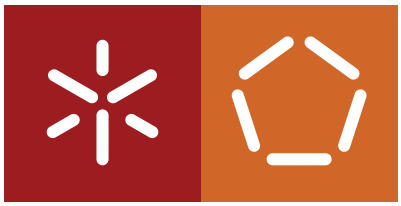
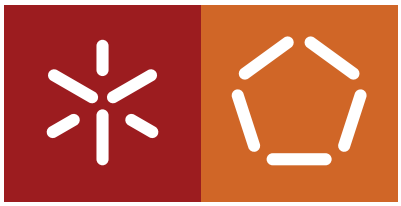


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Gabriela Sá Martins

Caracterização de Tráfego de Serviços de Streaming em Dispositivos Móveis

dezembro de 2021



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Gabriela Sá Martins

Caracterização de Tráfego de Serviços de Streaming em Dispositivos Móveis

Dissertação de Mestrado
Mestrado Integrado em Engenharia Informática

Trabalho efetuado sob a orientação do(a)
Prof. Doutor Paulo Manuel Martins de Carvalho
Prof. Doutora Maria Solange Pires Ferreira Rito Lima

dezembro de 2021

AGRADECIMENTOS

Em primeiro lugar, agradecer aos meus orientadores, Prof. Doutor Paulo Manuel Martins de Carvalho e Prof. Doutora Maria Solange Pires Ferreira Rito Lima, por todo o acompanhamento, pela enorme disponibilidade e por todo o conhecimento que generosamente partilharam comigo.

A toda a minha família, um grande obrigada. Aos meus pais serei eternamente grata por me permitirem seguir o meu caminho, apoiando-me e fornecendo-me todas as ferramentas necessárias. Aos meus irmãos, obrigada por me fazerem rir até nos momentos de maior tensão.

Aos meus colegas de curso que se tornaram amigos, agradecer o companheirismo neste caminho que fizemos juntos. Sem vocês, estes 5 anos não teriam sido a mesma coisa.

Aos meus grandes amigos, os que me acompanham há anos, muito obrigada por todas as palavras diárias de força.

Esta dissertação é o culminar de uma longa jornada, que representa uma enorme conquista para mim. Sem o incentivo de todas as pessoas que preenchem a minha vida, não seria possível. Muito obrigada a todos.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico.

Confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e respeitei o Código de Conduta Ética da Universidade do Minho.

ABSTRACT

In the current context, continuous technological development allows easy and fast access to a wide range of services and platforms, through mobile devices. For this reason, the volume and diversity of traffic have grown exponentially as well.

Knowledge of the traffic circulating in current networks is essential, either to help improving the management and configuration of the network elements and services, or for users to have the opportunity to better manage resources when using their mobile devices and applications.

Thus, this work intends to deepen the study of the characteristics of the traffic generated by mobile devices when accessing certain platforms/services. It is also expected to include a Machine Learning component to predict the user experience. In addition, it is intended to obtain a basis for comparison between the web and application versions of the same service.

KEYWORDS Computer Networks, Traffic Analysis, Machine Learning

RESUMO

No contexto atual, o contínuo desenvolvimento tecnológico permite um acesso fácil e rápido a variadíssimos serviços e plataformas, por meio de dispositivos móveis. Por este motivo, o volume e diversidade de tráfego tem crescido de forma exponencial também.

O conhecimento do tráfego que circula nas redes atuais torna-se indispensável, seja para ajudar a melhorar a gestão e configuração dos elementos e serviços de rede, seja para os utilizadores terem a oportunidade de gerir melhor os recursos na utilização dos seus dispositivos móveis e aplicações.

Assim, este trabalho pretende aprofundar o estudo das características do tráfego gerado por dispositivos móveis, no acesso a determinadas plataformas/serviços. Também se espera incluir uma componente de *Machine Learning* para previsão da experiência do utilizador. Além disso, pretende-se obter base de comparação entre as versões *web* e aplicacional de um mesmo serviço.

PALAVRAS-CHAVE Redes de Computadores, Análise de Tráfego, *Machine Learning*

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÓNIMOS

AP *Access Point*
ANN *Artificial Neural Network*

CPU *Central Processing Unit*

DBMS *Data Base Management System*
DNS *Domain Name System*
DoS *Denial of Service*

FTP *File Transfer Protocol*

HAS *HTTP Adaptive Streaming*
HTTP *Hypertext Transfer Protocol*

IPDV *IP Packet Delay Variation*
ISP *Internet Service Provider*

LightGBM *Light Gradient Boosting Machine*

MAE *Mean Absolute Error*
ML *Machine Learning*
MLP *Multilayer Perceptron*
MOS *Mean Opinion Score*
MSE *Mean Squared Error*

OWD *One Way Delay*

PR *Precision-Recall*

QoS *Quality of Service*
QoE *Quality of Experience*
QUIC *Quick UDP Internet Connection*

ROC *Receiver Operating Characteristic*
RTT *Round Trip Time*

SNR *Signal-to-Noise Ratio*

TCP *Transmission Control Protocol*

UDP *User Datagram Protocol*

VoD *Video on Demand*

XGBoost *EXtreme Gradient Boosting*

Parte I

1

INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO E MOTIVAÇÃO

Atualmente, vive-se num contexto onde, através de um simples clique, tem-se à disposição um enorme e variado leque de serviços. Segundo Cisco (2020), estima-se que até 2023, cerca de 46% da população mundial possua acesso a um *smartphone*. O crescimento exponencial que se tem observado a nível tecnológico, tende a continuar a aumentar e é notório o impacto que o mesmo terá na população, abordado em Gadzama et al. (2019).

Estes aspetos referidos refletem-se no enorme e diversificado volume de tráfego gerado a todo o momento. Consequentemente, o seu estudo e caracterização tornam-se fundamentais em vários aspetos.

Primeiramente, o conhecimento do tráfego que efetivamente circula nas redes de computadores permite, aos respetivos servidores, uma gestão e configuração otimizada da mesma, oferecendo vantagens no seu uso.

Por outro lado, é importante que os próprios utilizadores de dispositivos móveis tenham a possibilidade de usufruírem da melhor forma da sua rede. Nem sempre há a possibilidade de se conectarem a uma rede *Wi-Fi* para aceder à Internet, sendo que a localização e a capacidade dos APs (*Access Point*) podem condicionar este acesso, obrigando ao uso dos próprios dados móveis.

Cada vez mais, a utilização de *smartphones* tem-se afirmado em detrimento do uso de um PC *Laptop*, algo que obrigou o mercado a evoluir e apresentar melhores soluções para os seus produtos. Daqui, resultou então numa migração e adaptação das plataformas, inicialmente disponibilizadas apenas em versão *browser*, para versão aplicacional, pressupondo melhorias para os utilizadores, desde comodidade a qualidade.

Reunindo estes aspetos, torna-se indiscutível que conhecer detalhadamente o tráfego gerado pelos dispositivos móveis, no acesso a serviços e plataformas, é de extrema importância e deve ser alvo de estudos aprofundados.

1.2 OBJETIVOS

Os principais objetivos do presente projeto de mestrado centram-se na caracterização de tráfego real, gerado por utilizadores no acesso a serviços e aplicações atuais, através dos seus dispositivos móveis.

Para tal, decorrerá um estudo sobre mecanismos de captura de tráfego em dispositivos móveis, que permita o desenvolvimento de uma arquitetura de captura. Os métodos a utilizar devem ser adaptados aos casos de estudo, permitindo uma recolha fiel de dados.

As métricas a considerar também serão alvo de análise, de forma a proporcionar uma caracterização detalhada do serviço, servindo de base de comparação entre a utilização de versões aplicacionais face a *web*. Além disso, será necessário adequar as métricas em uso aos serviços a analisar.

Espera-se introduzir a utilização de técnicas de ML (*Machine Learning*), através de um modelo de previsão, que permitam automatizar a classificação do tráfego de rede, ao nível da qualidade de experiência do utilizador final. Para tal será necessário um vasto conjunto de dados.

O estudo engloba análises a todos os resultados obtidos, a partir dos processos de recolha anteriormente criados, que abranjam a interação entre servidores e clientes, bem como a obtenção de variadas informações sobre o fluxo de tráfego.

Espera-se que, no final do projeto de mestrado, seja possível efetuar uma descrição detalhada do tráfego que é gerado nos acessos realizados aos serviços/aplicações alvo de estudo.

Também se pretende elucidar os utilizadores de dispositivos móveis sobre aspetos de utilização mais vantajosos, decorrentes de possíveis comparações que poderão surgir entre resultados obtidos, relativamente ao uso da versão aplicacional e *web* de um mesmo serviço.

1.3 ESTRUTURA DO DOCUMENTO

Este documento apresenta sete capítulos.

No Capítulo 1 são expostas as principais motivações para o desenvolvimento do projeto, bem como os objetivos a cumprir e resultados que se esperam obter no final do mesmo.

No Capítulo 2 é abordado o estado de arte, onde se integram estudos e conceitos fundamentais no desenvolvimento desta investigação. Particularmente, a monitorização de tráfego, tipos de medições para recolha de dados, análise de métricas e caracterização de tráfego. É também introduzida alguma informação sobre ML e exemplos de como a sua utilização na área de redes tem sido realizada.

O Capítulo 3 apresenta a arquitetura que a utilizar para realização do projeto, fundamentada com a informação recolhida no estado de arte. Aqui também são apresentadas as ferramentas a utilizar na captura e análise de tráfego.

No capitulo 4 é descrito o modelo de previsão desenvolvido no decorrer deste trabalho. Engloba-se também uma explicação dos dados a utilizar, ferramentas, técnicas de ML e resultados do desempenho do modelo.

O Capítulo 5 descreve o serviço que será alvo de estudo e aspetos a ter em conta no mesmo. Aborda também o processo de recolha de tráfego, seja na versão aplicacional ou *browser*, e algumas métricas a analisar. Por fim, são incluídos os conteúdos a analisar em particular.

No capítulo 6 são apresentados os resultados da análise das capturas de tráfego efetuadas e algumas constatações sobre as mesmas. É também explicada a forma de utilização do modelo de ML desenvolvido para

fazer uma previsão para as capturas. Por fim, é incluída uma síntese comparativa da versão aplicacional com a versão *browser*.

Finalmente, o Capítulo 7 reúne conclusões finais sobre o projeto e, ainda, trabalho futuro a desenvolver.

2

TRABALHO RELACIONADO

O presente capítulo aborda o estudo inicialmente desenvolvido, que servirá de base para a conceção do projeto. Serão abordadas questões relativas à monitorização e caracterização de tráfego, tipos de medições e parâmetros de análise da qualidade de serviço e qualidade de experiência. Incluem-se também modelos de *Machine Learning*, e estudos que demonstram como estes podem ser englobados na análise de rede.

2.1 MONITORIZAÇÃO DE TRÁFEGO

Observando os valores presentes no relatório Cisco (2020), conclui-se que os mesmos refletem a grande evolução tecnológica, despoletada nos últimos anos e com tendência crescente. Por exemplo, se em 2018, 51% da população mundial possuía acesso à Internet, em 2023 este valor chegará a 66%.

Tal como abordado no artigo Callado et al. (2009), o enorme aumento do volume de tráfego gerado a todo o momento, levanta várias dificuldades, principalmente aos ISP's e administradores de redes. Estes têm que apresentar capacidade para suportar tal consumo de recursos, bem como lidar com o impacto do diferente tráfego gerado por novas aplicações, de forma eficiente.

Assim sendo, as estratégias de medição são fortes aliadas para se alcançarem estes objetivos, fornecendo o melhor serviço em relação ao custo mais benéfico. A sua importância deve-se, essencialmente, ao facto de serem uma ferramenta indispensável na área de redes, pois permitem a identificação de situações/comportamentos anómalos, como ataques de *DoS (Denial of Service)*, tráfego indesejado ou oscilações do mesmo, entre outros. Além disso, são fundamentais em vários outros aspetos, como a introdução de novos modelos de tráfego, distribuição do mesmo e otimização da capacidade da rede.

Tendo em conta estes pontos, torna-se evidente que a caracterização de tráfego fornece informação preciosa para aprimorar a gestão e operação da rede, para além de facilitar processos de diagnóstico ou deteção de anomalias na mesma. Por estes motivos, cada vez mais se tem dado atenção a este tópico, através da aplicação de maior esforço nos estudos sobre o tráfego que circula nas redes atuais, bem como no desenvolvimento de tecnologias que facilitem estes processos. Um exemplo disso é a tecnologia apresentada no artigo Song (2012), que consiste no desenvolvimento de um sistema de monitorização de tráfego, baseado em *sockets*.

2.1.1 *Tipos de Medições*

De acordo com Williamson (2001), ao efetuar medições de tráfego e para realizar o estudo do mesmo, existem alguns pontos essenciais a considerar.

Primeiramente, é importante ponderar se a análise deve ser realizada ***Online*** ou ***Offline***. No que concerne às medições *online*, as mesmas são efetuadas em tempo real, ou seja, durante a recolha dos dados. Por outro lado, no caso de medições *offline*, em tempo real dá-se a recolha e armazenamento de dados. Quando findada, procede-se à análise dos mesmos.

Outro aspeto relevante, também abordado em Williamson (2001), é a utilização de medição **ativa**, **passiva** ou **híbrida**. As mesmas serão apresentadas detalhadamente de seguida, de acordo com o artigo Mohan et al. (2011).

Medição Ativa

As medições ativas consistem na injeção de tráfego artificial na rede, recorrendo ao uso de pacotes de prova. Estes pacotes são enviados continuamente pela rede, com o objetivo de obter informação comportamental sobre a mesma, bem como conhecer as suas propriedades. Este tipo de medição destina-se mais a casos de deteção de vulnerabilidades, testes de performance, averiguar valores de atrasos na rede, entre outros.

É importante sublinhar que as medições ativas podem ter um impacto negativo na operação da rede, na medida em que é gerado tráfego adicional. Além disso, os pacotes são enviados independentemente do comportamento dos utilizadores, pelo que os resultados podem não representar estas interações. Por estes motivos, o uso deste tipo de medições deve ser cuidadoso e adequadamente planeado.

Medição Passiva

As medições passivas diferenciam-se por não criarem tráfego adicional na rede, procedendo-se à recolha de dados reais, ou seja, não injetados para fins de medição. A utilização deste tipo de medição aplica-se mais a casos onde se pretende obter informações sobre a utilização de largura de banda, distribuição de protocolos em uso, perda de pacotes, entre outros. Nestes casos, a medição passiva tende a ser mais precisa.

Apesar dos métodos passivos fornecerem uma representação real da rede, por não perturbarem o seu fluxo normal, a quantidade de dados recolhida e armazenada pode ser bastante elevada, também dependendo do números de pontos de captura utilizados na rede. Por este motivo, técnicas de compressão, filtragem ou amostragem, devem ser consideradas. Além disso, a enorme quantidade de dados pode dificultar análises efetuadas *online*.

De acordo com Callado et al. (2009), dentro das medições passivas, devem considerar-se dois níveis:

- **Medição ao nível do fluxo (Alto nível)** - Neste caso, o estudo passa por analisar, de forma geral, fluxos de tráfego na rede, sendo que um fluxo deve ser visto como a agregação de pacotes que partilham entre si os endereços/portas de origem e destino. É uma medição mais orientada à análise de portas de origem/destino, número de pacotes, duração do fluxo, entre outros.

- **Medição ao nível do pacote (Baixo Nível)** - Contrariamente, quando se trata deste tipo de medição, o foco é inspecionar o conteúdo do próprio pacote. Com o decorrer do tempo, este tipo de análise pode gerar uma quantidade de dados insustentável, pelo que deve ser considerado o uso de um DBMS (*Data Base Management System*). Normalmente, as técnicas utilizadas nestes casos baseiam-se na captura e análise dos cabeçalhos dos pacotes.

  Algumas métricas abrangidas por medição ao nível do pacote são o tamanho do mesmo, os protocolos aplicados, bem como endereços e portas de origem/destino.

Medição Híbrida

A medição híbrida caracteriza-se pela conjunção dos dois tipos de medição apresentados anteriormente - ativa e passiva. Em Mohan et al. (2011) é apresentado um exemplo de medição híbrida, cujo objetivo é avaliar os *delays* intermediários e *end-to-end*. Para isso, a medição integra a injeção de tráfego através de pacotes *probe* (ativa), e posterior monitorização do percurso dos mesmos pela rede (passiva).

2.1.2 *Volume de Dados e Amostragem*

Segundo Callado et al. (2009), no que concerne à análise de tráfego, as medições passivas podem recolher um volume de dados excessivo. Mesmo recorrendo a um DBMS, a utilização de técnicas de amostragem é necessária para se proceder a análises e estudos de forma mais eficiente, seja ao nível dos fluxos ou dos pacotes. Através da aplicação destes métodos, torna-se possível diminuir os requisitos de armazenamento e *hardware*, bem como os custos inerentes ao processamento de dados.

Obviamente, com a implementação deste tipo de procedimentos, torna-se iminente uma perda de informação resultante da redução de dados. Assim, estas técnicas devem ser aplicadas de forma inteligente, permitindo que a partição dos dados selecionados transmitam observações e conhecimentos reais sobre a rede, de forma escalável.

No artigo Duffield (2004) são detalhados vários métodos, utilizados nas medições, que permitem uma redução significativa de dados. De acordo com o mesmo, destacam-se os que se seguem.

Agregação

Tal como o nome indica, esta técnica baseia-se na combinação de dados, normalmente de forma aditiva. Ou seja, a agregação é utilizada quando se pretende obter informações mais compactas de acordo com certa característica, pressupondo que as outras componentes dos dados agregados possam perder a sua visibilidade. Um exemplo da utilização de agregação é quando se quer averiguar qual o total de tráfego proveniente de determinada fonte. A combinação de dados decorrerá de acordo com a origem do tráfego, sendo o resto dos atributos dispensáveis.

Filtragem

Esta técnica traduz-se na seleção dos dados que correspondem aos valores desejados. Ou seja, de acordo com determinado parâmetro é efetuada uma filtragem, sendo descartados os dados que não integram o conjunto pretendido. Este método é bastante utilizado quando se tenciona obter o tráfego proveniente de uma determinada fonte, ou com certa característica, protocolo, entre outros.

Amostragem

Como mencionado anteriormente, as técnicas de amostragem consistem na seleção de tráfego conforme o tipo que estiver em uso, sendo o resto dos dados descartados. As principais técnicas de amostragem são:

- **Sistemática**

  De forma simples, esta técnica consiste na recolha do primeiro pacote de um bloco de tráfego, associado ao tamanho da amostragem determinada. Uma implementação pode ser através de um contador, definido para um valor determinado. Por cada pacote, esse contador é decrementado e, quando atinge zero, é selecionado o primeiro pacote da amostra, reiniciando-se o processo.

  É importante ter em conta que este método pode ser tendencioso. Por exemplo, se existir algum padrão na rede que se relacione com o tamanho de amostragem definido, será sempre selecionado um pacote com as mesmas características, quando há variabilidade.

- **Aleatória**

  Esta estratégia pode colmatar a fragilidade da amostragem sistemática, apresentada anteriormente. Apesar de seguir a mesma linha de funcionamento, é utilizada uma variável aleatória para selecionar o pacote da amostragem, implicando um custo computacional maior.

- **Estratificada**

  Esta técnica opera de forma mais complexa, sendo necessário algum conhecimento sobre o tráfego. Os dados são divididos por grupos, designados estratos, de acordo com determinado atributo comum. Posteriormente, são selecionadas amostras de cada estrato, de forma aleatória ou não, dependendo do tipo de classe de entre as existentes nas amostragens estratificadas Kamienski et al. (2005).

  Uma das principais vantagens deste método é que permite uma representação mais real do tráfego, reduzindo significativamente o volume de dados e trazendo benefícios ao nível do processamento dos mesmos.

2.2 CARACTERIZAÇÃO DE TRÁFEGO

A caracterização de tráfego tem sido alvo de vários estudos, desempenhando cada vez mais um papel fundamental na área de redes de computadores. No âmbito deste projeto de mestrado, destaca-se a análise do

tráfego a dois níveis: QoS (Qualidade de Serviço) e QoE (Qualidade de Experiência). Nesta secção, serão abordados ambos os conceitos, bem como estudos relacionados e as principais métricas associadas.

2.2.1 *Qualidade de Serviço e Métricas*

A QoS é fundamental tanto para os utilizadores, como para os operadores, na medida em que reflete, de uma forma geral, qual a *performance* da rede ao prestar o serviço. Neste sentido, existem métricas relevantes que permitem inferir qual a qualidade do serviço. Além disso, cada vez mais têm sido apresentadas ferramentas de monitorização de QoS, como é o caso da exposta em Zeng et al. (2007), onde se efetua uma avaliação da QoS para serviços *web*.

Serão agora discutidos alguns artigos desenvolvidos no âmbito desta temática, onde são debatidas várias métricas.

Métricas de QoS

No artigo Hanemann et al. (2006) foram expostas algumas métricas relativas ao desempenho das redes, bem como uma análise para compreender as suas composições. Foram ainda avaliadas ferramentas relacionadas com estas medições, por exemplo perfSONAR. Segundo o mesmo, vários investigadores, utilizadores e operadores de redes tirariam grandes vantagens em conhecer métricas de desempenho da rede, permitindo otimizar a utilização da mesma, bem como corrigir uma possível deterioração do sinal.

Ao longo da pesquisa, os autores selecionaram métricas que consideram ser de enorme relevância no que diz respeito à avaliação do desempenho da rede, sendo divididas e descritas como:

- **Disponibilidade** - Consiste em observar a percentagem de tempo em que a rede está funcional, sem problemas relevantes, permitindo averiguar o quão robusta esta é.
- **Perda e Erro** - Traduz-se na medição da fração de pacotes perdidos ou que contenham *bits* errados. Essencialmente, revela condições de congestionamento da rede, possíveis erros de transmissão, entre outros.
- **Atraso/Latência** - Os autores englobam neste grupo métricas como ***OWD (One Way Delay)***, ***RTT (Round Trip Time)*** e ***IPDV (IP Packet Delay Variation)***, que traduzem, respetivamente, o atraso entre uma origem e um destino, o tempo de ida e volta entre eles, e variação no atraso entre pacotes consecutivos.

  Estas métricas permitem obter informação, por exemplo, em relação às condições de congestionamento no acesso à rede, ou sobre o efeito de alterações de *routing* efetuadas na mesma.
- **Largura de banda** - Por fim, esta métrica diz respeito à quantidade de dados máxima, normalmente em *bits*, que pode ser transmitida entre uma origem e um destino por unidade de tempo.

De acordo com Hanemann et al. (2006), existem outras questões que devem ser avaliadas na tentativa de decifrar causas para degradação do serviço fornecido, sendo elas *CPU (Central Processing Unit) load*, o consumo de memória, o sobreaquecimento dos dispositivos em utilização no acesso à rede, entre outras.

No artigo Gabriel Omomule et al. (2019), o enorme volume de dados gerado atualmente e o impacto que isso tem nos serviços e no desempenho dos mesmos, motivou a uma análise da usabilidade de redes *wireless* baseada na QoS. Para tal, os autores recorreram à recolha de uma enorme quantidade de pacotes, através da ferramenta Wireshark. Posteriormente, foram aplicadas métricas selecionadas para a análise, sendo elas correspondentes às descritas anteriormente do artigo Hanemann et al. (2006), às quais se adicionou:

- ***Throughput*** - Equivale à quantidade de dados efetivamente transferida (enviada ou recebida) por unidade de tempo.

Neste mesmo estudo foram ainda recolhidos dados como o **número total de pacotes** e a **distribuição de protocolos em uso**.

De outro ponto de vista, o artigo Chowdhury et al. (2008) apresenta uma ordem de prioridades para os parâmetros de QoS, de acordo com os serviços e os níveis protocolares em utilização. Parte desta informação é resumida na Tabela 1.

Camada / Serviço	*Application Layer*	*Network Layer*
Voz	Taxa de Chamadas Perdidas, OWD, *Jitter*, Taxa de Conclusão de Chamadas, Acessibilidade do Serviço.	Taxa de Sucesso de *Handover*, *Jitter*, *Handover Delay*, *Delay* Máximo de Transferência, *Bit Rate* Garantido e Máximo, Acessibilidade da Rede, Taxa de Erro/Perda de Pacotes.
Vídeo	Taxa de Chamadas Perdidas, OWD, Atraso de Sincronização Labial, Resolução, *Jitter*, Tempo de Resposta da Aplicação, Taxa de Conclusão de Chamadas, Acessibilidade do Serviço, Taxa de Erro/Perda de Pacotes.	Taxa de Sucesso de *Handover*, *Delay* Máximo de Transferência, *Handover Delay*, *Jitter*, *Bit Rate* Garantido e Máximo, Acessibilidade da Rede, Taxa de Erro de *Frame*, Capacidade e Atraso de *Buffering*, Prioridade Tratamento de Tráfego, Prioridade de Alocação/Retenção.
Dados	Taxa de Chamadas Perdidas, Taxa de Erro/Perda de Pacotes, Taxa de Transferência de Dados, *Web Browser/ End-to-End* OWD, *Jitter*.	Taxa de Sucesso de *Handover*, Taxa de Erro de *Frame*, Taxa de Transferência de Dados, Acessibilidade da Rede, *Bit Rate* Máximo, *Handover Delay*.

Tabela 1: Lista (ordem prioridade) de parâmetros de QoS por serviço Chowdhury et al. (2008).

Como se pode observar, existem métricas comuns a vários serviços, também referidas nos artigos anteriores, como é o caso de OWD, *Jitter*, Taxa de Erro/Perda de Pacotes, entre outros.

No entanto, existem ainda outros aspetos a ter em consideração quando se aborda análises de tráfego.

No artigo Mongkolluksamee et al. (2016), é apresentado um estudo que pretendia combinar a distribuição do tamanho de pacotes com padrões de comunicação extraídos, com o objetivo de identificar aplicações móveis. As técnicas foram aplicadas para serviços bastante atuais - *Facebook, Line, Skype, YouTube* e *Web*. Neste caso, foi abordada a importância dos **protocolos**, particularmente TCP (*Transmission Control Protocol*) e UDP (*User Datagram Protocol*), incluindo informações como:

- **Distribuição de portas em uso**, particularmente portas destino;
- **Número total de *bytes***;
- **Número total de pacotes**;
- **Número de fluxos de tráfego**;
- **Número de pacotes por fluxo**.

Reunindo os vários aspetos apontados, conclui-se que existem várias métricas a ter em consideração para se proceder a uma análise real e fiel do tráfego da rede, sendo que o tipo de serviço deve ser um fator chave na decisão das mesmas.

2.2.2 *Qualidade de Experiência e Métricas*

Quando se avalia o desempenho de um serviço, além da QoS, é crucial ter em conta a Qualidade de Experiência (QoE), um conceito mais ligado à interação com os utilizadores. Sendo a QoS relativa a questões mais técnicas, QoE é algo mais subjetivo, podendo variar entre utilizadores nas mesmas condições.

No artigo Liotou et al. (2016), apresenta-se uma vista geral sobre as métricas de QoE, bem como o mapeamento existente entre QoS e QoE. Segundo o mesmo, com a nova geração de utilizadores móveis, dá-se mais ênfase à perceção da qualidade de um serviço pelos utilizadores finais. Ao longo do artigo, os autores apresentaram atributos que consideram ter influência, no que concerne à QoE. Os mesmos condensam-se nas Tabelas 2 e 3.

Aspeto	Fatores de Influência de Qualidade
Transport/Network Layer	RTT, OWD, *Jitter*, Taxa de Perda de Pacotes, Período de Congestionamento, Tamanho do Pacote, Distribuição de Perda/Atraso de *Burstiness*.
Physical Layer	SNR, Taxa de Transferência, *Bottleneck* da Largura de Banda, *Bit Rate*, Probabilidade de Interrupção, Probabilidade de Erro do Pacote/*Bit*.
Fatores do Equipamento	*Codec*, Características do *Buffer* de *Dejittering*, Deteção de Atividade de Voz (VAD), Cancelamento de Eco/ Supressão de Ruído, Algoritmo de Ocultação de Perda de Pacote.
Fatores Comuns	Políticas e Custo de Cobrança, Suporte de Serviço, Privacidade, Segurança, Fidelidade, Usabilidade, Precisão, Eficiência, Contexto de uso, Nível de Ruído do Ambiente e Variação, Conforto.

Tabela 2: Fatores de influência na QoE segundo Liotou et al. (2016) - Independente do Serviço.

Aspeto	Fatores de Influência de Qualidade
Vídeo	Taxa de *Frames*, Taxa de *bits* do Vídeo, Conteúdo, Visibilidade de Perda de Pacotes, Sincronização de Vídeo e Áudio, Especificações do Monitor.
VoD (*Video on Demand*)	*Streaming* de Vídeo: Número e Duração de Eventos de *Stalling*, Duração Total do Vídeo, Atraso Inicial. HAS: Frequência/Altitude dos *Switches*, Tamanho do *Chunk/Buffer*.
Serviços de *Download*	*Web-browsing*: Tempo de *Download* da Página *Web*. FTP: Taxa de Dados, Tempo de *Download* do Ficheiro, Sincronização de Entrega.
Voz	Taxa de Perda de Pacotes, *Delay*, *Codec*, Taxa de Sucesso/Tempo de Configuração da Chamada, Tempo de Inicialização/Resposta.

Tabela 3: Fatores de influência na QoE segundo Liotou et al. (2016) - Dependente do Serviço.

Da informação acima, torna-se percetível que vários fatores, considerados pelos autores como influentes em QoE, estão relacionados ou coincidem com as métricas de QoS anteriormente explicitadas.

No entanto, para se efetuar uma avaliação de QoE quantitativamente existem duas metodologias principais que, de acordo com Liotou et al. (2016), são:

- **Testes Subjetivos**

  Este primeiro tipo de classificação consiste na avaliação direta de utilizadores na vida real, ou seja, acaba por estar associada à opinião de cada um na experiência de utilização do sistema, independentemente do tipo de serviço em questão. Assim, os utilizadores podem efetuar uma avaliação comparativa ou em escala absoluta. Neste caso, os resultados são influenciados por fatores relativos como a opinião, experiências anteriores, expectativas, perceção/descrição do utilizador em causa e, no geral, acabam por refletir a satisfação na utilização do sistema.

  Este tipo de testes são considerados dos mais confiáveis quando realizados com utilizadores imparciais, obviamente pressupondo uma boa preparação e controlo para a prática dos mesmos. No entanto, são caros e demorados.

- **Estimativa Objetiva de QoE**

  Para colmatar os problemas que surgem com os testes subjetivos, foram desenvolvidos métodos que avaliam a QoE sem a participação de utilizadores finais. Assim, os modelos objetivos são classificados, pelo artigo, usando critérios como:

  – **Utilização do Tráfego de Referência** - Diz respeito a identificar se o tráfego original, ou parte dele, é necessário para a estimativa de QoE;

  – **Modo do Modelo** - Sinaliza a injeção, ou não, de tráfego de teste no sistema a testar, podendo ser intrusivo/ativo ou não intrusivo/passivo;

  – ***Timeframe* do Modelo** - Avaliação *offline* (antes/depois do serviço) ou *online* (em serviço);

  – **Finalidade de Uso** - Por exemplo, planeamento/otimização da rede, monitorização do serviço em tempo real, entre outros;

  – **Método de Avaliação** - Dependendo da informação de *input* para efetuar a medição da QoE, pode ser:

    * *Media-layer* (baseado no sinal);
    * *Packet-layer/Bitstream* (baseado no cabeçalho dos pacotes ou dados de *payload*);
    * Planeamento Paramétrico (baseado em métricas da rede).

  Tendo estes aspetos em consideração, para se atribuir uma classificação à QoE, os modelos objetivos utilizam fatores descritos nas Tabelas 2 e 3.

- **Métodos Híbridos**

  São caracterizados por estimar de forma automática e objetiva a QoE. Para tal, são utilizadas, por exemplo, técnicas de *Machine Learning*, cujos modelos são treinados com valores de testes subjetivos, sendo efetuado o mapeamento das métricas da rede para valores MOS (*Mean Opinion Score*) . Posteriormente, os mesmos podem ser aplicados para previsões em tempo real.

*MOS (*Mean Opinion Score*)*

De acordo com Liotou et al. (2016), a escala de MOS, que abrange cinco níveis, é a métrica mais utilizada para pontuar a QoE. Inicialmente era apenas utilizada pelos testes **subjetivos**, onde os utilizadores finais lhe atribuíam um valor, em função da qualidade de experiência. No caso das medições **objetivas**, pode ser conferido um valor diretamente, ou efetuado um mapeamento de uma outra escala.

2.3 MACHINE LEARNING

Atualmente, a enorme quantidade de dados gerada a todo o momento é extremamente elevada, considerando-se esta a era de *big data*. No livro P. Murphy (2012), o autor considera que ML surgiu através da necessidade

de automatizar a análise deste colossal volume de dados. No mesmo, descreve-se ML como um conjunto de métodos capazes de detetar automaticamente determinados padrões nos dados, e posteriormente permitir a sua utilização para efetuar previsões futuras ou tomar determinadas decisões.

2.3.1 *Categorias de Aprendizagem*

É importante referenciar que, na aplicação de algoritmos de ML, existem diferentes tipos de aprendizagem. Os dois principais descrevem-se de seguida, de acordo com Shalev-Shwartz and Ben-David (2014) e P. Murphy (2012).

Aprendizagem Supervisionada

Este tipo de aprendizagem é utilizada quando são fornecidos dados e as suas respetivas *labels*, permitindo efetuar um mapeamento entre *inputs* (x) e *outputs* (y). Assim, durante a fase de treino, o algoritmo possui acesso à resposta correta, podendo efetuar, iterativamente, previsões sobre os dados de treino e corrigir as mesmas, até atingir bom desempenho.

Distinguem-se dois tipos de aprendizagem supervisionada:

- **Classificação** - Aplica-se quando a variável *output target* é categórica, ou seja, pertence a um conjunto finito $y_i \in \{1, ..., C\}$, sendo C o número de classes.

  Quando existem apenas duas classes, trata-se de um problema de **classificação binária**.

  Caso existam mais que duas, chama-se **classificação multiclasse**.

- **Regressão** - Ao contrário do caso anterior, aqui y_i é um valor real, tratando-se, portanto, de uma variável contínua.

Aprendizagem Não Supervisionada

Este tipo de aprendizagem distingue-se da anterior pois os dados de entrada (x) não possuem *labels*. Deste modo, o algoritmo recorre à padronização dos dados para criar conjuntos com características idênticas, uma vez que não estão disponíveis dados de treino que forneçam os *outputs* desejados.

A aprendizagem não supervisionada é considerada mais aplicável do que a supervisionada, uma vez que não necessita de um processo manual de atribuição de *labels*. Assim, na resolução de problemas de aprendizagem não supervisionada, são utilizados algoritmos de *clustering*, que agrupam dados em grupos de acordo com as suas distribuições.

2.3.2 *Modelos de Aprendizagem Supervisionada*

De seguida, descrevem-se alguns algoritmos de ML. Estes foram selecionados por serem muito comuns e populares em aprendizagem supervisionada para classificação, que será a utilizada neste trabalho.

Regressão Logística

Um dos algoritmos mais populares é a regressão logística, que pertence à família dos preditivos lineares.

No caso de uma regressão linear, ajusta-se uma equação linear, normalmente designada linha de regressão, capaz de encontrar uma relação entre variáveis de entrada (x) e uma variável de saída (y), correspondente a um valor contínuo.

Analogamente à regressão linear, a regressão logística é apropriada para tarefas de classificação, ajustando uma função logística para prever a probabilidade de ocorrer um determinado evento. Neste caso, a variável y já é discreta e pode ter dois ou mais valores, representando cada categoria (Shalev-Shwartz and Ben-David (2014)).

Redes Neuronais

Com inspiração no funcionamento dos neurónios no cérebro humano, surgiu o modelo de computação ANN (*Artificial Neural Network*). Uma ANN pode ser descrita como nodos interligados, que representam os neurónios conectados pelas extremidades. Tal como o cérebro humano, espera-se que esta estrutura seja capaz de aprender com os processos e armazenar conhecimento nas conexões entre neurónios. O seu funcionamento consiste na distribuição de processamento por uma densa interligação de pequenas unidades, os então denominadas neurónios.

Existem múltiplas variantes das Redes Neuronais, no entanto, nos casos mais simples, existe um conjunto de neurónios de entrada que são ativados por dados de entrada. Aqui, os dados sofrerão um processamento definido no modelo e a sua transformação será repassada à próxima camada da rede, através de conexões indexadas com um determinado peso. Após se efetuarem as iterações definidas, é ativado o neurónio de saída e termina o processo.

Um dos algoritmos de treino mais conhecidos nas ANNs denomina-se *backpropagation*. Neste caso, os dados de entrada propagam-se até à última camada de acordo com os parâmetros, funções e pesos definidos no modelo. No final deste processo, é calculado um erro resultante da comparação entre o resultado obtido e o desejado, tendo em consideração que se trata de um modelo supervisionado. Após isto, o erro é então retro propagado até à camada inicial, de acordo com funções de alteração dos pesos (Shalev-Shwartz and Ben-David (2014)).

Árvores de Decisão

Tal como o nome indica, este modelo preditivo consiste em prever uma *label* correspondente a um *input* x, através do percurso entre o nodo raiz e uma folha (nó sem filho). Ao longo do processo, para se selecionar qual o nó sucessor, há uma divisão do espaço de entrada, que ocorre de acordo com o valor de alguma *feature* pertencente a x, ou devido a alguma regra previamente estabelecida no modelo, sendo estas associadas a ramos da árvore de decisão. Esta sequência vai sendo repetida até, eventualmente, se atingir o final da árvore, onde estão os nós folha que contém uma *label* específica como resultado (Shalev-Shwartz and Ben-David (2014)).

Modelos Ensemble

Em Krawczyk et al. (2017), apresentam-se modelos *ensemble* para proceder a análises de *data streams*. Segundo os autores, estes modelos são extremamente promissores e eficientes por se tratarem da combinação de outros modelos preditivos. Ou seja, são utilizadas classificações de vários modelos individuais para melhorar o desempenho obtido.

Normalmente recorre-se a modelos *ensemble* para minimizar algumas causas de problemas em modelos de aprendizagem. Uma das principais causas é a variância dos dados. Para isso, uma das técnicas mais utilizadas é denominada ***bagging*** que reduz a variância. Por exemplo, nas árvores de decisão é comum existir algum *overfitting*. Por esse motivo, muitas vezes recorre-se ao algoritmo ***Random Forest***, que consiste em treinar uma série de árvores de decisão, retornando, no final, a classe que a maioria das árvores prever, sobre todas as existentes no conjunto.

A técnica de ***boosting*** é também conhecida por melhorar o desempenho de modelos, principalmente por reduzir tendências que possam existir. Resumidamente, o seu funcionamento passa por utilizar o resultado da sua última classificação para ajustar o peso dessa observação, de modo a minimizar o peso das classificações erradas.

No artigo Shehadeh et al. (2021) são descritos algoritmos que utilizam *boosting*. Dois deles tornaram-se muito populares recentemente, o ***XGBoost*** (*EXtreme Gradient Boosting*) e o ***LightGBM*** (*Light Gradient Boosting Model*). De acordo com os autores, ambos se baseiam na aplicação de um *boosting* de gradiente a árvores de decisão e destacam-se por qualidades como serem escaláveis, precisos e rápidos.

2.4 MACHINE LEARNING E A ANÁLISE DE TRÁFEGO

A área de *Machine Learning* pode ser útil no suporte às redes e serviços de comunicações e, cada vez mais, têm sido apresentados e implementados estudos que cruzam ambas, com diferentes objetivos. Por exemplo, com abordado em Alqudah and Yaseen (2020), modelos de ML têm contribuído para melhorar questões de segurança, uma vez que permitem detetar aspetos relativos a ameaças. Outras aplicações têm surgido para classificação de tráfego e identificação de aplicações através do mesmo - desenvolvido em Zander et al. (2005) e Reza et al. (2017).

Existem também outras abordagens interessantes que evidenciam a conexão entre as duas áreas.

O artigo Aggarwal et al. (2014) apresenta uma implementação de ML focada em estimar QoE de aplicações, através de medições passivas do tráfego. Segundo os autores, é desafiante para os operadores analisar a qualidade do seu serviço, essencialmente, devido à falta de controlo e às complexas camadas de protocolos que existem atualmente, que podem ter impacto negativo na qualidade dos serviços oferecidos. Para o desenvolvimento deste projeto, foram recolhidos dados em cenários controlados com tráfego móvel anónimo. O *dataset* de treino possui o tráfego gerado pela aplicação e, ainda, a métrica de QoE da mesma, fornecendo as ferramentas necessárias ao modelo para a aprendizagem. Neste caso, foram analisados serviços de VoD e VoIP, sendo as métricas da QoE a estimar correspondentes ao número de paragens do vídeo e MOS.

O artigo Zelasko et al. (2020) recorre também a métodos de ML para avaliar a qualidade de transmissão. Neste caso, foram utilizados como *input* quatro parâmetros de QoS, que funcionam como fatores chave na classificação, sendo eles o atraso, *jitter*, largura de banda e taxa de perda de pacotes. Além disso, foi recolhido um *dataset* com a experiência de utilização por um grupo de teste.

O estudo efetuado em Casas et al. (2017) apresenta uma abordagem para a mesma temática, mas centrada nas redes celulares móveis, com medições ao nível do *smartphone*. Para tal, foi efetuada uma recolha de dados, através de medições passivas ao tráfego gerado pelo *smartphone*. A par disto, foram obtidas opiniões de utilizadores sobre a QoE. Assim, tornou-se possível desenvolver e treinar modelos de ML, que permitissem prever a experiência geral do utilizador.

Em todos os estudos acima mencionados, foram utilizadas técnicas de aprendizagem supervisiona, uma vez que os *datasets* recolhidos correspondiam aos parâmetros necessários para tal.

Com os exemplos descritos, torna-se percetível que o desenvolvimento de técnicas e modelos de ML podem contribuir significativamente para melhorar, caracterizar e prever aspetos relativos a redes.

2.5 SUMÁRIO

No presente capítulo foram abordados estudos relacionados com o trabalho em desenvolvimento, bem como analisados e expostos conceitos teóricos importantes nas áreas.

Inicialmente abordaram-se alguns tópicos relacionados com a monitorização de tráfego, como tipos de capturas e técnicas de amostragem. Posteriormente, vários conceitos relacionados com a caracterização de tráfego foram analisados, divididos nas categorias de QoS e QoE.

De seguida, iniciou-se um estudo de ML onde se englobaram os diferentes tipos de aprendizagem e alguns modelos pertencentes a cada uma delas.

Finalmente, concluiu-se com a apresentação de alguns estudos que demonstram pontos interessantes no uso de técnicas de ML na área de redes, nomeadamente na análise de tráfego.

Deste modo, este capítulo fundamentou os passos que se seguiram neste trabalho, apresentados nos próximos capítulos.

Parte II

3

ARQUITETURA DE CAPTURA

Neste capítulo será descrita a arquitetura que será implementada para a recolha de tráfego, a par das ferramentas a utilizar para auxiliar a captura e análise de tráfego. Além disso, serão expostas as características que serão avaliadas, para se proceder à caracterização do tráfego, e especificados os dispositivos a utilizar.

3.1 METODOLOGIA DE MONITORIZAÇÃO

Relembrando que o principal objetivo do projeto passa por caracterizar tráfego gerado por aplicações móveis, é necessário utilizar um **Dispositivo Móvel**. Com a análise efetuada no Capítulo 2, é percetível que a necessidade de operar um **Dispositivo de Captura**, onde possam ser integradas ferramentas para recolha de dados.

3.1.1 *Tipos de Medição*

No âmbito deste projeto, o empenho central está em capturar e analisar tráfego no acesso a serviços e aplicações atuais, como descrito anteriormente. Assim sendo, esta tarefa torna-se possível através da exploração do tráfego real originado por estas aplicações. Por este motivo, será selecionada a **medição passiva**, uma vez que não é relevante qualquer tipo de injeção de pacotes.

Com referido, dentro da medição passiva podem existir medições ao nível do fluxo e/ou do pacote. Uma vez que também se pretende efetuar uma comparação entre as versões aplicacional e *web* de um mesmo serviço, a medição passiva ao nível do fluxo torna-se mais vantajosa, na medida em que permite observar comportamentos gerais da rede, análises mais estatísticas e, consequentemente, um melhor ângulo de comparação entre as duas abordagens.

No entanto, caso se revele necessário para obtenção de informação mais específica na análise de algumas métricas, há possibilidade de se recorrer à medição ao nível do pacote.

3.1.2 *Volume de Dados e Amostragem*

A escala do projeto a realizar, não implica uma quantidade de dados insustentável. Por este motivo, e como todos os dados recolhidos podem ser importantes para a análise, não faz sentido aplicar técnicas de amostragem,

que descartem parte dos dados. Ainda assim, será utilizada **filtragem** para averiguar algumas das métricas, quando for interessante e relevante para determinado parâmetro.

3.2 MÉTRICAS

As métricas a examinar são fundamentais para caracterizar a rede e, ainda, para fornecerem uma base de comparação entre as duas versões do mesmo serviço. Com base nos objetivos definidos e nos artigos discutidos na Secção 2.2, serão estudados dois níveis: QoS (Qualidade de Serviço) e QoE (Qualidade de Experiência).

Além disso, as métricas serão adaptadas ao serviço em estudo, como descrito na Tabela 1, de acordo com Chowdhury et al. (2008) - por exemplo o *delay*. As métricas abordadas em Mongkolluksamee et al. (2016) e citadas na Secção 2.2, serão também fundamentais para caracterizar a rede. Entre elas incluem-se o número total de pacotes e o número total de *bytes*.

Ao nível da QoS, também se irão integrar métricas como as descritas em Hanemann et al. (2006) - disponibilidade, *loss*, *delay*, largura de banda.

No que concerne à análise de QoE, pretende-se utilizar um modelo de previsão de ML para obter uma avaliação idealmente semelhante a MOS, tendo em consideração parâmetros relacionados com QoS.

3.3 DISPOSITIVO MÓVEL E SUAS CARACTERÍSTICAS

O dispositivo móvel a utilizar corresponde a um *smartphone*, e, como é comum, todos os dispositivos tecnológicos possuem uma série de processos constantemente ativos. Obviamente, quando se pretende caracterizar tráfego real de um serviço, toda a atividade que não tem origem no mesmo torna-se ruído, podendo perturbar significativamente o estudo em desenvolvimento. Assim, é fundamental que o dispositivo móvel evite ao máximo este tipo de obstáculos. Para tal, durante as capturas é necessário:

- fechar todas as aplicações abertas;
- desativar todas as aplicações não essenciais, para evitar que corram em segundo plano;
- evitar qualquer tipo de atualização/*download* durante o processo;
- proceder a limpezas prévias de *cache*;
- desativar todo o tipo de notificações;
- utilizar modos anónimos sempre que possível.

Ao respeitar estes procedimentos, o tráfego capturado irá corresponder mais ao real, permitindo análises e conclusões corretas.

Os únicos requisitos necessários a este dispositivo móvel são:

- possuir tecnologia *Wi-Fi*;

- possuir acesso a uma loja de aplicações;
- permitir o *download* e instalação das aplicações em estudo, bem como do *web browser*.

Para a realização deste projeto será utilizado um *smartphone* de sistema operativo *Android*. Será utilizado o *browser Google Chrome* e, para obtenção das aplicações, *Google Play Store*.

3.4 DISPOSITIVO DE CAPTURA E SUAS CARACTERÍSTICAS

O dispositivo de captura irá funcionar como equipamento intermediário entre a rede e o dispositivo móvel. Para além de ser responsável pela recolha de todo o tráfego gerado no dispositivo móvel, procederá também ao armazenamento dos dados, para a análise que se efetuará *offline*.

O dispositivo de captura tem como premissas:

- possuir porta *Ethernet*;
- possuir tecnologias de partilha de Internet;
- possuir/permitir o uso de ferramentas de captura de tráfego;
- possuir/permitir o uso de ferramentas de análise de tráfego capturado;
- possuir armazenamento suficiente para alocar os dados recolhidos.

Para desenvolvimento deste estudo será utilizado um computador portátil, que corresponde aos requisitos apresentados. Será utilizada a tecnologia de *Partilha de Internet* (ver Figura 1), onde estará apenas conectado o dispositivo móvel.

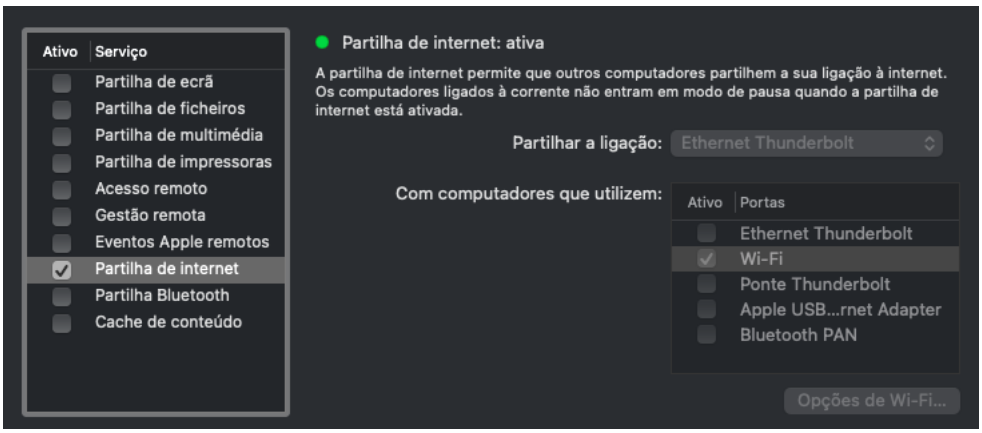


Figura 1: Partilha de Internat ativa.

O tráfego será capturado através da ferramenta Wireshark. Esta, além de permitir uma recolha de dados em tempo real e pela *interface* desejada, que irá corresponder à do dispositivo móvel, também dispõe de uma série de estatísticas sobre o tráfego capturado, permitindo por si só analisar várias métricas. Esta ferramenta possui

também um opção de filtragem do tráfego (ver Figura 2), algo que será bastante útil durante o desenvolvimento do estudo. Além do *Wireshark*, serão utilizadas ferramentas como STatistic and Tool e tcpdump.

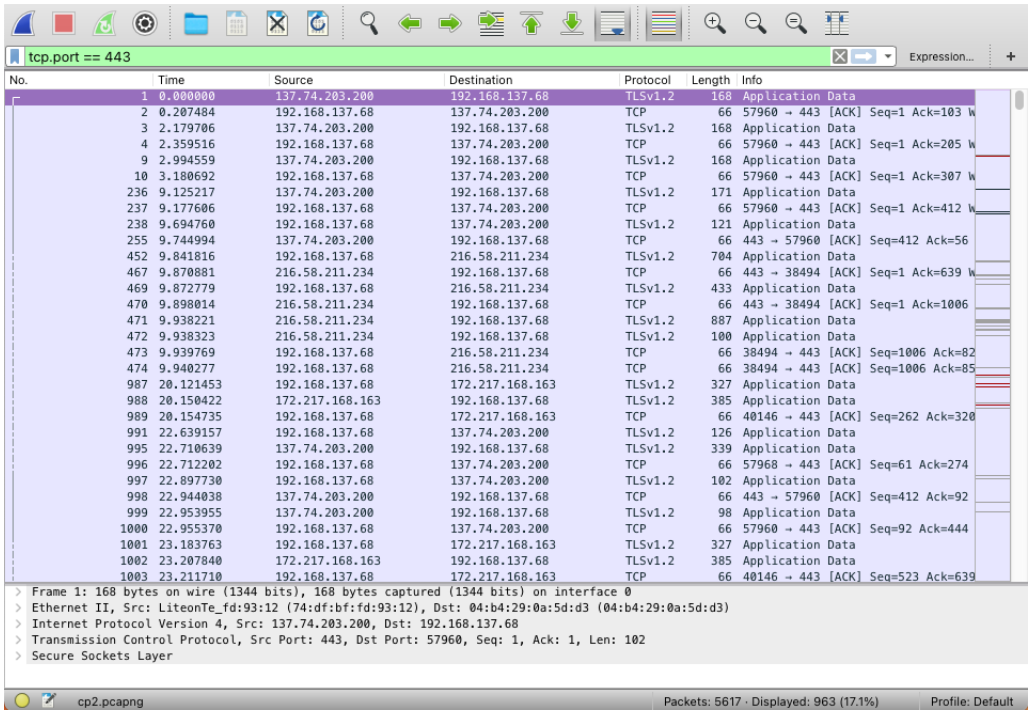


Figura 2: Exemplo de utilização da ferramenta Wireshark, com aplicação de filtro.

Relativamente à utilização de técnicas de ML para prever QoE, será utilizada a linguagem de programação Python. Todas as ferramentas e bibliotecas a utilizar nesta componente serão descritas no Capítulo 4.

3.5 ARQUITETURA FINAL

Finalmente, apresenta-se aquela que será a arquitetura final de testes. Como ilustrado na Figura 3, o computador portátil/dispositivo de recolha estará conectado à rede, através de cabo *Ethernet*. O mesmo terá instalada a ferramenta *Wireshark*, que estará a efetuar a recolha de tráfego em tempo real, da *interface* utilizada pelo dispositivo móvel. O dispositivo móvel estará conectado via *Wi-Fi* à partilha de Internet do dispositivo de captura. Deste modo, o dispositivo de captura estará pronto para proceder à recolha e armazenamento do tráfego gerado pelo dispositivo móvel, no acesso a aplicações/serviços.

Figura 3: Arquitetura final da plataforma de testes.

3.6 SUMÁRIO

No presente capítulo foram apresentados aspetos a ter em conta na realização das capturas de tráfego.

Inicialmente definiram-se os dispositivos a utilizar, bem como o tipo de medição (passiva) e a exclusão do uso de técnicas de amostragem, recorrendo-se apenas a filtragem. Depois, as métricas a serem analisadas foram descritas, dividindo-se estas nas categorias de QoS e QoE.

Posteriormente, delinearam-se os passos a seguir para a realização das capturas de tráfego, de modo a evitar ruído que possa impactar os resultados.

Finalmente, com base nos tópicos anteriores, apresentou-se a arquitetura final e as ferramentas auxiliares na captura e análise de tráfego, a utilizar neste projeto.

4

MODELO DE PREVISÃO

Neste capítulo apresenta-se a componente de ML desenvolvida neste projeto, cujo objetivo passa por obter um modelo capaz de prever a Qualidade de Experiência de um utilizador com base na Qualidade de Serviço. Serão abordadas ferramentas, bibliotecas, metodologia e modelos a utilizar. Por fim, serão apresentados os resultados obtidos para o modelo desenvolvido.

4.1 FERRAMENTAS E BIBLIOTECAS

Para a realização desta componente, foi selecionada a linguagem de programação ***Python***. Esta escolha baseou-se na facilidade para desenvolvimento de modelos de ML oferecida, uma vez que existem múltiplas bibliotecas que o suportam.

Para o desenvolvimento de código utilizou-se o ***JupyterLab***, que consiste num ambiente de desenvolvimento interativo para *notebooks*, *web-based*. Este destaca-se pela sua interface de utilização, flexibilidade e modularidade, permitindo facilmente dividir o código em blocos, acrescentar notas e comentários, entre outras vantagens (Jupyter).

No que concerne a bibliotecas, foram utilizadas algumas das mais populares e reconhecidas na área. Para importar, tratar e manipular dados utilizou-se a biblioteca **pandas**. No que diz respeito a visualização de dados recorreu-se às bibliotecas **seaborn** e **matplotlib**. Por fim, o desenvolvimento do ML, com tudo o que isso engloba, partiu do uso da biblioteca **scikit-learn**.

4.2 METODOLOGIA

Com a crescente utilização de técnicas de ML, surgiram alguns passos que normalmente são seguidos e aplicados.

Para a realização deste trabalho, seguiu-se genericamente a metodologia CRISP-DM Pete et al. (2000) (ver Figura 4), que procura tornar todo o processo mais eficiente.

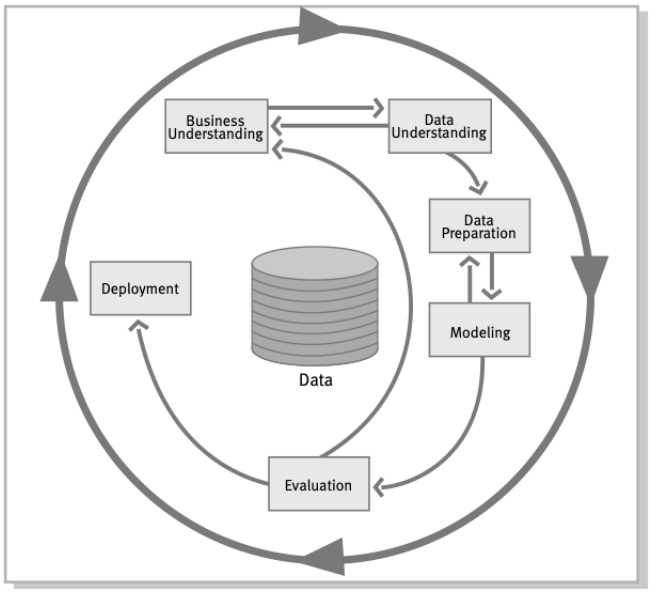


Figura 4: Fases do modelo de referência CRISP-DM (reproduzido de Pete et al. (2000)).

O primeiro passo consiste na definição concreta do problema a abordar. Neste caso, o mesmo passa por prever a QoE com base em parâmetros de QoS.

Posteriormente é necessário obter os dados a utilizar. Os mesmos devem ser estudados de forma a garantir a sua qualidade, algo fundamental para o bom desempenho de qualquer modelo. Além disso, normalmente é necessário recorrer a algum tipo de processamento que os permita melhorar, como a eliminação de valores em falta, normalização, entre outros.

Quando os passos anteriores são cumpridos e se dispõe um conjunto de dados consistente, dá-se lugar à modelação, ou seja, à seleção dos modelos de ML a aplicar. Os mesmos devem ser selecionados de acordo com as vantagens que apresentam para o problema definido. Pode também existir a otimização de hiperparâmetros para um desempenho ainda melhor.

Finalmente, procede-se a avaliação dos modelos implementados.

4.3 DESCRIÇÃO DOS DADOS E VARIÁVEL TARGET

O objetivo de prever QoE com base em QoS exige um *dataset* completo que, para além de conter informações sobre a qualidade da rede, pressupõe normalmente a opinião de vários utilizadores.

A criação de um *dataset* de qualidade e tamanho significativos demonstrou-se desde logo uma opção não concebível no desenvolvimento deste trabalho de mestrado, tal exigiria mais tempo para desenvolvimento de um ambiente de simulação controlado, ou para realização de experiências junto a um alargado número de diferentes utilizadores. Por este motivo, a melhor opção pareceu passar por utilizar dados públicos para desenvolver o melhor modelo de ML possível.

Deste modo, para treino e teste de ML recorreu-se ao *dataset* público utilizado no artigo Vasilev et al. (2018a), disponibilizado pelos autores em Vasilev et al. (2018b). Segundo os autores e citando o respetivo artigo, trata-se

de um *dataset* resultante de "um ambiente de simulação de alta fidelidade e totalmente controlável nos níveis de rede e *streaming*, (...), a plataforma de simulação é baseada no *Adaptive Multimedia Streaming Simulator Framework (AMust)* em ns-3 que implementa um cliente e servidor HTTP para LibDASH (...)". Na Figura 5 apresenta-se o ambiente de simulação utilizado.

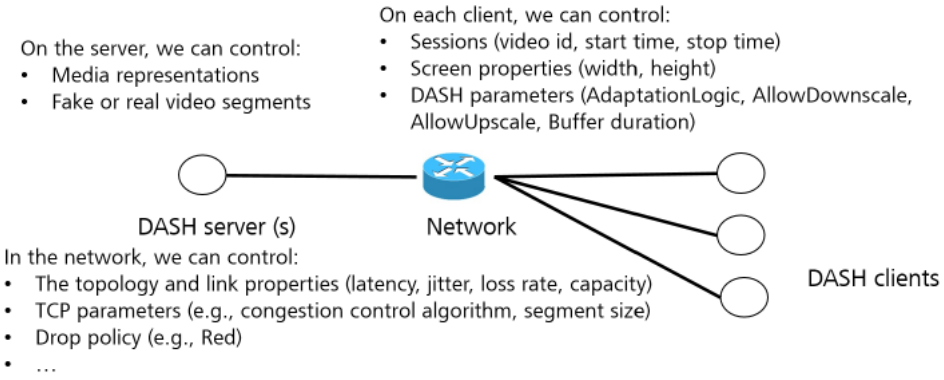


Figura 5: Ambiente de Simulação com AMust em ns-3 (reproduzido de Vasilev et al. (2018a)).

Foram simulados vários cenários e como conteúdo de *streaming*, foram utilizados três filmes bastante representativos. Toda a informação detalhada pode ser consultada no artigo Vasilev et al. (2018a).

O *dataset* final possui estatísticas para 69000 sessões de vídeo e 50 variáveis que os autores categorizaram em:

- informação de contexto sobre o congestionamento da rede e características do *stream*;
- métricas QoS;
- fatores QoE *target*;
- variáveis de QoE *hidden* (consideradas aquelas que não se podem obter diretamente).

Na Tabela 4 que se segue estão representadas todas as variáveis do *dataset* original.

Índice	Nome	Tipo	Descrição
1	RequestID	Contexto	Identificador da sessão de *streaming*
2	NbClients	Contexto	Nº máximo de *streams* a competir no *bottleneck*
3	BottleneckBW	Contexto	Capacidade do *bottleneck*
4	BottleneckDelay	Contexto	*Delay* da rede no *bottleneck*
5	BottleneckLoss	Contexto	Perda de pacotes no *bottleneck*
6	DASHPolicy	Contexto	Política DASH (por exemplo, ou nome do provedor do conteúdo
7	ClientResolution	Contexto	Resolução do ecrã do cliente ou tipo de dispositivo (por exemplo, *smartphone*)
8	RequestDuration	Métrica QoS	Duração do *stream*
[9, 13]	TCPOut/InPacket	Métrica QoS	Nº de pacotes TCP (In/Out)
[10, 14]	TCPOut/InDelay	Métrica QoS	*Delay* médio dos pacotes TCP (In/Out)
[11, 15]	TCPOut/InJitter	Métrica QoS	*Jitter* médio dos pacotes TCP (In/Out)
[12, 16]	TCPOut/InPloss	Métrica QoS	Taxa de *loss* dos pacotes TCP (In/Out)
17	TCPInputRetrans	Métrica QoS	Retransmissões dos pacotes TCP
18	StdNetworkRate	Métrica QoS	Desvio padrão da taxa da rede
[19:27]	[0,5,10,25,50,75,90,95,100] NetworkRate	Métrica QoS	x^{th} quantil para a taxa da rede (medido em intervalos de 2s)
28	StdInterATimesReq	Métrica QoS	Desvio padrão de tempos entre chegadas de pedidos de segmento
[29:37]	[0,5,10,25,50,75,90,95,100] InterATimesReq	Métrica QoS	x^{th} quantil para os tempos entre chegadas de pedidos de segmento
38	StartUpDelay	*Hidden*	Tempo inicial no cliente para começar a reproduzir o vídeo
39	AvgVideoDownloadRate	*Hidden*	Taxa média de *download* para segmentos de vídeo
40	StdVideoDownload	*Hidden*	Desvio médio da taxa de *download* para segmentos de vídeo
41	AvgVideoBufferLevel	*Hidden*	Comprimento médio do *buffer* de vídeo
42	StdVideoBufferLevel	*Hidden*	Desvio padrão do comprimento do *buffer* de vídeo
43	StallEvents	*Hidden*	Nº de eventos de *stall*
44	RebufferingRatio	*Target*	Porção do tempo gasto em eventos de *stall*
45	StallLabel	*Target*	Discretização da variável RebufferingRatio
46	TotalStallingTime	*Hidden*	Duração total dos eventos de *stall*
47	AvgTimeStallingEvents	*Hidden*	Duração média dos eventos de *stall*
48	AvgQualityIndex	*Hidden*	Índice normalizado médio de representações *downloaded*
49	AvgVideoBitRate	*Target*	Média de *bitrate* de vídeo consumida pelo *player*
50	AvgVideoQualityVariation	*Target*	Variação média de *bitrate* do vídeo
51	AvgDownloadBitRate	*Hidden*	Taxa média de *download* de segmentos de vídeo

Tabela 4: Dados originais do *dataset* (Vasilev et al. (2018b)).

A variável *StallLabel* resulta da agregação da *RebufferingRatio* em três variáveis discretas, sendo elas:

- *StallLabel* = *NoStall*, quando *RebufferingRatio* = 0 e não há ocorrência de *stalling*;
- *StallLabel* = *MildStall*, quando *RebufferingRatio* varia entre]0,0.1];
- *StallLabel* = *SevereStall*, quando *RebufferingRatio* superior a 0.1;

Para a realização deste trabalho, centrado no objetivo de prever QoE com base em QoS, as variáveis em uso foram reduzidas. Como *features* selecionaram-se as variáveis que dizem respeito a métricas de QoS. No entanto, tendo em consideração que nem todas as variáveis são de fácil/possível obtenção para as capturas de tráfego a efetuar neste trabalho, algumas foram eliminadas por esta dificuldade.

Como ***Target*** selecionou-se a variável *StallLabel*, uma vez que reflete a qualidade de experiência para um utilizador durante o *streaming* de vídeo. Assim, o *dataset* final a utilizar está descrito na Tabela 5.

TCPOutputPacket	Métrica QoS	Nº de pacotes TCP Out
TCPInputPacket	Métrica QoS	Nº de pacotes TCP In
TCPOutputDelay	Métrica QoS	*Delay* médio dos pacotes TCP Out
TCPInputDelay	Métrica QoS	*Delay* médio dos pacotes TCP In
TCPOutputJitter	Métrica QoS	*Jitter* médio dos pacotes TCP Out
TCPInputJitter	Métrica QoS	*Jitter* médio dos pacotes TCP In
TCPOutputPloss	Métrica QoS	Taxa de *loss* dos pacotes TCP Out
TCPInputPloss	Métrica QoS	Taxa de *loss* dos pacotes TCP In
TCPInputRetrans	Métrica QoS	Retransmissões dos pacotes TCP
StallLabel	***Target***	Discretização da variável RebufferingRatio

Tabela 5: Variáveis utilizadas no desenvolvimento de ML.

Para aplicar os modelos de ML desenvolvidos com este *dataset* às capturas de tráfego a efetuar neste trabalho, as variáveis a usar como *features* serão obtidas a partir do *Tstat*. A partir dos *LOG Files* que o mesmo gera, torna-se possível adquirir os valores médios necessários, diretamente ou através de aproximações. Maior detalhe sobre a obtenção destas variáveis será dado no momento de aplicação do modelo, no Capítulo 6.

4.4 PRÉ-PROCESSAMENTO DOS DADOS

O *dataset* utilizado, por ter sido criado a partir de um ambiente controlado e utilizado para um artigo, contém dados limpos. Assim, não se verificou a existência de dados em falta, valores nulos, ou necessidade de remover *outliers*. O único pré-processamento de dados efetuado está relacionado com a normalização ou padronização de dados. Tal será abordado na Secção 4.5.

4.4.1 *Variável Target*

Relativamente à variável *target* (*StallLabel*), de maneira a simplificar o desenvolvimento dos modelos, foi utilizado o seguinte mapeamento para atribuição de classes:

- *StallLabel* = *NoStalling* - Classe 0
- *StallLabel* = *MildStalling* - Classe 1
- *StallLabel* = *SevereStalling* - Classe 2

Quando analisada a distribuição desta variável, detetou-se um desbalanceamento de dados bastante significativo. Na Figura 6, representa-se o número de elementos de cada classe.

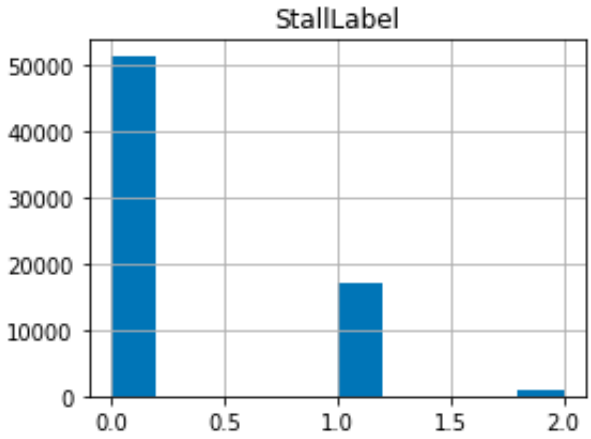


Figura 6: Distribuição da variável *target*.

- Classe 0 - 51155 (73.999%)
- Classe 1 - 17180 (24.852%)
- Classe 2 - 794 (1.149%)

Tendo em consideração estes dados, ponderou-se o agrupamento das Classes 1 e 2, tornando este problema num caso de classificação binária com a existência ou não de *stall*. No entanto, centrando no objetivo de prever a qualidade de experiência, entre a Classe 1 e a Classe 2 existe uma diferença significativa na experiência que oferecem ao utilizador durante o *streaming* de vídeo. Por este motivo, optou-se por se manter as três classes para classificação.

4.4.2 *Visualização de Dados*

No sentido de compreender o impacto de algumas variáveis face à variável *target StallLabel*, procedeu-se a algumas técnicas que permitem visualizar os dados.

Uma das técnicas mais conhecidas é a representação da relação entre variáveis através de um *Heatmap*. Na Figura 7 apresenta-se o *heatmap* de correlação entre as variáveis em uso. A sua interpretação é feita considerando que cada quadrado pode variar entre [-1,1] e representa a correlação linear entre as duas variáveis específicas. Quanto mais perto de 1, maior a correlação positiva (uma variável aumenta, a outra também). Se for mais perto de -1, trata-se de uma correlação negativa (uma variável aumenta, a outra diminui). Assim, valores de correlação perto de 0, significam que não existe uma correlação significativa entre as variáveis, e quanto mais perto de -1 ou 1, mais forte a correlação.

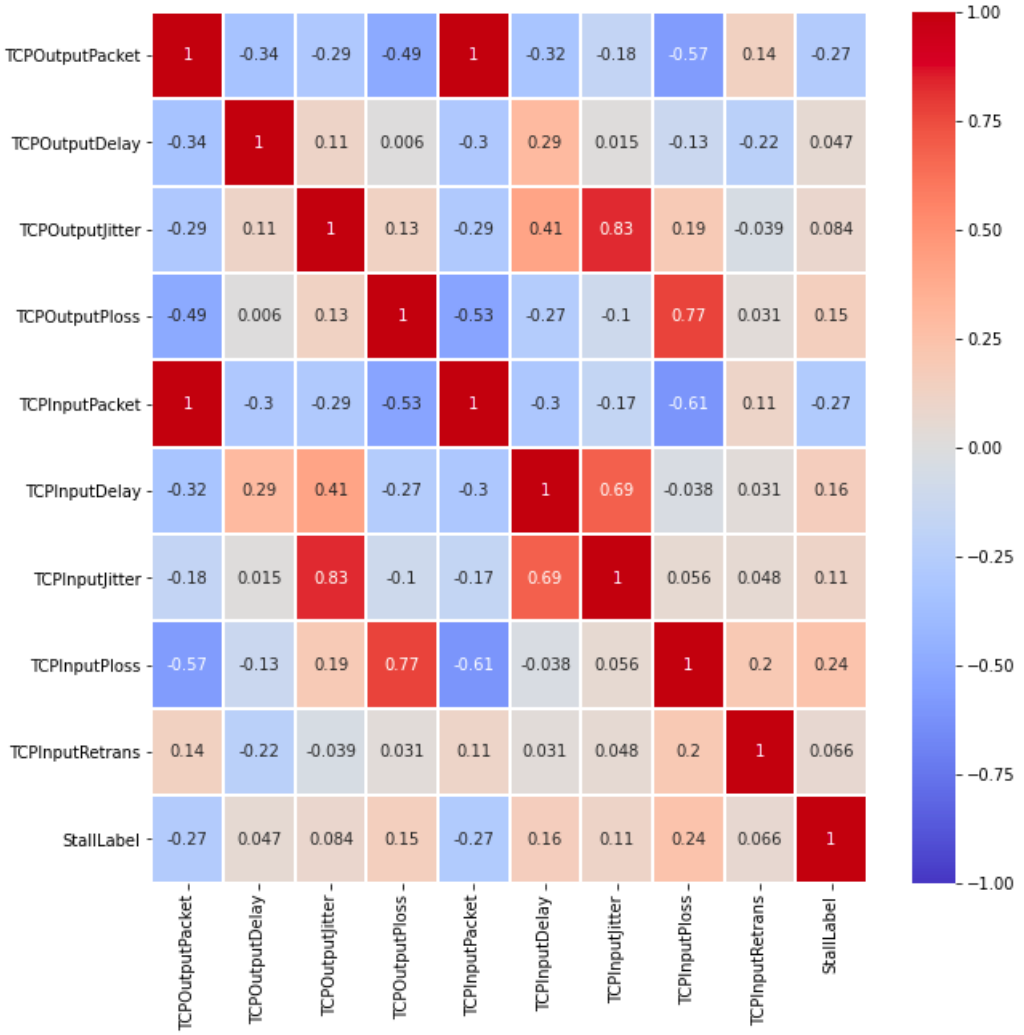


Figura 7: *Heatmap*.

Analisando o resultado, no que diz respeito à variável *target*, é possível concluir que não possui alta correlação com nenhuma variável. As duas variáveis mais importantes são o número de pacotes *TCP In/Out* (*TCPOutput-Packet* e *TCPInputPacket*), com uma pequena correlação negativa de -0.27. Como correlação positiva (0.24), a variável que mais se destaca é a taxa de perda de pacotes *input* (*TCPInputPloss*). Portanto, quanto maior esta taxa de perda, a *StallLabel* tende a aumentar também, neste caso, a pertencer a uma das classe com *stall*.

No âmbito deste trabalho, torna-se também interessante avaliar o *heatmap* a nível de correlação entre as variáveis de QoS. Por exemplo, é possível perceber que, na generalidade, existe uma elevada correlação positiva entre uma mesma métrica quando comparando o *In* e *Out*. Por exemplo, *TCPInputPloss* e *TCPOutputPloss* apresentam alta correlação positiva, correspondendo à mesma métrica no sentido *In* e *Out*.

Através da biblioteca **scikit-learn**, utilizou-se o *SelectKBest*, para obter as cinco *features* com maior pontuação, ou seja, que mais contribuem para obter a variável *target*.

O resultado, que concorda com o já observado também no *heatmap*, apontou para as variáveis *TCPOutputPacket*, *TCPOutputPloss*, *TCPInputPacket*, *TCPInputDelay* e *TCPInputPloss* (ver Figura 8).

```
The best 5 features:
 - TCPOutputPacket
 - TCPOutputPloss
 - TCPInputPacket
 - TCPInputDelay
 - TCPInputPloss
```

Figura 8: *KBest features*.

4.5 MODELAÇÃO

Nesta secção apresentam-se os modelos de ML implementados, bem como o uso de ferramentas para normalizar e balancear dados. Identificado o problema de classificação e tendo conhecimento sobre os dados em utilização, decidiu-se utilizar algumas alternativas diferentes para a modelação.

Primeiramente, recorreu-se a *feature scaling* para normalizar e padronizar os dados. Uma das opções foi *MinMaxScaler*, uma técnica de pré-processamento que aplica a cada coluna uma transformação na escala de dados entre 0 e 1, preservando a distribuição original dos dados. Uma segunda opção foi o *StandardScaler*, este que também se aplica a todas as colunas mas de forma diferente ao anterior. A sua utilização resulta numa média igual a 0 e distribuição de dados com desvio padrão e variância igual a 1.

Normalmente os algoritmos baseados em árvores, não necessitam de *feature scaling* por serem invariantes a escala. No entanto, todos os estimadores utilizados foram testados com e sem normalização e padronização.

Além disto, tal como abordado previamente e demonstrado na Figura 6, os dados em uso são consideravelmente desbalanceados. Para tentar combater problemas que esta característica pode levantar, utilizaram-se técnicas de atribuição de diferentes pesos das classes, *oversampling* e *undersampling*.

O *oversampling* e *undersampling* são técnicas de *sampling*, aplicadas aos dados de treino, que procuram balancear as classes. Mais particularmente, o *oversampling* consiste em aumentar os dados das classes minoritárias, para igualar a classe maioritária. Contrariamente, *undersampling* procura reduzir todas as classes à mesma quantidade da classe minoritária. Para o caso de oversampling utilizou-se SMOTE (*Synthetic Minority Oversampling Technique*), uma técnica que gera novas instâncias para as classes minoritárias. Foi possível utilizar esta técnica de *oversampling* sintético uma vez que o *dataset* não possui elevadas dimensões. Para *undersampling* recorreu-se ao *Tomek Links*, que remove dados das classes maioritárias de acordo com a proximidade a instâncias das classes minoritárias.

Como modelos foram definidos alguns dos mais populares que se adequavam ao problema de classificação, nomeadamente, Regressão Logística, *Random Forest*, Redes Neuronais (MLP (*Multilayer Perceptron*)), *XGBoost* e *LightGBM*.

Na Tabela 6 apresentam-se as combinações que foram testadas entre estimadores, *scalers* e técnicas de balanceamento de classes.

Estimadores	***Scalers***	**Balanceamento**
Regressão Logística		Nenhum
Random Forest	Nenhum	Balanceamento do peso das classes
Redes Neuronais (MLP)	*MinMaxScaler*	*Oversampling*
XGBoost	*StandardScaler*	*Undersampling*
LightGBM		*Oversampling + Undersampling*

Tabela 6: Alternativas a utilizar para modelos de ML.

Para execução dos modelos, o *dataset* foi dividido em dados de treino (70%) e dados de teste (30%). Os dados de treino utilizados para treinar os modelos e o dados de teste para testar o desempenho do mesmo.

4.6 MÉTRICAS E AVALIAÇÃO DE MODELO

Para se proceder à avaliação dos modelos desenvolvidos, ponderou-se a utilização de algumas das mais populares métricas. No artigo Korotcov et al. (2017), são expostas algumas destas métricas, das quais se destacam as apresentadas de seguida.

Matriz de Confusão

A matriz de confusão é um output muito utilizado para avaliar a performance do modelo, onde existe uma comparação entre os valores reais e os previstos. Além disso, serve de ponto de partida para outras métricas. Por exemplo, no caso mais geral de um problema de classificação binária com as *labels* 0 (negativo) e 1 (positivo), a matriz de confusão traduz-se em algo semelhante ao demonstrado na Figura 9,

		Valor Real	
		Positivo	Negativo
Valor Previsto	Positivo	TP	FP
	Negativo	FN	TN

Figura 9: Matriz de confusão.

onde:

- Verdadeiros Positivos (TP) – Quando a classe prevista é 1 e a classe real é 1;
- Verdadeiros Negativos (TN) – Quando a classe prevista é 0 e a classe real é 0;
- Falsos Positivos (FP) – Quando a classe prevista é 1 e a classe real é 0;
- Falsos Negativos (FN) – Quando a classe prevista é 0 e a classe real é 1.

Accuracy

A métrica de *accuracy* traduz-se na seguinte equação:

$$Accuracy = \frac{\text{Número de previsões corretas}}{\text{Número total de previsões efetuadas}} \tag{1}$$

Esta métrica, apesar de simples e muito usada, nem sempre representa um valor realmente representativo do desempenho do modelo. Por exemplo, se para um problema de classificação binária estiver a ser usado um *dataset* não balanceado com percentagem muito baixa de dados pertencentes a uma classe, se o modelo apenas acertar nos valores da classe em maioria, a *accuracy* continuará a ter um valor elevado, mesmo não sendo capaz de prever nenhum valor da classe em minoria. Por este motivo, é importante ter atenção às condições em que se aplica esta métrica para obter uma avaliação real.

F1 Score

A métrica F1 é calculada a partir da Média Harmónica entre a precisão e o *recall* (sensibilidade),

$$F1 = 2 * \frac{1}{\frac{1}{\text{Precisão}} + \frac{1}{\text{Recall}}} = 2 * \frac{\text{Precisão} * \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{2}$$

onde:

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

Esta métrica é bastante mais precisa e robusta que a anterior, uma vez que nos permite ter em consideração a performance real do modelo, em todas as classes.

Curva de ROC (Receiver Operating Characteristic)

A Curva de ROC é uma forma precisa de avaliar o modelo de classificação resultante da visualização de um gráfico construído com os parâmetros da Taxa de Verdadeiros Positivos e a Taxa de Falsos Positivos, que traduzem o rácio de TP/Positivos Totais e FP/Negativos Totais, respetivamente. Assim, o gráfico é projetado de acordo com os valores destas taxas em diferentes *thresholds* da classificação.

AUC (Area Under Curve)

No seguimento da métrica anterior, a AUC mede a área do espaço bidimensional sob a curva ROC. Esta é uma ótima métrica para avaliar modelos onde existe um desbalanceamento de classes, uma vez que é sensível aos mesmos. Apesar de ser mais utilizada em classificação binária, pode ser adaptada para outros tipos de classificação.

MAE (Mean Absolute Error)

O MAE permite-nos calcular a magnitude de erros nas previsões do modelo, ou seja, o quão longe dos valores reais estão os seus outputs, não considerando a direção dos mesmos. Representa-se por:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - y_i^p|, \text{ onde } y_i^p \text{ representa o valor de } y \text{ previsto} \tag{5}$$

MSE (Mean Squared Error)

O MSE é representado por:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_i^p)^2, \text{ onde } y_i^p \text{ representa o valor de } y \text{ previsto} \tag{6}$$

Como se pode observar, o MSE traduz-se na média do quadrado da diferença entre os valores previstos e os reais, ou seja, a sua variância.

Após se avaliarem todas as métricas anteriores, decidiu-se utilizar como métrica decisiva **Micro F1-score**, uma vez que esta nos permite obter um resultado mais real, principalmente tendo em consideração que se trata de um *dataset* desbalanceado. Notando que Micro F1-score diz respeito a F1-score com a atribuição de *micro* ao parâmetro *average* no cálculo da métrica, necessário por se tratar de um problema de *multi-class*. Com *micro*, a métrica é calculada globalmente contando o total de TP, FN, FP.

Assim, de entre todas as combinações referidas na Tabela 6, a métrica Micro F1-score será utilizada para comparar todos os modelos e selecionar o que demonstrar melhor desempenho. Posteriormente, esse modelo será alvo de uma otimização de hiperparâmetros, onde será feita uma análise mais aprofundada de resultados.

4.7 RESULTADOS

Na Tabela 7 apresentam-se os quinze melhores modelos obtidos de entre todas as combinações (ver Tabela 6), de acordo com a métrica Micro F1-score. Como já referido, esta é uma métrica que representa de forma justa o desempenho dos modelos. Por exemplo, caso o modelo não consiga prever nada da classe 2 minoritária, a *accuracy* continuaria a ser elevada desde que tenha um bom desempenho para a classe 0 e 1.

Estimador	*Scaler*	Balanceamento	Micro F1-score
Xgboost	-	-	**0.8456**
Lightgbm	-	-	0.8452
Random Forest	-	-	0.8390
Lightgbm	-	Oversampling + undersampling	0.8370
Random Forest	-	Balanceamento do peso das classes	0.8366
Xgboost	-	Undersampling	0.8312
Lightgbm	-	Undersampling	0.8261
Xgboost	-	Oversampling + Undersampling	0.8244
Random Forest	-	Undersampling	0.8244
Lightgbm	-	Oversampling	0.8226
Random Forest	-	Oversampling	0.8196
Xgboost	-	Oversampling	0.8195
Rede Neuronal (MLP)	MinMaxScaler	-	0.8013
Rede Neuronal (MLP)	MinMaxScaler	Undersampling	0.7939
Rede Neuronal (MLP)	MinMaxScaler	Oversampling	0.7750

Tabela 7: Melhores modelos de ML obtidos.

Como é possível observar, o modelo *XGBoost* obteve o melhor desempenho, sem utilização de qualquer *scaler* ou técnica de *sampling*. Neste caso, os modelos de *boosting* demonstraram obter melhores resultados de forma global.

Relativamente ao uso de *scalers*, era previsível que não se sentisse o seu impacto nos estimadores baseados em árvores. Como referido anteriormente, estes são insensíveis a *scalers*. No entanto, a sua utilização permitiu melhorar os resultados da MLP e também da regressão logística.

Neste caso e para este *dataset*, o uso de técnicas de *sampling* não se revelaram importantes na sua aplicação. Em nenhum caso, a utilização de uma destas técnicas melhorou o desempenho de um modelo, face ao desempenho que o mesmo apresentava com a utilização dos dados originais.

Concluída esta etapa, o modelo *XGBoost*, sem uso de *scalers* e técnicas de *sampling*, foi selecionado para um processo de otimização de hiperparâmetros, com o objetivo de melhorar mais o seu desempenho.

4.7.1 *Otimização de Hiperparâmetros*

O objetivo desta otimização de hiperparâmetros passa por alcançar um modelo *XGBoost* com a métrica micro F1-score superior ao valor atingido previamente (0.8456).

Para este processo recorreu-se ao *RandomizedSearchCV*. Neste caso, quando se passam os parâmetros com valores a utilizar, em vez de se testar todas as combinações entre todos os valores (como é o caso do *GridSearchCV*), as combinações são efetuadas de forma aleatória. Desta forma é possível testar um maior número de parâmetros com menos tempo de treino, obtendo-se resultados igualmente bons.

Os principais parâmetros que se optou por testar, de acordo com a documentação de desenvolvimento do modelo, dizem respeito a:

- *n_estimators* - número de árvores a utilizar no modelo;

- *colsample_bytree* - fração de *features* a ser utilizada em cada árvore;
- *colsample_bylevel* - fração de *features* a ser utilizada em cada nível;
- *learning_rate* - está relacionado com a velocidade de convergência do algoritmo, uma vez que define qual o tamanho do passo a dar durante a aprendizagem do mesmo;
- *max_depth* - profundidade máxima permitida, ou seja, número máximo de nodos entre a raiz e a folha mais longínqua;
- *min_child_weight* - peso mínimo ou número de instâncias mínimas necessárias para criar um novo nodo na árvore;
- *subsample* - fração de amostragem das instâncias de treino a utilizar em cada passo;
- *gamma* - valor mínimo de redução de *loss* para particionar o respetivo nó folha.

Na Tabela 8 apresentam-se os intervalos de valores testados para otimização do modelo. Os intervalos definidos procuram fornecer uma grande variabilidade para experimentação de valores. Alguns dos parâmetros selecionados têm como objetivo evitar situações como *overfitting*, controlando a complexidade do modelo desenvolvido. Para outros parâmetros foram utilizados intervalos de valores comuns na implementação e desenvolvimento deste algoritmo.

Hiperparâmetro	**Intervalo**
n_estimators	[10, 300]
colsample_bytree	[0.5, 0.8]
colsample_bylevel	[0.8, 1]
learning_rate	[0.001, 0.1]
max_depth	[5, 400]
min_child_weight	[5, 20]
subsample	[0.5, 1]
gamma	[0.01, 0.2]

Tabela 8: Intervalos de valores atribuídos aos hiperparâmetros.

Com o uso do *RandomizedSearchCV* adicionou-se um método de *cross-validation*. Um dos motivos mais recorrentes para o seu uso é averiguar se o modelo é capaz de generalizar a sua aprendizagem, utilizando diferentes porções de dados de teste e treino em diferentes iterações. Neste caso, utilizou-se *StratifiedKFold*, uma técnica de *cross-validation* que consiste na divisão dos dados em K *folds*. Uma vez que esta técnica preserva a distribuição dos dados nas divisões, demonstra ser mais adequada para usar com dados não balanceados.

4.7.2 *Resultados*

Na Figura 10 apresentam-se os valores dos parâmetros ótimos obtidos.

```
{'subsample': 0.7,
 'n_estimators': 300,
 'min_child_weight': 10,
 'max_depth': 10,
 'learning_rate': 0.1,
 'gamma': 0.1,
 'colsample_bytree': 0.7,
 'colsample_bylevel': 0.89}
```

Figura 10: Melhores parâmetros obtidos na otimização de hiperparâmetros.

Com a otimização de hiperparâmetros, a métrica Micro F1-score teve um aumento pouco significativo para 0.8483.

Antes de analisar as restantes métricas obtidas neste modelo, é importante visualizar as variáveis com maior impacto nele. Na Figura 11 apresenta-se o gráfico da importância de cada variável. Para este gráfico, foi definido o tipo de importância *weight*, que diz respeito ao número de vezes que a *feature* apareceu numa árvore no modelo. Como se pode observar, as variáveis relacionadas com *jitter* e atraso têm maior impacto no modelo desenvolvido.

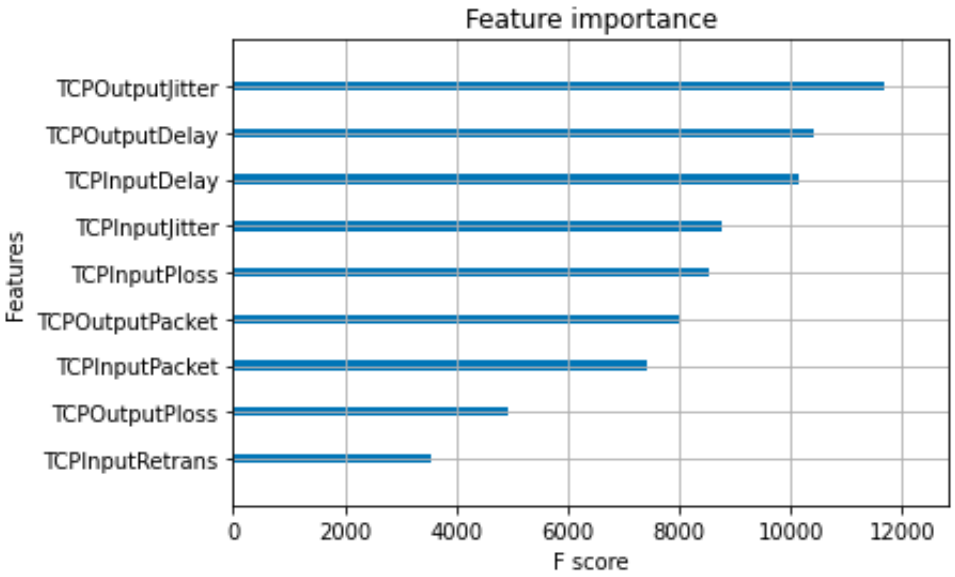


Figura 11: Importância de cada *feature* no modelo.

De seguida, na Figura 12, apresenta-se o relatório de classificação do modelo.

Classification Report :

	precision	recall	f1-score	support
0	0.88	0.94	0.91	15335
1	0.74	0.61	0.67	5174
2	0.69	0.23	0.34	230
accuracy			0.85	20739
macro avg	0.77	0.59	0.64	20739
weighted avg	0.84	0.85	0.84	20739

Figura 12: Relatório de classificação.

Avaliando os valores obtidos é possível perceber que a classe maioritária (0 - *NoStalling*) apresenta resultados melhores que as outras duas, sendo os piores resultados correspondentes à classe minoritária (2 - *SevereStalling*). Este comportamento deverá estar associado ao desbalanceamento de classes existente, uma vez que a distribuição de elementos nas classes é diretamente proporcional aos resultados obtidos, ou seja, quantos mais valores possui a classe, melhor o resultado.

Como demonstrado na Tabela 6, inicialmente foram utilizadas e testadas técnicas de *sampling* para combater este problema. No entanto, estas não demonstraram melhorar os modelos (ver Tabela 7).

Para complementar a análise, visualizaram-se as curvas de ROC (Figura 13) e curvas de PR (*Precision-Recall*) (Figura 14).

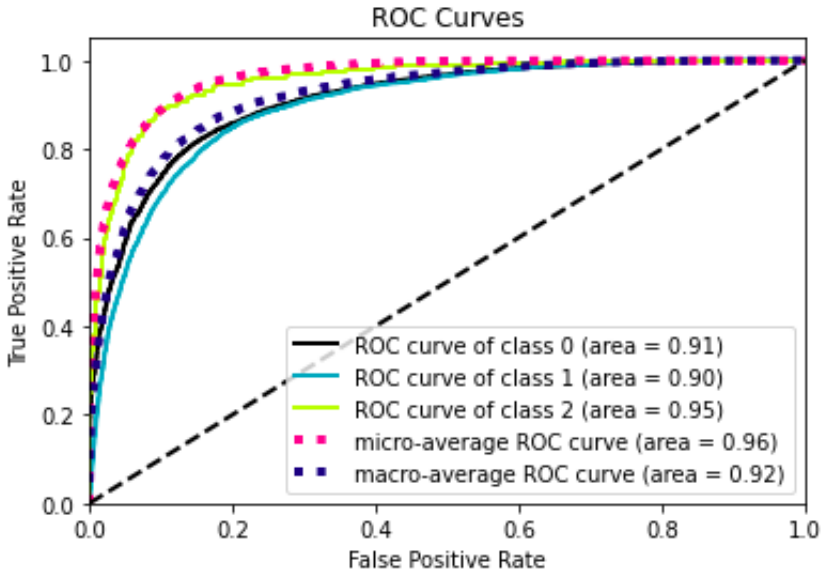


Figura 13: Curvas de ROC.

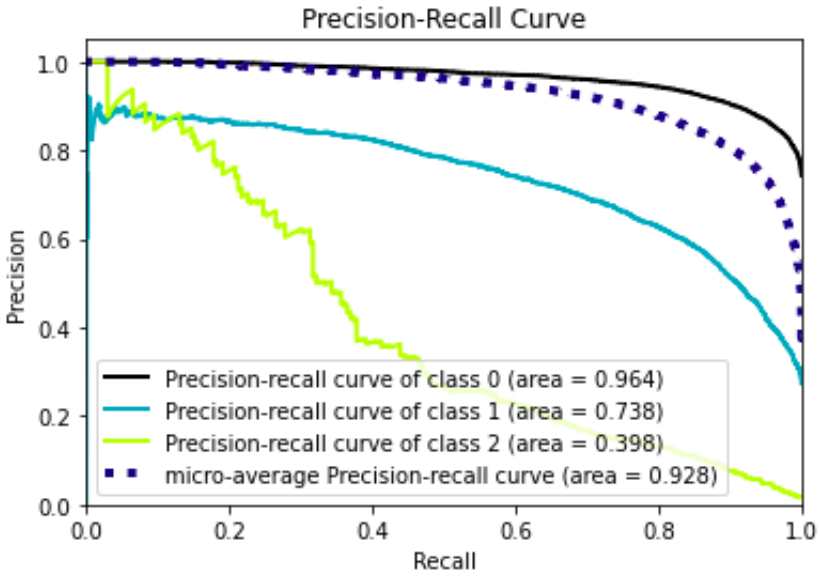


Figura 14: Curvas de PR.

As curvas de ROC, previamente descritas na Secção 4.6, e as curvas PR, que representam graficamente a relação entre precisão e *recall* para diferentes *thresholds*, confirmam os dados obtidos no relatório de classificação. Uma vez que quanto mais perto de 1 estiverem as curvas, melhor o desempenho do modelo, conclui-se também através desta visualização um pior resultado na previsão para a classe minoritária.

Apesar disso, de forma geral, consideram-se os resultados deste modelo satisfatórios.

4.8 SUMÁRIO

Neste capítulo apresentou-se a componente de ML desenvolvida neste trabalho, cujo objetivo passa por criar um modelo de ML capaz de prever QoE com base em fatores de QoS.

Em primeiro lugar, foram apresentadas as ferramentas, bibliotecas e metodologia utilizadas. Depois, foram apresentados os dados de treino e teste provenientes de um *dataset* público, dada a incapacidade de obtenção de um *dataset* próprio. Após a definição da variável *target*, procedeu-se a um pré-processamento de dados e visualização dos mesmos.

Na fase de modelação, foram expostos os vários estimadores a utilizar e, de seguida, selecionou-se a métrica decisiva para escolher o melhor modelo.

Por fim, obteve-se o melhor modelo e efetuou-se uma otimização de hiperparâmetros. Deste modo, foi possível obter um modelo com bons resultados, capaz de ser aplicado às capturas de tráfego a realizar.

5

CASOS DE ESTUDO

O presente capítulo tem como objetivo a apresentação de possíveis casos a analisar e qual o selecionado para a realização deste trabalho. Inicialmente serão nomeadas plataformas aplicacionais em destaque no passado ano e, posteriormente, os motivos que levaram à seleção de uma delas. Pretende-se também neste capítulo expor qual o conteúdo da plataforma que será alvo de estudo, bem como métricas a utilizar e cuidados a ter durante a recolha de tráfego, na versão aplicacional e *browser*.

5.1 APLICAÇÕES MAIS UTILIZADAS

Consultando o relatório SensorTower (2020), é possível obter imensa informação sobre o mercado das aplicações durante o ano de 2020. Em 2020 foram efetuados 143 biliões de *downloads* nas duas lojas de aplicações de renome, *App Store* e *Google Play*. Sendo que o *top* 5 de aplicações mais descarregadas, mundialmente, em 2020 são: TikTok, WhatsApp, Facebook, Zoom e Instagram.

Também examinando vários *rankings* atuais, como os fornecidos por comscore, verifica-se que a aplicação com maior alcance de utilizadores é o YouTube, sendo a mais representativa na categoria de *streaming* de vídeo e no volume de tráfego a nível global.

5.2 SERVIÇO SELECIONADO - YOUTUBE

Para decidir qual o serviço a utilizar e analisar ao longo deste projeto, questões como a alta utilização e o interesse que podem representar foram consideradas. Optou-se por selecionar apenas um, de modo a permitir um estudo mais aprofundado, a nível de QoS e QoE.

O serviço YouTube foi o selecionado por vários motivos. Em primeiro lugar por ser a plataforma com maior alcance entre utilizadores. Além disso, tendo em consideração o objetivo de prever QoE com base em QoS, uma plataforma de *streaming* de vídeo gera maior facilidade para os utilizadores percecionarem quebras na qualidade da rede. Para a componente de ML deste projeto, utilizou-se um *dataset* público de *streaming* de vídeo e, portanto, faz todo o sentido que o YouTube seja a plataforma de eleição para análise.

O YouTube revela também muitos aspetos interessantes por ser uma plataforma dedicada a vídeo. Por exemplo, a transmissão de vídeo é efetuada através de protocolos como HTTP/TLS/TCP ou HTTP/QUIC/UDP

confiáveis. É também um serviço que utiliza HAS (*HTTP Adaptive Streaming*), beneficiando o utilizador final de maior qualidade e eficiência, aspetos abordados em Biernacki and Tutschku (2014).

5.3 RECOLHA DE DADOS

Para a análise do serviço YouTube, efetuaram-se e analisaram-se recolhas do *streaming* de vídeos. Os vídeos em causa e as condições de captura serão apresentados nesta secção.

5.3.1 *Vídeos Utilizados*

Inicialmente ponderou-se a seleção de vídeos de acordo com o grau de popularidade, uma vez que a própria plataforma fornece uma secção apenas dedicada a tendências.

No entanto, optou-se por seguir um caminho diferente e selecionar os vídeos de acordo com os tipos de cenas e tempos de reprodução. O objetivo é perceber de que maneira é que estes fatores podem impactar o tráfego gerado.

Enumeram-se de seguida os vídeos utilizados:

1. *Big Buck Bunny 60fps 4K - Official Blender Foundation Short Film* (Blender (2014))

   O primeiro vídeo consiste num *cartoon* com uma mistura de cenas *high* e *low motion*. Possui uma duração total de 10:34 minutos (ver Figura 15).

Figura 15: Primeiro vídeo de análise (Blender (2014)).

2. *Filmic Pro Cinematic Video Test Aestethic Video Random Nature clips B roll 4K 24fps* (FORMARAN (2021))

   O segundo vídeo consiste num pequeno vídeo de cenas *low motion*. Possui uma duração total de 02:02 minutos (ver Figura 16).

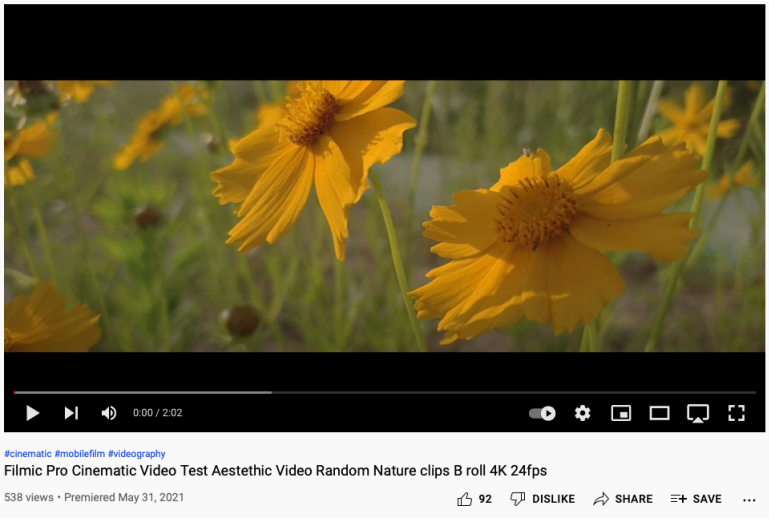


Figura 16: Segundo vídeo de análise (FORMARAN (2021)).

3. *Spider-Man PS4 - Spiderman vs 6 Villain Bosses Epic Cutscene (Marvel's Spiderman Game 2018) PS4 Pro* (Aesthetics (2018))

   O terceiro vídeo consiste num pequeno vídeo de cenas *high motion*. Possui uma duração total de 03:17 minutos (ver Figura 17).

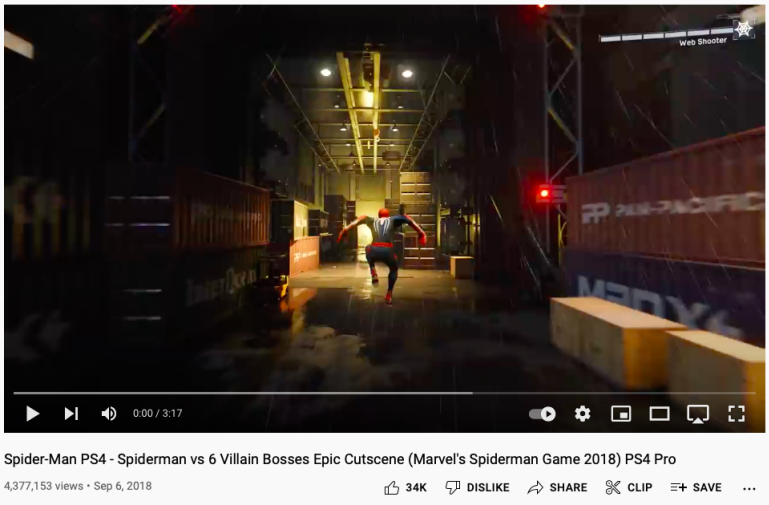


Figura 17: Terceiro vídeo de análise (Aesthetics (2018)).

Na Figura 18 representa-se uma síntese dos vídeos a utilizar neste trabalho. É importante referir que a resolução a utilizar será a mais baixa possível em todos os vídeos, de modo a que as oscilações da rede não interfiram na comparação entre a versão aplicacional e a versão *browser*. Por este motivo, a *Frame Rate* não será a máxima nas capturas com cenas de *high motion*, passando a rondar os 30*fps*. No caso do vídeo com cenas *low motion*, mantém-se a *Frame Rate* (24*fps*).

Vídeo	*Motion Scene*	Duração (min)	*Frame Rate* (máxima)
1	*High* e *Low*	10:34	60fps
2	*Low*	02:02	24fps
3	*High*	03:17	60fps

Figura 18: Resumo dos vídeos utilizados.

5.3.2 *Processo de Captura: Aplicação e* Browser

Relativamente aos processos de captura de tráfego, após uma breve análise sobre a utilização do YouTube pela aplicação e pelo *browser*, concluiu-se que não existem diferenças significativas entre ambas. Por este motivo, e pressupondo que todos os passos descritos na Secção 3.3 foram cumpridos, os processos de recolha passarão por:

- **Aplicação:**
  1. entrar na aplicação YouTube;
  2. limpar a *cache*;
  3. ativar a funcionalidade de navegação anónima, para que o histórico de utilização não tenha impacto na recolha;
  4. sair da aplicação e fechá-la;
  5. iniciar a captura no dispositivo de captura, através do *Wireshark*;
  6. abrir a aplicação;
  7. procurar o vídeo desejado pela barra de pesquisas;
  8. selecionar o vídeo e definir a qualidade do mesmo para a mais baixa possível (144p) - oscilações na qualidade de reprodução têm um grande impacto no tráfego gerado;
  9. esperar até o vídeo terminar;
  10. finalizar o processo de captura no *Wireshark*.
- ***Browser***:
  1. abrir o *browser*;
  2. fechar todos os separadores abertos e limpar a *cache*;
  3. abrir o navegador anónimo, para não existir ruído do histórico anterior;
  4. iniciar a captura no dispositivo de captura, através do *Wireshark*;
  5. abrir o *site* YouTube;

6. procurar o vídeo desejado pela barra de pesquisas;
7. selecionar o vídeo e definir a qualidade do mesmo para a mais baixa possível (144p);
8. esperar até o vídeo terminar;
9. finalizar o processo de captura no *Wireshark*.

5.3.3 *Métricas*

Antes de mais, é importante notar que o YouTube é uma plataforma que possui anúncios, que podem aparecer antes/durante/depois dos vídeos. Caso ocorram durante as recolhas, estes não serão englobados, uma vez que o seu tráfego será descartado antes de se efetuar qualquer tipo de análise.

Além disso, é crucial que a qualidade dos vídeos seja colocada a mais baixa possível, neste caso 144p, de forma a que as recolhas não revelem o impacto dos processos comuns de *buffering*. Deste modo, a comparação entre versão aplicacional e *browser* será mais realista, uma vez que estes processos não vão contribuir para o enviesamento de resultados.

Assim, as métricas a utilizar serão de acordo com o abordado na Secção 3.2, sendo algumas respetivas a análises da carga de trabalho da rede, outras para caracterizar a mesma. Destacam-se as seguintes ao nível da QoS:

- número total de pacotes;
- tamanho dos pacotes;
- quantidade de tráfego total;
- distribuição de protocolos em uso;
- largura de banda;
- RTT;
- débito/*throughput*;

Ao nível da avaliação de QoE, pretende-se efetuar uma previsão inspirada na medição MOS, recorrendo a métricas de QoS. Para tal, desenvolveu-se a componente de ML descrita na Secção 4. Nesta componente, utilizaram-se métricas comuns de QoS, mais particularmente relacionadas com o protocolo TCP - número de pacotes, *delay*, *jitter*, *loss*, retransmissões - para prever a qualidade de experiência por parte do utilizador. Esta reflete-se através de três classes, que representam as quebras durante o *streaming de vídeo*.

5.4 SUMÁRIO

Neste capítulo apresentou-se o serviço a ser analisado, nomeadamente o YouTube. Para a recolha de dados foram selecionados os vídeos a utilizar, para capturas de tráfego na versão aplicacional e na versão *browser*.

Posteriormente, as regras para o processo de captura foram definidas, sendo o principal objetivo reduzir ao máximo o impacto do ruído e oscilações na qualidade de serviço, que prejudiquem a comparação aplicação e *web*.

Por fim, apresentaram-se as métricas a analisar, primeiro numa vertente de QoS e depois QoE, a partir do modelo de ML previamente desenvolvido.

6

ANÁLISE DE RESULTADOS

Nesta secção apresentam-se os resultados da análise às capturas de dados recolhidas. Inicialmente será apresentada informação sobre os vídeos alvo de estudo (ver Capítulo 5). Depois serão discutidas algumas métricas obtidas.

É importante relembrar que este trabalho se centra na caracterização de tráfego de aplicações em dispositivos móveis, comparando também a versão aplicacional e *browser* da plataforma. Por este motivo, algumas métricas serão apresentadas em termo de comparação entre os valores gerados pelos mesmos vídeos nas diferentes versões.

6.1 VOLUME DOS VÍDEOS EM ANÁLISE

Existem várias propriedades dos vídeos que podem diferenciar substancialmente o tráfego gerado durante o seu *streaming*.

Durante o *streaming* de vídeos no YouTube são utilizadas ferramentas de codificação e descodificação, os designados *codecs*. Estes têm a capacidade de comprimir/codificar e armazenar imagem e áudio de vídeos e, posteriormente, reverter o processo para reprodução pelo utilizador. Os *codecs* podem variar de acordo a duração do vídeo, qualidade selecionada pelo utilizador e até pela via pela qual o vídeo é reproduzido (aplicação ou *browser*).

Assim, numa primeira análise torna-se interessante identificar os *codecs* dos vídeos a analisar, definidos no Capítulo 5. Para tal, utilizou-se a ferramenta *open source youtube-dl* (youtube dl), que permite efetuar o *download* dos vídeos do YouTube e obter informações sobre os *codecs*.

Nas Figuras 19, 20 e 21 apresentam-se os resultados para os três vídeos selecionados, na versão aplicacional e *browser*. É importante relembrar que se utilizou uma qualidade de 144p, para que a comparação entre aplicação e *browser* não fosse enviesada por oscilações na qualidade da rede nos momentos de captura.

```
% youtube-dl -F https://youtu.be/aqz-KE-bpKQ
[youtube] aqz-KE-bpKQ: Downloading webpage
[info] Available formats for aqz-KE-bpKQ:
format code  extension  resolution note
249          webm       audio only tiny   49k , webm_dash container, opus @ 49k (48000Hz), 3.75MiB
250          webm       audio only tiny   65k , webm_dash container, opus @ 65k (48000Hz), 4.95MiB
251          webm       audio only tiny  128k , webm_dash container, opus @128k (48000Hz), 9.73MiB
140          m4a        audio only tiny  129k , m4a_dash container, mp4a.40.2@129k (44100Hz), 9.80MiB
256          m4a        audio only tiny  195k , m4a_dash container, mp4a.40.5@195k (24000Hz), 14.76MiB
258          m4a        audio only tiny  387k , m4a_dash container, mp4a.40.2@387k (48000Hz), 29.34MiB
160          mp4        256x144    144p   57k , mp4_dash container, avc1.4d400c@  57k, 30fps, video only, 4.32MiB
278          webm       256x144    144p   75k , webm_dash container, vp9@  75k, 30fps, video only, 5.72MiB
394          mp4        256x144    144p   87k , mp4_dash container, av01.0.00M.08@  87k, 30fps, video only, 6.64MiB
133          mp4        426x240    240p  124k , mp4_dash container, avc1.4d4015@ 124k, 30fps, video only, 9.44MiB
242          webm       426x240    240p  148k , webm_dash container, vp9@ 148k, 30fps, video only, 11.24MiB
395          mp4        426x240    240p  149k , mp4_dash container, av01.0.00M.08@ 149k, 30fps, video only, 11.33MiB
134          mp4        640x360    360p  244k , mp4_dash container, avc1.4d401e@ 244k, 30fps, video only, 18.51MiB
396          mp4        640x360    360p  272k , mp4_dash container, av01.0.01M.08@ 272k, 30fps, video only, 20.64MiB
243          webm       640x360    360p  312k , webm_dash container, vp9@ 312k, 30fps, video only, 23.67MiB
135          mp4        854x480    480p  377k , mp4_dash container, avc1.4d401f@ 377k, 30fps, video only, 28.55MiB
397          mp4        854x480    480p  459k , mp4_dash container, av01.0.04M.08@ 459k, 30fps, video only, 34.74MiB
244          webm       854x480    480p  522k , webm_dash container, vp9@ 522k, 30fps, video only, 39.54MiB
247          webm       1280x720   720p  956k , webm_dash container, vp9@ 956k, 30fps, video only, 72.38MiB
136          mp4        1280x720   720p 1188k , mp4_dash container, avc1.4d401f@1188k, 30fps, video only, 89.87MiB
398          mp4        1280x720   720p60 1238k , mp4_dash container, av01.0.08M.08@1238k, 60fps, video only, 93.68MiB
302          webm       1280x720   720p60 1567k , webm_dash container, vp9@1567k, 60fps, video only, 118.57MiB
298          mp4        1280x720   720p60 2009k , mp4_dash container, avc1.4d4020@2009k, 60fps, video only, 152.01MiB
399          mp4        1920x1080  1080p60 2074k , mp4_dash container, av01.0.09M.08@2074k, 60fps, video only, 156.96MiB
303          webm       1920x1080  1080p60 2572k , webm_dash container, vp9@2572k, 60fps, video only, 194.57MiB
299          mp4        1920x1080  1080p60 3418k , mp4_dash container, avc1.64002a@3418k, 60fps, video only, 258.62MiB
400          mp4        2560x1440  1440p60 6128k , mp4_dash container, av01.0.12M.08@6128k, 60fps, video only, 463.57MiB
308          webm       2560x1440  1440p60 6872k , webm_dash container, vp9@6872k, 60fps, video only, 519.89MiB
401          mp4        3840x2160  2160p60 10843k , mp4_dash container, av01.0.13M.08@10843k, 60fps, video only, 820.28MiB
315          webm       3840x2160  2160p60 19509k , webm_dash container, vp9@19509k, 60fps, video only, 1.44GiB
18           mp4        640x360    360p  564k , avc1.42001E, 30fps, mp4a.40.2 (44100Hz), 42.71MiB
22           mp4        1280x720   720p 1317k , avc1.64001F, 30fps, mp4a.40.2 (44100Hz) (best)
```

Figura 19: Informações do *download* do vídeo 1.

```
% youtube-dl -F https://youtu.be/6GTWNZlEcHU
[youtube] 6GTWNZlEcHU: Downloading webpage
[info] Available formats for 6GTWNZlEcHU:
format code  extension  resolution note
249          webm       audio only tiny   56k , webm_dash container, opus @ 56k (48000Hz), 840.17KiB
250          webm       audio only tiny   73k , webm_dash container, opus @ 73k (48000Hz), 1.08MiB
140          m4a        audio only tiny  129k , m4a_dash container, mp4a.40.2@129k (44100Hz), 1.90MiB
251          webm       audio only tiny  143k , webm_dash container, opus @143k (48000Hz), 2.10MiB
160          mp4        256x144    144p   67k , mp4_dash container, avc1.4d400c@  67k, 24fps, video only, 1008.98KiB
278          webm       256x144    144p   78k , webm_dash container, vp9@  78k, 24fps, video only, 1.15MiB
242          webm       426x240    240p  146k , webm_dash container, vp9@ 146k, 24fps, video only, 2.14MiB
133          mp4        426x240    240p  164k , mp4_dash container, avc1.4d4015@ 164k, 24fps, video only, 2.40MiB
243          webm       640x360    360p  270k , webm_dash container, vp9@ 270k, 24fps, video only, 3.96MiB
134          mp4        640x360    360p  312k , mp4_dash container, avc1.4d401e@ 312k, 24fps, video only, 4.58MiB
244          webm       854x480    480p  501k , webm_dash container, vp9@ 501k, 24fps, video only, 7.34MiB
135          mp4        854x480    480p  807k , mp4_dash container, avc1.4d401e@ 807k, 24fps, video only, 11.81MiB
247          webm       1280x720   720p 1019k , webm_dash container, vp9@1019k, 24fps, video only, 14.92MiB
136          mp4        1280x720   720p 1625k , mp4_dash container, avc1.64001f@1625k, 24fps, video only, 23.79MiB
248          webm       1920x1080  1080p 1727k , webm_dash container, vp9@1727k, 24fps, video only, 25.28MiB
137          mp4        1920x1080  1080p 3143k , mp4_dash container, avc1.640028@3143k, 24fps, video only, 46.01MiB
271          webm       2560x1440  1440p 5457k , webm_dash container, vp9@5457k, 24fps, video only, 79.88MiB
313          webm       3840x2160  2160p 10963k , webm_dash container, vp9@10963k, 24fps, video only, 160.48MiB
18           mp4        640x360    360p  441k , avc1.42001E, 24fps, mp4a.40.2 (44100Hz)
22           mp4        1280x720   720p 1800k , avc1.64001F, 24fps, mp4a.40.2 (44100Hz) (best)
```

Figura 20: Informações do *download* do vídeo 2.

```
% youtube-dl -F https://youtu.be/e2bzrGxTFCk
[youtube] e2bzrGxTFCk: Downloading webpage
[info] Available formats for e2bzrGxTFCk:
format code  extension  resolution note
249          webm       audio only tiny   44k , webm_dash container, opus @ 44k (48000Hz), 1.04MiB
250          webm       audio only tiny   58k , webm_dash container, opus @ 58k (48000Hz), 1.38MiB
251          webm       audio only tiny  117k , webm_dash container, opus @117k (48000Hz), 2.77MiB
140          m4a        audio only tiny  129k , m4a_dash container, mp4a.40.2@129k (44100Hz), 3.06MiB
278          webm       256x144    144p   88k , webm_dash container, vp9@  88k, 30fps, video only, 2.08MiB
160          mp4        256x144    144p   89k , mp4_dash container, avc1.4d400c@  89k, 30fps, video only, 2.11MiB
242          webm       426x240    240p  194k , webm_dash container, vp9@ 194k, 30fps, video only, 4.58MiB
133          mp4        426x240    240p  209k , mp4_dash container, avc1.4d4015@ 209k, 30fps, video only, 4.95MiB
243          webm       640x360    360p  356k , webm_dash container, vp9@ 356k, 30fps, video only, 8.41MiB
134          mp4        640x360    360p  445k , mp4_dash container, avc1.4d401e@ 445k, 30fps, video only, 10.50MiB
244          webm       854x480    480p  636k , webm_dash container, vp9@ 636k, 30fps, video only, 15.01MiB
135          mp4        854x480    480p  693k , mp4_dash container, avc1.4d401f@ 693k, 30fps, video only, 16.35MiB
298          mp4        1280x720   720p60 1277k , mp4_dash container, avc1.4d4020@1277k, 60fps, video only, 30.14MiB
247          webm       1280x720   720p 1347k , webm_dash container, vp9@1347k, 30fps, video only, 31.78MiB
302          webm       1280x720   720p60 1990k , webm_dash container, vp9@1990k, 60fps, video only, 46.95MiB
136          mp4        1280x720   720p 2768k , mp4_dash container, avc1.4d401f@2768k, 30fps, video only, 65.31MiB
303          webm       1920x1080  1080p60 3851k , webm_dash container, vp9@3851k, 60fps, video only, 90.86MiB
299          mp4        1920x1080  1080p60 4474k , mp4_dash container, avc1.64002a@4474k, 60fps, video only, 105.55MiB
18           mp4        640x360    360p  681k , avc1.42001E, 30fps, mp4a.40.2 (44100Hz), 16.08MiB
22           mp4        1280x720   720p 2897k , avc1.64001F, 30fps, mp4a.40.2 (44100Hz) (best)
```

Figura 21: Informações do *download* do vídeo 3.

Na Tabela 9 apresentam-se os resultados relativos ao volume de dados de áudio e vídeo dos vídeos selecionados para análise.

Vídeo	Plataforma	Volume Áudio (MB)	Volume Vídeo (MB)	Volume Total (MB)
1	Aplicação	9.8	5.72	15.52
	Browser	9.73	5.72	15.45
2	Aplicação	1.90	1.15	3.05
	Browser	2.10	1.15	3.25
3	Aplicação	3.06	2.08	5.14
	Browser	2.77	2.08	4.85

Tabela 9: Volume de dados obtido (vídeo 1, 2 e 3).

6.2 ANÁLISE DAS CAPTURAS

Após se obter o volume de dados, segue-se uma exploração dos dados das capturas efetuadas. Estas capturas englobam sessões de *streaming* no YouTube dos três vídeos utilizados (ver Secção 5.3.1), na versão aplicacional e versão *browser*. É importante salientar que os processos de captura seguiram os passos descritos na Secção 5.3.2, em ambas as versões.

Como já referido previamente, estas capturas dizem respeito aos momentos de *streaming* de vídeo. Nenhuma captura inclui a ocorrência de qualquer tipo de publicidade, muitas vezes existente nos vídeos.

Neste trabalho são apresentados seis cenários de captura, sendo cada um deles relativo a apenas uma captura de um vídeo. Para tornar mais simples a leitura de informação, os seis cenários apresentados foram divididos entre A e F. Na Figura 22 apresentam-se os mesmos.

Cenário	Vídeo	Contexto
A	1	Aplicação
B	1	*Browser*
C	2	Aplicação
D	2	*Browser*
E	3	Aplicação
F	3	*Browser*

Figura 22: Cenários de captura.

6.2.1 *Resultados da Análise Estática*

Para iniciar a análise das capturas efetuadas, começou-se por obter algumas métricas relativas aos pacotes e volume dos mesmos.

Na Figura 23 apresentam-se resultados relativo ao número de pacotes, tamanho médio dos pacotes, número total de *bytes* e débito para cada cenário de captura.

Cenário	Nº Pacotes	Tamanho Médio Pacotes (Bytes)	Nº Total de Bytes (MB)	Débito (Mbps)
A	17906	1103	19.758	0.239
B	13905	939	12.795	0.162
C	5702	1017	5.801	0.324
D	4109	763	3.134	0.176
E	7754	1080	8.374	0.333
F	6944	919	6.379	0.244

Figura 23: Resultados da primeira análise sobre os pacotes e volume de dados.

Observando os valores apresentados, pode concluir-se que:

- para o mesmo vídeo, a versão aplicacional apresenta sempre um maior número de pacotes do que a versão *browser*;
- o tamanho médio de pacotes é superior na versão aplicacional. Enquanto os pacotes relativos à utilização do YouTube no *browser* não ultrapassam os 1000 *bytes*, na plataforma aplicacional este valor é ultrapassado;
- em consequência dos dois pontos anteriores, a versão aplicacional apresenta um total de *bytes* e débito superiores, quando comparada com o mesmo vídeo reproduzido no *browser*;
- no contexto aplicacional, os pacotes seguem um padrão mais linear, por exemplo, o tamanho médio de pacotes na versão *browser* varia entre intervalos maiores e na *app* mantém-se dentro dos mesmos valores;
- a diferença entre vídeos de *high motion* e *low motion* reflete-se mais na utilização do *browser*, principalmente no tamanho médio dos pacotes.

Os protocolos em uso foram também alvo de análise, nomeadamente o TCP e UDP. Na Figura 24 estão representados os números de pacotes referentes a cada um destes protocolos, e a percentagem que representam face ao número total de pacotes verificado nessa sessão.

Cenário	Nº Pacotes TCP	Nº Pacotes UDP
A	1061 (5.9%)	16776 (93.7%)
B	2275 (16.4%)	11588 (83.3%)
C	191 (3.3%)	5443 (95.5%)
D	368 (9%)	3733 (90.8%)
E	415 (5.4%)	7321 (94.4%)
F	824 (11.9%)	6102 (87.9%)

Figura 24: Resultados da análise sobre os protocolos TCP e UDP.

Outros protocolos para além do TCP e UDP são utilizados. No entanto, a sua presença é quase nula e, por isso, não foram representados, sendo o DNS (*Domain Name System*) o que mais se destaca entre estes protocolos.

Para uma melhor compreensão e comparação entre estes valores, apresentam-se os mesmos representados graficamente na Figura 25.

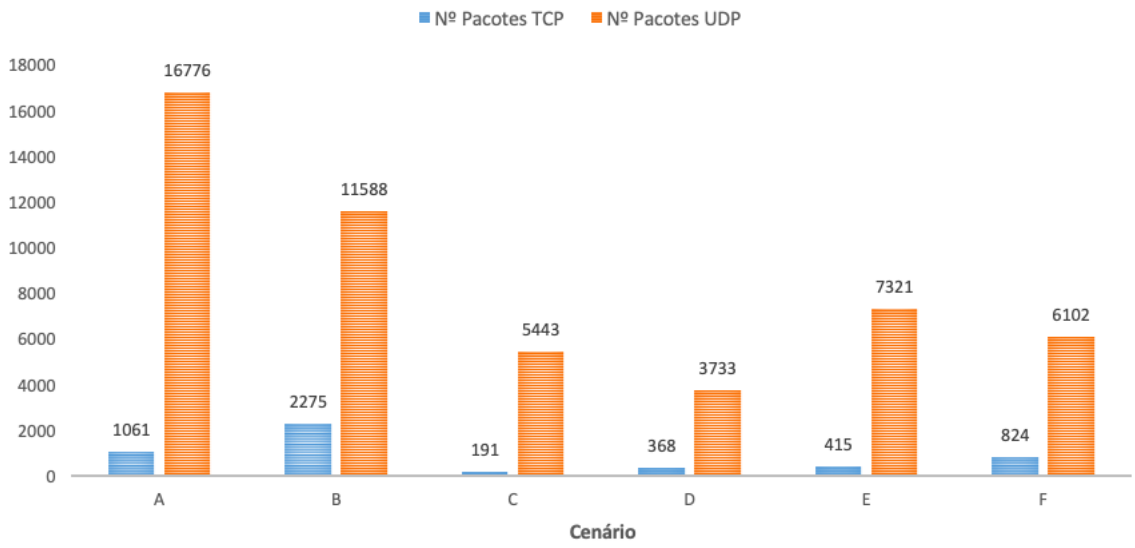


Figura 25: Representação da utilização dos protocolos TCP e UDP.

Analisando os valores obtidos não se afere um comportamento distinto entre aplicação e *browser*. Torna-se evidente que o protocolo UDP representa uma clara maioria comparativamente ao TCP.

No sentido de se analisar mais particularmente o uso destes protocolos, fez-se uma pesquisa sobre a utilização do QUIC (*Quick UDP Internet Connection*) nos pacotes UDP.

O QUIC diz respeito a um protocolo da camada de transporte de rede, que procura manter conexões rápidas com maior segurança. Enquanto o TCP oferece um maior controlo e confiança no envio e receção de dados, o UDP torna-se mais vantajoso pela sua rapidez. Assim, o QUIC representa uma combinação dos pontos mais vantajosos de cada um, utilizando passos do protocolo TCP, com uma latência mais reduzida e consequentemente conexões mais rápidas.

Na Figura 26 apresentam-se os valores referentes ao uso do QUIC nos pacotes UDP. Por exemplo, no caso do cenário A, dos 16776 pacotes UDP, 15701 são QUIC e, portanto, 87.7% de todos os pacotes utilizam QUIC.

Cenário	QUIC
A	15701 (87.7%)
C	4507 (79%)
E	6284 (81%)

Figura 26: Representação da utilização do QUIC.

Como se pode verificar, o uso do QUIC está presente apenas na versão aplicacional. Na versão *browser*, este não é utilizado. O QUIC tem crescido exponencialmente e o seu uso no YouTube é conhecido. No entanto, na versão *browser* o seu uso não foi registado em nenhuma captura efetuada no decorrer deste trabalho.

No decorrer desta análise, procedeu-se a uma caracterização média dos RTT. Para tal utilizou-se o *Tstat*, ferramenta que permite obter o RTT médio do cliente e do servidor de uma conexão TCP completa.

Para esta análise, selecionaram-se os IP's dos cinco servidores com que se estabeleceram maior número de conexões. Infelizmente, em nenhuma captura referente aos vídeos 2 e 3, se obtiveram conexões TCP completas suficientes para proceder a esta análise.

No entanto, na Figura 27 apresentam-se os valores relativos aos cenários A e B, reproduzindo o vídeo 1 na aplicação e *browser*.

Vídeo	Contexto	Cliente	Servidor	RTT médio cliente (ms)	RTT médio servidor (ms)
1	Aplicação (Cenário A)	192.168.2.2	216.58.215.174	16.646	8.939
			157.240.212.16	8.565	12.528
			142.250.185.10	17.011	22.751
			142.250.201.78	18.245	97.276
			40.101.137.34	39.960	10.184
	Browser (Cenário B)	192.168.2.2	212.113.172.17	5.653	5.517
			142.250.178.182	15.770	14.624
			157.240.212.16	8.170	12.520
			216.58.215.142	19.590	12.387
			13.225.15.250	7.796	5.113

Figura 27: RTT - servidor e cliente.

Através da observação dos valores de RTT, conclui-se que estes tendem a ser menores na versão *browser* comparativamente à plataforma aplicacional. Um comportamento que se verifica em relação ao RTT médio do servidor e do cliente também. Notando que para além destes dois cenários, outras capturas a vídeos demonstraram seguir este padrão.

6.2.2 *Resultados da Análise Temporal*

Com o objetivo de perceber o comportamento da sessão relativamente a métricas abordadas na análise estática mas numa vertente evolutiva, obteve-se a partir do *Wireshark* gráficos que descrevem o fluxo de pacotes e *bytes* transmitidos por segundo. Apesar deste último gráfico não representar diretamente a largura de banda, permite uma observação indireta da mesma, uma vez que a largura de banda está associada à a capacidade máxima de transmissão da rede em determinado momento.

Nas Figuras 28 e 29, representam-se graficamente estes valores para o cenário A e B, respetivamente.

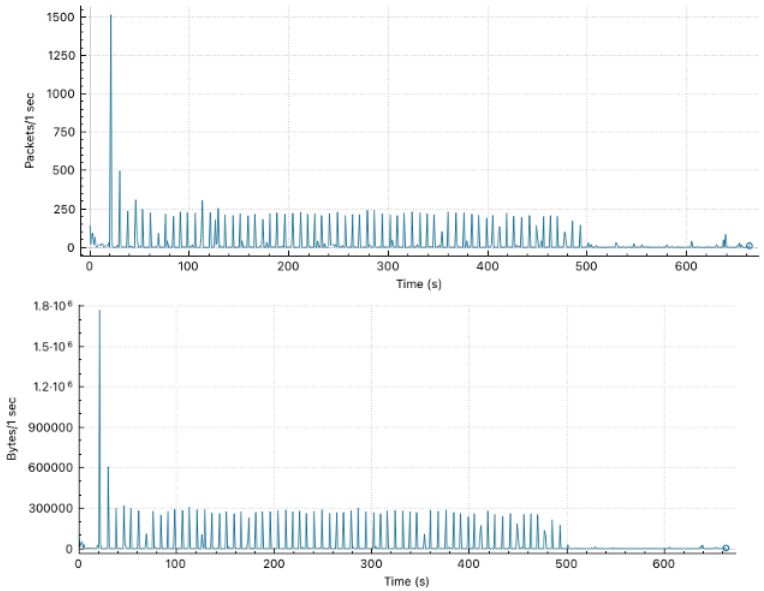


Figura 28: Análise temporal do cenário A.

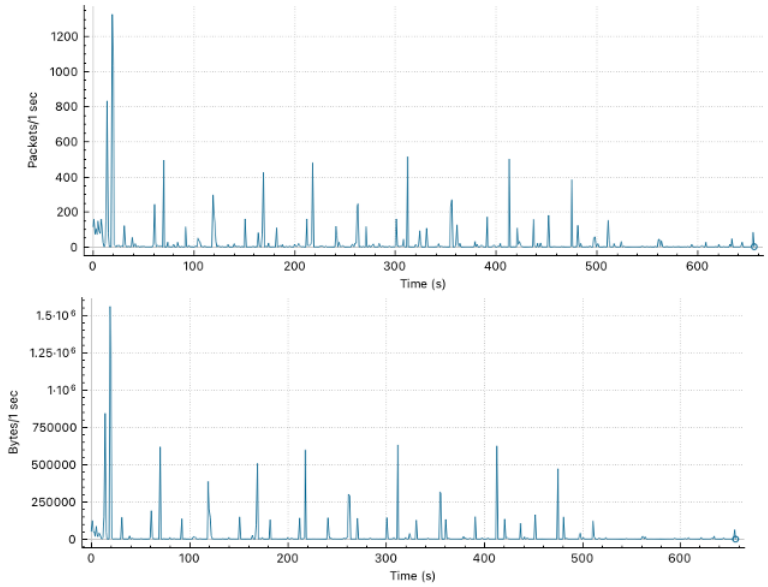


Figura 29: Análise temporal do cenário B.

Nas Figura 28 e 29 é possível visualizar a análise do mesmo vídeo em contextos diferentes (aplicação e *browser*). Observando a distribuição de valores representada, torna-se clara uma diferença no comportamento do *streaming* de vídeo no YouTube em diferentes versões. Inicialmente, ambas têm o mesmo comportamento, com a ocorrência de um pico de dados. No entanto, ao longo da sessão, na aplicação mantém-se um fluxo com valores que seguem um padrão semelhante. Contrariamente, no *browser*, existe maior discrepância entre valores.

Nas Figuras 30, 31, 32 e 33 está representada a mesma análise para os restantes cenários.

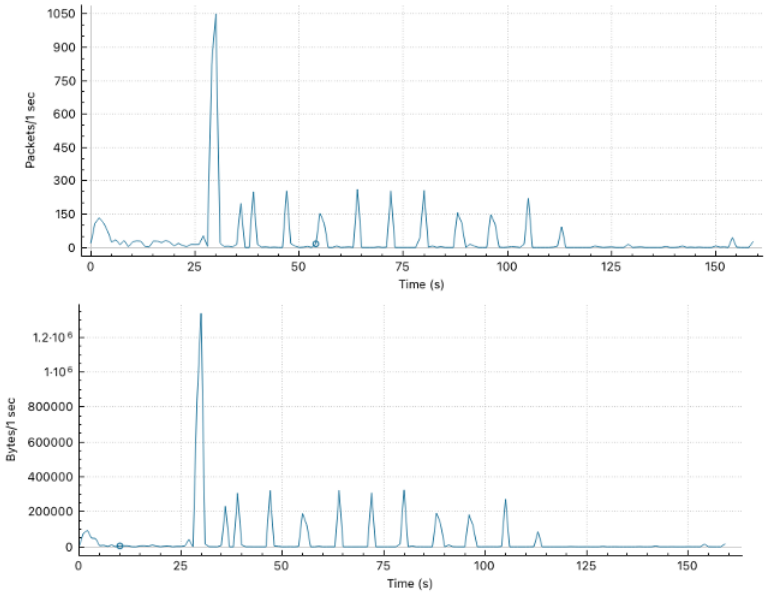


Figura 30: Análise temporal do cenário C.

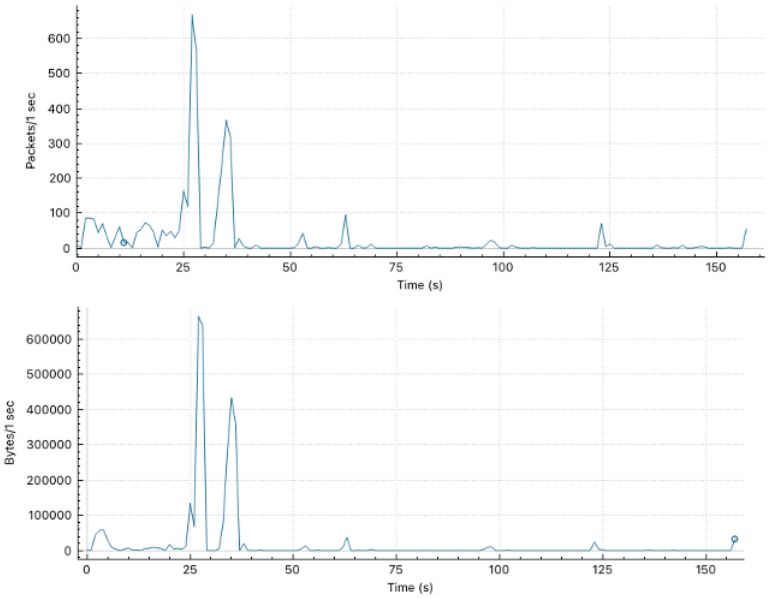


Figura 31: Análise temporal do cenário D.

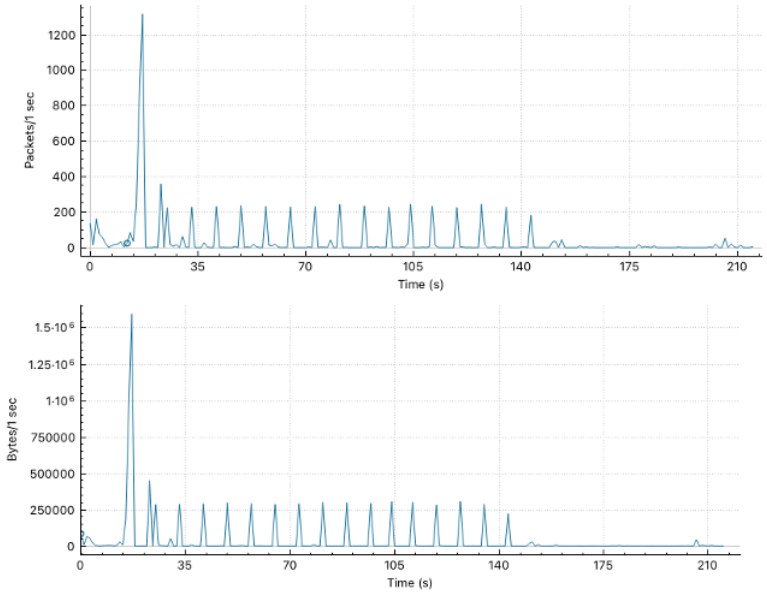


Figura 32: Análise temporal do cenário E.

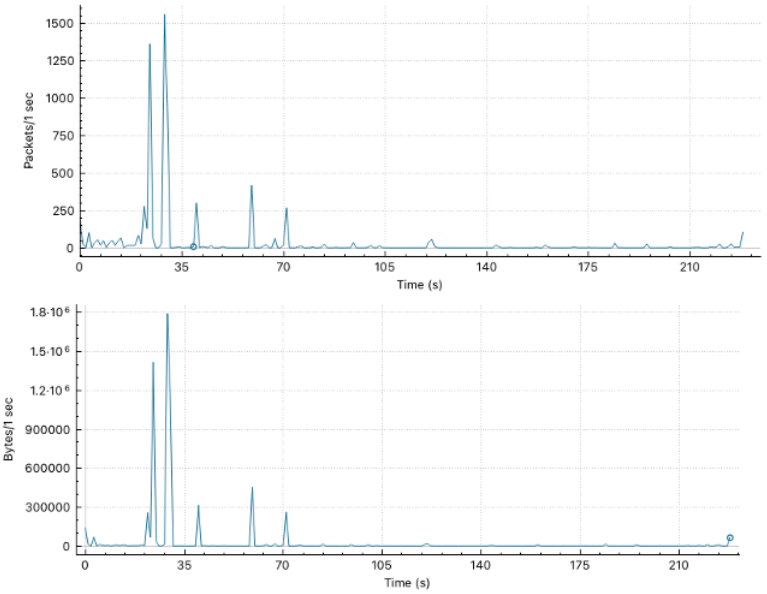


Figura 33: Análise temporal do cenário F.

Como se pode observar, para os restantes casos verifica-se o mesmo dos cenários anteriores. Na versão aplicacional, parece existir um comportamento mais linear, com a ocorrência de iguais picos de dados em determinados intervalos de tempo. Na utilização da plataforma através do *browser*, não há regularidade nos intervalos de tempo nem nos valores destes picos de dados.

Tanto na versão aplicacional como na versão *browser* ocorrem picos de dados nos primeiros momentos do *streaming*, algo que deverá estar relacionado com o estabelecimento da conexão e transferência de dados inicial.

6.2.3 *Aplicação do modelo de ML*

Como apresentado no Capítulo 4, desenvolveu-se um modelo capaz de prever QoE com base em fatores de QoS. Neste caso, a QoE está representada e dividida por três classes. Relembrando:

- Classe 0 - *StallLabel* = *NoStall*, quando *RebufferingRatio* = 0 e não há ocorrência de *stalling*;
- Classe 1 - *StallLabel* = *MildStall*, quando *RebufferingRatio* varia entre]0,0.1];
- Classe 2 - *StallLabel* = *SevereStall*, quando *RebufferingRatio* superior a 0.1;

Como referido, para desenvolver o modelo utilizou-se um *dataset* público dada a incapacidade de criação de um *dataset* robusto e alargado. Como tal, é esperado que exista uma margem de erro na previsão do modelo com base nos valores destas capturas.

Cada variável necessária para o modelo foi obtida através da análise gerada pelo *Tstat*. O *Tstat* cria ficheiros, denominados *LOG Files*, onde se representam resultados para cada fluxo existente na conexão. Um dos ficheiros gerados pela ferramenta, diz respeito às conexões TCP completas e divide métricas por servidor e cliente. Tendo em consideração que as variáveis utilizadas no modelo dizem respeito a métricas TCP por sessão, para as capturas efetuadas neste trabalho, estas métricas podem ser obtidas através da agregação dos vários fluxos analisados pelo *Tstat*.

Optou-se por prever uma classificação apenas para o cenário A e B, porque estes possuem um número elevado de conexões TCP completas. Para os outros cenários não se verificam conexões TCP completas suficientes.

Para aplicar o modelo aos cenários, são necessários os dados da sessão relativos às métricas que se seguem, onde também se explica quais os valores a usar em cada, assumindo que alguns dos valores não se conseguem obter das capturas diretamente e, portanto, recorreu-se a aproximações.

- *TCPOutputPacket* - assume-se o número de pacotes TCP do cliente;
- *TCPInputPacket* - utiliza-se o número de pacotes TCP do servidor;
- *TCPOutputDelay* - por simplificação utiliza-se o RTT do cliente. Apesar de existir uma diferença entre os dois, uma vez que o RTT incluiu o envio do pacote e a receção do respetivo *acknowledgment*, não foi possível obter concretamente este valor e, portanto, optou-se por esta aproximação;
- *TCPInputDelay* - seguindo a lógica do ponto anterior, assume-se o RTT do servidor;
- *TCPOutputJitter* - aproximação a partir da variação dos RTTs do cliente;

- *TCPInputJitter* - aproximação a partir da variação dos RTTs do servidor;
- *TCPOutputPloss* - taxa obtida a partir do cálculo do número de pacotes retransmitidos em função do número de pacotes total do cliente;
- *TCPInputPloss* - mesmo processo do ponto anterior mas do servidor;
- *TCPInputRetrans* - utiliza-se o número de pacotes TCP retransmitidos do servidor.

Após se agregar os valores destas variáveis de todos os fluxos TCP (ver Figuras 34 e 36), obteve-se os resultados para cada sessão (ver Figuras 35 e 37), aos quais se torna possível aplicar o modelo desenvolvido.

	TCPOutputPacket	TCPOutputDelay	TCPOutputJitter	TCPOutputPloss	TCPInputPacket	TCPInputDelay	TCPInputJitter	TCPInputPloss	TCPInputRetrans
0	390	25.58385	1.908227	0.041667	364	23.457288	28.036423	0.028083	8

Figura 34: Resultados das variáveis do cenário A.

```
A classe prevista para o cenário A é [2].
```

Figura 35: Classe prevista para o cenário A.

	TCPOutputPacket	TCPOutputDelay	TCPOutputJitter	TCPOutputPloss	TCPInputPacket	TCPInputDelay	TCPInputJitter	TCPInputPloss	TCPInputRetrans
1	833	33.385195	1.897605	0.041667	823	10.371076	16.218242	0.029864	16

Figura 36: Resultados das variáveis do cenário B.

```
A classe prevista para o cenário B é [1].
```

Figura 37: Classe prevista para o cenário B.

Como se pode verificar, para a versão aplicacional a classe prevista foi a 2 e para a versão *browser* a 1. No entanto, é importante ter em consideração que os dados utilizados no treino e teste do modelo, dizem respeito a tráfego gerado em ambiente controlado. Neste caso, são valores capturados de uma rede real. Esta diferença pode ter impacto na previsão do modelo.

Outro fator que pode impactar a previsão do modelo é a baixa qualidade de reprodução utilizada nas capturas. Relembrando que esta baixa qualidade foi definida para que oscilações presentes na rede nos momentos de capturas, não impactassem na comparação entre *aplicação* e *browser*.

6.3 YOUTUBE: APLICAÇÃO E WEB

A análise efetuada no decorrer deste capítulo permite observar que existem algumas diferenças ao longo da sessão no YouTube versão aplicacional e versão *browser*.

As análises estáticas com métricas relativas a *workload*, demonstram que a versão aplicacional possui, no geral, um maior uso de dados do que a versão *web*, com valores como o débito e o número total de *bytes* das sessões a serem superiores na *app*.

Outra das principais diferenças detetadas consiste na utilização de QUIC apenas na versão aplicacional, algo que pode impactar os resultados discutidos anteriormente.

No que diz respeito a QoE, o modelo de previsão atribuiu melhor classe à sessão *browser*, apesar de existir uma margem de erro associada.

No global, a utilização do YouTube pelo *browser* apresenta um menor consumo de dados. Por este motivo, se o utilizador estiver a utilizar um pacote de dados móveis limitado, recomenda-se o uso *web*. No entanto, se estiver conectado a *Wi-Fi* ou possuir dados móveis ilimitados, a versão aplicacional é a melhor opção uma vez que é desenvolvida de acordo com as necessidades da plataforma, com o objetivo de melhorar a eficiência e experiência do utilizador, com uma interface mais orientada ao mesmo.

6.4 SUMÁRIO

Neste capítulo foram analisados os resultados obtidos a partir da captura dos vídeos do YouTube previamente selecionados.

Numa fase inicial procedeu-se a uma análise do volume de dados e, posteriormente, apresentaram-se os resultados a nível de *workload* e QoS. Estes resultados foram divididos por diferentes cenários correspondentes aos diferentes casos de estudo.

De seguida, realizou-se a extração das métricas necessárias para aplicação do modelo de ML, para obter uma previsão de QoE com base em parâmetros de QoS das capturas efetuadas.

Finalmente, concluiu-se com uma sucinta comparação entre a utilização do YouTube na versão aplicacional e *browser*, incluindo uma recomendação ao utilizador.

7

CONCLUSÃO

Ao longo da realização deste estudo foram recolhidas informações cruciais para fundamentar o desenvolvimento do mesmo. Devido ao enorme crescimento tecnológico despoletado nos últimos anos, as quantidades de tráfego que circulam nas redes de computadores são colossais, tornando imperativa a necessidade de conhecimento sobre as mesmas.

Com a realização do estudo de arte, foi possível reunir aspetos fundamentais para monitorizar a rede, proceder a recolhas de tráfego e analisar os dados obtidos. Assim, tornou-se viável desenvolver uma arquitetura de captura, que permitisse a obtenção de valores o mais reais possível. Este passo foi indispensável para que os resultados a obter sejam confiáveis. A par disso, com estes estudos surgiu inspiração para a aplicação de ML e obtenção de dados passíveis de utilizar para treino e teste do modelo.

De seguida, iniciou-se então o desenvolvimento do modelo de previsão de acordo com as possibilidades existentes, nomeadamente a nível de dados. Várias decisões fundamentadas foram tomadas para seleção de técnicas de amostragem, tratamento de dados e estimadores, até se obter um modelo final otimizado e pronto a prever QoE com base em métricas de QoS.

Uma análise ao mercado de aplicações motivou a seleção do serviço a analisar - YouTube. Com esta decisão efetuada, foi necessário observar o comportamento da sua versão aplicacional e versão *browser*. Deste modo, identificaram-se as diferenças existentes entre as duas versões do mesmo serviço, permitindo esboçar o processo de aquisição de dados. Com o serviço selecionado, os vídeos em análise foram então alvo de capturas.

Por fim, apresentaram-se os resultados obtidos das capturas. Estes resultados consistiram numa análise a nível de métricas estáticas de *workload* que permitiram observar que a versão aplicacional possui um maior consumo de dados. Além disso, análises da utilização dos protocolos TCP e UDP foram realizadas, permitindo concluir que na aplicação, uma grande parte dos pacotes UDP dizem respeito ao QUIC, algo que não se verifica na plataforma *web*. Por fim, uma análise do RTT demonstrou que estes tendem a ser superiores na *app*.

Uma análise temporal foi também realizada, revelando uma diferença entre o comportamento da aplicação e do *browser* no decorrer das sessões de *streaming* de vídeo. Enquanto na versão aplicacional existe uma distribuição mais linear do número de pacotes e *bytes* por segundo, no *browser* estes valores tendem a variar mais.

Finalmente, utilizou-se o modelo de previsão desenvolvido para classificar as sessões dentro das classes que refletem a QoE do utilizador.

Em suma, concluiu-se que a versão *browser* é recomendada quando existe um limite de dados móveis a utilizar. Caso contrário, não se justifica a não utilização da versão aplicacional, desenvolvida para uma melhor interação com o utilizador.

7.1 TRABALHO FUTURO

A realização deste trabalho de mestrado levanta pontos interessantes de se abordar no futuro. O primeiro aspeto seria incluir um maior número de cenários para análise de resultados.

Por outro lado, perceber quais os passos existentes na conexão, desde o primeiro pedido até ao final da reprodução do vídeo, seria também algo importante a analisar.

Por fim, a obtenção de um *dataset* próprio, alargado e robusto, para utilizar no desenvolvimento, treino e teste do modelo de previsão acrescentaria bastante ao trabalho.

BIBLIOGRAFIA

Zanar Aesthetics. Spider-man ps4 - spiderman vs 6 villain bosses epic cutscene (marvel's spiderman game 2018) ps4 pro, 2018. URL https://www.youtube.com/watch?v=e2bzrGxTFCk.

Vaneet Aggarwal, Emir Halepovic, Jeffrey Pang, Shobha Venkataraman, and He Yan. Prometheus: Toward quality-of-experience estimation for mobile apps from passive network measurements. *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications, HotMobile 2014*, 2014.

Nour Alqudah and Qussai Yaseen. Machine Learning for Traffic Analysis: A Review. *Procedia Computer Science*, 170:911–916, 2020. URL https://doi.org/10.1016/j.procs.2020.03.111.

Arkadiusz Biernacki and Kurt Tutschku. Performance of HTTP video streaming under different network conditions. *Multimedia Tools and Applications*, 72(2):1143–1166, 2014.

Blender. Big buck bunny 60fps 4k - official blender foundation short film, 2014. URL https://www.youtube.com/watch?v=aqz-KE-bpKQ.

Arthur Callado, Carlos Kamienski, Géza Szabó, Balázs Péter Gero, Judith Kelner, Stênio Fernandes, and Djamel Sadok. A survey on internet traffic identification. *IEEE Communications Surveys and Tutorials*, 11(3):37–52, 2009.

Pedro Casas, Alessandro D'Alconzo, Florian Wamser, Michael Seufert, Bruno Gardlo, Anika Schwind, Phuoc Tran-Gia, and Raimund Schatz. Predicting QoE in cellular networks using machine learning and in-smartphone measurements. *2017 9th International Conference on Quality of Multimedia Experience, QoMEX 2017*, 2017.

Mostafa Zaman Chowdhury, Mohd Noor Islam, Young Min Seo, and Young Ki Lee. Characterizing Parameters and Application of Wireless Networks. *Power*, pages 760–764, 2008.

Cisco. Cisco annual internet report (2018–2023). *Cisco*, pages 1–41, 2020. URL https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf.

comscore. URL https://www.comscore.com.

Nick Duffield. Sampling for passive Internet measurement: A review. *Statistical Science*, 19(3):472–498, 2004.

RAPHYAP FORMARAN. Filmic pro cinematic video test aestethic video random nature clips b roll 4k 24fps, 2021. URL https://www.youtube.com/watch?v=6GTWNZlEcHU.

Taiwo Gabriel Omomule, Omomule T G, Olarinde D O, Ugwu C C, and Falana T A. QoS Performance Metrics for Analyzing Wireless Network Usability. (December), 2019. URL https://www.researchgate.net/publication/338101205.

Wadzani Gadzama, Bitrus Joseph, and Taraba State. Global Smartphone Ownership , Internet Usage And Their. (September):0–10, 2019.

Andreas Hanemann, Athanassios Liakopoulos, Maurizio Molina, and D. Martin Swany. A study on network performance metrics and their composition. *Campus-Wide Information Systems*, 23(4):268–282, 2006.

Jupyter. URL https://jupyter.org.

Carlos Kamienski, Tatiene Souza, Stenio Fernandes, Guthemberg Silvestre, and Djamel Sadok. Caracterizando propriedades essenciais do tráfego de redes através de técnicas de amostragem estratificada. 01 2005.

Alexandru Korotcov, Valery Tkachenko, Daniel P. Russo, and Sean Ekins. Comparison of deep learning with multiple machine learning methods and metrics using diverse drug discovery data sets. *Molecular Pharmaceutics*, 14(12):4462–4475, 2017. doi: 10.1021/acs.molpharmaceut.7b00578.

Bartosz Krawczyk, Leandro L. Minku, João Gama, Jerzy Stefanowski, and Michal Wozniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2017.02.004. URL https://www.sciencedirect.com/science/article/pii/S1566253516302329.

Eirini Liotou, Dimitris Tsolkas, and Nikos Passas. A roadmap on QoE metrics and models. *2016 23rd International Conference on Telecommunications, ICT 2016*, pages 1–5, 2016.

Venkat Mohan, Y R Janardhan Reddy, and K Kalpana. Active and Passive Network Measurements : A Survey. *Computer Science and Information Technologies*, 2(4):1372–1385, 2011.

Sophon Mongkolluksamee, Vasaka Visoottiviseth, and Kensuke Fukuda. Combining communication patterns & traffic patterns to enhance mobile traffic identification performance. *Journal of Information Processing*, 24(2): 247–254, 2016.

Kevin P. Murphy. *Machine learning: A probabilistic perspective*. MIT Press, Cambridge, Mass.[u.a.]., 2012.

perfSONAR. URL https://www.perfsonar.net/.

Chapman Pete, Clinton Julian, Kerber Randy, Khabaza Thomas, Reinartz Thomas, Shearer Colin, and Rudiger Wirth. Crisp-Dm 1.0. *CRISP-DM Consortium*, page 76, 2000.

Python. URL https://www.python.org.

Mohammad Reza, Mohammad Javad, Seyed Raouf, and Reza Javidan. Network Traffic Classification using Machine Learning Techniques over Software Defined Networks. *International Journal of Advanced Computer Science and Applications*, 8(7), 2017.

SensorTower. Q4 2020 store intelligence data digest. 2020. URL https://go.sensortower.com/rs/351-RWH-315/images/Sensor-Tower-Q4-2020-Data-Digest.pdf.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, USA, 2014.

Ali Shehadeh, Odey Alshboul, Rabia Emhamed Al Mamlook, and Ola Hamedat. Machine learning models for predicting the residual value of heavy construction equipment: An evaluation of modified decision tree, lightgbm, and xgboost regression. *Automation in Construction*, 129:103827, 2021. ISSN 0926-5805. doi: https://doi.org/10.1016/j.autcon.2021.103827. URL https://www.sciencedirect.com/science/article/pii/S0926580521002788.

Guoqin Song. The study and design of network traffic monitoring based on socket. *Proceedings - 4th International Conference on Computational and Information Sciences, ICCIS 2012*, pages 845–848, 2012.

Tstat TCP STatistic and Analysis Tool. URL http://tstat.polito.it/links.shtml.

tcpdump. URL https://www.tcpdump.org.

Vladislav Vasilev, Jeremie Leguay, Stefano Paris, Lorenzo Maggi, and Merouane Debbah. Predicting QoE Factors with Machine Learning. *IEEE International Conference on Communications*, 2018-May, 2018a. ISSN 15503607. doi: 10.1109/ICC.2018.8422609.

Vladislav Vasilev, Jeremie Leguay, Stefano Paris, Lorenzo Maggi, and Merouane Debbah. Qos-qoe dataset - predicting qoe factors with machine learning (ieee icc 2018), 2018b. URL http://jeremie.leguay.free.fr/qoe/index.html.

C. Williamson. Internet traffic measurement. *IEEE Internet Computing*, 5(6):70–74, 2001.

Wireshark. URL https://www.wireshark.org/.

YouTube. URL https://www.youtube.com.

youtube dl. URL https://github.com/ytdl-org/youtube-dl.

Sebastian Zander, Thuy Nguyen, and Grenville Armitage. Automated traffic classification and application identification using machine learning. *Proceedings - Conference on Local Computer Networks, LCN*, 2005 (December 2005):250–257, 2005.

Dariusz Zelasko, Pawel Plawiak, and Joanna Kolodziej. Machine learning techniques for transmission parameters classification in multi-agent managed network. *Proceedings - 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGRID 2020*, pages 699–707, 2020.

Liangzhao Zeng, Hui Lei, and Henry Chang. Monitoring the QoS for Web services. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4749 LNCS:132–144, 2007.