Algoritmo de monitorização e gestão de transportes

Márcio Steven Neves Duarte Lopes

Dissertação do Mestrado em Engenharia Industrial apresentado à Escola Superior de Tecnologia e de Gestão do Instituto Politécnico de Bragança.

Trabalho orientado por:

Prof. Ana I. Pereira

Prof. José Luís M. Lima

Prof. Sara Paiva

Bragança

2021

Algoritmo de monitorização e gestão de transportes

Márcio Steven Neves Duarte Lopes

Dissertação do Mestrado em Engenharia Industrial apresentado à Escola Superior de Tecnologia e de Gestão do Instituto Politécnico de Bragança.

Trabalho orientado por:

Prof. Ana I. Pereira

Prof. José Luís M. Lima

Prof. Sara Paiva

Bragança

2021

A Escola Superior de Tecnologia e de Gestão não se responsabiliza pelas opiniões expressas neste relatório.

Dedicatória

A minha filha, Melissa.

Agradecimentos

A conclusão do mestrado significa o fecho de mais um objetivo escolhido para quem optou ir atrás do conhecimento através do ambiente académico. Este caminho, de forma geral, apresentou inúmeros dificuldades mas também igual satisfações para a construção de um ser instruído e preparado para o futuro.

Agradeço à Professor Ana Isabel Pereira, a minha orientadora, e ao Professor José Lima, meu co-orientador, pela enorme disponibilidade, boa disposição, ajuda e dedicação imensurável e contributo no desenvolvimento desta dissertação.

Aos professores Sara Paiva e Sérgio Lopes, do Instituto Politécnico de Viana do Castelo, pela presença, orientação e contribuição desta dissertação, através de reuniões por vídeo-chamada.

À Escola Superior de Tecnologia e Gestão de Bragança por ser o instituto que ajudou-me a crescer a nível intelectual, através da disponibilização de ferramentas e recursos para a minha aprendizagem.

Aos meus amigos e companheiros e irmã, pela paciência, companheirismo, força e suporte que sempre prestaram.

À minha companheira, pelo apoio incondicional, paciência, carinho e compreensão.

E por último, um agradecimento sem limites aos melhores pais que algum dia poderia desejar, por serem compreensíveis, pelo apoio incondicional, incentivo, força e bastante carinho e paciência demonstrados.

Sem deixar de mencionar, a minha família pela ajuda e palavras de encorajamento.

Abstract

The construction of smart and sustainable cities encompasses different sectors, from mobility, energy or any service necessary for people's lives.

In this thesis the focus is on mobility within smart cities or future smart cities and the main objective is to develop an algorithm that can monitor and manage transport, especially public transport. Two machine learning algorithms were used in this project: K-means Clustering and Support Vector Machine. The objective of the first algorithm is to identify public transport routes behaviors and patterns. The second is used to predict the arrival time of transport on a given route.

The K-means algorithm can identify the flow of passengers, and separate clusters with more passengers.

The SVM algorithm is able to forecast the arrival time of the transport with a maximum error of 2 minutes.

Keywords: Transport Monitoring; Transport Management; Forecast Algorithm; Machine Learning; Support Vector Machine; K-means Clustering

Resumo

A construção de cidades inteligentes e sustentáveis engloba diferentes sectores, desde a mobilidade, a energia ou qualquer serviço necessário à vida das pessoas.

Nesta dissertação o foco é a mobilidade dentro das cidades inteligentes, ou futuras cidades inteligentes, e objetivo principal é desenvolver uma metodologia que possa fazer a monitorização e gestão de transportes, principalmente de transportes públicos.

Utilizou-se dois algoritmos de *machine learning* neste projeto: *K-means Clustering* e *Support Vector Machine*. O objetivo do primeiro algoritmo é identificar padrões de rotas de transportes públicos. E o segundo serve para fazer a previsão do tempo de chegada do transporte duma determinada rota.

O algoritmo de *K-means* consegue identificador o fluxo de passageiros e fazer a separação de *clusters* com mais passageiros.

O algoritmo de SVM consegue fazer a previsão da hora de chegada do transporte com um erro máximo de 2 minutos.

Palavras-chave: Monitorização de Transportes; Gestão de Transportes; Algoritmo de Previsão; Machine Learning; Support Vector Machine; K-means Clustering

Conteúdo

Lista de Tabelas

Lista de Figuras

Acrónimos

ANN *Artificial Neural Network.*

CORSIM *Corridor Simulation.*

DL *Deep Learning.*

EQM Erro Quadrático Médio.

GIS *Geographical Information System.*

GPS *Global Positioning System.*

HMADA *Hybrid Moving Average and Dynamic Adjustment.*

IA Inteligência Artificial.

KNN *K-Nearest Neighbor.*

MADA *Moving Average Dynamic Adjustment.*

MAM *Moving Average Model.*

ML *Machine Learning.*

RBF *Radial Basis Function.*

RFID Identificação por Radiofrequência.

SCD *Smart Card Data.*

SVD *Signal Voronoi Diagram.*

SVM *Support Vector Machines.*

TI Tecnologias da Informação.

UI Interface de Utilizador.

Capítulo 1

Introdução

A mobilidade tem-se demonstrado um dos grandes desafios que vários países enfrentam nos dias de hoje e Portugal é um deles. Em Portugal, residentes nas áreas metropolitanas preferem automóveis. Apenas 11,1% das pessoas do Porto e 15,8% em Lisboa usam os transportes públicos e/ou coletivos como principal meio de transporte. E isso acontece devido a vários fatores [22]:

- Conforto dos utilizadores,
- Rapidez do transporte coletivo,
- Rede de transportes públicos sem ligação direta ao destino,
- Ausência de alternativa,
- Serviços de transporte público sem a frequência ou fiabilidade (com valores acima dos 25%).

Neste trabalho será abordado o estudo de uma solução, ou soluções, para mitigar um desses fatores, nomeadamente a frequência e a fiabilidade dos transportes públicos coletivos.

A meta principal para o início do desenvolvimento deste trabalho é encontrar soluções que, de certo modo, atraiam e motivem a população portuguesa, e não só, a aderirem mais ao uso de transportes coletivos.

Uma das propostas é estudar o comportamento e a identificação de padrões nos meios de transportes coletivos, inicialmente autocarros, através da recolha de dados do mesmo. Os dados são compostos por diversas variáveis, nomeadamente:

- Horas de partida e chegada
- Local de partida e chegada
- Quantidade de passageiros quem entram e saem
- Distância entre as paragens
- Velocidade do veículo nas viagens

Com base nestes dados, pretende-se desenvolver um algoritmo ou modelo que faça a identificação de padrões críticos na trajetória do meio de transporte e que possa ser indicativo para possíveis alterações. Esse modelo deve poder determinar o segmento no qual começa e termina as situações críticas. Um padrão que pode ser considerado crítico é o número elevado de passageiros numa determinada hora, ao longo de várias semanas/meses.

Outra proposta será o desenvolvimento de um algoritmo ou modelo de previsão. Este irá prever o tempo de chegada de meios de transportes públicos coletivos através dos estudo dos dados do histórico de viagens das frotas. Essa previsão será útil para os utilizadores, pois essa informação deverá estar disponibilizado através de plataformas *on-line*.

Desta forma, pretende-se providenciar ferramentas/aplicações que informam o utilizador em tempo real da hora de chegada do seu transporte público. E mesmo que este esteja afastado da paragens desejada, a aplicação deverá informar o tempo que o utilizador demora até a paragem e o tempo de chegada do transporte. Isto irá evitar que o utilizador fique muito tempo à espera na paragem.

E com isso, este trabalho visa o desenvolvimento de estratégias de monitorização e gestão de transportes, principalmente transportes públicos. Embora haja aplicações

como a Google Maps e a Moovit para monitorizar os horários dos meios de transportes públicos, estes só funcionam na sua totalidade em grandes cidades, deixando assim as zonas pouco urbanizadas com pouca informação em relação aos mesmos. Normalmente, as informações sobre os horários dos transportes públicos, nas zonas pouco urbanizadas, estão disponíveis em páginas web das câmaras municipais ou em postes nas paragens.

A presente proposta é desenvolver uma aplicação que não só pode disponibilizar o horário dos transportes públicos em tempo real (ou previsão com dados do histórico de viagens), e simultaneamente recolhe informações de todos os veículos da frota para poder propor soluções de melhores rotas, baseado no comportamento dos utilizadores.

1.1 Objetivos

O objetivo principal deste projeto é desenvolver uma estratégia computacional capaz de calcular a previsão do tempo de chegada de transportes públicos. Este trabalho tem foco em zonas com baixa densidade populacional em que a informação digital sobre transportes públicos é mal difundida à população. Este trabalho tem os seguintes objetivos:

- Estudar o problema e estado de arte;
- Apresentar propostas;
- Implementar uma estratégia computacional;
- Analisar os resultados obtidos;
- Concluir e discutir resultados.

1.2 Estrutura do documento

Este documento encontra-se organizada em cinco capítulos, e está distribuído da seguinte forma:

- O **Capítulo 1 Introdução** tem a função de apresentar o tema em estudo, a que situação enquadra-se e indicar os objetivos a serem cumpridos durante o trabalho.

- O **Capítulo 2 Estado da arte** como o nome já sugere, descreve o estado da arte sobre a problemática em estudo. Apresenta métodos, modelos e aplicações que vários pesquisadores desenvolveram no âmbito do tema deste trabalho.

- O **Capítulo 3 Inteligência artificial** faz referência a alguns fundamentos teóricos considerados relevantes para a correta compreensão deste trabalho. Primeiro fez-se uma pequena introdução a inteligência artificial e as suas ramificações. Depois apresenta o ramo da inteligência artificial estudada, *Machine Learning* e os seus diferentes tipos de *Machine Learning* (ML), plataformas onde é possível desenvolver algoritmos de ML e os algoritmos utilizados neste trabalho.

- **Capítulo 4 Caso de estudo e resultados numéricos** descreve os dados utilizados neste trabalho, nomeadamente as variáveis que irão ser utilizados para a construção do modelo de gestão e previsão do tempo de chegada. É apresentado o horário e paragens do autocarro em estudo, e também dados simulados e reais do fluxo de passageiros de cada paragem. Em primeiro ponto são descritos os dados utilizados para o modelo de identificação de padrões e os resultados obtidos através do mesmo. E no último ponto são descritos os dados utilizados no modelo de previsão do tempo de chegada e os resultados provenientes deste.

- **Capítulo 5 Conclusão e trabalhos futuros** tem como finalidade apresentar uma analise crítica dos resultados obtidos no capítulo anterior, tanto para o modelo de identificação de padrões como para o modelo de previsão. E, para finalizar, são apresentados as conclusões gerais sobre este trabalho e sugestões para trabalhos futuros.

Capítulo 2

Estado da arte

O Capítulo 2 retrata alguns artigos e trabalhos relacionados com o tema deste trabalho, desde métodos para a previsão do tempo de chegada de transportes até ao modo como é disponibilizada essa informação ao público.

2.1 Métodos para previsão de tempo

O Conceito *Smart City* abrange mais do que a mobilidade, as plataformas digitais ou a sustentabilidade. Isto é, a mobilidade é um dos pilares para a caracterização de uma *Smart Cities*. O objectivo fundamental de uma *Smart City* é a incorporação de todas estas áreas a fim de melhorar a vida dos cidadãos no mundo [1].

Este trabalho aborda a ramificação da mobilidade pública. Os objectivos das empresas de transportes visam a redução de custos de operação através de soluções inteligentes, enquanto fornecem um serviço de qualidade aos seus clientes. Diferentes propostas de soluções surgem em diversas partes do mundo e cabe às empresas decidirem se a solução é a conveniente para o seu modelo de negócio.

Muitos investigadores expressam as suas soluções para o atual problema no sistema de transporte. E o foco é como modernizar o sistema que está extremamente vinculado a hábitos antigos. Algumas das soluções estão expressas nesta revisão de literatura.

Zhou et al. [36] desenvolveram um algoritmo de previsão do tempo de chegada dos

autocarros baseado num modulo chamadado de *Smart Card Data* (SCD). Recolheram dados de uma semana, incluindo o SCD e o atual tempo de chegada dos autocarros. Foram no total 1011 ensaios, onde 446 dos dados foram usados para o treino do algoritmo de previsão e 565 usados para o treino do mesmo.

Para reduzir a taxa de erro de previsão no SCD, incluíram a diferença de tempo entre o tempo de passagem do *Smart Card* (SC) e o tempo real da chegada do autocarro, a densidade de passagem dos SCs, e a capacidade de assentos.

O cálculo do tempo da chegada do autocarro é feito através da distribuição do intervalo entre a passagem dos SCs. A premissa do modelo consiste na estimativa da chegada do autocarro usando o tempo de passagem entre dois SCs e o tempo da última passagem do cartão em diferentes estações/paragens. A taxa de erro desse algoritmo é de 10%. Para uma previsão mais exata, o fator de carga e agendamentos podem ser considerados como parâmetros.

Yu et al. [34] implementaram um modelo dinâmico que utiliza o histórico de viagens para a previsão do tempo de chegada dos autocarros. É utilizado o *Global Positioning System* (GPS) para fazer a recolha de dados em intervalos regulares. O GPS instalado irá enviar dados a cada 15/20 segundos. Eles fizeram um modelo de previsão baseado na análise de clusters e ajustes polinomial.

São usados dados recolhidos do ano anterior para testar o sistema. Para evitar o efeito de padrões diferentes na precisão da previsão, recorre-se ao uso da distância euclidiana com análise de cluster para o ponto mais próximo. Usaram o método de agrupamento hierárquico e o método da distância euclidiana por causa da sua baixa taxa de erro.

O modelo de previsão recolhe os dados do histórico de viagens, processa-os e depois guarda-os numa base de dados. De seguida, o modelo de previsão verifica a correspondência de padrões e estima o tempo de chegada.

Tal como o trabalho de Yhu [34], Zhu et al. [37] implementaram um modelo que faz a previsão da hora de chegada do autocarro em tempo real utilizando GPS. Foi desenvolvido em dois modelos: um baseado em pontos e outro em rotas.

Para o cálculo da previsão com base em pontos, a velocidade instantânea, as informações do GPS e a distância são utilizadas para definir o tempo de chegada.

Para a previsão do tempo de chegada com base em rotas, é usado o tempo de viagem do autocarro, o atraso nas paragens e interseções. O tempo de espera é calculado separadamente. Um dos métodos matemáticos utilizados é o erro médio absoluto percentual.

É feita a comparação dos dois modelos acima citados, e é concluído que o método de previsão baseado em pontos oferece uma taxa de erro mínima e é mais confiável em comparação com o método baseado em rotas.

Li et al. [19], propuseram um modelo linear para a previsão. Dividiram o tempo total em duas partes: linear e residual. Para a previsão, incluíram o estado do trânsito, tempo de espera, interseção e a hora de partida dos veículos. O processo de calibração dos parâmetros é usado para a previsão.

Primeiramente, são classificados e determinados os padrões de trânsito através de históricos. Na segunda etapa, é estimado o tempo de permanência, viagem e de serviço. E por último, é aplicado o método dos mínimos quadrados para otimizar os parâmetros.

Eles usaram dados históricos e classificaram oito clusters diferentes, e uma abordagem estatística foi desenvolvida para a identificação de diferentes parâmetros a serem incluídos no cluster. Também desenvolveram um aplicativo da web para identificar a praticidade e eficiência.

Os investigadores Chien et al. [6], sugeriram um modelo que funciona com utilização da Rede Neural Artificial ou *Artificial Neural Network* (ANN). Usaram dois tipos de ANN, *linked-based* ANN e *stop-based* ANN. No desenvolvimento deste modelo, usaram uma rede multi-camada totalmente interligada de *feed-forward* com um algoritmo neuronal de retro propagação – *Back Propagation*. Usaram uma ANN melhorada (*Enhanced* ANN) para aperfeiçoar o algoritmo.

Embora este algoritmo preveja o tempo de chegada usando os dados de treino, também considera os dados de tempo real. Para melhorar a precisão deste, é feita a integração dos dois modelos referidos acima, com um algoritmo adaptativo.

Para implementar esse algoritmo, foi utilizado o modelo de simulação *Corridor Simulation* (CORSIM). Para a recolha de dados, foi utilizado um detetor de Identificação por Radiofrequência (RFID) para detetar o veículo específico e a sua identificação única.

A principal vantagem desta abordagem é que é possível prever a paragem singular ou múltipla de veículos numa determinada paragem, devido ao uso dos dois tipos ANN mencionados acima.

Ainda dentro da categoria de redes neuronais, Zorkany et al. [38] implementaram uma rede neuronal híbrida que faz a previsão do tempo de chegada do autocarro. Neste sistema que propuseram, foi feita o uso de dois modelos: um baseado em ANN e outro com o filtro de Kalman.

Este algoritmo recebe os dados de um módulo, processa esses dados e guarda-os numa base de dados. Depois disso, o algoritmo verifica se há correspondência nas regras já estabelecidas, e se alguma regra é compatível com o ANN, e prevê o tempo de chegada através do uso do filtro de Kalman.

Este modelo neuronal é composto por quatro camadas: camada de entrada de dados, duas camadas escondidas e um a camada de saída de dados.

A camada de entrada de dados é composta por sete nós. A primeira camada escondida apresenta dez nós e o segunda três nós. A camada de saída apresenta somente um nó.

Para fazer a previsão é usado uma versão modificada do filtro de Kalman. Dados recolhidos dos últimos três dias similares das últimas três semanas semelhantes são usadas para fazer a previsão.

Yu et al. [33], fazem uma outra abordagem para a previsão do tempo de chegada utilizando um algoritmo de ML. O modelo faz a previsão do tempo de chegada de autocarros numa mesma paragem, em que a rota de viagem seja diferente. Este modelo dinâmico consiste em duas partes: o modelo *Support Vector Machines* (SVM) para estimar o tempo de viagem base e filtro de Kalman para fazer ajustes com informações mais recentes. As previsões são medidas usando o erro absoluto médio, o erro médio percentual absoluto e a raiz do erro quadrático médio.

Para o cálculo do tempo real de chegada e tempo de execução, o método manual é

utilizado. É comparado o desempenho de quatro métodos matemáticos diferentes, dos quais a regressão linear que apresenta o pior desempenho. O algoritmo de *K-Nearest Neighbor* (KNN) apresenta um melhor desempenho que a regressão linear. KNN e ANN são semelhantes, mas o ANN é ligeiramente melhor. O SVM oferece o melhor resultado entre todos.

O coeficiente de correlação dos quatros métodos são: regressão linear com 0.84, KNN com 0.85, ANN com 0.87 e SVM com 0.90.

Os investigadores, Bai et al. [3], implementaram um modelo híbrido que também faz uso de um algoritmo de ML, focando-se apenas em SVM. Este modelo faz a previsão através do histórico de viagens obtidos no campo de estudo. Primeiramente, é usado um modelo SVM previamente treinado para prever o tempo de viagem do histórico de viagens obtidos no campo. Depois, é usado o algoritmo de filtragem de Kalman para melhorar a exatidão do tempo de viagem previsto pelo modelo de SVM.

O modelo de previsão compila, primeiramente, os dados recolhidos anteriormente e entrega-o ao SVM. Este faz o cálculo do tempo de chegada. Posteriormente, esses dados são enviados para o algoritmo de filtragem de Kalman, onde os dados do SVM e os do tempo real são usados para prever os dados de chegada dos autocarros de uma forma precisa.

Bai et al. [3] implementaram um modelo híbrido que também faz uso de um algoritmo de ML já mencionado anteriormente. Este modelo faz a previsão através do histórico de viagens obtidos no campo de estudo. Primeiramente, é usado um modelo SVM previamente treinado para prever o tempo de viagem do histórico de viagens obtidos no campo. Depois, é usado o algoritmo de filtragem de Kalman para melhorar a exatidão do tempo de viagem previsto pelo modelo de SVM.

O modelo de previsão compila, primeiramente, os dados recolhidos anteriormente e entrega-o ao SVM. Este faz o cálculo do tempo de chegada. Posteriormente, esses dados são enviados para o algoritmo de filtragem de Kalman, onde os dados do SVM e os do tempo real são usados para prever os dados de chegada dos autocarros de uma forma precisa.

No trabalho de Padmanaban et al. [26] há uma breve referencia ao filtro de Kalman embora não mencionem o algoritmo utilizado. A previsão da chegada do autocarro é em tempo real. Este sistema usa os dados históricos para fazer a previsão. Para fazer a recolha de dados, utilizaram a rota Nº 21-L Chennai. Foram recolhidos dados de uma semana de dois autocarros. Os dados foram: a latitude, longitude e a respetiva hora através de um modulo GPS. Depois disso, é usada a fórmula de Haversine para calcular a distância.

Eles desenvolveram um procedimento automatizado para calcular o tempo de demora nas paragens. Posteriormente, utilizaram um algoritmo, desenvolvido por eles, para fazer a previsão do tempo de chegada de um terceiro autocarro através de dados recolhidos anteriormente.

Para a previsão, foi feita uma divisão nos tempos: tempo de viagem e tempo de atrasos. E existem também diversas categorias dentro dos atrasos, tal como, atrasos nos semáforos, atrasos nas paragens, etc. E para melhorar precisão da previsão é utilizado o filtro de Kalman.

Os dados históricos e os do tempo real são combinados para melhorar o desempenho do algoritmo de previsão.

Em outro caso, levando em conta que a maior parte da população utiliza smartphones, Zhou et al. [35] implementaram um modelo de previsão do tempo de chegada do autocarro que faz o uso de smartphones. Este sistema é divido em três componentes.

O utilizador insere o destino e a rota de interesse. Por outro lado, o utilizador dentro do autocarro envia dados para um servidor de base de dados. Neste servidor é feita a distinção de quem está dentro do autocarro e de quem não está. Depois, esses dados são enviados para outro servidor, chamado de *back-end* e este é responsável para fazer todos os cálculos.

Os dados recebidos de cada utilizador são processado e é enviado uma resposta adequada a cada um deles, consoante a pesquisa feita por cada um deles. E para tal, o utilizador tem de ter a aplicação instalada no smartphone.

Este sistema é totalmente dependente dos utilizadores, porque se estes não existirem

não há como efectuar a previsão.

O modelo de Gong et al. [15] é um modelo híbrido, com dados do histórico e GPS com *Hybrid Moving Average and Dynamic Adjustment* (HMADA), para prever a hora de chegada do autocarro.

É selecionado um conjunto de dados do histórico de viagens e dados em tempo real. Os dados de tempo real não são utilizados diretamente na previsão. É feito um pré-processamento desses dados para que estes possam ser utilizados no modelo de previsão. E para isso, é utilizado o *Moving Average Model* (MAM), *Moving Average Dynamic Adjustment* (MADA) e um modelo híbrido baseado no HMADA.

É sugerido que é preferível o uso do HMADA em relação a MAM e MADA.

É sugerido também que os dados fornecidos pelo GPS devem ser pré-processados antes de serem utilizados no modelo de previsão.

E para fechar os estudos que envolvem a utilização de GPS, Maiti et al. [21] que desenvolveram um sistema baseado em dados do histórico de viagem, relativamente aos tempos de chegada. Afirmam que esse sistema é mais rápido que o ANN e SVM.

Nesse modelo proposto, eles recolhem dados previamente adquiridos, dados de análise e pré-processamento. De seguida, é removido o ruído, valores ausentes e é feita a correção das estações do autocarro. Depois, com esses dados já prontos, é feita a previsão da hora de chegada do autocarro.

Esse procedimento de previsão da hora de chegada é dado através da velocidade de deslocação. E para fazer a recolha de dados, é usado o dispositivo de Connect-Port X5 R. Este dispositivo recolhe dados da velocidade, localização, tempo de espera, tempo de passagem de cartão, o número de passageiros que entram e saem. Smartphones com GPS podem substituir esse dispositivo.

Existiram também estudos para desenvolver um algoritmo de previsão utilizando um software designado de CORSIM. Chien et al. [7] implementaram dois modelos de previsão: modelo de histórico de viagens e modelo de dados em tempo real. Fizeram a recolha de dados do New York City Thruway. Recolheram dados do histórico nas paragens dos terminais rodoviários.

Para melhorar as previsões, são eliminados os resultados de um autocarro específico caso os dados desse forem muito divergentes do habitual num período particular. São processados e filtrados poucos dados de acordo com o requisito do algoritmo de previsão. São utilizados dois métodos para testes, um baseado nos caminhos feitos pelos transportes e outro em ligações entre os transportes. Para a implementação do sistema é utilizada uma versão melhorada do CORSIM.

Não são utilizados os dados do tempo real diretamente, este tem de ser previamente tratada antes de ser implementado no sistema. Isto porque pode haver casos em o padrão seja totalmente divergente do habitual e esse dado é trabalhado antes de ser utilizado ou por vezes eliminado.

Com as diferentes experiências feitas, Chien et al. [7] chegaram à conclusão que o método de previsão baseado em ligações entre os transportes é mais fiável e preciso em comparação ao método baseado em caminhos feitos pelos transportes. O método baseado em caminhos só é fiável quando a informação do tráfico é uniforme e caminho feito pelo transporte é curto.

Diferente das pesquisas anteriores, Weng et al. [31] implementaram um modelo de estimativa de velocidade dos autocarros. Este modelo é dividido em três partes: processamento e pré-processamento de dados, correspondência de dados e estimativa de velocidade de viagem, e correção da velocidade de viagem.

Na primeira fase, eles fazem a correspondência entre os dados do GPS e os do *Geographical Information System* (GIS) para estimar a localização do veículo e o trajeto de viagem do mesmo, gerando assim um arco de posicionamento. Na segunda fase, o movimento do arco é diagnosticado e é adicionado partes em falta. Na terceira fase, é realizado o cálculo estimado do tempo de chegada do veículo. Na quarta fase, é determinada a direção do veículo. Na quinta fase, é feito o cálculo da velocidade de deslocamento entre dois pontos. E, para finalizar, os dados de campo e os dados do modelo são comparados. A taxa de precisão deste modelo é de 88,4%.

2.2 Aplicativos de transportes públicos

É impossível falar de aplicativos que disponibilizam os horários de transportes públicos sem mencionar o Google *Transit*, da Google *Maps*. O Google *Transit* permite que os utilizadores visualizem as opções de transporte público no Google *Maps*. Este serve para ajudar os utilizadores em todo o planeta a planear as melhores rotas para chegar aonde estão viajando. Ao combinar os dados de programação e rota com o poder do Google *Maps*, suas informações de tráfego se tornam facilmente obtidas por milhões de utilizadores do Google em dezenas de idiomas em computadores e dispositivos móveis.

Quando as agências de transporte público participantes compartilham seus dados estáticos de transporte público (por exemplo, rotas, paragens e horários) com o Google *Transit*, essas informações são integradas ao Google *Maps* e podem ser consultadas pelos utilizadores do Google.

Depois que as agências compartilham as rotas, estatísticas e informações do horário, o Google *Transit* permite que o utilizador adicione atualizações de trânsito em tempo real por meio do *Realtime Transit*. As atualizações em tempo real melhoram a experiência dos passageiros com os últimos horários de partida e chegada, alertas de serviço e posições de veículos. Para que as informações de estatísticas de trânsito estejam disponíveis por meio do Google *Maps*, o utilizador está qualificado para enviar dados para o *Realtime Transit* [16].

Outra aplicação que é muito utilizada é a Moovit [24]. O Moovit é líder em soluções de Mobilidade como Serviço (Maas - Mobility as a Service) e é das mais populares atualmente. O Moovit foi lançado em 2012 como aplicativo gratuito de Android, iOS e na web para guiar as pessoas de maneira simples e eficiente, combinando diversos meios de transporte. Ao longo dos anos conquistou mais de 1 bilhão de utilizadores em 3400 cidades de 112 países, e está disponível em 45 idiomas.

A aplicação realiza análises de mobilidade urbana, coordenando tendências, oferece soluções para pagamento dos bilhetes de viagem através dos smartphones, dando ênfase a melhoria da saúde publica, isto é, evitar o contacto dos passageiros com o motoristas

ou o operador que vende bilhetes fisicamente, e ainda disponibilizando todos os serviços de mobilidade da região em tempo real.

Uma aplicação pouco difundida foi desenvolvida por Liu et al. [20], chamada WiLocator. Propuseram uma ferramenta chamada de *Signal Voronoi Diagram* (SVD) que usa os pontos de acesso Wi-Fi (APs). O WiLocator é composto por três partes: smartphones com acesso a Wi-Fi, um servidor *back-end* e uma Interface de Utilizador (UI).

Os smartphones enviam os dados para o servidor *back-end* através do Wi-Fi. O servidor é utilizado para o armazenamento, processamento e previsão de dados. A UI estabelece a comunicação entre o WiLocator e os seus utilizadores do serviço.

O WiLocator é dividido em três secções: a posição do autocarro baseado no SVD, previsão do tempo de chegada e a geração do mapa de trânsito; na segunda parte, é feita a análise dos dados do GPS e é calculada a proposição da posição; depois disso, é estimada a velocidade do veículo; e por último, se o limite de velocidade não for atingido, a velocidade de deslocamento é corrigida [20].

Também existe um aplicativo web, desenvolvido por Ferris et al. [12], que é um sistema de monitorização de transporte público em tempo real baseada no transporte público em Seattle.

Criaram um aplicativo e um web site onde está disponibilizada toda a informação dos horários para os utilizadores, tal como informações sobre a empresa de transportes. E neste serviço está presente um mapa que mostra a localização exata dos transportes.

O sistema que foi implementado oferece diversos módulos de serviços e integrados usando uma *framework* de inversão de *Spring*, um banco de dados de comparações com um *framework* de hibernação.

Este sistema transmite dados em tempo real da localização do autocarro, e faz uma estimativa do tempo de chegada numa determinada paragem.

2.3 Programas, plataformas e livrarias utilizadas

Inicialmente, depois de se ter organizado o conjunto de amostras, o primeiro *software* a ser utilizado para realizar os estudos com o *K-means* e o *SVM* foi o MatLab.

Com um conjunto pequeno de amostras, é feito um estudo preliminar na aplicação dentro do MatLab, o *Classification Learner*. *Classification Learner* treina modelos para classificar dados. Esta aplicação explora os dados, selecionando recursos, especificando esquemas de validação, treinando modelos e avaliar os resultados. O treinamento automatizado identifica o melhor tipo de modelo de classificação, incluindo árvores de decisão, análise discriminante, Support Vector Machine (SVM), regressão logística, *K-means*, naive Bayes, e classificação de rede neural [23].

Posteriormente, foi utilizada a plataforma do Google *Colaboratory*, que usa a linguagem de programação *Python*, para codificar o método de K-means e SVM explorar os dados simulados e obtidos de transportes públicos para o estudo da previsão do tempo de chegada e gestão da frota dos mesmos.

Capítulo 3

Inteligência artificial

A Inteligência Artificial é a ciência que estuda e procura compreender o fenómeno da inteligência. O campo da Inteligência Artificial (IA), vai além de apenas tenta entender, mas também se esforça para construir entidades inteligentes. A IA é uma das ciências mais recentes. O surgimento dos primeiros usos de *Machine Learning* (ML) começou logo após a Segunda Guerra Mundial, e foi nomeado assim em 1956. A IA atualmente está presente numa ampla variedade de aplicações, de áreas de propósito geral, como ensino e percepção, a áreas mais específicas, como xadrez, prova de teoremas matemáticos, escrita de poesia, diagnóstico de doenças, entre outros [29].

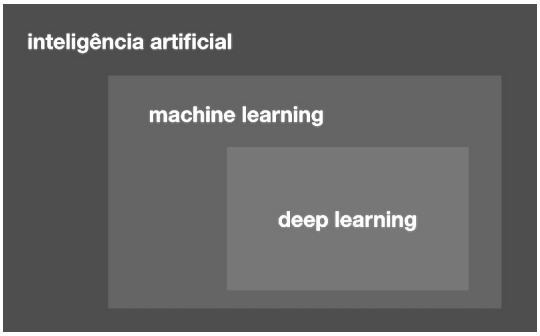

Figura 3.1: Ligação entre IA e ciências da computação [29]

Machine Learning (ML) e *Deep Learning* (DL) são sub-categorias do IA, como é representado na Figura 3.1.

3.1 Pré-processamento e análise de dados

Todos as técnicas de ML usam dados. Esses conjuntos de dados podem ser recolhidos e editados por humanos ou reunidos de forma autónoma por outras ferramentas de softwares. E esses mesmos conjuntos de dados são usados para treinar um modelo [27].

Por exemplo, os sistemas de controle recolherem dados de sensores à medida que os sistemas operam e usam esses dados para identificar parâmetros ou treinar o sistema. Os conjuntos de dados podem ser, por vezes, muito grandes e isso implica investir em infraestrutura de armazenamento de dados e dos bancos de dados. E dado a esse fenómeno, é evidente o crescimento aplicações de ML [27].

E é preciso enfatizar que um modelo de ML é tão bom quanto os dados usados para criá-lo, e a seleção de dados de treino é praticamente uma área em si [27].

Esses dados normalmente são divididos em três conjuntos:

- Um conjunto de treino, que vai ser utilizado para criar o modelo de ML. Normalmente é 80% dos dados conhecidos;
- Um conjunto de validação que contém alguns dados do subconjunto de treino (normalmente escolhidos aleatoriamente) com propósito de validar o modelo com dados conhecidos. Normalmente contém 20% dos dados do conjunto de treino;
- Um conjunto de teste que contém dados novos, isto é, que não estão incluídos no conjunto de treino. Usa-se os restantes 20% dos dados que não foram utilizados para o treino para avaliar o comportamento do modelo com dados novos. [10].

Os dados de treino precisam de ser preparados (aleatórios e não-duplicados) e verificar se há desequilíbrios que podem gerar um impacto no treinamento [10].

Figura 3.2: Preparação do conjunto de dados em ambiente de *Machine Learning* [2]

3.2 *Machine Learning*

Nos últimos anos, o ML transformou-se num dos ramos mais importantes e prolíficos da Tecnologias da Informação (TI) e da IA. Centenas de artigos publicados todos os meses, contribuem para um dos mais intensos processos de democratização da história da TI. Mas porquê o ML é tão importante e valioso? Desde sempre, os seres humanos constroem ferramentas e máquinas para simplificar o seu trabalho e reduzir o esforço geral necessário para concluir diferentes tarefas. E o ML é uma delas [4].

Portanto, o principal objetivo do ML é estudar, projetar e melhorar modelos matemáticos que podem ser treinados (uma vez ou continuamente) com dados relacionados com o contexto (fornecidos por um ambiente genérico), para inferir o futuro e tomar decisões sem conhecimento completo de todos os elementos que influenciam (fatores externos) [4].

Exemplo de aplicações de ML são os assistentes digitais, tal como, o Google Assistente, a Siri da Apple, a Kortana do Windows, etc.; as recomendações de páginas web com base em compras realizadas online; o padrão de aspiração de robôs de aspiração; detetores de *spams* em e-mails; analisadores de imagem médica para a deteção de tumores; e os primeiros carros autónomos[10].

A cada dia que passa, a quantidade de dados para serem analisados aumenta.

O ML segue a lógica representada na figura 3.3, onde através de um algoritmo usa dados de treino (*Training Data*), para criar o modelo. Que depois, ao fornecer dados de teste (*Input Data*) gera-se os dados de saída (*Output*) [18].

Existem diferentes tipos de ML que servem para resolver diferentes tipos de problemas, isto é, cada conjunto de dados é diferente logo cada um requer um tipo de ML.

- ***Supervised Learning*** - é muito semelhante ao processo de aprendizagem do ser humano. A principal tarefa é fornecer uma medida precisa do seu erro (diretamente comparável com os valores de saída). Isto é, é realizado sob treino, em que o algoritmo é constituído por entradas e pares de saída corretos [18]. A partir dessas informações, a aplicação pode corrigir seus parâmetros de forma a reduzir a magnitude de uma função de perda global. Após cada iteração, se o algoritmo for flexível

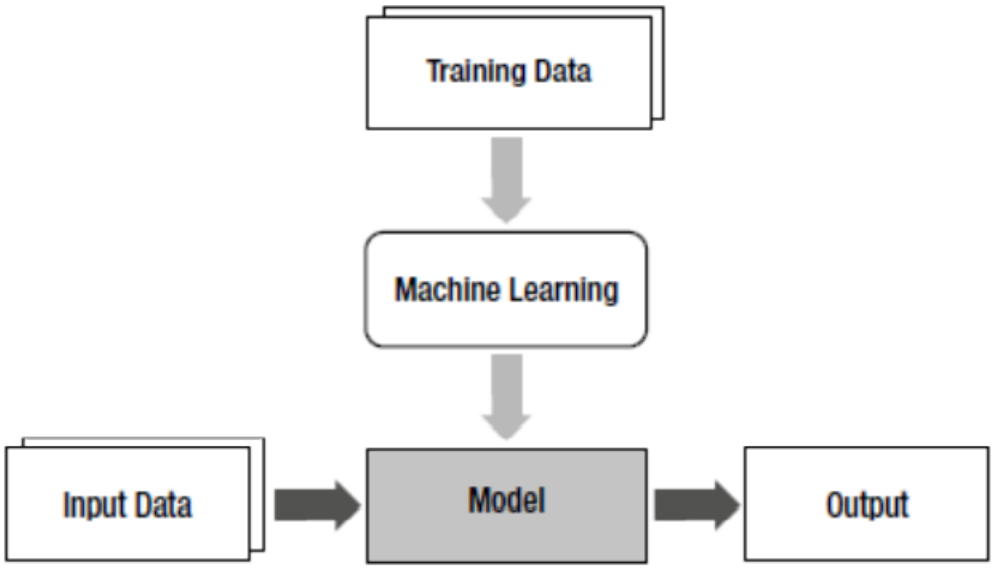

Figura 3.3: Representação do processo de *Machine Learning* [18]

o suficiente e os elementos de dados forem coerentes, a precisão geral aumenta e a diferença entre os valores previstos e esperados torna-se próxima de zero. É claro que, em um cenário supervisionado, o objetivo é treinar um sistema para desenvolver uma capacidade de generalização e evitar um problema comum denominado *overfitting*, que causa um *super-learning* devido a uma capacidade excessiva. Um dos principais efeitos desse problema é a capacidade de prever correctamente apenas as amostras utilizadas para treino, enquanto o erro para as restantes é sempre muito elevado [4]. *Supervised Learning* divide-se em diferentes técnicas com diferentes algoritmos:

  - **Regressão:** *Linear regression*
  - **Classificação:** *Support Vector Machine*, *Naive Bayes*, *Logistic Regression*

- ***Unsupervised Learning*** - usa algoritmos de ML para analisar e agrupar conjuntos de dados não rotulados. Esses algoritmos descobrem padrões ocultos e ou faz agrupamentos de dados sem a necessidade de intervenção humana. A sua capacidade de descobrir semelhanças e diferenças nas informações torna a solução ideal

para análise exploratória de dados, estratégias de vendas, segmentação de clientes e reconhecimento de imagem [11]. Este é constituído por diferentes algoritmos:

  - ***Clustering*:** *K-means*, *Hierarchical model* e *Mixture model*.

- ***Semi-Supervised Learning*** - Neste ambiente, os dados podem-se apresentar rotulados e não rotulados. Esta abordagem é, portanto, chamada de *semi-supervised learning* e pode ser adotada quando é necessário categorizar uma grande quantidade de dados com alguns exemplos rotulados [4]. Este tipo é constituído por diferentes algoritmos, tal como:

  - **Classificação e Controle**: método de Monte Carlo, método da Diferença Temporal e Direct Policy Search.

- ***Reinforcement Learning*** - é um tipo de ML que aprender o que fazer de modo a maximizar um sinal numérico de recompensa. Por exemplo, não é dito ao aprendiz que ações devem ser executadas, mas, em vez disso, este deve descobrir que ações geram mais recompensas ao experimentá-las. As ações podem afetar não apenas a recompensa imediata, mas também a próxima situação e, apesar disso, todas as recompensas subsequentes. Essas duas características (busca por tentativa e erro e recompensa no final) são as duas características distintivas mais importantes da *Reinforcement Learning* [30].

  O *reinforcement learning* é particularmente eficiente quando o ambiente não é determinístico. Em ambientes dinâmico é impossível ter uma medida de erro precisa [4]. Este tipo de ML engloba alguns algoritmos, tal como:

  - **Classificação e *Clustering***: *Low density separation*, *Graph based method* e *Heuristic Approach*.

3.3 Métodos *Elbow* e *K-means*

O método de *Elbow*, ou cotovelo, é usado para determinar o número ideal de clusters.

k refere-se ao número de centróides necessários em um conjunto de dados. Um centróide é o local que representa o centro do cluster. O algoritmo K-means identifica o número k de centróides e, em seguida, atribui cada ponto de dados para o cluster mais próximo, enquanto mantém os centróides o menor possível.

Se k aumentar, a distorção média diminuirá, cada cluster terá menos elementos constituintes e as elementos estarão mais próximas de seus respectivos centróides. No entanto, as melhorias na distorção média diminuirão à medida que k aumenta.

O valor de k no qual distorção diminui mais é chamado de *Elbow* (ver na Figura 3.4 - *Elbow point*), no qual deve-se parar de dividir os dados em grupos adicionais [8].

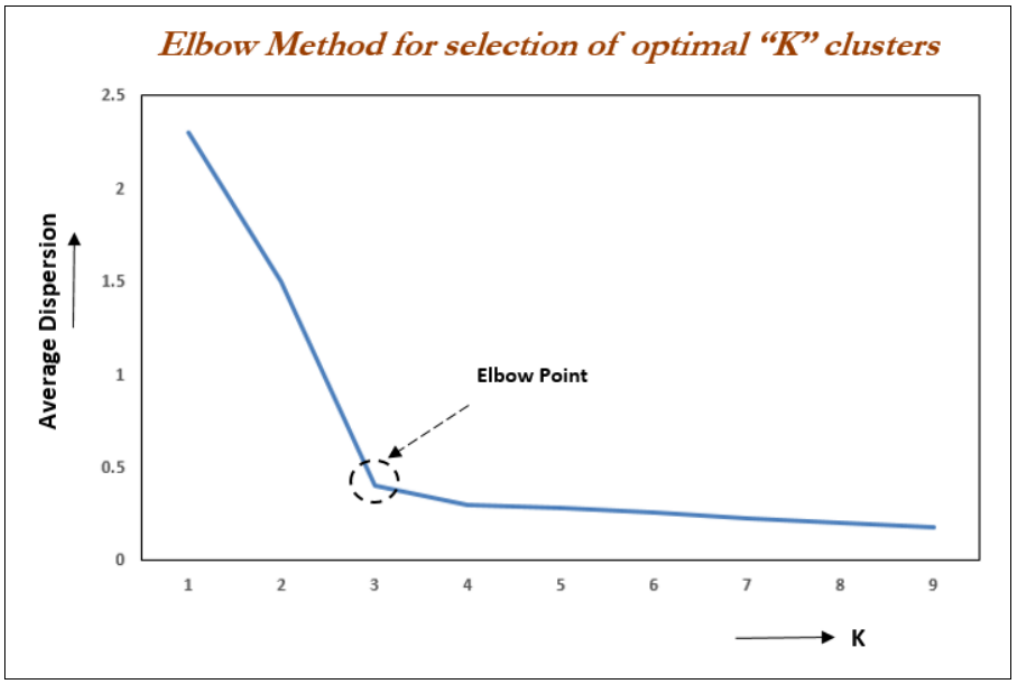

Figura 3.4: Seleção de k ideal com o método de *Elbow* [8]

K-means é um algoritmo de *unsupervised* ML. Normalmente, algoritmos *unsupervised*, ou sem supervisão, fazem inferências a partir de conjuntos de dados usando apenas vetores de entrada, sem referir-se a resultados conhecidos ou rotulados [14].

O objetivo do *k-means* é bastante simples, é agrupar pontos de dados similares e descobrir padrões subjacentes. Este método procura um número fixo k de clusters num conjunto de dados [5].

O k pode ser definido manualmente, isto é, observando os dados e escolhendo o número de clusters. Mas nem sempre isso é viável devido a quantidade de dados. Daí o uso do método de *Elbow*.

3.4 Método *Support Vector Machines*

Support Vector Machines ou SVM, são modelos de *supervised learning* com algoritmos que analisam dados usados para classificação e análise de regressão. O objetivo do SVM é produzir um modelo, com base nos dados de treino, que preveja os valores de destino. Em SVM, o mapeamento não linear de dados de entrada num espaço de recursos de grande dimensão é feito com funções do *Kernel* [27].

O SVM tem como objetivo encontrar um hiperplano num espaço N-dimensional (N - é o numero de recursos, como é mostrado na Figura 3.5) que classifica distintamente os pontos dos dados [13].

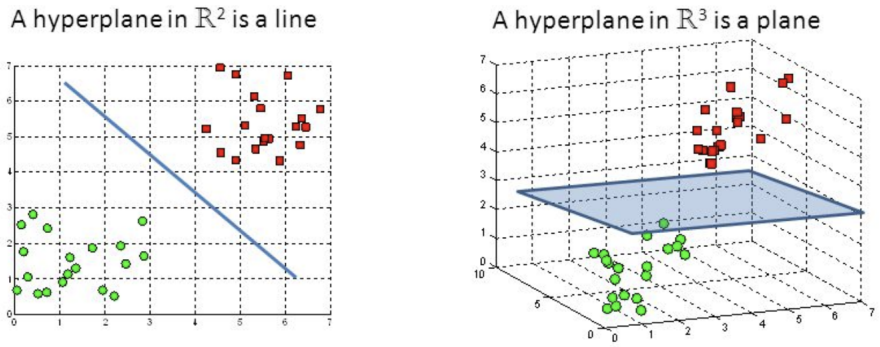

Figura 3.5: Hiperplanos em 2D ou 3D [13]

Para separar um determinado conjunto de pontos, em duas partes por exemplo, há inúmeros hiperplanos possíveis que podem ser escolhidos. Hiperplanos são designados de fronteiras de decisão que guiam a classificação de pontos. Novos pontos que estejam situados num dos lados do hiperplano são classificados consoante esses dados [13].

Quando os pontos não são linearmente separáveis, podendo induzir uma má classificação de SVM linear, é possível aplicar uma técnica denominada de *Kernel Trick*. Essa

técnica é capaz de classificar pontos que não são linearmente separáveis num espaço dimensional superior, transformando assim esses em dados linearmente separáveis, como é demonstrado na Figura 3.6, podendo assim aplicar o SVM normalmente [32].

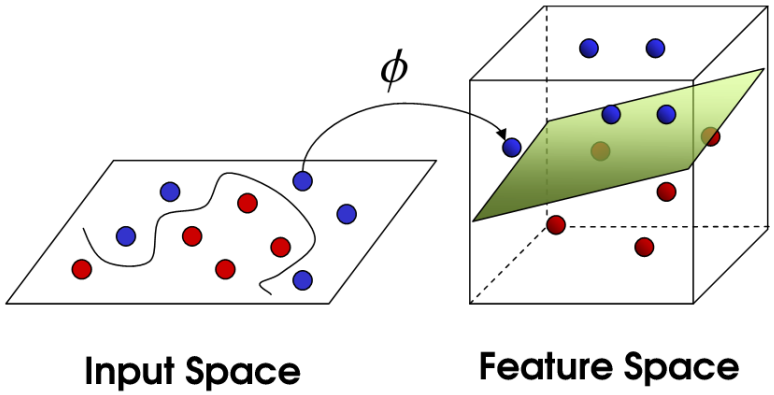

Figura 3.6: *Kernel Trick* [17]

Existem diferentes tipos de *Kernel*: linear, polinomial, sigmóide e *Radial Basis Function* (RBF).

3.5 *Machine Learning*: MatLab

A MathWorks, da MatLab, vende vários pacotes para o estudo de ML. Este produto fornece algoritmos para análise de dados junto com ferramentas gráficas para visualizar os dados. As ferramentas de visualização são uma parte crítica de qualquer sistema de ML [27].

Estes podem ser usados para aquisição de dados, por exemplo, para reconhecimento de imagem ou como parte de sistemas para controle autónomo de veículos, ou para diagnóstico e depuração durante o desenvolvimento [27].

Todos esses pacotes podem ser integrados entre si e com outras funções do MatLab para produzir sistemas de ML mais robustos. As caixas de ferramentas mais utilizadas são:

- Toolbox de Estatísticas e ML

- Toolbox de rede neural
- Toolbox do sistema de visão computacional
- Toolbox de identificação do sistema
- MatLab para *Deep Learning*
- Toolbox de Fuzzy Logic

3.5.1 Toolbox de Estatísticas e de *Machine Learning*

Neste trabalho somente foi explorado a toolbox de Estatísticas e de ML.

Essa toolbox fornece métodos de análise de dados para identificar tendências e padrões em grandes quantidades de dados. Esses métodos não requerem um modelo para analisar os dados. As funções da toolbox podem ser amplamente divididas em ferramentas de classificação, ferramentas de regressão e ferramentas de agrupamento [27].

- Os métodos de classificação são usados para colocar os dados em diferentes categorias. Por exemplo, dados, na forma de uma imagem, podem ser usados para classificar uma imagem de um órgão como tendo um tumor. A classificação é usada também para reconhecimento de escrita, identificação facial entre outras aplicações. Os métodos de classificação incluem algoritmos como o SVM, *Decision Trees* e redes neuronais [27].

- Os métodos de regressão permitem construir modelos de dados para prever dados futuros. Os modelos podem então ser atualizados à medida que novos dados se tornam disponíveis. Se os dados forem usados apenas uma vez para criar o modelo, então é chamado um método de *batch* ou lote. Um método de regressão que incorpora dados que estejam sempre a atualizar é chamado de método recursivo [27].

- O clustering encontra agrupamentos nos dados. O reconhecimento de objetos é uma aplicação de métodos de agrupamento. Por exemplo, se se quiser encontrar um carro

em uma imagem, procure os dados que estão associados à parte de uma imagem que é um carro. Embora os carros tenham formas e tamanhos diferentes, eles têm muitas características em comum [27].

Esta toolbox tem muitas funções para oferecer suporte a essas áreas e muitas que não se encaixam perfeitamente nessas categorias. A toolbox de Estatísticas e de ML contém ferramentas excelentes para começar a explorar algumas estratégias de ML [27].

Classification Learner

Classification Learner é uma app que está incorporada dentro da toolbox de Estatísticas e de ML.

Em ML, a escolha do modelo de classificação correto costuma ser uma questão de tentativa e erro. A app *Classification Learner* torna isso mais fácil [23].

É possível explorar os dados, selecionar recursos, especificar esquemas de validação cruzada, treinar modelos e avaliar os resultados [23].

O treino pode ser realizado de forma automática para identificar o melhor tipo de modelo de classificação, incluindo *Decision Trees*, análise discriminante, SVM, regressão logística, KNN e classificação de conjunto [23].

Fornecendo um conjunto de dados conhecidos de *input* (observações ou exemplos) e respostas conhecidas aos dados (ou seja, rótulos ou classes) é treinado um modelo, que posteriormente pode gerar previsões para dados novos [23].

3.6 *Machine Learning*: Python

Python é uma linguagem de programação de alto nível. A sua sintaxe fácil de aprender e capacidade de portabilidade torna-o popular atualmente, nomeadamente na área de *Data Science* [28].

De acordo com estudos e pesquisas, o Python é a quinta linguagem de programação mais importante, bem como a linguagem mais popular para ML e *Data Science* [28].

O Python apresenta diversos pontos positivos:

- **Fácil de aprender e entender:** a sintaxe do Python é simples, portanto, é relativamente fácil compreender a linguagem, principalmente para iniciantes.
- **Linguagem multi-funcional:** Python é uma linguagem de programação multi-funcional porque oferece suporte à programação estruturada, à programação orientada a objetos e também à programação funcional.
- **Grande número de módulos:** Python tem um grande número de módulos para cobrir todos os aspectos da programação. Esses módulos estão facilmente disponíveis para uso, tornando o Python uma linguagem extensível.
- **Suporte da comunidade de código aberto:** Por ser uma linguagem de programação de código aberto, o Python é apoiado por uma grande comunidade de desenvolvedores. Devido a isso, os bugs são facilmente corrigidos pela comunidade Python. Esta característica torna o Python muito robusto e adaptativo.
- **Escalabilidade:** Python é uma linguagem de programação escalável porque fornece uma estrutura para suportar programas grandes do tipo *shell-scripts*.

Embora o Python seja uma linguagem de programação popular e poderosa, tem fraquezas, e uma delas é velocidade lenta de execução. A velocidade de execução do Python é lenta em comparação com as linguagens compiladas porque Python é uma linguagem interpretada. Esta pode ser a principal área de melhoria para a comunidade Python [28].

3.6.1 NumPy

É um componente útil que torna o Python uma das linguagens favoritas para a *Data Science*. Basicamente, significa *Numerical Python* e consiste em objetos de array multi-dimensionais [28]. Ao usar o NumPy, podemos realizar as seguintes operações:

- Operações matemáticas e lógicas em matrizes.
- Transformação de Fourier.

- Operações associadas à álgebra linear.

Também podemos ver o NumPy como a substituição do MatLab porque o NumPy é mais usado junto com o Scipy (*Scientific Python*) e o Mat-plotlib (biblioteca de plot) [28].

3.6.2 Pandas

É outra biblioteca útil do Python. O Pandas é basicamente usado para manipulação, organização e análise de dados. Foi desenvolvido por Wes McKinney em 2008 [28]. Com a ajuda do Pandas, no processamento de dados, pode-se realizar as cinco etapas a seguir:

- Carregar
- Preparar
- Manipular
- Modelar
- Analisar

Representação de dados em Pandas:

Toda a representação dos dados no Pandas é feita com a ajuda das três estruturas de dados [28]. Essas estruturas são:

- ***Series*:** é basicamente um **ndarray** uni-dimensional com um rótulo de eixo, o que significa que é como um array simples com dados homogéneos. Por exemplo, a série a seguir é uma coleção de inteiros 1,5,10,15,24,25,...
- ***Data frame*:** é a estrutura de dados mais útil e usada para quase todos os tipos de dados representação e manipulação em Pandas. É basicamente uma estrutura de dados bi-dimensional que pode conter dados heterogéneos. Geralmente, os dados tabulares são representados por meio de *Data Frame*. Por exemplo, uma tabela que mostra os dados dos alunos com seus nomes, números, idade e sexo.

- ***Panel***: é uma estrutura de dados tridimensional contendo dados heterogéneos. É muito difícil representar o *Panel* em representação gráfica, mas pode ser ilustrado como um array de textitData Frame.

Podemos entender essas estruturas de dados porque a estrutura de dados de dimensão superior é o array da estrutura de dados de dimensão inferior [28].

3.6.3 Scikit-learn

É outra biblioteca útil para *Data Science* e ML em Python é Scikit-learn [28]. A seguir estão alguns recursos do Scikit-learn que o tornam tão útil:

- É baseado em NumPy, SciPy e Matplotlib.
- É um código aberto e pode ser reutilizado sob licença BSD.
- É acessível a todos e pode ser reutilizado em vários contextos.
- Ampla gama de algoritmos de ML cobrindo as principais técnicas de ML, como classificação, agrupamento, regressão, redução dimensional, seleção de modelo, etc., podem ser implementados com a ajuda desta biblioteca.

Capítulo 4

Caso de estudo e resultados numéricos

4.1 Caso de Estudo

Esta secção pretende apresentar os parâmetros e os dados utilizados neste trabalho.

Os dados foram obtidos através do estudo da linha U1 dos STUB (Serviço de Transportes Urbanos de Bragança), onde foram escolhidos os parâmetros necessários para a realização deste trabalho.

Teve-se em consideração os seguintes parâmetros:

- Hora na origem (convertidas em segundos e em minutos);
- Hora de chegada (convertidas em segundos e em minutos);
- Local de origem;
- Local de chegada;
- Fluxo de passageiros (somatório de passageiros que entram e saem).

As horas na origem e de chegadas foram obtidos através do histórico de viagens de um autocarro que percorre a linha U1; esses valores estão disponibilizados nas Tabelas 4.2 e

4.3. Essas horas foram convertidas em segundos para um conjunto de experiências, e em outras experiências, o tempo foi convertido em minutos.

O propósito de realizar experiências em duas unidades de tempo serve para validar a precisão do algoritmo, apesar que usualmente as previsões do tempo de chegada dos meios de transportes públicos são feitas em minutos.

Os dados dos locais na origem e de chegada e as horas estipuladas de chegada foram obtidos através da tabela de horários dos STUB disponível no site da Câmara Municipal de Bragança, como é mostrado na Figura 4.1.

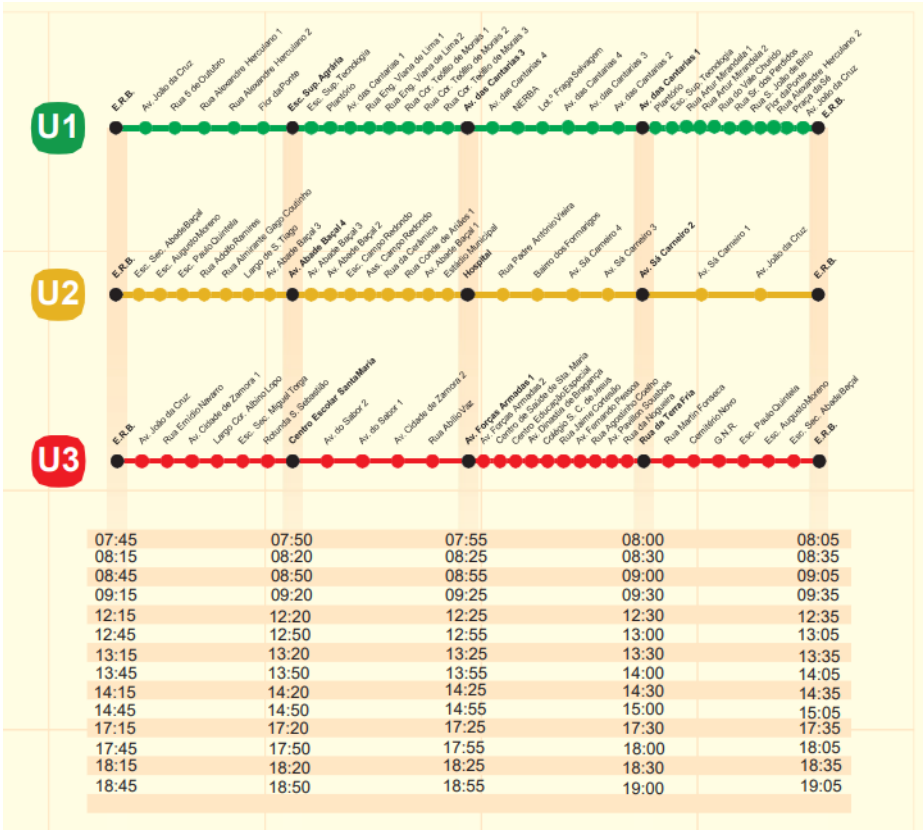

Figura 4.1: Tabela de horários das três linhas urbanas de Bragança [9]

A linha U1 é composta por 34 paragens, sendo que algumas delas se repetem no retorno do autocarro, totalizando assim 26 paragens distintas. Na tabela 4.1 estão representadas as paragens com os seus respetivos números atribuídos para a construção da base de dados.

Número do Local	Local
1	E.R.B
2	Av. João da Cruz
3	Rua 5 de Outubro
4	Rua Alexandre Herculano 1
5	Rua Alexandre Herculano 2
6	Flor da Ponte
7	Esc. Sup. Agrária
8	Esc. Sup. Tecnologia
9	Plantório
10	Av. Das Cantarias 1
11	Rua Eng. Viana de Lima 1
12	Rua Eng. Viana de Lima 2
13	Rua Cor. Teófilo de Morais 1
14	Rua Cor. Teófilo de Morais 2
15	Rua Cor. Teófilo de Morais 3
16	Av. Das Cantarias 3
17	Av. Das Cantarias 4
18	NERBA
19	Lotº Fraga Selvagem
20	Av. Das Cantarias 2
21	Rua Artur Mirandela 1
22	Rua Artur Mirandela 2
23	Rua do Vale Chorido
24	Rua Sr. Dos Perdidos
25	Rua S. João de Brito
26	Praça da Sé

Tabela 4.1: Paragens da linha U1

A linha U1 faz o percurso apresentado na Figura 4.2, tendo origem na Estação Rodoviária de Bragança (E.R.B.), até ao NERBA,e depois retorna à E.R.B.

Nas Tabelas 4.2 e 4.3 mostra-se como estão organizados os parâmetros da base de dados para as primeiras experiências.

Na Tabela 4.2 as horas são expressas conforme descrito na fórmula 4.1.

$$(horas \times 3600) + (minutos \times 60) + segundos \tag{4.1}$$

Por exemplo, a primeira viagem do dia começa às 7:45, isto é 27900 segundos, e termina

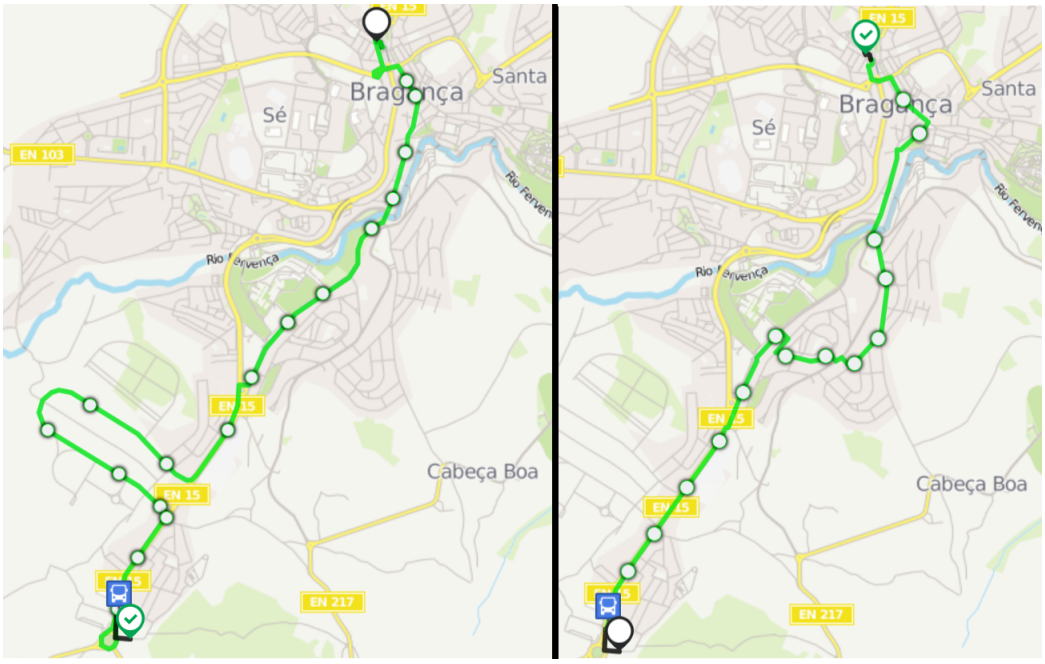

Figura 4.2: Trajeto de viagem da linha U1 [24]

às 19:05, que corresponde a 68700 segundos. Tendo em conta que as 00:00 equivalem a 0 segundos e 23:59 equivale a 86340 segundos.

O fluxo de passageiros representa o somatório da quantidade de passageiros que entraram (IN) e saíram (OUT) numa determinada paragem.

$$Fluxo = IN + OUT$$

Índice	**Hora na origem (seg.)**	**Hora de chegada (seg.)**	**Local de origem**	**Local de chegada**	**Fluxo passageiros**
1	27840	27900	1	2	10
...	...	...	...	...	...
1518	68670	68700	2	1	15

Tabela 4.2: Conjunto de 1548 amostras (horas expressas em segundos)

Os dados da Tabela 4.2 e da 4.3, representam 23 voltas completas numa mesma linha, em torno de 26 paragens, durante dois dias. Do índice 1 até ao índice 759 representam-se os dados do primeiro dia, e de 760 até 1518 representam-se os dados do segundo dia. E

isto pode ser observado na Figura 4.3, onde cada pico verde representa o numero de cada ciclo de viagem.

No gráfico, laranja representa o fluxo de passageiros ao longo do dia de viagem.

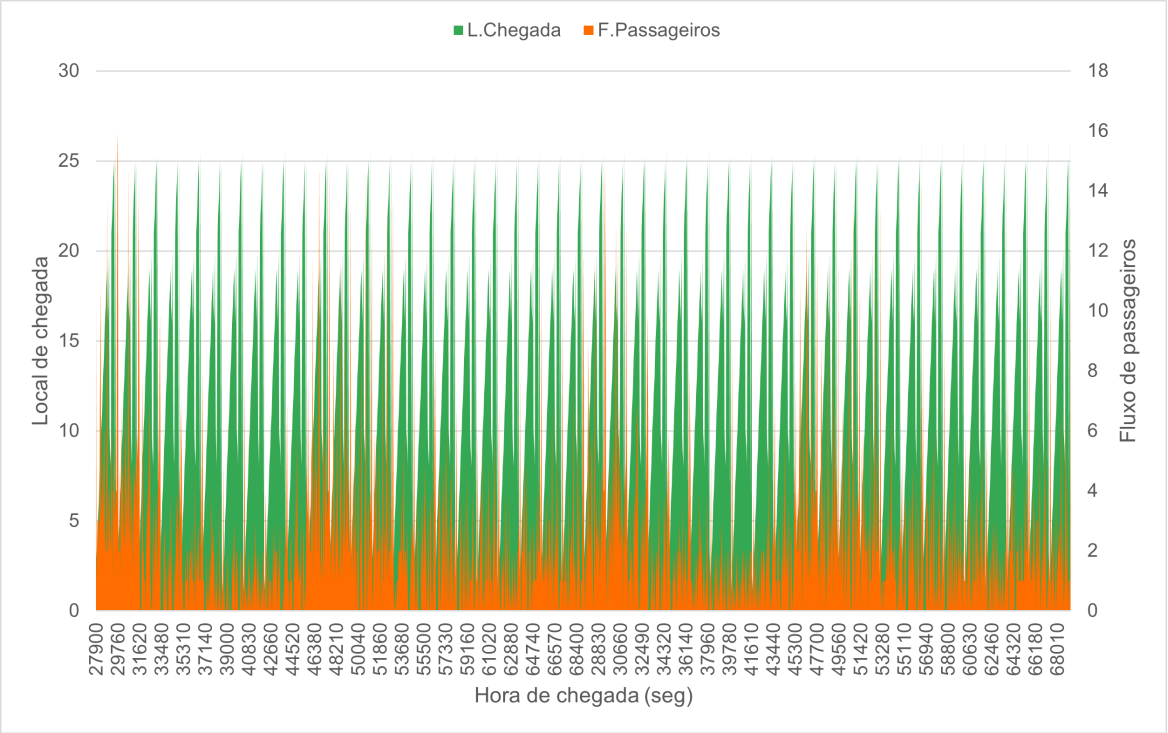

Figura 4.3: Representação de três parâmetros da Tabela 4.2

Na Tabela 4.3, onde o tempo é expresso em minutos, foi utilizada a seguinte fórmula:

$$(Horas \times 60) + Minutos \tag{4.2}$$

Por exemplo, a primeira viagem do dia começa as 7:45, isto é 465 minutos, e última viagem termina as 19:05 que corresponde a 1145 minutos. Tendo em conta que as 00:00 equivale a 0 minutos e 23:59 equivale a 1439 minutos. Os dados representam 23 voltas completas numa mesma linha, em torno de 26 paragens, durante dois dias.

Como mencionado anteriormente, os dados representam 23 voltas completas numa mesma linha, em torno de 26 paragens, durante dois dias.

A Figura 4.4 é idêntica à Figura 4.3, com a diferença na representação do tempo de chegada. Anteriormente era em segundos e nesta figura é em minutos.

Índice	Hora origem (min.)	Hora chegada (min.)	Local origem	Local chegada	Fluxo passageiros
1	464	465	1	2	10
...	...	...	...	...	...
1518	1144	1145	2	1	15

Tabela 4.3: Conjunto de 1548 amostras (horas expressas em minutos)

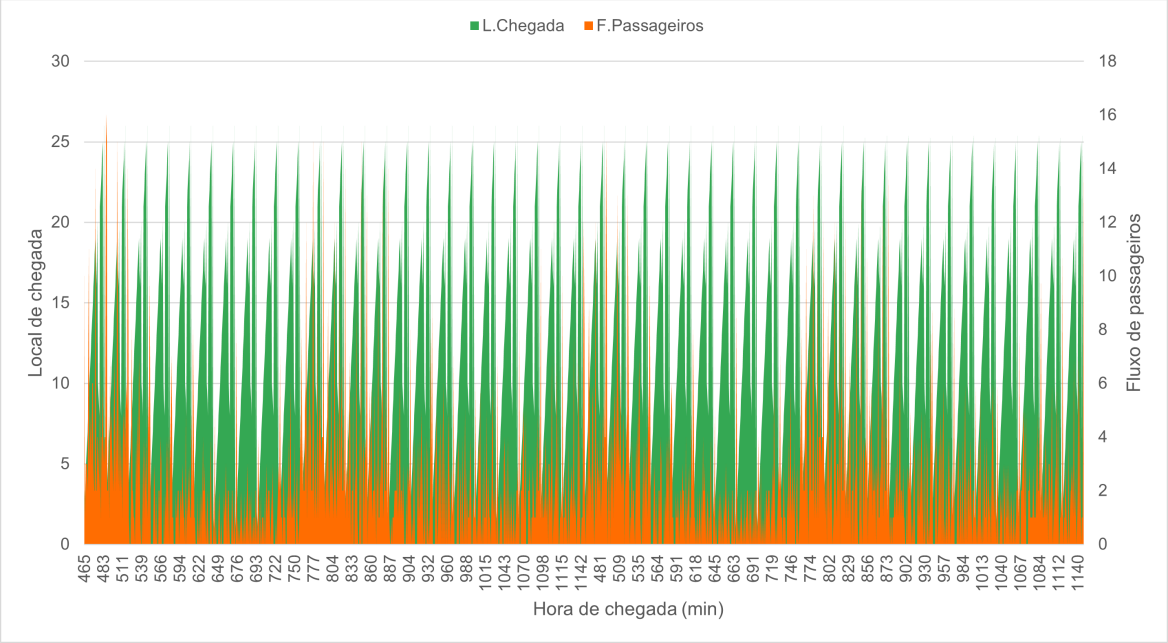

Figura 4.4: Representação de três parâmetros da Tabela 4.3

Também nesta figura, são visíveis os picos das 23 voltas dos dois dias totalizando 46 voltas. A representação do fluxo de viagens é igual à da Figura 4.3.

4.2 Resultados Numéricos

Esta secção apresenta os resultados obtidos pelas diversas experiências realizados com o método de *K-means* e *Support Vector Machine*.

4.2.1 Experiências com o Método de *K-means*

Com o método de *K-means* foram realizados duas experiências. E o que difere essas duas experiências é o modo de como é expresso a unidade de tempo. A **experiência Nº1** é expressa em segundos enquanto que a **experiência Nº2** é expressa em minutos. E foram utilizados 3 dos 5 parâmetros apresentados na Tabela 4.2:

- Tempo de chegada
- Local de chegada
- Fluxo de passageiros

As **Experiências Nº1** e **Nº2** foram realizados em Python (Google *Colaboratory*).

Experiência Nº1

A **Experiência Nº 1** é composta por um conjunto de 24 amostras retirados da Tabela 4.2. O objetivo desta experiência é delinear e validar o funcionamento do algoritmo de *K-means*. Essas amostras estão representadas na Tabela 4.4.

Índice	**Hora de chegada**	**Local de chegada**	**Fluxo de passageiros**
1	27900	2	3
...	...	...	...
24	28830	1	2

Tabela 4.4: Conjunto de 24 amostras da **Experiência Nº 1**

Essas amostras, em contexto real, equivalem a dados de um dia, 3 voltas completas do mesmo autocarro em 6 paragens distintas, em três períodos do dia.

- O primeiro período começa no índice 1 e vai até ao índice 8
- O segundo período começa no índice 9 e vai até ao índice 16
- O último período começa no índice 17 e acaba no índice 24

A primeira viagem do dia começa às 7:45, isto é 27900 segundos, e termina às 8:30 que corresponde a 28830 segundos.

Nestas experiências, é visualizado como funciona o algoritmo de *K-means* juntamente como o método do *Elbow*.

Primeiramente, **na Experiência Nº1**, é calculado o número ideal de clusters com o método de *Elbow*. Na Figura 4.5 está representado o curvatura da linha do *Elbow*, ou curva da distorção, entre os dados do tempo de chegada e do fluxo de passageiros.

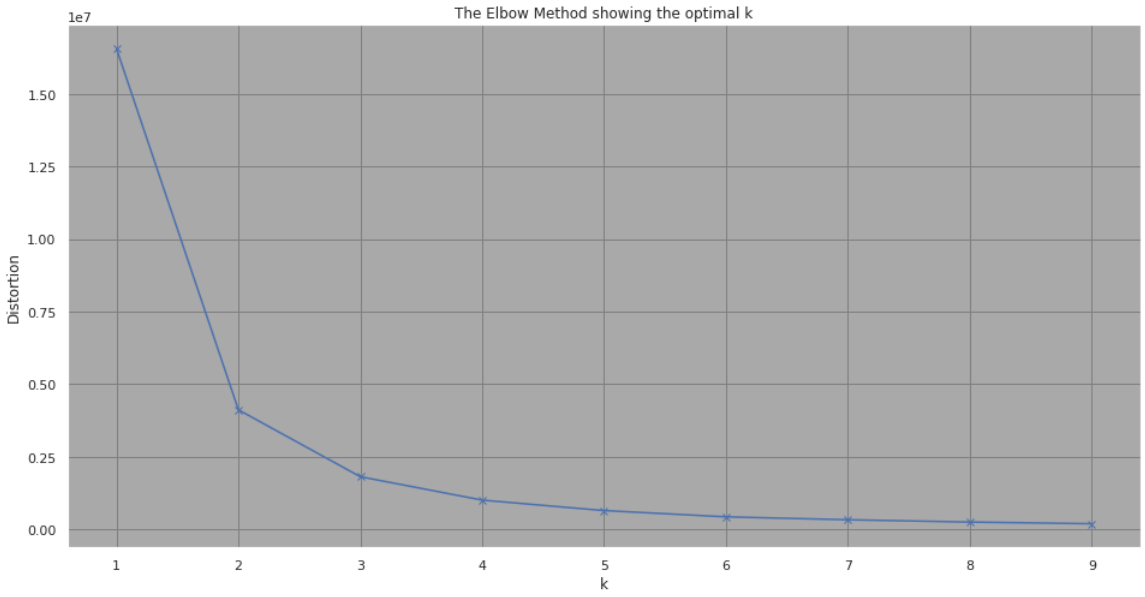

Figura 4.5: Curva de distorção do método de *Elbow* da **Experiência Nº1**

O eixo horizontal representa o valor de número de clusters (N) e o eixo vertical representa o valor da distorção.

A curva de distorção da Figura 4.5 *Elbow* estabiliza-se em N=3, logo o nosso conjunto de pontos deve ser divido em 3 clusters.

Sabendo o número ideal de clusters, é executado o algoritmo de *K-means*. É gerado um mapa bidimensional, Figura 4.6, onde o eixo horizontal representa as horas de serviço do transporte público, e o eixo vertical representa o número do fluxo de passageiros.

Então, como se observa nesse mapa, os pontos estão dispersos em três grupos distintos, logo era de se prever que o método de *Elbow* iria definir três clusters. Os círculos em preto

são designados de centros dos clusters (C), ou centroide. O centroide é o centro geométrico de um objeto convexo e pode ser designado como uma generalização da média [25].

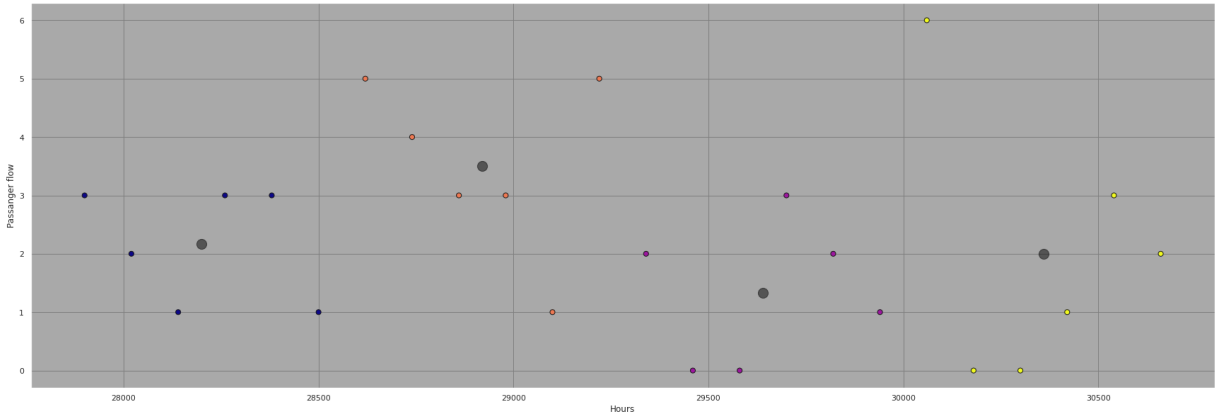

Figura 4.6: Clusters do método de *K-means* da **Experiência Nº1**

A Figura 4.6 é a representação gráfica do resultado do algoritmo de *K-means* entre o tempo de chegada (eixo horizontal) e o fluxo de passageiros (eixo vertical).

Com a análise desta representação é possível identificar o período em que o transporte público opera, a dispersão do fluxo entre as horas de trabalho, e o máximo e mínimo do fluxo de passageiros.

Nesta experiência, os centroides tem coordenadas distintas. E essas coordenadas estão representadas na Tabela 4.5.

Centroide	**Coordenada Horizontal**	**Coordenada Vertical**
C1	29640	1,333
C2	28200	2,167
C3	28920	1,500
C4	30360	2,000

Tabela 4.5: Coordenadas dos centroides de *K-means* da **Experiência Nº1**

O $C1$ e o $C3$ apresentam valores de coordenadas vertical próximos, isto acontece porque a média do fluxo para esses dois clusters são idênticos. E o mesmo acontece para o $C2$ e $C4$.

Experiência Nº2

Nesta experiência o conjunto de dados contém 1518 amostras, e esta tem a finalidade de visualizar o comportamento do algoritmo com um número maior de dados.

Estas amostras representam 23 voltas completas numa mesma linha, em torno de 26 paragens, durante dois dias.

A primeira viagem do dia começa às 7:45, isto é 27900 segundos e termina às 19:05, o que corresponde a 68700 segundos.

Índice	**Hora de chegada**	**Local de chegada**	**Fluxo de passageiros**
1	27900	1	10
2	27960	2	1
3	28020	3	3
...	...	...	...
1516	68610	6	1
1517	68640	26	4
1518	68700	2	15

Tabela 4.6: Conjunto de 1518 amostras da **Experiência Nº 2** (*K-means*)

Aplicando o método do *Elbow* a esse conjunto de amostras, o método devolve um valor de N=4, como é visualizado na Figura 4.7.

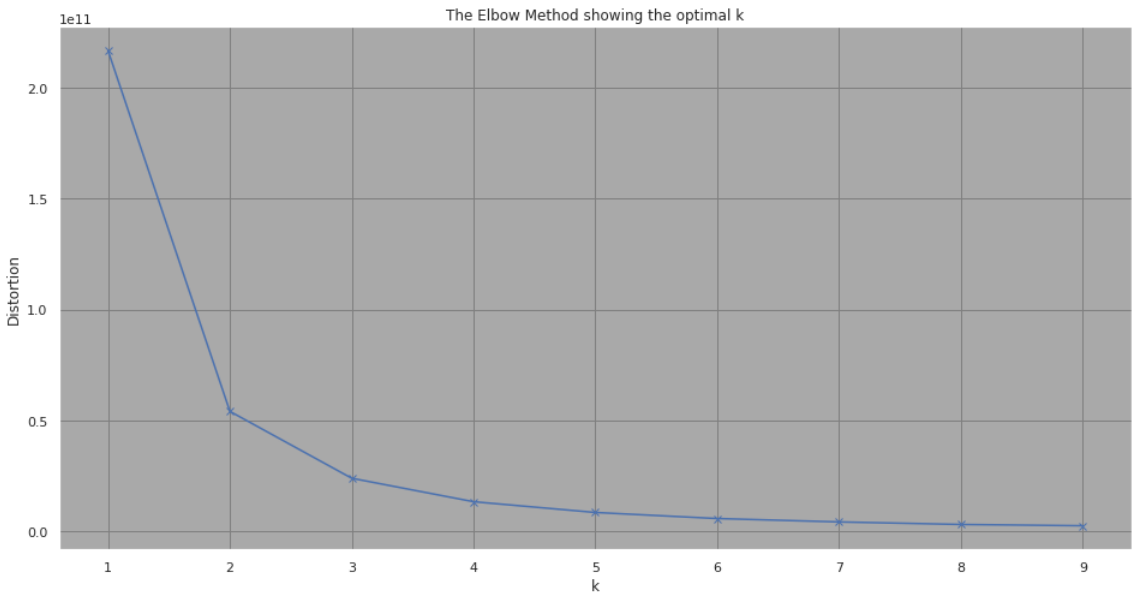

Figura 4.7: Curva de distorção do método de *Elbow* da **Experiência Nº2**

- O primeiro cluster, identificado pela cor violeta, representa o fluxo de passageiro entre às 7:45 até às 10:35. Há um pico no fluxo logo pela manhã e isto deve-se ao número de utentes que vai para o trabalho/aulas; logo de seguida esse fluxo baixa para os valores médios.

- No cluster seguinte, com os limites entre às 10:45 até às 13:35, apresenta um comportamento inverso do primeiro. O fluxo de passageiros é ascendente ao longo das horas, isto devido a hora do almoço, ou para alguns alunos o final das aulas. E esse pico de crescimento abrange também o terceiro cluster até por volta das 15:05.

- Os limites do terceiro cluster estão entre as 13:45 e as 16:05. Este apresenta um comportamento similar ao primeiro cluster, isto é, um comportamento descendente do fluxo ao longo das horas.

- E, finalmente, o quarto cluster, delimitado entre as 16:15 até às 19:05, representa as horas do final do dia. E como esperado, há um aumento no fluxo de passageiros dado a hora de retorno dos utilizadores às sua residências.

Na Figura 4.8, o eixo horizontal representa as horas de trabalho, em que a primeira

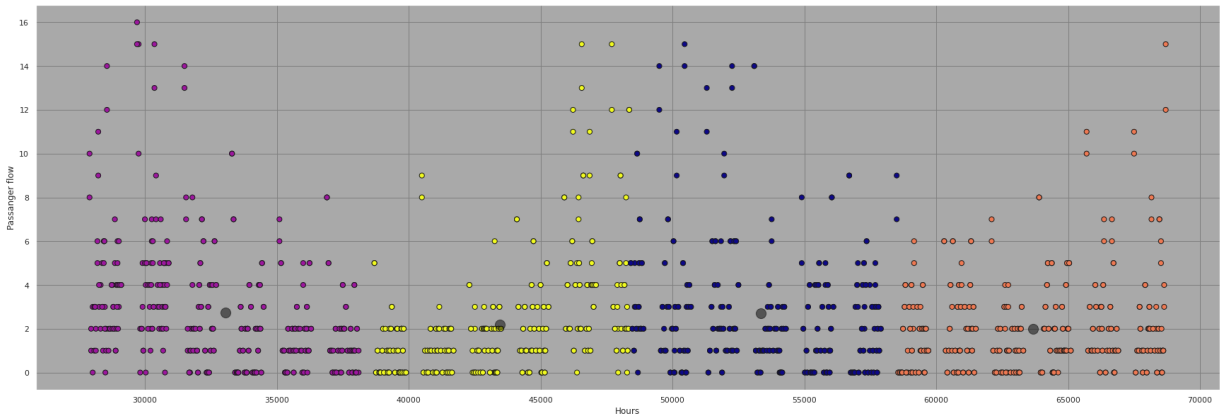

Figura 4.8: Clusters do método de *K-means* da **Experiência Nº2** com k=4

viagem começa às 7:45 e a última viagem começa às 18:45 e acaba às 19:05, e o eixo vertical representa o fluxo de passageiros, que varia entre os 0 e 16.

As amostras dos dois dias de viagem, representados na Figura 4.8, estão relativamente sobrepostos. Com uma maior densidade de amostras fica melhor a identificação de padrões nos dados em estudo. Pode-se identificar as ditas horas de pico, em que são as horas em que há uma maior movimentação de passageiros, que são:

- Logo pelo início da manhã
- Antes e depois da hora de almoço
- E no final do dia.

A Tabela 4.7 representa os centroides da **experiência Nº**4 com o k=4.

Centroide	**Coordenada Horizontal**	**Coordenada Vertical**
C1	54280	2,514
C2	33049	2,750
C3	43849	2,285
C4	64153	2,050

Tabela 4.7: Coordenadas dos centroides de *K-means* da **Experiência Nº2** com k=4

Também é possível identificar as horas com pouca densidade de passageiros que entram e saem, que são os períodos entre as horas de pico.

Se for escolhido um k=3, temos os dados agrupados em 3 clusters, como é visível na Figura 4.9. E esses clusters são divididos nas horas de baixa densidade. O primeiro cluster vai do início da primeira viagem até por volta das 10 da manhã. O segundo das 10 horas da manhã até por volta das 16 horas. E o ultimo cluster tem início às 16 horas e vai até à última viagem do autocarro.

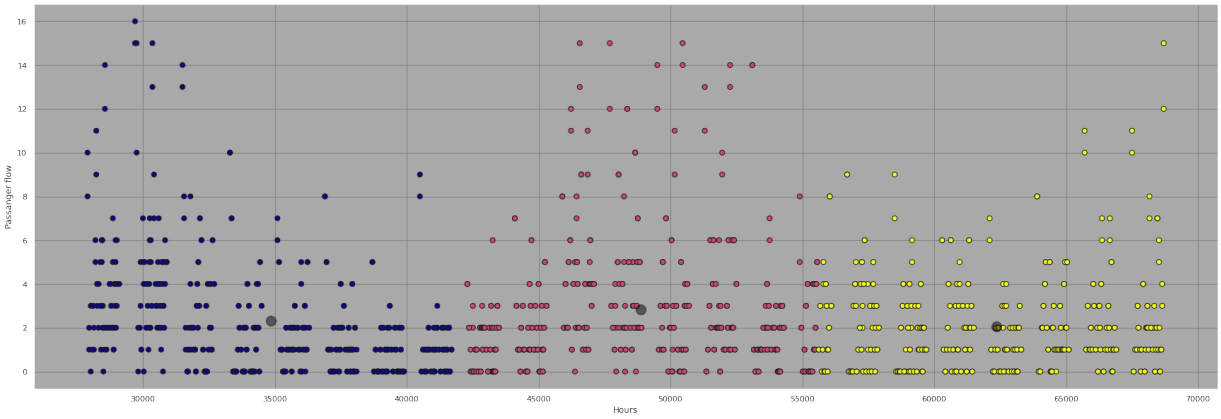

Figura 4.9: Clusters do método de *K-means* da **Experiência Nº2** com k=3

E na Tabela 4.8 estão representados os respetivos centroides dos três clusters representados na Figura 4.9.

Centroide	**Coordenada Horizontal**	**Coordenada Vertical**
C1	34849	2,330
C2	48904	2,831
C3	62384	2,053

Tabela 4.8: Coordenadas dos centroides de *K-means* da **Experiência Nº2** com k=3

É visível que segundo cluster apresenta um maior fluxo de passageiros devido ao valor da sua coordenada vertical. O segundo cluster com maior fluxo é o primeiro e por último é o terceiro cluster.

Por outro lado, se é escolhido manualmente um k=2 teremos dois clusters de dados. Como se pode ver na Figura 4.10, o primeiro clusters está representado em amarelo e o segundo em azul.

O cluster em amarelo engloba os dados da viagem do início do dia até por volta do

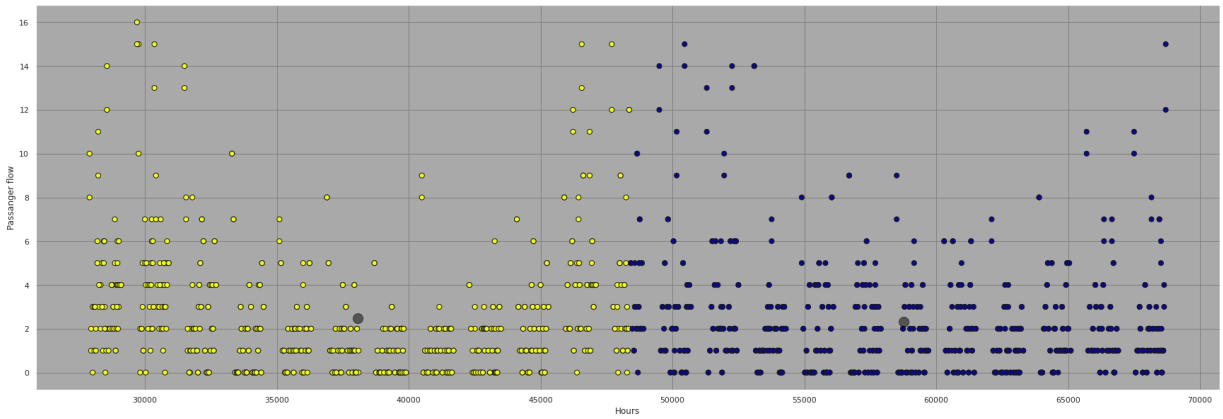

Figura 4.10: Clusters do método de *K-means* da **Experiência Nº2** com k=2

meio dia, e o segundo cluster, em azul, engloba os dados do meio dia até ao final do dia de viagem.

Logo o número de centroides é dois e os valores deste estão representados na Tabela 4.9.

Centroide	**Coordenada Horizontal**	**Coordenada Vertical**
C1	38074	2,482
C2	58761	2,330

Tabela 4.9: Coordenadas dos centroides de *K-means* da **Experiência Nº2** com k=2

Comparando as coordenadas verticais, podemos verificar que a média do fluxo de passageiros foi maior no primeiro cluster do que no segundo cluster. Isto é, o movimento de passageiros é maior durante o período da manhã.

4.2.2 Experiências com o Método de SVM

Com o método *K-means* foram realizados duas experiências. O que difere nessas experiências é a quantidade de dados.

Já com o método de SVM foram realizados quatro experiências.

As **experiências Nº3** e **Nº4** utilizam uma quantia reduzida de amostras. Uma delas a unidade de tempo é expressa em segundos enquanto outra é expressa em minutos.

As **experiências Nº5** e **Nº6** utilizam o conjunto de amostras completo. Como nas duas experiências anteriores, a **experiência Nº5** está expressa em segundos enquanto que a **experiência Nº6** está expressa em minutos.

No primeiro caso a unidade do tempo está expressa em segundos, logo o resultado das previsões são em segundos, e no segundo caso, a unidade do tempo está expressa em minutos, logo o resultado das previsões é expresso em minutos.

Antes das primeiras experiências com o método de SVM, é definido o conjunto dos *labels* ou parâmetros de saída, que representa os dados que queremos prever. Os restantes dados são denominados de *features* ou parâmetros de entrada.

Esses dois conjuntos, *labels* e *features*, são divididos em duas partes. Essas partes são designadas por conjunto de treino e conjunto de teste, sendo que o conjunto de treino é composto por dados de treino ou modelação e dados de validação. Os dados de validação

são valores escolhidos dentro do conjunto de treino, podendo ou não ser aleatórios. Já o conjunto de teste são dados novos em comparação com os de treino.

As **Experiências Nº3**, **Nº4**, **Nº5** e **Nº6** foram realizados em Python (Google *Colaboratory*).

Experiência Nº3

Como acontece na **Experiência Nº1**, a **Experiência Nº3** é realizada com poucos dados para validar o comportamento do algoritmo.

Esta experiência utiliza dados simulados, que representam o comportamento de um autocarro. Este autocarro começa o trajeto na Estação Rodoviária de Bragança às 7:45 e termina na rotunda do NERBA (Associação Empresarial do Distrito de Bragança) às 8:49.

Esses dados estão agrupados em dois conjuntos, *features* e o *labels* (ver Tabela 4.10) e estão organizados da seguinte forma:

- O conjunto de treino é composto pelos elementos do índice 1 ao índice 33
- O conjunto de validação é composto pelo elemento do índice 33
- O conjunto de teste é composto pelos elemento do índice 34

Na Tabela 4.10 estão representados os elementos dos *features*. Os *features* ou os parâmetros de entrada são:

- Horas na origem (em segundos)
- Local de origem
- Local de chegada
- Fluxo de passageiros

	Hora na origem (seg)	**Local de origem**	**Local de chegada**	**Fluxo de passageiros**	**Hora de chegada**
Índice	***Features* de treino**				***Labels* de treino**
1	27900	1	2	10	27980
2	28020	2	3	1	28090
3	28140	3	4	3	28200
...	...	...	...	...	...
31	31500	14	15	6	31680
32	31620	15	16	3	31800
33	31740	16	17	2	31920
Índice	***Features* de validação**				***Labels* de validação**
33	31740	16	17	2	31920
Índice	***Features* de teste**				***Labels* de teste**
34	31860	17	18	2	32040

Tabela 4.10: Conjunto de *features* e *labels* da **Experiência Nº3**

Também está representado o conjunto dos *labels* da **Experiência Nº3** que representa a hora de chegada.

As *labels* representam o conjunto no qual queremos fazer a previsão. Nesta experiência, o objetivo é prever o tempo de chegada do autocarro. Então, com o método de SVM, é feita a previsão do tempo de chegada da primeira viagem do dia seguinte.

Executando o algoritmo de SVM, utilizando diferentes tipos *Kernel*, é obtido o valor previsto e com esse valor é calculado o Erro Quadrático Médio (EQM). Esses valores estão representados na Tabela 4.11.

Erro Absoluto	**Linear**	**Poly**	**RBF**	**Sigmoid**
Validação (seg)	0	0	0	962
Teste (seg)	10	10	10	1125

Tabela 4.11: Resultados da **Experiência Nº3**

Nota-se que o valor de EQM para o calculo da previsão utilizando valores de validação é zero para o *Kernel* Linear, Poly e RBF.

Já o mesmo não acontece com a previsão utilizando valores do conjunto de teste, pois esses dados não são conhecidos pelo modelo de previsão. O algoritmo faz uma previsão

aproximada, daí a diferença de 10 segundos para o *Kernel* Linear, Poly e RBF.

O algoritmo ótimo é aquele que apresenta o menor erro absoluto. Por exemplo, se estamos a prever a chegada de um autocarro numa determinada hora, o erro de previsão aceitável tem que ser baixo, isto é, se a previsão está para as 16:00 com um erro absoluto de 1 minuto, quer dizer que esse autocarro pode chegar entre as 15:59 e 16:01, salvo se acontecer algum acidente de percurso do mesmo.

Já o *Kernel* Sigmoid apresenta valores de EQM altos tanto para a previsão com o conjunto de validação e como o de teste. Isto dá-se porque este *Kernel* não é aconselhável para este tipo de dados.

Experiência Nº4

A **Experiência Nº4** é idêntica à **Experiência Nº3**, com a diferença no modo de como são expressas as horas. Nesta experiência, os parâmetros de *Hora na origem* e o da *Hora na chegada* são expressas em minutos.

A Tabela 4.12 representa o conjunto das *features* e das *labels* da **Experiência Nº4**.

	Hora na origem (min)	**Local na origem**	**Local de chegada**	**Fluxo de passageiros**	**Hora de chegada**
Índice	***Features* de treino**				***Labels* de treino**
1	465	1	2	10	466
2	467	2	3	1	468
3	469	3	4	3	470
...	...	...	...	...	...
31	525	14	15	6	528
32	527	15	16	3	530
33	529	16	17	2	532
Índice	***Features* de validação**				***Labels* de validação**
33	529	16	17	2	532
Índice	***Features* de teste**				***Labels* de teste**
34	531	17	18	2	534

Tabela 4.12: Conjunto de *features* e *labels* da **Experiência Nº4**

Depois de serem feitas as previsões, é calculado o valor do EQM. É calculado o EQM

do valor previsto utilizando dados do conjunto de validação e, depois, com os dados do conjunto de teste, como está representado na Tabela 4.13.

Erro Absoluto	**Linear**	**Poly**	**RBF**	**Sigmoid**
Validação (min)	0	0	0	66
Teste (min)	2	2	2	68

Tabela 4.13: Resultados da **Experiência Nº4**

O valor do EQM em relação ao conjunto validação, com *Kernel* linear, poly e RBF é zero, como acontece na **Experiência Nº3**. Como dito anteriormente, isso é indicativo de que o algoritmo consegue prever valores já conhecidos pelo modelo. O que já não acontece com o *Kernel* sigmoid, pelo mesmo motivo mencionado anteriormente, tanto em relação ao conjunto de validação como para o de teste.

O EQM em relação ao conjunto de teste, é expectável que seja diferente de zero pois, como já mencionado, o modelo não conhece os valores do conjunto de teste. Este faz uma previsão aproximada do valor de chegada, e este valor é igual para o *Kernel* Linear, Poly e RBF.

Experiência Nº5

Na **Experiencia Nº5**, o conjunto de amostras é maior que o das experiências Nº3 e Nº4. Isto porque já se sabe como o Método de SVM funciona.

Nesta experiência foram utilizados dados reais para a construção do modelo de previsão SVM. As *Horas na origem* e *Horas de chegada* estão expressas em segundos. A configuração dos *features* e *labels* é idêntica às duas experiências anteriores. Na Tabela 4.14 estão representados os *features* e os *labels*.

Na tabela 4.15, estão representados os resultados da **Experiência Nº5**. Nesta experiência, com o *Kernel* Linear, os cálculos não devolvem valores concretos devido à natureza das amostras serem em segundos.

CNC - Cálculos não conclusivos

Os resultados com *Kernel* Poly e RBF para o EQM da previsão com valores de validação são zero, pois o modelo já conhece os dados que se quer prever. E o EQM da previsão

	Hora na origem (seg)	**Local de origem**	**Local de chegada**	**Fluxo de passageiros**	**Hora de chegada**
Índice	***Features* de treino**				***Labels* de treino**
1	27840	1	2	10	27900
2	27930	2	3	1	27960
3	27990	3	4	3	28020
...	...	...	...	...	...
1515	68570	25	6	1	68580
1516	68595	6	26	1	68610
1517	68625	26	2	2	68640
Índice	***Features* de validação**				***Labels* de validação**
1517	68625	26	2	4	68640
Índice	***Features* de teste**				***Labels* de teste**
1518	68670	2	1	15	68700

Tabela 4.14: Conjunto de *features* e *labels* da **Experiência Nº5**

Erro Absoluto	**Linear**	**Poly**	**RBF**	**Sigmoid**
Validação (seg)	CNC	0	0	37950
Teste (seg)	CNC	60	60	38040

Tabela 4.15: Resultados da **Experiência Nº5**

com os valores de teste são de 60 segundos para ambos.

Novamente, para o *Kernel* Sigmoid, os resultados são elevados em comparação com o segundo e terceiro *Kernel*, tanto para o EQM a previsão com valores de validação como para valores de teste.

Experiência Nº6

Na **Experiência Nº6** só difere da experiência anterior na utilização da unidade de tempo, neste os campos de *Hora na origem* e *Hora de chegada* estão expressas em minutos. A Tabela 4.16 representa o conjunto das *features* e o conjunto das *labels*.

Já em minutos, o comportamento do modelo de previsão é totalmente diferente. Neste é possível obter a previsão da hora de chega como *Kernel* Linear e logo é possível calcular o EQM; zero minuto o para a previsões com o conjunto de validação e de 2 minutos para previsões com o conjunto de teste.

	Hora na origem (min)	**Local na origem**	**Local de chegada**	**Fluxo de passageiros**	**Hora de chegada**
Índice	***Features* de treino**				***Labels* de treino**
1	464	1	2	10	465
2	465	2	3	1	466
3	466	3	4	3	467
...	...	...	...	...	...
1515	1142	25	6	1	1143
1516	1143	6	26	1	1143
1517	1143	26	2	2	1144
Índice	***Features* de validação**				***Labels* de validação**
1517	1143	26	2	4	1144
Índice	***Features* de teste**				***Labels* de teste**
1518	1144	2	1	15	1145

Tabela 4.16: Conjunto de *features* e *labels* da **Experiência Nº6**

Erro Absoluto	**Linear**	**Poly**	**RBF**	**Sigmoid**
Validação (min)	0	2	10	640
Teste (min)	2	2	11	641

Tabela 4.17: Resultados da **Experiência Nº6**

Com o *Kernel* Poly, o EQM com o conjunto de validação tal como a de teste é de 2 minutos. Neste caso, o algoritmo já começa a ser pouco credível pois porque ele não consegue acertar previsões com dados já conhecidos.O EQM para previsões com o conjunto de validação deveria ser de 0 minutos.

Com o RBF acontece o mesmo com o Poly, e EQM de previsão já começa a ser grande pois a média de tempo de viagem de uma paragem y para um paragens y é de 1 a 2 minutos.

E novamente o sigmoid apresenta valores elevados, em relação aos outros 3 *Kernel*.

Capítulo 5

Conclusões e trabalhos futuros

Este capítulo reflete sobre as conclusões acerca do tema deste trabalho, e também propõe trabalhos futuros relacionados com o tema.

5.1 Conclusões

Com a utilização do algoritmo de *K-means*, comprovou-se que este é capaz de identificar padrões nos dados adquiridos de um transporte público. Podem-se identificar nas rotas zonas com maior afluência de passageiros e a hora em que isto acontece, uma das causas de atrasos nos transportes públicos.

Tendo os padrões identificados, podemos afirmar se uma determinada rota está bem consoante o fluxo de passageiros, ou se há um fluxo exagerado de passageiros ao longo de vários meses sucessivos e dar ao início ao processo de alteração dessa rota.

No futuro, será importante combinar o algoritmo de *K-means* com algoritmos de otimização de rotas, de forma a que se faça a leitura dos dados, calcule os clusters e devolva propostas para rotas alternativas.

Em relação ao segundo algoritmo, *Support Machine Vector*, é possível realizar a previsão do tempo de chegada de qualquer meio transporte. Sabendo os dados da rotina do mesmo, tal como a hora na origem, local na origem, local de chegada, fluxo de passageiros, coordenadas GPS, distância euclidiana, entre outros. Este dados de rotina são

designados de *features*. Neste trabalho optou-se pelo uso de 4 *features*, hora na origem, local na origem, local de chegada e fluxo de passageiros.

Já existem estudos que comprovam que o SVM é um ótimo método para fazer a previsão de tempos de chegadas de transportes através do estudo das *features*.

Dentro do algoritmo de previsão, foram estudadas 4 tipos de *Kernel*: *Linear*, *Poly*, RBF e *Sigmoid*. Para o tipo de dados/*features* em estudo nesta trabalho, o *Kernel* mais indicado é o linear, o poly e o RBF, pois o sigmoid apresenta uma taxa de erro elevada em relação ao outros três.

Tem que se ter em conta também que o *Linear* é indicado para dados relativamente pequenos. Por exemplo, a previsão com dados em segundos, as amostras seria de cinco algarismos, já em minutos seriam no máximo de quatro algarismos. Essa ligeira diferença faz com que o algoritmo não consiga calcular o resultado de previsão.Já com as amostras em minutos, o modelo apresenta um erro total de 2 minutos.

Entre o *Poly* e o RBF, embora que ambos apresentam bons resultados nos ensaios com as amostras em segundos (taxa de erro de 60 segundos), o *Poly* destaca-se pois nos ensaios em minutos apresenta melhor taxa de erro em relação ao RBF, 2 e 11 minutos respetivamente.

Os dados utilizados neste trabalho foram provenientes de um módulo desenvolvido por um colega, Diogo Martins, na sua dissertação.

5.2 Trabalhos Futuros

Para este trabalho, ainda ficam várias formas de expandir e melhorar os algoritmos de gestão e previsão. Sugestões para trabalhos futuros são:

- Aprofundar e aprimorar a capacidade do método de *K-means*. Desenvolver formas para categorizar padrões semelhantes e atuar imediatamente sobre os problemas, sugerindo propostas de melhoria.

- Em relação ao métodos de SVM, explorar mais a fundo o *Kernel* de RBF, pois

este engloba dois parâmetros (C e *gamma*) que modificam o comportamento do hiperplano de decisão.

- Explorar outras variantes para serem englobadas na *features* do RBF, por exemplo: a estação do ano, as condições meteorológicas, datas com atividades na localidade onde funcionam os transportes públicos, etc.

Bibliografia

[1] Fundação AIP. Portugal smart cities summit - machine learning. https://portugalsmartcities.fil.pt/, 2020.

[2] A. Amidi and S. Amidi. Cs 229 - machine learning - tips and tricks. https://stanford.edu/~shervine/l/pt/teaching/cs-229/dicas-truques-aprendizado-maquina#.

[3] C. Bai, Z.-R. Peng, Q.-C. Lu, and J. Sun. Dynamic bus travel time prediction models on road with multiple bus routes. *Computational intelligence and neuroscience, pp. vol. 2015, p. 63*, 2015.

[4] G. Bonaccorso. *Machine Learning Algorithms*. Packt Publishing Ltd., Birmingham, UK, 2017.

[5] A. Bulezyuk. Project introduction: Machine learning model: Python sklearn & keras. https://www.education-ecosystem.com/andreybu/REaxr-machine-learning-model-python-sklearn-kera/oPGdP-machine-learning-model-python-sklearn-kera/, Agosto 2017.

[6] S. I. Chien, Y. Ding, and C. Wei. Dynamic bus arrival time prediction with artificial neural networks. *Journal of Transportation Engineering, pp. vol. 128, no. 5 , pp. 429–438*, 2002.

[7] S. I.-J. Chien and C. M. Kuchipudi. Dynamic travel time prediction with realtime and historic data. *Journal of transportation engineering, pp. vol. 129, no. 6 , pp. 608–616*, 2003.

[8] P. Dangeti. *Statistics for Machine Learning*. Packt Publishing Ltd., Birmingham, UK, 1st edition, 2017.

[9] Município de Bragança. Stub-serviço de transportes urbanos de bragança. `https://www.cm-braganca.pt/servicos-e-informacoes/logistica-e-mobilidade/mobilidade/stub-servico-de-transportes-urbanos-de-braganca`.

[10] IBM Cloud Education. Machine learning. `https://www.ibm.com/cloud/learn/machine-learning`, Julho 2020.

[11] IBM Cloud Education. Unsupervised learning. `https://www.ibm.com/cloud/learn/unsupervised-learning`, Setembro 2020.

[12] B. Ferris, K. Watkins, and A. Borning. Onebusaway: results from providing realtime arrival information for public transit. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. pp. 1807–1816*, 2010.

[13] R. Gandhi. Support vector machine — introduction to machine learning algorithms. `https://bit.ly/2Z5dbZ1`, Junho 2018.

[14] M. J. Garbade. Understanding k-means clustering in machine learning. `https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1`, Setembro 2018.

[15] J. Gong, M. Liu, and S. Zhang. Hybrid dynamic prediction model of bus arrival time based on weighted of historical and real-time gps data. *Control and Decision Conference (CCDC), 2013 25th Chinese, pp. pp. 972–976*, 2013.

[16] Google. Google transit basics. `https://support.google.com/transitpartners/answer/1111471?hl=en`.

[17] J. Jordan. Support vector machines. `https://www.jeremyjordan.me/support-vector-machines/`, Junho 2017.

[18] S. Kim. *MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence*. Apress, Madrid, 1st edition, 2017.

[19] F. Li, Y. Yu, H. Lin, and W. Min. Public bus arrival time prediction based on traffic information management system. *Service Operations, Logistics, and Informatics (SOLI), 2011 IEEE International Conference, pp. pp. 336–341*, 2011.

[20] W. Liu, J. Liu, H. Jiang, B. Xu, H. Lin, G. Jiang, and J. Xing. Wilocator: Wifisensing based real-time bus tracking and arrival time prediction in urban environments. *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference, pp. pp. 529–538*, 2016.

[21] S. Maiti, A. Pal, A. Pal, T. Chattopadhyay, and A. Mukherjee. Historical data based real time prediction of vehicle arrival time. *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference, pp. pp. 1837–1842*, 2014.

[22] M. Marujo. Só uma minoria usa transportes públicos: Não são rápidos nem confortáveis. `https://www.dn.pt/cidades/so-uma-minoria-usa-transportes-publicos-9542110.html`, Julho 2018.

[23] MathWorks. Classification learner. `https://www.mathworks.com/help/stats/classificationlearner-app.html`, 2021.

[24] Inc Moovit. Moovit app. `https://moovit.com/pt`.

[25] L. Morissette and S. Chartier. The k-means clustering technique: General considerations and implementation in mathematica. *Tutorials in Quantitative Methods for Psychology*, 2013.

[26] R. Padmanaban, K. Divakar, L. Vanajakshi, and S. Subramanian. Development of a real-time bus arrival prediction system for indian traffic conditions. *IET intelligent transport systems, pp. vol. 4, no. 3, pp. 189–200*, 2010.

[27] M. Paluszek and S. Thomas. *MATLAB Machine Learning Recipes: A Problem-Solution Approach*. Apress, New York, NY, 2nd edition, 2019.

[28] Tutorials Points. *Machine Learning With Python*. Tutorials Point (I) Pvt. Ltd, India, 2019.

[29] S. J. Russel and P. Norvig. *Inteligência Artificial: Un Enfoque Moderno*. Pearson Educación, S.A, Madrid, segunda edición edition, 2004.

[30] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 2nd edition, 2018.

[31] J. Weng, C. Wang, H. Huang, Y. Wang, and L. Zhang. Real-time bus travel speed estimation model based on bus gps data. *Advances in Mechanical Engineering, pp. vol. 8, no. 11, p. 1687814016678162*, 2016.

[32] D. Wilimitis. The kernel trick in support vector classification. `https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f`, Dezembro 2018.

[33] B. Yu, W. H. K. Lam, and M. L. Tam. Bus arrival time prediction at bus stop with multiple routes. *Transportation Research Part C: Emerging Technologies, pp. vol. 19 , no. 6, pp. 1157–1170*, 2011.

[34] H. Yu, R. Xiao, Y. Du, and Z. He. A bus-arrival time prediction model based on historical traffic patterns. *Computer Sciences and Applications (CSA), 2013 International Conference, pp. pp. 345–349*, 2013.

[35] P. Zhou, Y. Zheng, and M. Li. How long to wait?: predicting bus arrival time with mobile phone based participatory sensing. *Proceedings of the 10th international conference on Mobile systems, applications, and services, pp. pp. 379–392*, 2012.

[36] Y. Zhou, L. Yao, Y. Chen, Y. Gong, and J. Lai. Bus arrival time calculation model based on smart card data. *Transportation Research Part C: Emerging Technologies, pp. vol. 74, pp. 81–96*, 2017.

[37] T. Zhu, F. Ma, T. Ma, and C. Li. The prediction of bus arrival time using global positioning system data and dynamic traffic information. *Wireless and Mobile Networking Conference (WMNC), 2011 4th Joint IFIP, pp. pp. 1–5*, 2011.

[38] M. Zorkany, M. Zaki, I. Ashour, and B. Hisham. Online bus arrival time prediction using hybrid neural network and kalman filter techniques. *International Journal of Modern Engineering Research, pp. vol. 3, no. 4, pp. 2035–2041*, 2013.

Apêndice A

Outro(s) Apêndice(s)

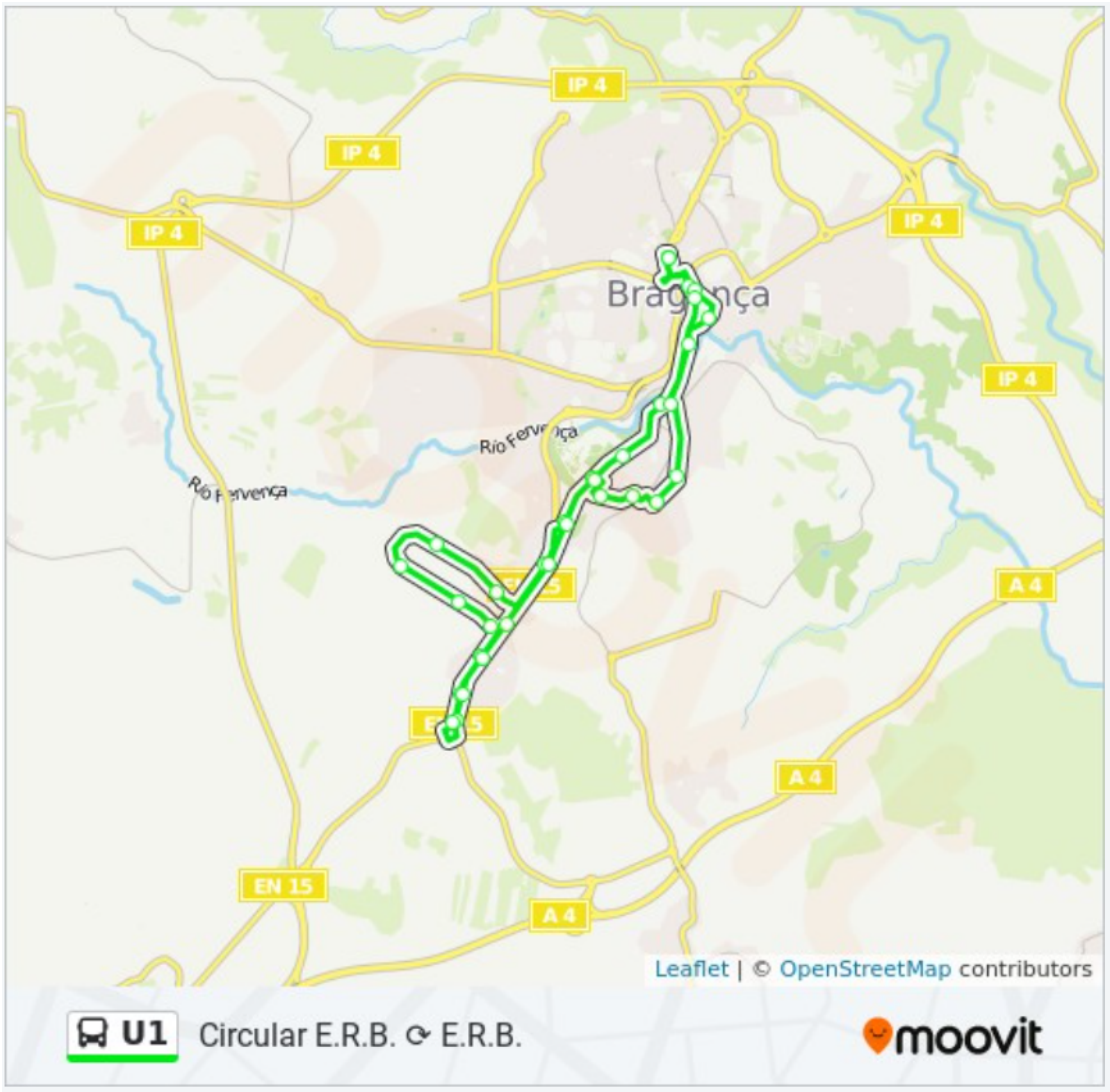

Figura A.1: Trajeto de viagem da linha U1 (Ida e volta) [24]

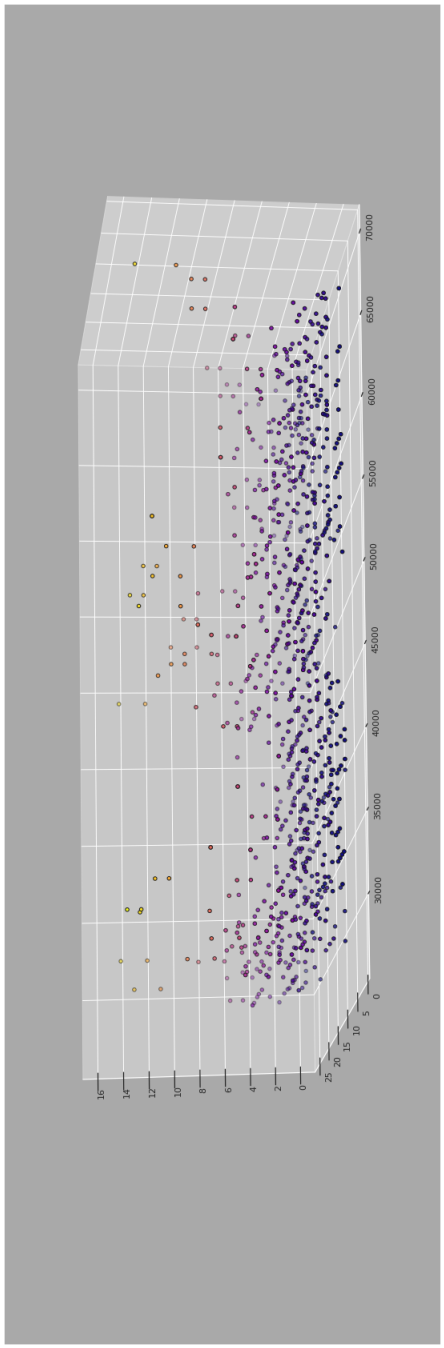

Figura A.2: Representação 3D das 1518 amostras

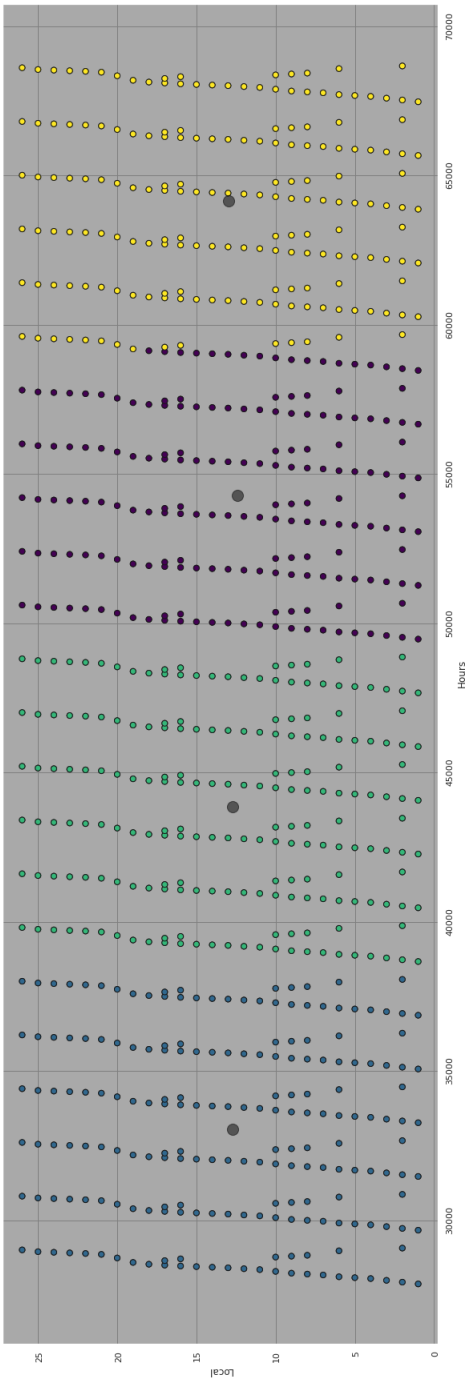

Figura A.3: Clusters do método de *K-means* da Experiência Nº2.1

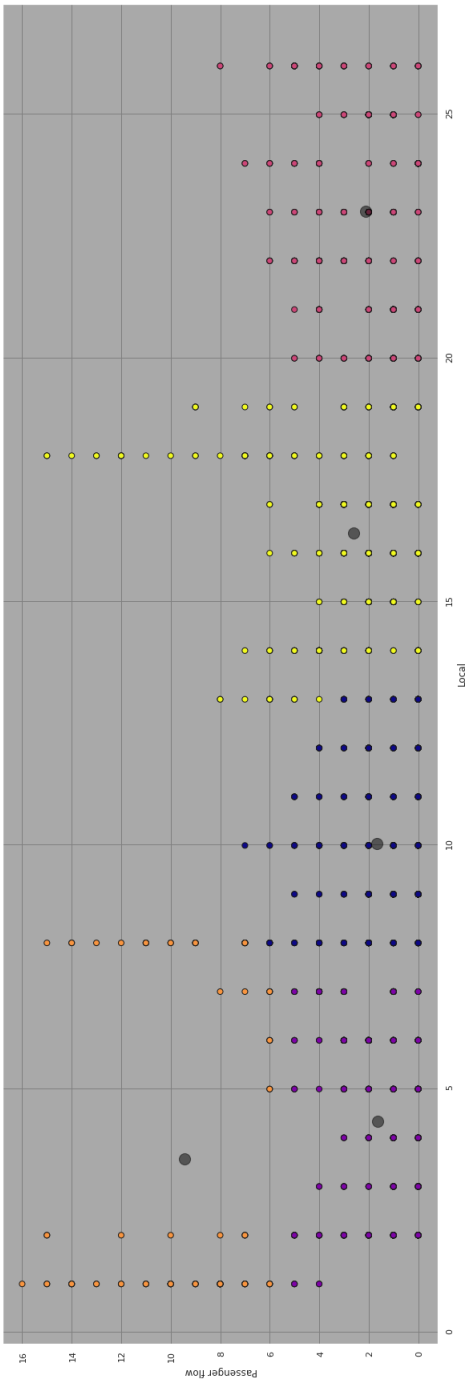

Figura A.4: Clusters do método de *K-means* da Experiência Nº2.2