para uma melhor visualização dos resultados, construiu-se um gráfico para cada modelo com a comparação do conjunto atual e previsto. As Figuras 3.17 e 3.18 apresentam um excerto dos resultados dos respetivos modelos.

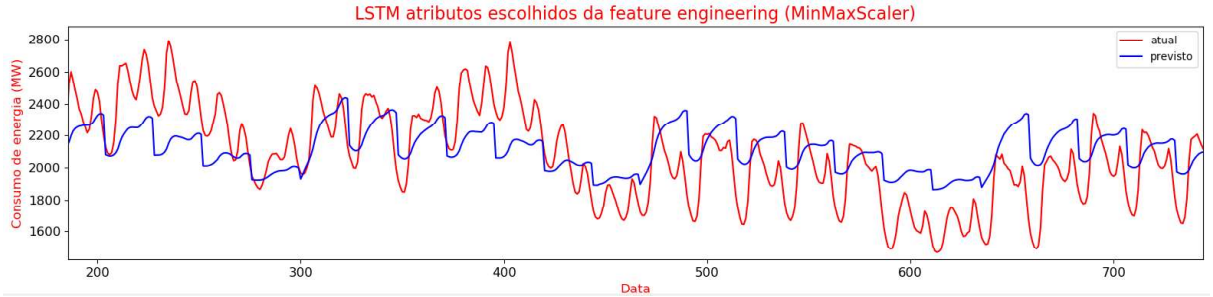


Figura 3.17: Excerto da comparação do conjunto de dados atual e previsto LSTM com *feature engineering* (atributos escolhidos)

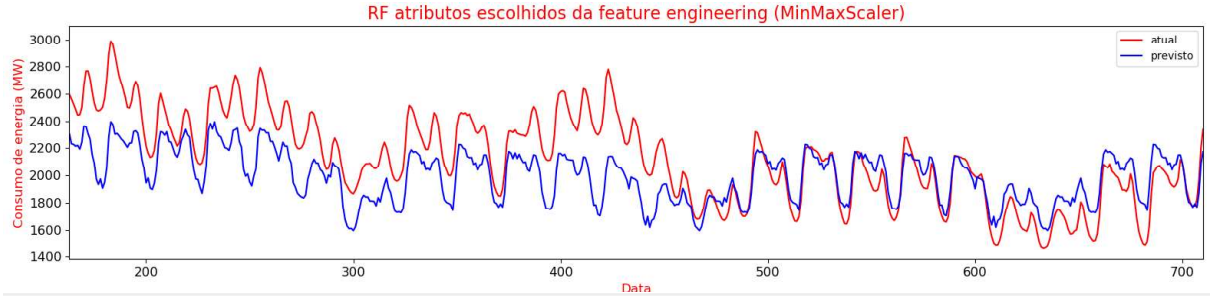


Figura 3.18: Excerto da comparação do conjunto de dados atual e previsto RF com *feature engineering* (atributos escolhidos)

Tendo em conta o Quadro 3.8, pode-se verificar que os resultados não são muito diferentes em relação ao teste anterior, continuando o LSTM a ser o modelo que melhor resultado obteve. Com a escolha dos atributos, isto é, eliminando o "year" e "day", o RF obteve melhor resultado, ao contrário do que aconteceu com o modelo LSTM, onde, curiosamente, o resultado foi pior, confirmando, assim, o que foi dito no final da secção anterior.

Relativamente às Figuras 3.17 e 3.18, não existe uma grande diferença em relação às do teste anterior, também devido à pequena diferença nos resultados das avaliações.

Em relação ao tempo de treino, em todos os testes efetuados, não diferiu muito. No geral o treino do RF foi sempre mais rápido que os restantes modelos, seguido do ARIMA e do LSTM, o que vem comprovar efetivamente o que foi referido no capítulo do estado de arte, que os algoritmos de *Deep Learning* necessitam de bastante poder computacional para serem treinados.

Os três modelos a seguir têm em conta o tipo de normalização utilizado. Para isso, serão utilizados os conjuntos de dados que melhores resultados obtiveram em cada modelo nos testes efetuados anteriores. Para o LSTM será utilizado o conjunto de dados com todos os atributos da *Feature Engineering*, para o ARIMA o conjunto de dados original e para o RF o conjunto com os atributos escolhidos.

3.3.4 Normalização (MinMaxScaler)

Para este teste, foi utilizado o tipo de normalização MinMaxScaler, com valores compreendidos entre 0 e 1, o mesmo tipo de normalização utilizado pelos testes anteriores. Segue-se o Quadro 3.9 com os resultados.

Quadro 3.9: Resultados dos modelos com a normalização MinMaxScaler

	LSTM	ARIMA	RF
RMSE	143	626	223
Tempo de treino	4.94s	0.1s	0.06s

Para uma melhor visualização dos resultados, construiu-se um gráfico para cada modelo com a comparação do conjunto atual e previsto. As Figuras 3.19, 3.20 e 3.21 apresentam um excerto dos resultados dos respetivos modelos.

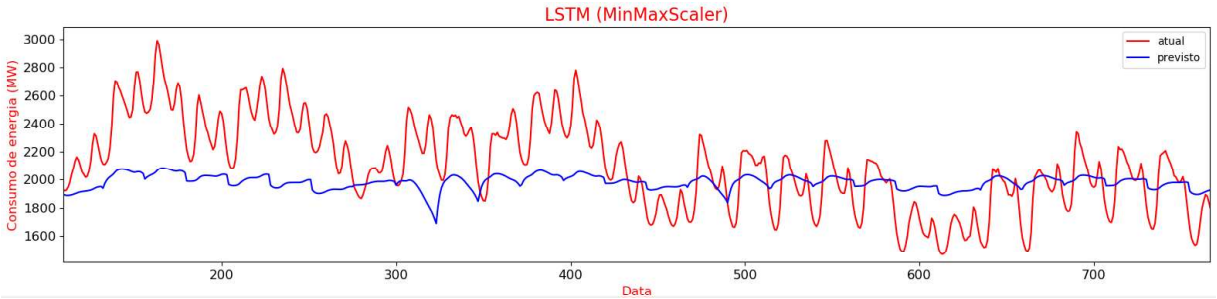


Figura 3.19: Excerto da comparação do conjunto de dados atual e previsto LSTM com a normalização MinMaxScaler

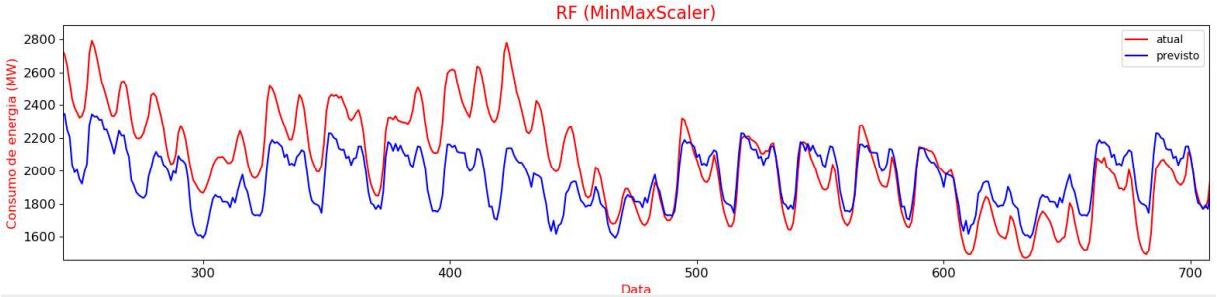


Figura 3.20: Excerto da comparação do conjunto de dados atual e previsto RF com a normalização MinMaxScaler

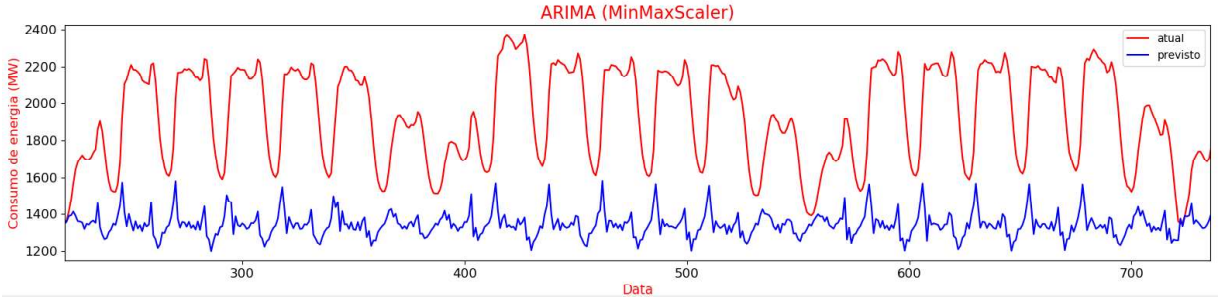


Figura 3.21: Excerto da comparação do conjunto de dados atual e previsto ARIMA com a normalização MinMaxScaler

Tendo em conta os resultados do Quadro 3.9, O LSTM continua a ter melhor resultado que os restantes modelos. Verificando as Figuras 3.19, 3.20 e 3.21, o que parece apresentar melhores resultados é o RF, porém, como já foi dito anteriormente, muitos dos valores previstos estão distantes dos originais e, daí, os resultados serem mais baixos que os do LSTM.

3.3.5 Normalização (RobustScaler)

Neste teste, foi utilizado o tipo de normalização RobustScaler para os três modelos. Segue-se a tabela com os resultados.

Quadro 3.10: Resultados dos modelos com a normalização RobustScaler

	LSTM	ARIMA	RF
RMSE	88	243	285
Tempo de treino	4.9s	0.1s	0.05s

Para uma melhor visualização dos resultados, construiu-se um gráfico para cada modelo com a comparação do conjunto atual e previsto. As seguintes figuras apresentam um excerto dos resultados dos respetivos modelos.

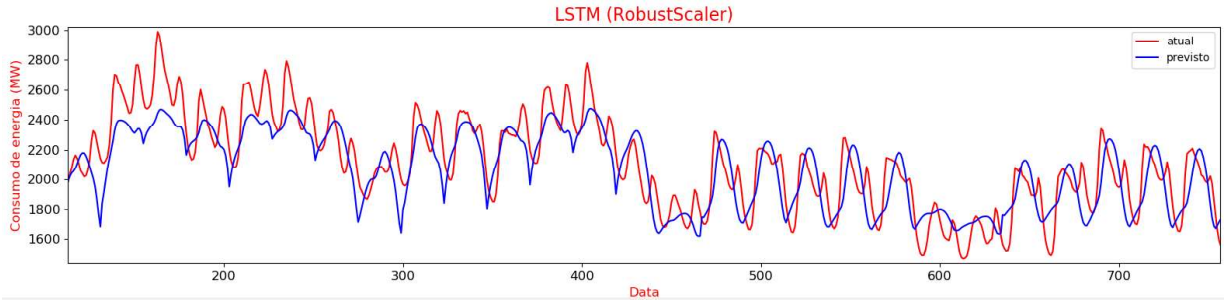


Figura 3.22: Excerto da comparação do conjunto de dados atual e previsto LSTM com a normalização RobustScaler

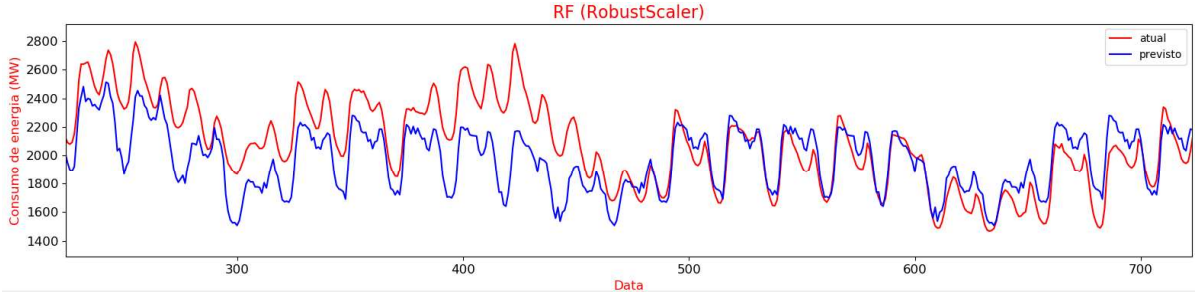


Figura 3.23: Excerto da comparação do conjunto de dados atual e previsto RF com a normalização RobustScaler

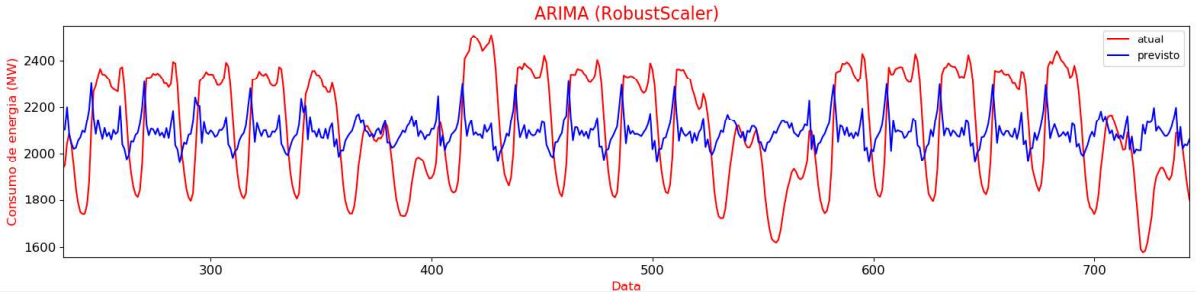


Figura 3.24: Excerto da comparação do conjunto de dados atual e previsto ARIMA com a normalização RobustScaler

Tendo em conta os resultados apresentados no Quadro 3.10, os modelos LSTM e ARIMA obtiveram um melhor resultado em relação ao teste da normalização anterior. O mesmo não se pode dizer sobre o RF, que obteve um pior resultado.

Verificando as Figuras 3.22, 3.23 e 3.24, pode-se concluir que houve uma grande diferença nos modelos LSTM e ARIMA em relação ao tipo de normalização anterior. Pode-se verificar, no caso do LSTM, que o modelo começa a aprender os diversos padrões que o conjunto de dados atual tem.

3.3.6 Normalização (StandardScaler)

No último teste, foi utilizado o tipo de normalização StandardScaler para os três modelos. O Quadro 3.11 apresenta os resultados dos modelos.

Quadro 3.11: Resultados dos modelos com a normalização StandardScaler

	LSTM	ARIMA	RF
RMSE	97	266	226
Tempo de treino	4.9s	0.1s	0.05s

Para uma melhor visualização dos resultados, construiu-se um gráfico para cada modelo com a comparação do conjunto atual e previsto. As Figuras 3.25, 3.26 e 3.27 apresentam um excerto dos resultados dos respetivos modelos.

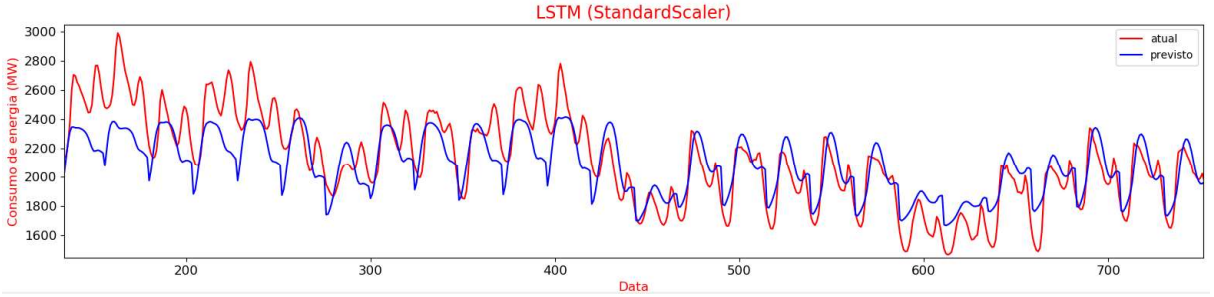


Figura 3.25: Excerto da comparação do conjunto de dados atual e previsto LSTM com a normalização StandardScaler

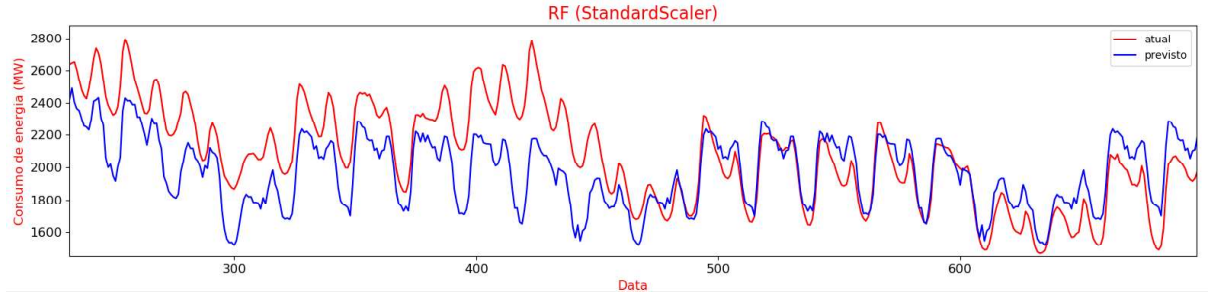


Figura 3.26: Excerto da comparação do conjunto de dados atual e previsto RF com a normalização StandardScaler

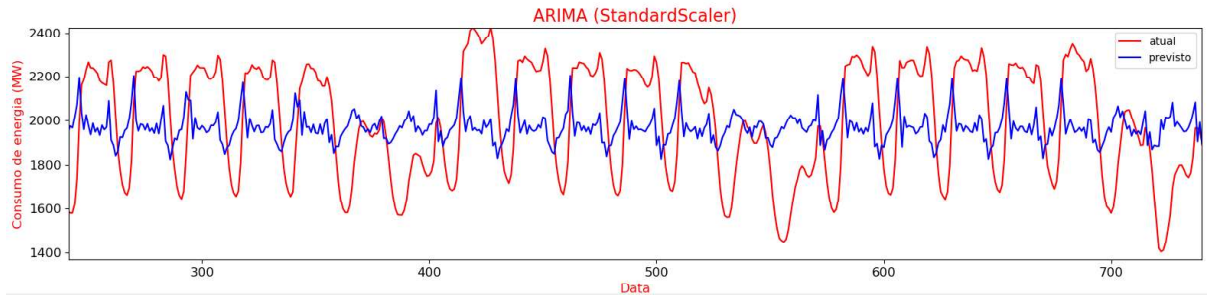


Figura 3.27: Excerto da comparação do conjunto de dados atual e previsto ARIMA com a normalização StandardScaler

Tendo em conta os resultados apresentados do Quadro 3.11, o algoritmo que melhor resultado obteve foi, novamente, o LSTM, seguido do RF e, por último, o ARIMA. Neste tipo de normalização, em comparação com o RobustScaler, os modelos LSTM e ARIMA tiveram pior resultado, embora semelhante. No caso do RF, teve melhor resultado com o StandardScaler, muito parecido com o teste da normalização MinMaxScaler.

Verificando as Figuras 3.25, 3.26 e 3.27, não houve grande diferença em relação ao tipo de normalização RobustScaler.

Em relação aos tempos de treino, foram, mais uma vez, semelhantes, pelo que se pode concluir que, para estes modelos, o tipo de normalização a utilizar não tem muita influência no tempo de treino dos mesmos.

Porém, em relação ao resultado, pode-se concluir que o tipo de normalização a utilizar tem, em alguns casos, grande influência no resultado.

Em suma, no caso do LSTM, obteve melhores resultados com o conjunto de dados com todos os atributos provenientes da *Feature Engineering* e com o tipo de normalização RobustScaler, no caso do ARIMA, com o conjunto de dados original e com o tipo de normalização RobustScaler e, por último, no caso do RF com o conjunto de dados com os atributos escolhidos e com o tipo de normalização MinMaxScaler. O Quadro 3.12 apresenta o resumo dos resultados dos modelos testados anteriormente.

Quadro 3.12: Resumo dos resultados dos diferentes modelos testados

	LSTM	ARIMA	RF
Sem *feature engineering*	389	626	498
Com *feature engineering* (todos os atributos)	143	-	260
Com *feature engineering* (atributos escolhidos)	152	-	223
Normalização (MinMaxScaler)	143	626	223
Normalização (RobustScaler)	88	243	285
Normalização (StandardScaler)	97	266	226

Com base nos resultados anteriores, pode-se concluir que o modelo que obteve melhores resultados, no geral, foi o LSTM e, por isso, foi o modelo escolhido para a previsão de energia elétrica.

3.4 Ajuste do modelo escolhido

Escolhido o modelo que obteve melhores resultados, o LSTM, foi necessário ajustá-lo, com o objetivo de fazer previsões mais acertadas.

Uma das características que as redes neuronais têm é que não existe uma regra geral que diga que é necessário um determinado número de épocas ou número de camadas escondidas para oferecer melhores previsões. Devido a isso, para encontrar o valor ideal dos parâmetros, é necessário testar diversos valores para os parâmetros que a rede neuronal requer.

Poderão existir algumas técnicas que ajudam a encontrar bons valores nas primeiras tentativas como, por exemplo, se um conjunto de dados tiver uma grande complexidade, é preferível ter um número maior de camadas escondidas, com o objetivo de o modelo ser capaz de treinar com maior eficácia os padrões do conjunto.

Tendo em conta o modelo com melhor resultado da secção anterior, foram construídos vários modelos com os diferentes parâmetros: número de épocas, número de camadas, número de *units*, tamanho do *batch*, o tipo de função de ativação, o *Dropout* e o número de dados anteriores.

3.4.1 Épocas

Nos modelos, não só de *machine learning*, mas também de *deep learning*, é recorrente utilizar-se este parâmetro. O número de épocas (*epochs*) diz respeito ao número de vezes que o modelo vai percorrer o conjunto de dados para treinar. Normalmente o número de épocas a utilizar é alto, como 100 ou 200 ou até ser capaz de minimizar o erro (Brownlee, 2019).

Para este teste foram utilizados números de épocas entre o 1 e o 100. O Quadro 3.13 apresenta os resultados obtidos.

Quadro 3.13: Resultados dos modelos tendo em conta o número de épocas

Épocas	RMSE	Tempo de treino
1	88	00:00:05
5	59	00:00:16
10	47	00:00:30
20	28	00:00:56
30	27	00:01:25
40	27	00:01:53
50	26	00:02:20
60	27	00:02:59
70	27	00:03:14
80	27	00:03:58
90	27	00:04:19
100	27	00:05:00

Tendo em conta os resultados do Quadro 3.13, pode-se concluir que, quanto maior o número de épocas, menor é o RSME. A partir da época 30, o RSME mantém-me, isto significa que, treinar a partir dessa época torna-se insignificante. Com os resultados pode-se concluir, também, que quanto maior o número de épocas, mais tempo o modelo demora a treinar, o que é compreensível, visto que o modelo vai ter de percorrer novamente o conjunto de dados em cada época.

3.4.2 *Units*

O número de *units* representa o número de *neurons* que a rede neuronal vai ter. Para este teste, testaram-se para números de *units* entre 1 e 500, todos eles para 100 épocas. Segue-se o Quadro com os resultados.

Quadro 3.14: Resultados dos modelos tendo em conta o número de *units*

Units	RMSE	Tempo de treino
1	27	00:05:00
10	13	00:05:37
50	10	00:05:59
100	9	00:06:45
200	9	00:08:24
300	9	00:11:47
400	9	00:15:26
500	9	00:19:37

Com base nos resultados do Quadro 3.14, à semelhança do que aconteceu com o número de épocas, quanto maior for o número de *units*, menor o valor do RMSE. A partir do número 100 de *units*, o valor do RMSE mantém-se, pelo que não vale a pena aumentar o número de *units*. Em relação ao tempo de treino, quanto maior o número de *units*, maior é o tempo.

3.4.3 Camadas escondidas (*Hidden layers*)

Como foi dito no estado da arte, uma rede neuronal pode ter várias camadas, sendo que se houver uma ou mais camadas para além das camadas de entrada e saída, o modelo é considerado como modelo de *deep learning*. Para este teste foram testados para um número de camadas entre 0 e 5, com 100 épocas e 100 *units*. Segue-se o Quadro 3.15 com os resultados.

Quadro 3.15: Resultados dos modelos tendo em conta o número de camadas escondidas

Camadas escondidas	RMSE	Tempo de treino
0	9	00:05:45
1	9	00:09:05
2	9	00:11:49
3	9	00:16:43
4	9	00:19:51
5	10	00:23:21

Tendo em conta os resultados do Quadro 3.14, pode-se concluir que o aumento do número de camadas escondidas não faz diminuir o valor de RMSE, até pelo contrário, com 5 camadas o valor de RMSE aumentou. Isto pôde ter acontecido devido ao facto de este conjunto de dados não ter detalhe suficiente para justificar a utilização de camadas escondidas. Por isso, para este conjunto de dados, é irrelevante utilizar ou não camadas escondidas. Em relação ao tempo de treino, quanto maior o número de camadas escondidas, maior o tempo.

3.4.4 *Batch*

O parâmetro *Batch*, que faz parte de diversos modelos de *machine learning*, define o número de dados que vão ser processados pelo modelo antes dos pesos da rede neuronal serem atualizados. Isto é, no final de cada *batch* ser processado, as previsões são calculadas com as variáveis de saída e, com base nesses cálculos é calculado o erro. Através desse erro, são atualizados os pesos do modelo, com o objetivo de ir aumentando a eficácia do modelo.

Para este teste, foram feitos modelos com tamanhos de *batch* entre 1 e 10000. Em cada modelo, foi utilizada 1 época, 1 *unit* e 2 camadas escondidas. Estes não foram os parâmetros com menor RMSE, porém se fossem utilizados os parâmetros com menor RMSE, os testes demorariam bastante tempo e o valor de RMSE muito provavelmente não alteraria muito. Segue-se, o Quadro com os resultados.

Quadro 3.16: Resultados dos modelos tendo em conta o tamanho do *batch*

Tamanho do batch	RMSE	Tempo de treino
1	58	00:09:17
5	67	00:02:15
10	72	00:01:08
20	81	00:00:36
50	94	00:00:17
100	123	00:00:10
200	135	00:00:07
500	148	00:00:05
1000	153	00:00:04
10000	154	00:00:03

Com base nos resultados do Quadro 3.16, pode concluir-se que o tamanho do *batch* influencia na previsão. Quanto maior for o tamanho do *batch*, maior é o valor de RMSE, isto é, menor será a precisão da previsão. Relativamente ao tempo de treino, pode-se verificar que quanto menor o tamanho do *batch*, maior o tempo de treino. Este tempo vai ao encontro do

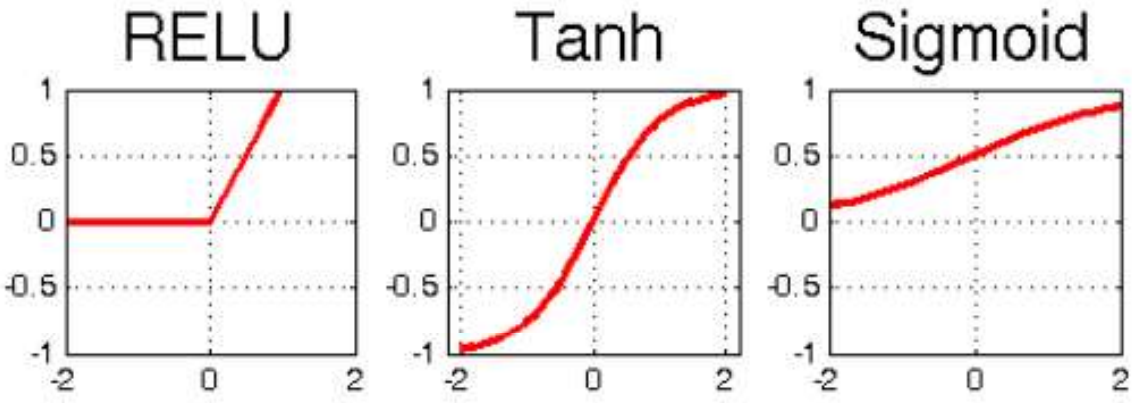
expectável, visto que quanto menor o tamanho do *batch*, mais vezes serão efetuados os cálculos intermédios, pelo que o treino demorará mais tempo.

3.4.5 Função de ativação

A função de ativação, que faz parte de qualquer rede neuronal, tem como objetivo calcular o *output* de um nó que irá servir de *input* para o(s) próximo(s). Esta permite evitar que informação desnecessária passe para os próximos *neurons* e, devido a isso, as funções de ativação têm um papel bastante importante e têm influência direta no sucesso ou insucesso do modelo (Hamdan, 2018).

Existem diversas funções de ativação, sendo a mais conhecida e utilizada na maior parte das redes neuronais a *ReLu* (*Rectified Linear Units*), devido a ser uma das mais simples e eficaz (Ramachandran, Zoph e Le, 2017).

A Figura 3.28 mostra o comportamento dos três tipos de função de ativação utilizados.



[10]

Figura 3.28: Comportamento das funções de ativação *ReLu*, *Tanh* e *Sigmoid*

Para este teste foram testados modelos com as três funções de ativação mais conhecidas: *Tanh* (*Hyperbolic Tangent*), *Sigmoid* e *ReLu* (*Rectified Linear Units*). Os restantes parâmetros utilizados foram: 1 época, 2 camadas escondidas, 100 *units* e 20 como tamanho de *batch*. Segue-se o Quadro 3.17 com os resultados obtidos.

Quadro 3.17: Resultados dos modelos tendo em conta a função de ativação

Função de ativação	RMSE	Tempo de treino
Tanh	15	00:37:15
Sigmoid	20	00:02:43

[10] Hamdan, 2018

ReLu	16	00:02:55

Tendo em conta os resultados do Quadro 3.16, pode-se concluir que a função de ativação que melhor resultado teve foi a *Tanh*, seguido da *ReLu*, onde obteve um resultado bastante próximo e, por último, a *Sigmoid*. Em relação ao tempo de treino, a *Tanh* foi a que menor tempo teve, seguido da *Sigmoid* e da *ReLu*, com uma grande diferença em relação à primeira.

3.4.6 *Dropout*

Na validação de modelos de *machine learning*, por vezes, verifica-se que a previsão do conjunto do teste se ajusta muito bem ao conjunto do teste original, o que, à partida, se conclui que o modelo está a prever bem. Porém, nem sempre é verdade, uma previsão do conjunto poderá ajustar-se bem e a previsão de novos dados poderá ter dados muito diferentes do esperado. A isto chama-se de *overfitting* (sobreajuste), um conceito bastante importante e que deve ser sempre evitado.

Para evitar que o *overfitting* aconteça, foi utilizado um parâmetro, que a biblioteca keras oferece, chamado *Dropout*. Este é uma técnica bastante simples onde se eliminam temporariamente uma percentagem de *units* aleatórias, juntamente com as suas conexões a outras *units*, durante o treino do modelo, como se pode verificar na Figura 3.29. Com isto, o modelo é forçado a aprender com menos *units*, tornando-o, assim, mais robusto. Fica mais bem preparado para qualquer eventual mudança nos dados, oferecendo melhores previsões em diversas situações (Srivastava Hinton, Krizhevsky, Sutskever, e Salakhutdinov, 2014).

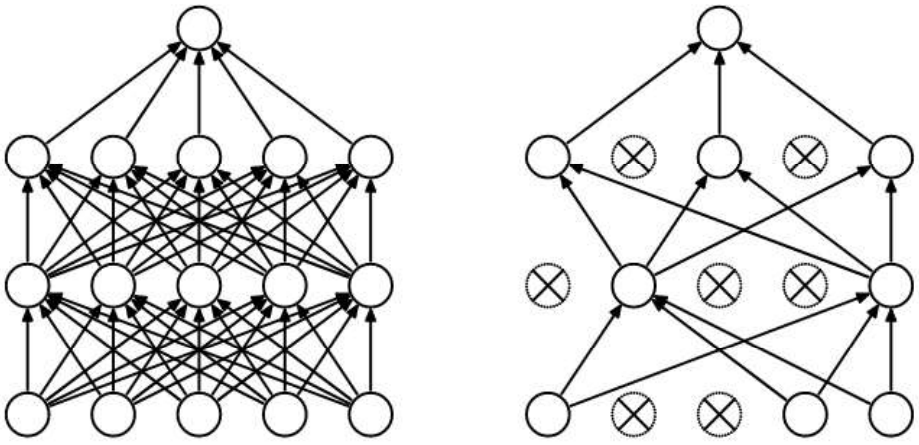


Figura 3.29: Exemplo de uma rede neuronal sem (esquerda) e com (direita) *Dropout*

Para este teste, o importante não foi encontrar o valor do *dropout* que oferecesse o melhor valor de RMSE, mas sim o valor de *dropout* que não originasse *overfitting* do modelo. Para isso, foram construídos quatro modelos, o primeiro sem qualquer *dropout*, o segundo com um *dropout* de 0.1, o terceiro com 0.2 e o último com 0.5, o que equivale a 0%, 10%, 20% e 50% de *units* que foram temporariamente removidas, respetivamente.

Todos os modelos foram construídos com os seguintes parâmetros: 100 épocas, tamanho do *batch* de 20 dados, função de ativação *Tanh* e 100 *units*.

Uma técnica para verificar se um modelo está com *overfitting* é verificar a perda do treino e do teste ao longo das épocas. Se, ao longo das épocas, o valor da perda do treino crescer, é sinal de *overfitting*, se o valor da perda do treino e do teste decrescer ao longo do tempo, não ocorreu *overfitting*.

Seguem-se, na Figura 3.30, o gráfico com as perdas do modelo sem *dropout*.

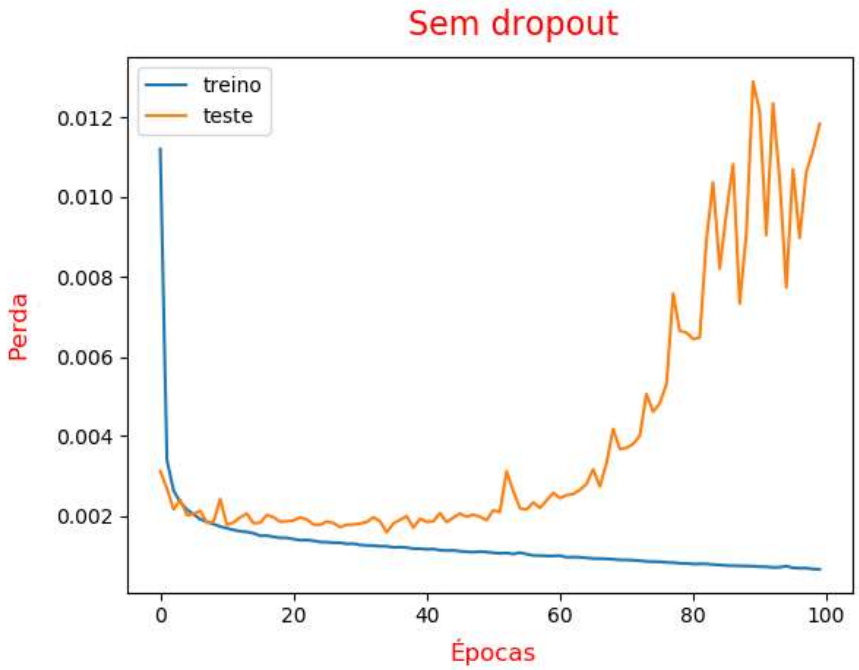


Figura 3.30: Perdas dos conjuntos de treino e teste sem aplicação do *dropout*

Tendo em conta a Figura 3.30, pode-se verificar que, de facto, sem *dropout* existe *overfitting* no treino do modelo, pois, por volta da época 40, a perda do conjunto de teste começa a aumentar significativamente, enquanto que a do treino diminui gradualmente.

A Figura 3.31 refere-se ao resultado das perdas do modelo com um *dropout* de 10%.

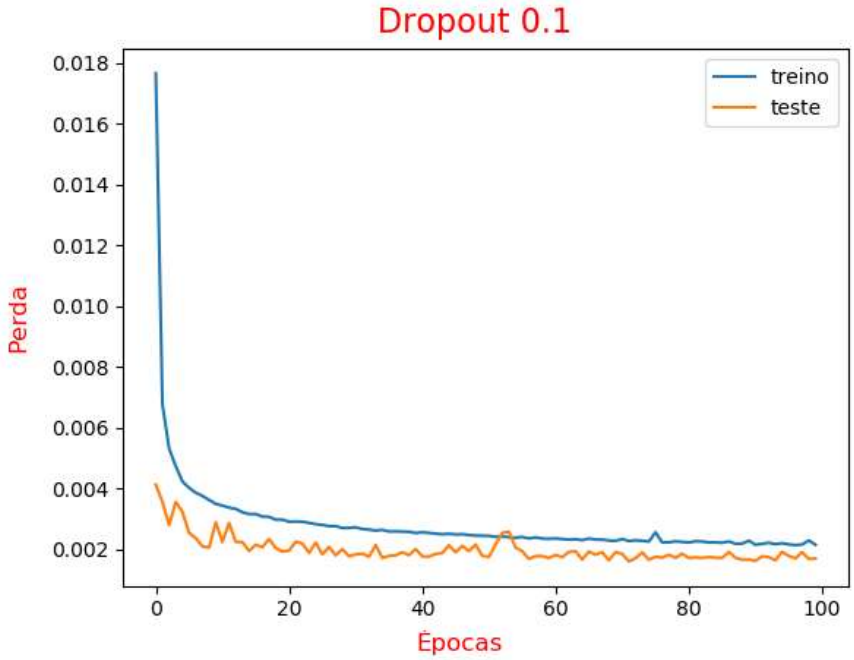


Figura 3.31: Perdas dos conjuntos de treino e teste com a aplicação de um *dropout* de 0.1

Relativamente ao à Figura 3.31, pode-se concluir que o *dropout* tem, de facto, grande influência no treino do modelo. Entre a época 40 e 60, existe um pico relativamente à perda de teste que até se sobrepõe à perda do treino, o que poderá ser um indicativo de que ainda existe *overfitting* no modelo.

Seguidamente, na figura 3.32, apresenta-se o gráfico relativamente à perda do modelo com o dropout de 20%.

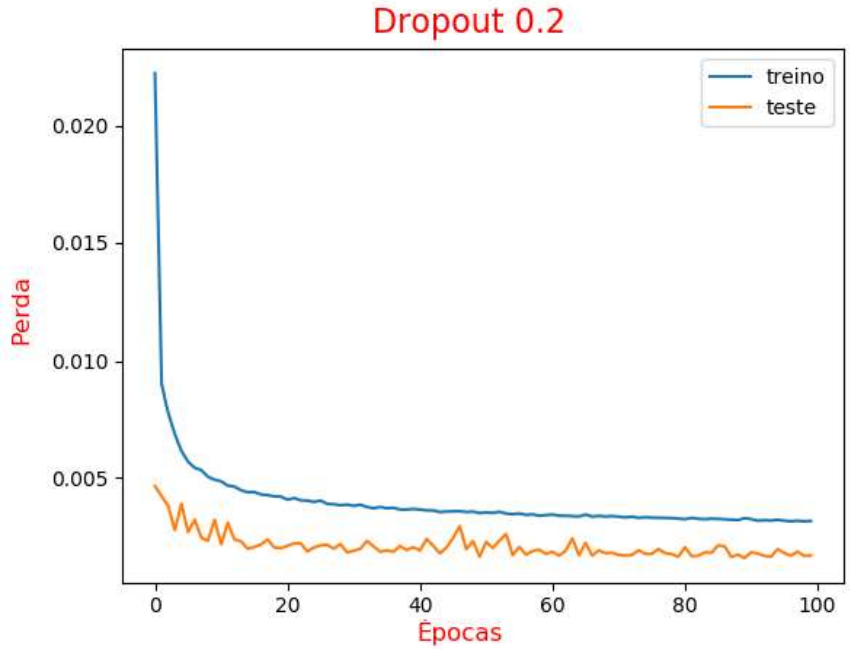


Figura 3.32: Perdas dos conjuntos de treino e teste com a aplicação de um *dropout* de 0.2

Relativamente à Figura 3.32, pode-se verificar que a perda do teste diminui gradualmente e não se sobrepõe em nenhum momento à perda do treino, o que significa que não existe *overfitting*.

A figura seguinte representa as perdas relativamente ao modelo com um *dropout* de 50%.

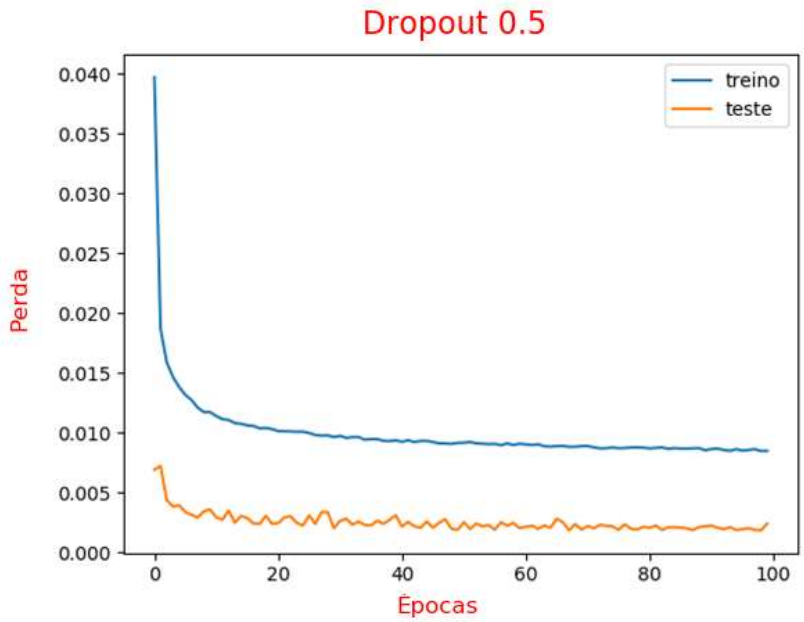


Figura 3.33: Perdas dos conjuntos de treino e teste com a aplicação de um *dropout* de 0.5

Tendo em conta a Figura 3.33, pode-se verificar que também não existe *overfitting*. Porém, os valores das perdas do conjunto de treino e do teste estão um pouco distanciadas, o que poderá ser um sinal de que o modelo está com *underfitting* (subajuste), isto é, o modelo não se consegue adaptar bem ao conjunto de dados com os quais foi treinado, oferecendo más previsões de novos valores.

Com base nas figuras anteriores relativas ao *Dropout*, pode-se concluir que o melhor valor deste, para este conjunto de dados, é 0.2 (20%), para que o modelo não tenha *overfitting* nem *underfitting*.

3.4.7 Dados anteriores

A biblioteca Keras, para treinar um modelo LSTM, requer sempre os dados de entrada no formato de uma matriz tridimensional, apresentada na Figura 3.34. A primeira dimensão significa o tamanho do *batch*, explicado anteriormente, a segunda refere-se aos dados anteriores (*time steps*) que a rede neuronal vai ter de relembrar em cada conjunto de *batch* e a última representa os atributos do conjunto de dados (Brownlee, 2018).

O número de dados anteriores é um dos parâmetros mais importantes da arquitetura LSTM, pois um número maior de dados anteriores, em teoria, oferece melhores previsões, visto que a

rede vai ter como base um maior número de dados anteriores para usar na atualização dos pesos internos no treino do modelo. Porém, um maior número de dados anteriores significa que a matriz vai ter uma maior dimensão, o que significa que o modelo vai levar mais tempo a treinar e serão necessários mais recursos computacionais no treino.

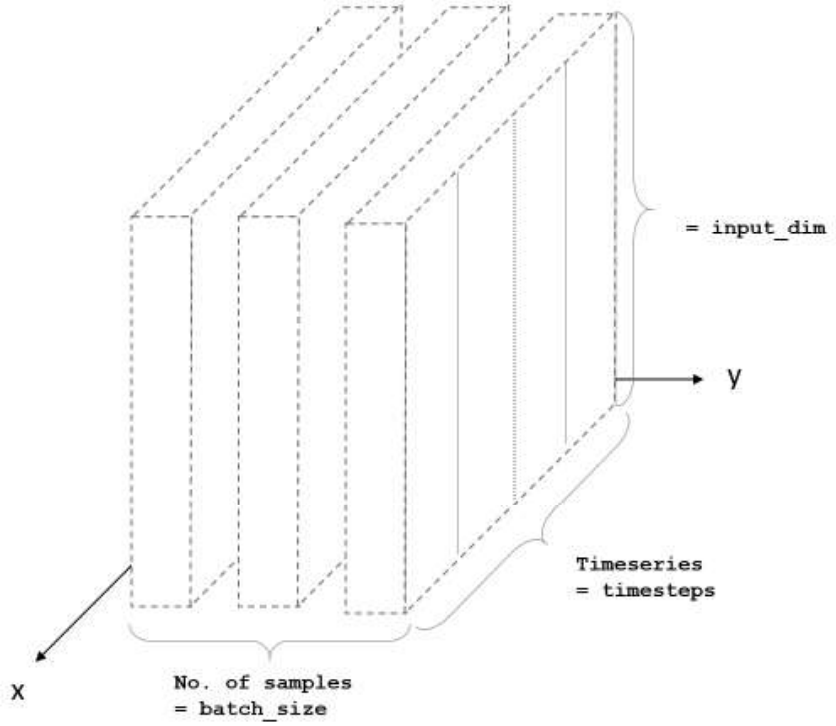


Figura 3.34: Representação do formato dos dados de entrada requerida pela biblioteca Keras relativamente ao modelo LSTM[11]

Para este parâmetro, foram testados modelos com um número de dados anteriores entre 1 e 1000, com o objetivo de perceber se o valor de RMSE altera consoante o número de dados anteriores. Em cada modelo, para que o treino fosse geralmente rápido e apresentasse valores de RMSE diferentes nos vários modelos, foi utilizada 1 época, 1 *unit* e 2 camadas escondidas, um tamanho de *batch* de 20 e um *dropout* de 0.2.

Segue-se o Quadro 3.18 com os resultados dos diversos modelos.

Quadro 3.18: Resultados dos modelos tendo em conta o número de dados anteriores

Dados anteriores	RMSE	Tempo de treino
1	74	00:00:29
5	78	00:00:31

[11] Fonte: https://mc.ai/understanding-input-and-output-shape-in-lstm-keras/ [Consultado em: 28-08-2020]

10	75	00:00:32
20	81	00:00:37
50	110	00:00:43
100	86	00:00:56
200	86	00:01:30
500	83	00:02:58
1000	16	00:09:07

Tendo em conta o Quadro 3.18, pode-se concluir que, de facto, o número de dados anteriores tem influência no tempo de treino: quanto maior for o número de dados anteriores, maior o tempo de treino do modelo. Em relação ao valor de RMSE, não se pode tirar muitas conclusões, pois não existe um padrão. Porém, o modelo com 1000 dados anteriores obteve um melhor resultado, o que poderá indicar que um grande número de dados anteriores leva a um melhor resultado.

É de referir que, não só se pode ter em conta o parâmetro que dá melhor resultado, no mais baixo valor de RMSE, mas também para o tempo de treino, poder computacional necessário para treinar o modelo e o tipo de previsão. Por exemplo, se o objetivo é treinar modelos para fazer previsões de até um ou dois dias e o sistema computacional tiver pouco poder de processamento, seria melhor escolher um baixo número de dados anteriores.

3.5 Previsão de novos dados

Tendo já uma visão geral dos parâmetros que dão um melhor resultado, o tempo de treino e a quantidade de poder computacional que utilizam, passou-se para a parte da previsão de novos dados, para além dos que estão presentes no conjunto de dados.

A biblioteca Keras somente consegue fazer a previsão de 1 dado. Não oferece nenhuma função onde receba um número de previsões que o utilizador quer e retorne o resultado da quantidade dessas previsões. Por isso, foi necessário construir uma função que retorne as previsões com base no número das mesmas que o utilizador quiser.

A função criada utiliza o conjunto de dados de teste (devido a ser o conjunto de dados mais recente) como base para a previsão de um dado no futuro, designadamente, 1 hora no futuro, pois o conjunto de dados tem dados a cada hora. Seguidamente, acrescenta o dado da previsão numa lista de resultados a retornar, acrescentando, também, esse dado no final do conjunto de dados de teste e elimina o primeiro dado do mesmo conjunto, o mais antigo, para que o conjunto permaneça com o mesmo tamanho, tamanho esse definido no parâmetro de dados anteriores.

Segue-se a Figura 3.35 com o pseudocódigo da função criada.

```
1: função (conjunto de teste, modelo treinado, número de previsões)
2:     results = [] //array vazio
3:     valores_teste = conjunto de teste
4:     ciclo for (número de previsões)
5:         previsão para o conjunto de testes
6:         acrescentar ao results o último valor da previsão
7:         remoção do primeiro dado de valores_teste e acrescentar a previsão no fim
8:     fim do ciclo for
9:     Inversão da normalização dos valores em results
10:    return results
```

Figura 3.35: Pseudocódigo da função para a previsão de valores futuros

Esta função tem uma única desvantagem, que é ao longo das previsões, o modelo vai utilizando os dados já previstos como base para a previsão dos próximos valores. Se se fizer uma previsão de, por exemplo, 5000 dados, é provável que, a partir de X dados a previsão venha a ter dados pouco realistas. Porém, é a única forma de obter dados futuros com base no tempo utilizando o LSTM.

Com a função criada fez-se, então, o modelo final a utilizar na REST API. Com base nos testes efetuados na secção anterior, utilizou-se 100 *units*, 100 épocas, 100 como tamanho do *batch*, 1000 dados anteriores, 2 camadas escondidas, Tanh como função de ativação e um *dropout* de 0.2.

Segue-se, na Figura 3.36, um excerto de uma previsão de 1 semana, que são 168 horas e novas previsões para além do conjunto de testes.

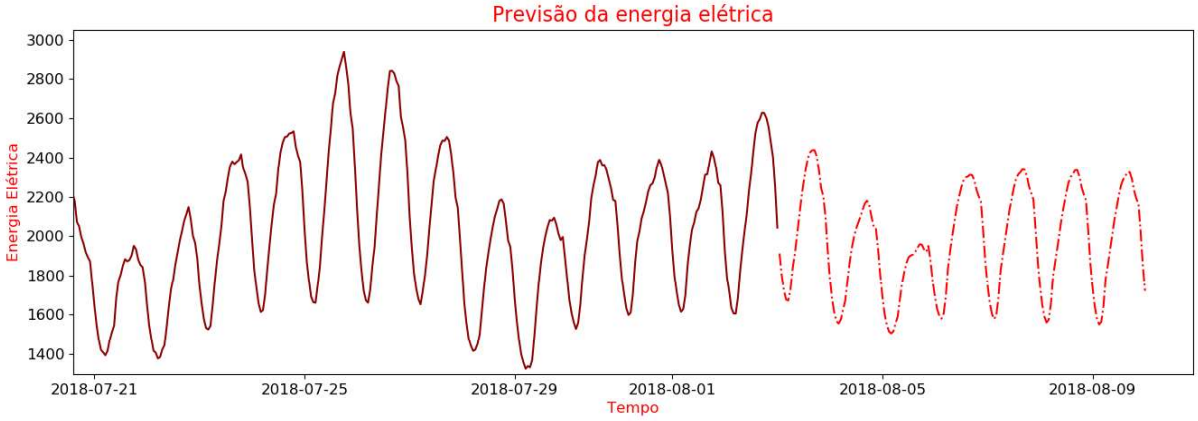


Figura 3.36: Excerto de uma previsão de 3 dias após o conjunto de teste

Ao verificar a Figura 3.36, consegue-se concluir que o modelo previu bastante bem, aprendendo alguns dos padrões que o conjunto de dados original tinha.

3.6 Visualização dos dados

Nos dias de hoje, muitas empresas utilizam, ainda, ferramentas como folhas de cálculo, com o objetivo de, não só guardar os diversos dados, mas também mostrá-los em gráficos ou tabelas, efetuar algumas estatísticas, entre outras operações. Por vezes, o uso deste tipo de ferramentas não é a melhor opção, devido à dificuldade na criação dos diversos gráficos e estatísticas e até mesmo devido à segurança.

Com o passar dos anos, foram surgindo várias ferramentas web, que substituíam algumas das funcionalidades que folhas de cálculo tinham, nomeadamente a construção de gráficos e tabelas. Daí, surgiram as ferramentas de *Business Intelligence* (BI), as quais ofereciam uma interface gráfica muito mais apelativa ao utilizador, a capacidade de se conectar a mais do que uma base de dados, adicionar utilizadores com permissões, entre outras capacidades.

Com isto, para uma melhor visualização dos dados da previsão de consumos de energia, utilizou-se uma ferramenta de BI denominada *Redash* para mostrar as previsões de energia elétrica produzidas pelo modelo. Esta foi escolhida devido à experiência de utilização e pelo facto de ter uma licença de código aberto.

Para que, não só esta ferramenta de BI, mas como a maior parte delas, seja capaz de mostrar dados, é necessário coloca-los numa base de dados relacional. Por isso, foi feito uma REST API, com o objetivo de gerar previsões, utilizando o modelo escolhido, para que estas sejam facilmente consumidas pelo *Redash*.

O objetivo da API é, através de um conjunto de dados de consumos de energia elétrica no formato ".csv", fazer previsões para os próximos tempos, utilizando o modelo escolhido na secção anterior, e mostrá-los numa ferramenta de *Business Intelligence*. Este projeto permite, também, a atualização do modelo de *machine learning* com novos dados que cheguem ao conjunto de dados, com o objetivo de fazer previsões mais precisas ao longo do tempo.

A arquitetura é constituída nomeadamente por 3 componentes, que se interligam entre si: a aplicação web, a base de dados e a ferramenta de *business intelligence*. A arquitetura da API é apresentada através da Figura 3.37.

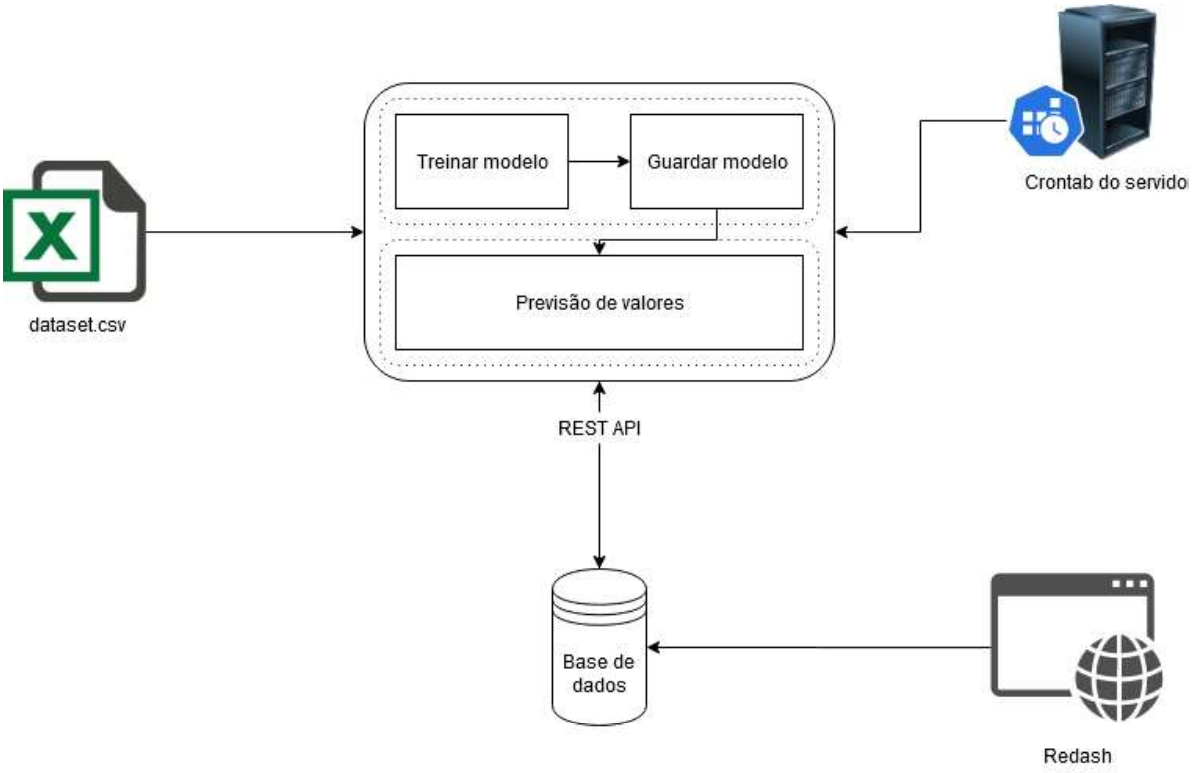


Figura 3.37: Arquitetura da REST API

O primeiro componente é uma aplicação web que disponibiliza uma API, em REST. Para o desenvolvimento desta aplicação, foi utilizada uma *framework* web, escrita na linguagem de programação Python, denominada *Flask*. Esta *framework* foi utilizada devido a ser escrita na mesma linguagem em que foram escritos os modelos de *machine learning*, pela experiência de utilização e pela grande comunidade de utilizadores que a utilizam.

A API é composta por duas partes, cada uma com o seu *endpoint*: a primeira tem como objetivo treinar o modelo com o conjunto de dados com os novos dados recolhidos e guardá-los na base de dados e o segundo de retornar a quantidade de valores previstos que o utilizador passar no *endpoint* e guardar estes dados, também, na base de dados.

O primeiro *endpoint* foi feito para, no caso de chegarem novos dados ao conjunto de dados, ser possível treinar novamente o modelo, pois como foi referido na secção anterior, a partir de certo número de previsões, estas começam a ser pouco realistas. Treinando novamente o modelo, resolve este problema, sendo possível obter, nos futuros próximos dados previstos, melhores previsões.

Segue-se, na Figura 3.38, um exemplo da resposta do *endpoint* de treino do modelo, no formato JSON.

```
GET api/train

{
  "success": true,
  "message": "train completed"
}
```

Figura 3.38: Resposta do *endpoint* de treino do modelo

O segundo *endpoint* retorna os valores previstos, dado o número de dias requeridos pelo utilizador, guardando esses valores na base de dados.

A Figura 3.39 refere-se a um exemplo de resposta do *endpoint* de 5 dados, isto é, 5 horas, no formato JSON.

```
GET api/forecast/5

{
  "data": [
    {
      "timestamp": "Fri, 03 Aug 2018 01:00:00 GMT",
      "value": 1856.8
    },
    {
      "timestamp": "Fri, 03 Aug 2018 02:00:00 GMT",
      "value": 1726.7
    },
    {
      "timestamp": "Fri, 03 Aug 2018 03:00:00 GMT",
      "value": 1636.3
    },
    {
      "timestamp": "Fri, 03 Aug 2018 04:00:00 GMT",
      "value": 1587.9
    },
    {
      "timestamp": "Fri, 03 Aug 2018 05:00:00 GMT",
      "value": 1599.0
    }
  ],
  "success": true
}
```

Figura 3.39: Exemplo de resposta do *endpoint* de previsão de 5 dados

Para automatizar o processo de treino e previsão, foi utilizada uma ferramenta própria de algumas distribuições do sistema operativo Linux denominada *Crontab*. Esta ferramenta permite adicionar comandos automaticamente em determinados períodos de tempo, definidos pelo utilizador. É uma ferramenta simples, de fácil de utilização e muito útil para esta API,

pois permite agendar principalmente os treinos dos modelos numa hora em que o servidor não tenha muita utilização, sem ser necessária qualquer intervenção manual.

O segundo componente da arquitetura da API é a base de dados. O sistema de gestão de base de dados escolhido foi o *PostgreSQL* devido, à experiência de utilização e à compatibilidade com o *Redash*.

A base de dados tem como objetivo guardar os valores, tanto do histórico dos dados dos consumos de energia recolhidos, como os valores previstos, para que a ferramenta de BI possa recolher esses dados e mostrá-los.

Para ligação da base de dados à API, foi utilizada uma biblioteca de mapeamento objeto-relacional chamada *SQLAlchemy*, devido a ter compatibilidade com a *framework* web *Flask*, o que torna a criação das tabelas e respetivas pesquisas bastante fácil e simples.

O esquema da base de dados, representado na Figura 3.40, é constituído por duas tabelas independentes, uma para o histórico de valores e outra para a previsão dos mesmos.

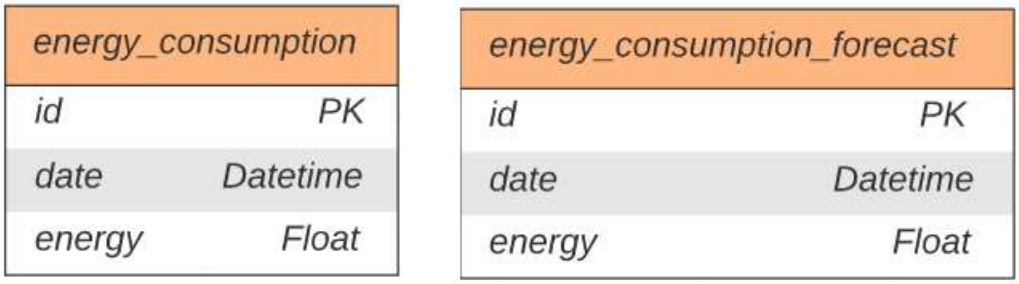


Figura 3.40: Tabelas utilizadas na API

O último componente refere-se à visualização dos dados, onde foi utilizado a ferramenta de *business intelligence* escolhida, *Redash*. A instalação foi bastante fácil, pois esta ferramenta de BI é disponibilizada em *Docker*, o que permite que seja instalada em qualquer ambiente sem a preocupação da instalação das várias dependências.

Esta ferramenta de BI é muito completa, sendo possível criar diversos tipos de gráficos, como por exemplo, gráfico de piza, de barras, de linhas, entre outros. Também é possível adicionar utilizadores com permissões, adicionar dados de múltiplas bases de dados, criar alertas e fazer todo o tipo de pesquisas nas bases de dados.

A figura 3.41 apresenta o *dashboard* com os dados do consumo de energia do conjunto original e os dados da previsão.

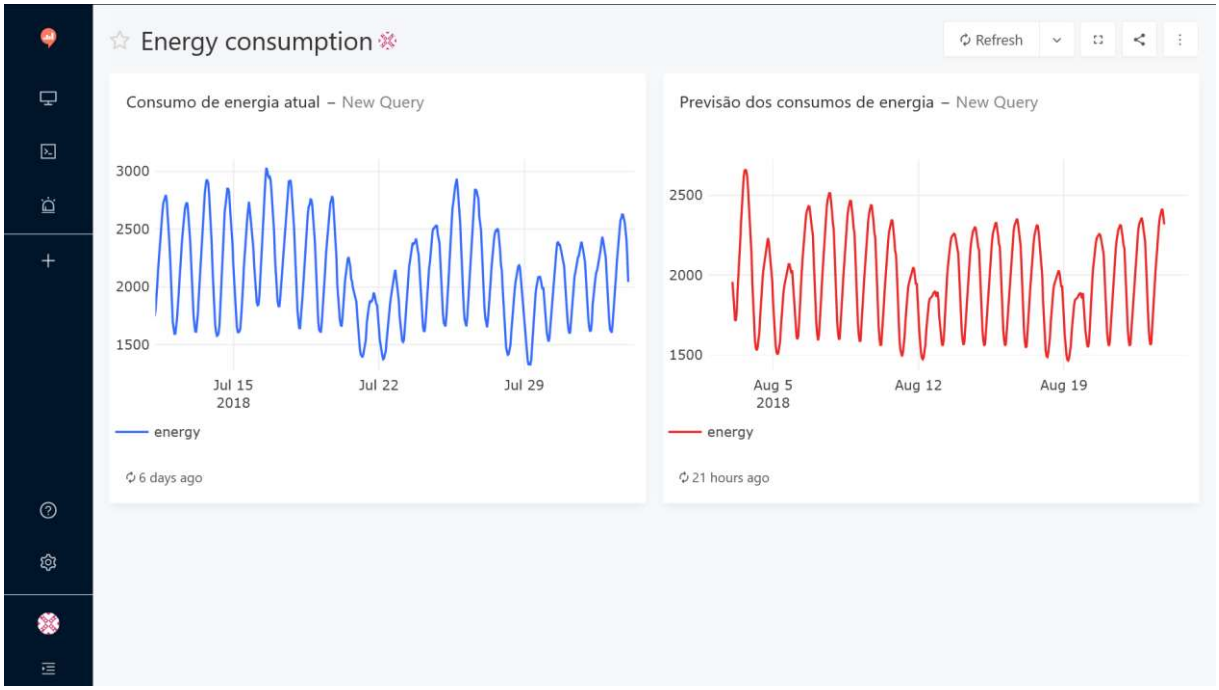


Figura 3.41: *Dashboard* com o conjunto de dados original e a previsão no *Redash*

Foram, também, adicionados alguns gráficos e indicadores que poderiam ser úteis para o utilizador final, com o objetivo de mostrar que, não só esta ferramenta, mas como todas as ferramentas de BI são importantes no que diz respeito à visualização dos dados, oferecendo uma melhor tomada de decisão por parte dos municípios e podendo, até, superar as típicas ferramentas como folhas de cálculo.

A Figura 3.42 apresenta um *dashboard* com algumas estatísticas relativamente ao conjunto de dados original.

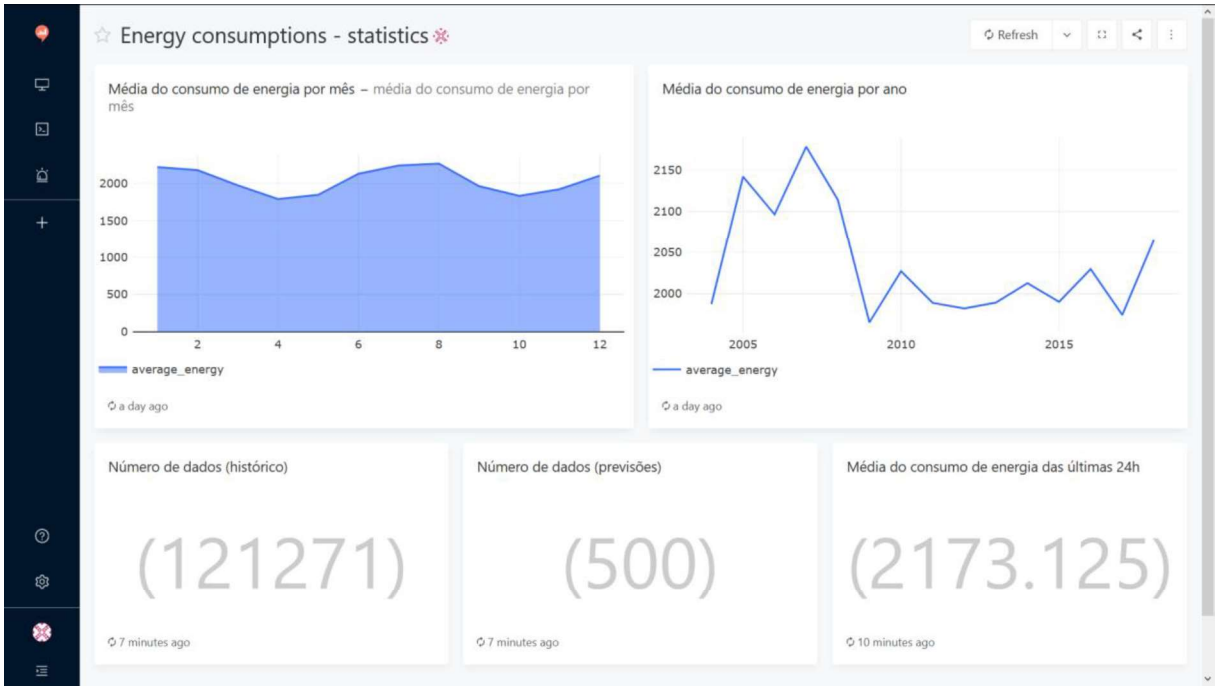


Figura 3.42: *Dashboard* com algumas estatísticas sobre o conjunto de dados original

4. Conclusões

O objetivo do presente trabalho centrou-se na aplicação de modelos de *machine learning* em dados de energia elétrica provenientes de uma cidade, para fazer previsões dos mesmos para os próximos tempos. Com estas previsões, seria possível mostrá-las numa ferramenta de *business intelligence*, permitindo que o município tivesse uma melhor visão dos dados e, consequentemente, uma melhor tomada de decisão.

Com o trabalho desenvolvido, pode-se afirmar que os objetivos definidos foram todos cumpridos com sucesso.

Ao longo do desenvolvimento do trabalho, ficou muito mais saliente de que somente com o uso de técnicas de *machine learning* não é possível obter contribuições efetivas. Por exemplo, sem IoT, não existiriam dados suficientes para darem origem ao *Big Data*, o que tornaria difícil transformar a cidade numa cidade inteligente. Sem *Big Data* não seria possível construir modelos de *machine learning* mais robustos e sem *business intelligence* seria mais trabalhoso para o utilizador final mostrar os dados em gráficos e algumas estatísticas que pudessem ser úteis ao município.

O processamento dos dados teve uma grande influência, não só nos resultados dos vários modelos, mas também, na análise dos respetivos dados, permitindo retirar algumas estatísticas e informações úteis dos mesmos como, por exemplo, saber que a temperatura tem influência nos consumos de energia.

Um dos processos que, por vezes, é subvalorizado no processamento dos dados é a *feature engineering*. Porém, este foi muito importante no conjunto de dados utilizado, pois permitiu criar mais atributos, o que fez com que o modelo tivesse em conta essas informações adicionais durante o treino, permitindo reduzir substancialmente o erro e, assim, melhorar as previsões de dados futuros.

Com base nos resultados obtidos nos modelos testados, o algoritmo que obteve melhores resultados em praticamente todos os testes foi o LSTM, seguido do RF e do ARIMA. Este último ficou mais distanciado, provavelmente porque somente consegue fazer previsões em conjuntos de dados univariados. É de salientar, mais uma vez, que, embora um algoritmo oferecesse resultados menos positivos, não significa que é pior que os restantes algoritmos. Para o conjunto de dados utilizado, o LSTM ofereceu melhores resultados que, por exemplo, o ARIMA, mas o ARIMA poderá oferecer melhores resultados do que o LSTM ou o RF num outro conjunto de dados.

O ajuste do algoritmo escolhido, o LSTM, permitiu adaptar o modelo de forma a que, para o conjunto de dados a utilizar, conseguisse obter ainda melhores resultados, oferecendo melhores previsões em futuros dados. Com isto, pode-se concluir que o algoritmo LSTM é, de facto, uma boa opção para fazer previsões de dados de energia elétrica, provenientes de uma cidade.

Na visualização dos dados, pode-se concluir que o *Redash* foi, também, uma boa escolha como ferramenta de *business intelligence*, oferecendo ao utilizador gráficos agradáveis e intuitivos, o que faz com que este seja capaz de retirar informações acerca de tendências no aumento ou diminuição dos consumos de energia elétrica ao longo do tempo e outros indicadores.

Com a resolução deste projeto foram adquiridas diversas competências, não só na área do *machine learning*, mas também em outras como a mineração de dados, análise de dados, ciência de dados e *business intelligence*. Foi bastante trabalhoso, mas muito enriquecedor, pois a previsão de valores futuros com base no tempo é um dos problemas mais difíceis na área do *machine learning* e, devido a isso, foi possível aprender diversas técnicas utilizadas pela comunidade científica, que ainda estão numa fase muito inicial e que tendem a crescer cada vez mais.

4.1 Limitações e desafios

Ao longo deste projeto houve diversos desafios, começando pela escolha e estudo dos algoritmos de *machine learning* e implementação dos mesmos, onde foi necessário estudar, também, as bibliotecas dos respetivos algoritmos. As bibliotecas mais desafiantes foram as respeitantes ao LSTM, o Tensorflow 2.0 e Keras.

Um outro desafio foi relativamente ao treino do modelo escolhido, pois, por questões de *hardware*, não foi possível testar certos valores de alguns parâmetros, visto que as redes neuronais necessitam de muito poder computacional.

A única limitação na resolução deste projeto foi o facto de o conjunto de dados ser univariado, isto é, ter apenas um atributo, que era o valor do consumo de energia elétrica por hora. Se tivesse mais atributos como, por exemplo, a temperatura ou a qualidade do ar, provavelmente a previsão dos novos dados seria mais precisa.

4.2 Trabalho futuro

A previsão de dados futuros, utilizando conjuntos de dados com base no tempo, é do tipo de previsões mais desafiante e complexa que existe atualmente, por ser um problema ainda muito recente. Devido a isso, como trabalho futuro, pretende-se continuar a aplicar outros algoritmos de *machine learning* neste tipo de dados, com base no tempo, com o objetivo de encontrar e testar diferente técnicas que poderão ser úteis para comunidade científica.

Seria interessante aplicar o algoritmo LSTM a dados de outras cidades e também a outro tipo de dados relacionados com *smart cities* como, por exemplo, dados da qualidade do ar ou da água, com o objetivo de saber se este algoritmo conseguia prever tão bem como nos dados do consumo de energia.

Seria igualmente interessante e desafiante aplicar modelos de *machine learning* a outros cenários, com o objetivo de encontrar previsões que ajudem o município a ir ao encontro dos seus objetivos, melhorando, assim, a qualidade de vida dos seus cidadãos.

REFERÊNCIAS

Ejaz, W., Naeem, M., Shahid, A., Anpalagan, A. & Jo, M., 2017. Efficient energy management for the internet of things in smart cities. *IEEE Communications Magazine*, *55*(1), pp.84-91.

Pérez-Chacón, R., Luna-Romera, J., Troncoso, A., Martínez-Álvarez, F. & Riquelme, J., 2018. Big data analytics for discovering electricity consumption patterns in smart cities. *Energies*, *11*(3), p.683.

Chatterjee, S., Sk, S.C., Singh, M.K. & Sanyal, J., 2019. Machine Learning Based Prediction of Energy Consumption. *Machine Learning*, *7*(5).

Lima, C.M.N., 2015. *Previsão de consumo de energia elétrica em contexto de smart grids* (Doctoral dissertation).

Albertin, A.L. & de Moura Albertin, R.M., 2017. A internet das coisas irá muito além as coisas. *GV EXECUTIVO*, *16*(2), pp.12-17.

Mohammadi, M., Al-Fuqaha, A., Sorour, S. & Guizani, M., 2018. Deep learning for IoT big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, *20*(4), pp.2923-2960.

Dåderman, A., & Rosander, S. (2018). Evaluating frameworks for implementing machine learning in signal processing: A comparative study of CRISP-DM, semma and kdd.

Bolívar, M.P.R., 2015. Smart cities: Big cities, complex governance?. In *Transforming city governments for successful smart cities* (pp. 1-7). Springer, Cham.

Anthopoulos, L.G., 2015. Understanding the smart city domain: A literature review. In *Transforming city governments for successful smart cities* (pp. 9-21). Springer, Cham.

Su, K., Li, J. & Fu, H., 2011, September. Smart city and the applications. In *2011 international conference on electronics, communications and control (ICECC)* (pp. 1028-1031). IEEE.

Rana, N.P., Luthra, S., Mangla, S.K., Islam, R., Roderick, S. & Dwivedi, Y.K., 2019. Barriers to the development of smart cities in Indian context. *Information Systems Frontiers*, *21*(3), pp.503-525.

Arasteh, H., Hosseinnezhad, V., Loia, V., Tommasetti, A., Troisi, O., Shafie-Khah, M. & Siano, P., 2016, June. Iot-based smart cities: a survey. In 2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC) (pp. 1-6). IEEE.

Chourabi, H., Nam, T., Walker, S., Gil-Garcia, J.R., Mellouli, S., Nahon, K., Pardo, T.A. & Scholl, H.J., 2012, January. Understanding smart cities: An integrative framework. In 2012 45th Hawaii international conference on system sciences (pp. 2289-2297). IEEE.

Arroub, A., Zahi, B., Sabir, E. & Sadik, M., 2016, October. A literature review on Smart Cities: Paradigms, opportunities and open problems. In 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM) (pp. 180-186). IEEE.

Patel, K.K. & Patel, S.M., 2016. Internet of things-IOT: definition, characteristics, architecture, enabling technologies, application & future challenges. International journal of engineering science and computing, 6(5).

Madakam, S., Ramaswamy, R. & Tripathi, S., 2015. Internet of Things (IoT): A literature review. Journal of Computer and Communications, 3(05), p.164.

Dorsemaine, B., Gaulier, J.P., Wary, J.P., Kheir, N. & Urien, P., 2015, September. Internet of Things: a definition & taxonomy. In 2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies (pp. 72-77). IEEE.

Kim, T.H., Ramos, C. & Mohammed, S., 2017. Smart city and IoT.

Ahlgren, B., Hidell, M. & Ngai, E.C.H., 2016. Internet of things for smart cities: Interoperability and open data. IEEE Internet Computing, 20(6), pp.52-56.

De Mauro, A., Greco, M. & Grimaldi, M., 2016. A formal definition of Big Data based on its essential features. Library Review, 65(3), pp.122-135.

Youssra, R. & Sara, R., 2018. Big data and big data analytics: concepts, types and technologies. Int J Res Eng, 5(9), pp.524-528.

De Francisci Morales, G., Bifet, A., Khan, L., Gama, J. & Fan, W., 2016, August. Iot big data stream mining. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 2119-2120). ACM.

Zhou, L., Pan, S., Wang, J. & Vasilakos, A.V., 2017. Machine learning on big data: Opportunities and challenges. Neurocomputing, 237, pp.350-361.

Hasan, S., Shamsuddin, S.M. & Lopes, N., 2014. Machine learning big data framework and analytics for big data problems. Int. J. Advance Soft Compu. *Appl*, *6*(2).

Stackowiak, R., Rayman, J. & Greenwald, R., 2007. Oracle data warehousing & business intelligence SO. John Wiley & Sons.

Santos, B., Sério, F., Abrantes, S., Sá, F., Loureiro, J. & Wanzeler, C. (2019). Open Source Business Intelligence Tools: Metabase and Redash. 467-474.

Al-Ali, A.R., Zualkernan, I.A., Rashid, M., Gupta, R. & Alikarar, M., 2017. A smart home energy management system using IoT and big data analytics approach. IEEE Transactions on Consumer Electronics, 63(4), pp.426-434.

El Naqa, I. & Murphy, M.J., 2015. What is machine learning?. In *Machine Learning in Radiation Oncology* (pp. 3-11). Springer, Cham.

Surden, H., 2014. Machine learning and law. Wash. L. Rev., 89, p.87.

Grus, J., 2019. Data science from scratch: first principles with python. O'Reilly Media.

Mueller, J.P. & Massaron, L., 2019. Python for data science for dummies. For dummies.

Witten, I.H., Frank, E., Hall, M.A. and Pal, C.J., 2016. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann.

Barazandeh, I. & Gholamian, M.R., 2016. Knowledge discovery and data mining applications in the healthcare industry: a comprehensive study. In E-Health and Telemedicine: Concepts, Methodologies, Tools, and Applications (pp. 1097-1118). IGI Global.

Tan, Y., Zhang, C., Ma, Y. & Mao, Y., 2015, June. Knowledge discovery in databases based on deep neural networks. In 2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA) (pp. 1217-1222). IEEE.

Lison, P., 2015. An introduction to machine learning. Language Technology Group (LTG), 1, 35.

Portugal, I., Alencar, P. & Cowan, D., 2018. The use of machine learning algorithms in recommender systems: A systematic review. Expert Systems with *Applications*, *97*, pp.205-227.

Shalev-Shwartz, S. & Ben-David, S., 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Marsland, S., 2014. Machine learning: an algorithmic perspective. Chapman and Hall/CRC.

Noor, K. & Jan, S., 2017. Vehicle price prediction system using machine learning techniques. International Journal of Computer Applications, 167(9), pp.27-31.

Awad, W.A. & ELseuofi, S.M., 2011. Machine learning methods for spam e-mail classification. International Journal of Computer Science & Information Technology (IJCSIT), 3(1), pp.173-184.

Sathya, R. & Abraham, A., 2013. Comparison of supervised and unsupervised learning algorithms for pattern classification. International Journal of Advanced Research in Artificial Intelligence, 2(2), pp.34-38.

Peng, H., Wang, J., Pérez-Jiménez, M.J. & Riscos-Núñez, A., 2015. An unsupervised learning algorithm for membrane computing. Information Sciences, 304, pp.80-91.

Monamo, P., Marivate, V. & Twala, B., 2016, August. Unsupervised learning for robust Bitcoin fraud detection. In 2016 Information Security for South Africa (ISSA) (pp. 129-134). IEEE.

Sutton, R.S. & Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Mocanu, E., Nguyen, P.H., Kling, W.L. & Gibescu, M., 2016. Unsupervised energy prediction in a smart grid context using reinforcement cross-building transfer learning. Energy and Buildings, 116, pp.646-655.

Szepesvári, C., 2010. Algorithms for reinforcement learning. Synthesis lectures on artificial intelligence and machine learning, 4(1), pp.1-103.

Siami-Namini, S., Tavakoli, N. & Namin, A.S., 2018, December. A comparison of ARIMA and LSTM in forecasting time series. In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 1394-1401). IEEE.

Walczak, Steven & Cerpa, Narciso. (2003). Artificial Neural Networks. 10.1016/B0-12-227410-5/00837-1.

Yue, B., Fu, J., & Liang, J. (2018). Residual recurrent neural networks for learning sequential representations. Information, 9(3), 56.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654-669.

Hu, X., & Balasubramaniam, P. (Eds.). (2008). Recurrent neural networks (Vol. 400). InTech.

Kim, J., Kim, J., Thu, H. L. T., & Kim, H. (2016). Long short term memory recurrent neural network classifier for intrusion detection. In 2016 International Conference on Platform Technology and Service (PlatCon) (pp. 1-5). IEEE.

Liu, C., Hoi, S. C., Zhao, P., & Sun, J. (2016). Online arima algorithms for time series prediction.

Wang, W. C., Chau, K. W., Xu, D. M., & Chen, X. Y. (2015). Improving forecasting accuracy of annual runoff time series using ARIMA based on EEMD decomposition. Water Resources Management, 29(8), 2655-2675.

Yang, N., Li, T., & Song, J. (2007, October). Construction of decision trees based entropy and rough sets under tolerance relation. In International Conference on Intelligent Systems and Knowledge Engineering 2007. Atlantis Press.

Genuer, R., Poggi, J. M., & Tuleau-Malot, C. (2010). Variable selection using random forests. Pattern recognition letters, 31(14), 2225-2236.

Wang, L. & Sng, D., 2015. Deep learning algorithms with applications to video analytics for a smart city: A survey. arXiv preprint arXiv:1512.03131.

Robinson, C., Dilkina, B., Hubbs, J., Zhang, W., Guhathakurta, S., Brown, M.A. & Pendyala, R.M., 2017. Machine learning approaches for estimating commercial building energy consumption. Applied energy, 208, pp.889-904.

Melzi, F., Same, A., Zayani, M. & Oukhellou, L., 2017. A dedicated mixture model for clustering smart meter data: identification and analysis of electricity consumption behaviors. Energies, 10(10), p.1446.

Edwards, R.E., New, J. & Parker, L.E., 2012. Predicting future hourly residential electrical consumption: A machine learning case study. Energy and Buildings, 49, pp.591-603.

Ahmad, M.W., Mourshed, M. & Rezgui, Y., 2017. Trees vs Neurons: Comparison between random forest and ANN for high-resolution prediction of building energy consumption. Energy and Buildings, 147, pp.77-89.

Seyedzadeh, S., Rahimian, F.P., Glesk, I. & Roper, M., 2018. Machine learning for estimation of building energy consumption and performance: a review. Visualization in Engineering, 6(1), p.5.

Amasyali, K. & El-Gohary, N., 2016. Building lighting energy consumption prediction for supporting energy data analytics. Procedia Engineering, 145, pp.511-517.

Vinagre, E., Pinto, T., Ramos, S., Vale, Z. & Corchado, J.M., 2016, September. Electrical energy consumption forecast using support vector machines. In 2016 27th International Workshop on Database and Expert Systems Applications (DEXA) (pp. 171-175). IEEE.

Camara, A., Feixing, W., & Xiuqin, L. (2016). Energy consumption forecasting using seasonal ARIMA with artificial neural networks models. International Journal of Business and Management, 11(5), 231.

Who We Are. (1999). Retrieved June 12, 2020, from https://www.pjm.com/about-pjm/who-we-are.aspx

De Cian, E., Lanzi, E., & Roson, R. (2007). The impact of temperature change on energy demand: a dynamic panel analysis.

Zaidi, Nayyar. (2015). Feature Engineering in Machine Learning. 10.13140/RG.2.1.3564.3367.

Nahuis, S. L. C., Guyeux, C., Arcolezi, H. H., Couturier, R., Royer, G., & Lotufo, A. D. P. (2019, April). Long short-term memory for predicting firemen interventions. In 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT) (pp. 1132-1137). IEEE.

Wan, X. (2019, June). Influence of feature scaling on convergence of gradient iterative algorithm. In Journal of Physics: Conference Series (Vol. 1213, No. 3, p. 032021). IOP Publishing.

Roondiwala, M., Patel, H., & Varma, S. (2017). Predicting stock prices using LSTM. International Journal of Science and Research (IJSR), 6(4), 1754-1756.

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. Geoscientific model development, 7(3), 1247-1250.

Reitermanova, Z. (2010). Data splitting. In WDS (Vol. 10, pp. 31-36).

Brownlee, J. (2019, October 25). Difference Between a Batch and an Epoch in a Neural Network. Retrieved July 16, 2020, from https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

Hamdan, M. K. (2018). VHDL auto-generation tool for optimized hardware acceleration of convolutional neural networks on FPGA (VGT).

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. arXiv preprint arXiv:1710.05941.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

Brownlee, J. (2018). Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python. Machine Learning Mastery.