ELE8704 - Transmission de données et réseaux internet

Projet de session

Expérience de capture de paquets

Ana Manresa Gonzalez, 2223369

Anas Bouhemhem, 1953369

Laure Pourcel, 2164239

Louis Fleurquin-Asselin, 1880414

Polytechnique Montréal

05/12/2022

Table des matières

1 Liste des acronymes **2**

2 Introduction **3**

3 Collecte des données **4**

3.1 Vue d'ensemble 4

3.2 Prise de mesures 4

3.3 Résultats 4

4 Statistiques **5**

4.1 Protocoles 5

4.2 Taille des paquets 11

4.3 Gigue 14

5 Apprentissage Machine **15**

5.1 Preprocessing 15

5.2 Division des statistiques entre entrainement, test et validation 16

5.3 Modèles d'apprentissage machine 16

6 Conclusion **20**

7 Annexe **21**

Liste des figures

4.1 Graphique du nombre de paquets par protocole pendant un appel 6
4.2 Graphique du nombre de paquets par protocole pour le téléchargement de la librairie Lapack . . 6
4.3 Graphique du nombre de paquets par protocole pour le visionnage d'une vidéo 7
4.4 Graphiques de répartition des différentes destinations pour chaque expérience 8
4.5 Graphique de répartition des différentes destinations pour chaque expérience sans les 2 destinations les plus communes 10
4.6 Moyenne de la taille des paquets selon l'expérience 11
4.7 Écart-type de la taille des paquets selon l'expérience 12
4.8 Graphique des différentes tailles des paquets pour chaque expérience 13
4.9 Valeur de la gigue selon les différentes expériences 14
5.1 Dataframe de travail 15
5.2 Dataframe après l'encodage 16
5.3 Valeurs numériques prédites par la régression linéaire pour les protocoles 17
5.4 Valeurs numériques prédites par l'apprentissage machine pour la classification des "Target" . . . 18
5.5 Matrice de confusion pour le feature "Protocol" 19
5.6 Matrice de confusion pour le feature "Target" 19
7.1 Graphique de répartition des différentes sources pour chaque expérience 21
7.2 Graphique de répartition des différentes sources pour chaque expérience sans les 2 sources les plus communes 22

1 Liste des acronymes

E-mail ..... Electronic mail

IP ..... Internet Protocol

LAPACK ..... Linear Algebra Package

QUIC ..... Quick UDP Protocol

TCP ..... Transmission Control Protocol

VOIP ..... Voice Over Internet Protocol

WI-FI/WIFI ... Wireless Fidelity

NAT ..... Network Address Translators

2 Introduction

Dans ce projet, nous étudions le trafic IP pour différentes applications. Nous nous intéressons plus précisément à 3 applications : un Appel, le téléchargement d'un fichier et le visionnage d'une vidéo en streaming. Pour chacun de ces cas d'usage nous réalisons une collecte de données avec une connexion filaire, Ethernet, et une connexion sans-fil (Wi-Fi). Diverses statistiques sont présentées afin d'illustrer le fonctionnement du transfert de données. De plus, des modèles d'apprentissage machine sont inclus afin de classifier les paquets selon plusieurs critères afin de mieux visualiser leur fonctionnement.

3 Collecte des données

3.1 Vue d'ensemble

La prise de mesures s'est déroulée en plusieurs étapes. La première partie a consisté à enregistrer des paquets en utilisant une connexion Wi-Fi. Trois prises de mesures ont été effectuées : téléchargement d'une vidéo YouTube, téléchargement de la librairie Lapack et appel vidéoconférence. Par la suite, les mêmes prises de mesures ont été effectuées à nouveau, mais avec une connexion filaire Ethernet. Afin de s'assurer de la fiabilité des résultats, le même ordinateur a été utilisé pour tous les tests, et ce, au même endroit. La connexion Ethernet et la connexion Wi-Fi ont fait appel au même modem. Le câble Ethernet utilisé a une longueur de 4.5 m (15 ft). Pour les mesures en wifi, l'ordinateur a été placé à une distance d'environ deux mètres du modem. De plus, pour chaque test, seule l'application en question a été utilisée, c'est-à-dire que toute application n'étant pas reliée au test et pouvant être fermée a été fermée. Également, tout onglet de navigateur internet n'étant pas relié à la prise de mesure a été fermé. Ainsi, le nombre d'applications effectuant des transferts de données en arrière-plan est minimal et on diminue la probabilité d'obtenir des données aberrantes. Le logiciel Wireshark a été utilisé pour enregistrer les paquets.

3.2 Prise de mesures

3.2.1 Appel vidéo

Les appels vidéo se sont déroulés sur l'application Microsoft Teams. Chaque appel a une durée de deux minutes. La même personne a été appelée dans les deux cas.

3.2.2 Vidéo YouTube

La vidéo visionnée est *Montreal city in 5K by Sky Semijon*SEMIJON s. d. Elle dure une minute et 57 secondes et a été visionnée en haute résolution (1920 par 1080 pixels). Mozilla Firefox a été utilisé comme navigateur.

3.2.3 Librairie Lapack

Pour le téléchargement de fichier, il a été question de la librairie Lapack 3.7.1 NETLIB s. d. Le téléchargement s'est effectué avec Mozilla Firefox.

3.3 Résultats

Une fois chaque prise de données effectuée, les résultats ont été exportés vers un fichier au format .csv. Par la suite, le fichier a été converti en format tableur .xlsx pour pouvoir être traité avec Excel. Les données ont donc pu être triées selon divers attributs (longueur, source, destination, protocole) ou encore être exportées directement dans une archive .ipynb afin d'utiliser des librairies telles que Pandas et Sklearn pour effectuer la

suite du traitement des données.

4 Statistiques

4.1 Protocoles

TCP et UDP sont des protocoles de communication réseau. En termes simples, ce sont deux "méthodes" de communication par transmission de données numériques, sur l'IP, le protocole qui régit toutes les communications sur Internet.

Ils diffèrent à deux niveaux : la qualité et la vitesse. TCP est adapté aux communications nécessitant une transmission de données parfaite et sans perte, au prix d'une lenteur très relative. Ce protocole est dit "robuste" et est utilisé par la plupart des applications (navigateur, transfert de fichiers, e-mail, etc.).

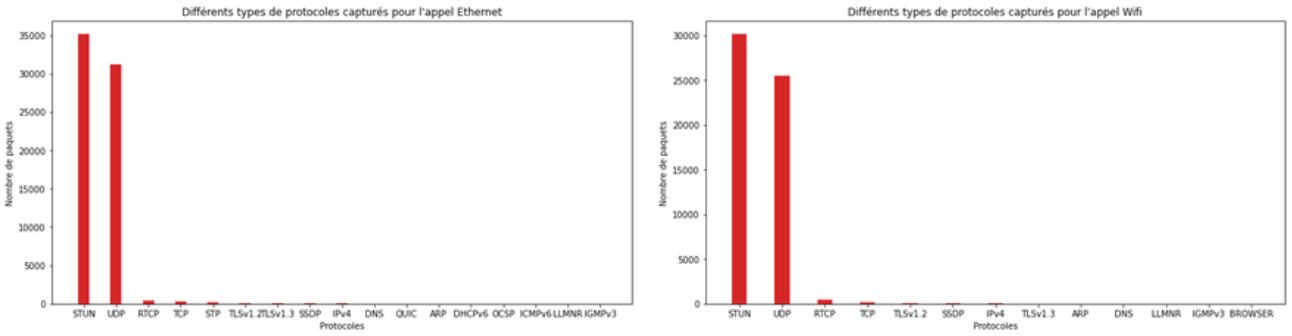
L'UDP privilégie la vitesse à la fiabilité et comme il ne nécessite pas de connexion constante, il libère des ressources système des deux côtés, ce qui rend les applications plus légères. Il est principalement utilisé dans les applications en temps réel telles que les jeux vidéo ou les appels audio, ainsi que des services streaming. Comme mentionné précédemment, nous avons effectué des captures de paquets des trois principaux types de données. Nous estimons que les valeurs les plus fréquentes de protocoles pour chaque type de paquets seront les suivantes, pour chaque type de test :

4.1.1 Appels

Protocoles : principalement UDP.

Les appels ne nécessitent pas la robustesse du protocole TCP, car si un paquet est perdu, il n'est pas souhaitable de le récupérer a posteriori. Par contre, lors d'un appel la rapidité de transmission est un élément très important afin d'avoir une conversation fluide.

Pour confirmer notre hypothèse, nous affichons le nombre de paquets pour chaque protocole au sein de nos données collectées :



(a) Appel Ethernet (b) Appel Wi-Fi

FIGURE 4.1 – Graphique du nombre de paquets par protocole pendant un appel

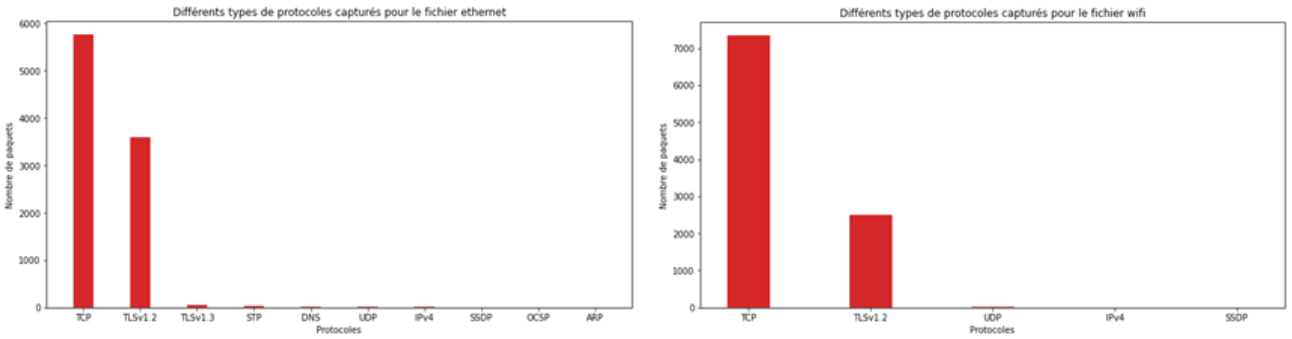
Dans le cas de l'appel, nous pouvons constater une importante quantité de paquets avec le protocole Simple Traversal of UDP through NAT, aussi appelé STUN. Ce protocole permet la communication de stations sur des réseaux privés vers des réseaux publics via la couche de transport UDP, une étape essentielle dans la construction d'une infrastructure de téléphonie IP. Nous pouvons conclure qu'il s'agit d'un protocole basé sur UDP, résultat qui confirme que la majorité du trafic utilise UDP.

4.1.2 Téléchargements de fichiers

Protocoles : principalement TCP.

Lors du téléchargement d'un fichier il est primordial de récupérer tous les paquets correctement, afin que le fichier ne soit pas corrompu et donc inutilisable. Afin de réaliser le contrôle d'erreurs et d'éviter la perte de paquets nous supposons que le protocole prédominant est TCP.

Nous vérifions cette hypothèse en affichant le nombre de paquets transmis en fonction des protocoles pour les données collectées lors du téléchargement de la librairie lapack :



(a) Fichier Ethernet (b) Fichier Wi-Fi

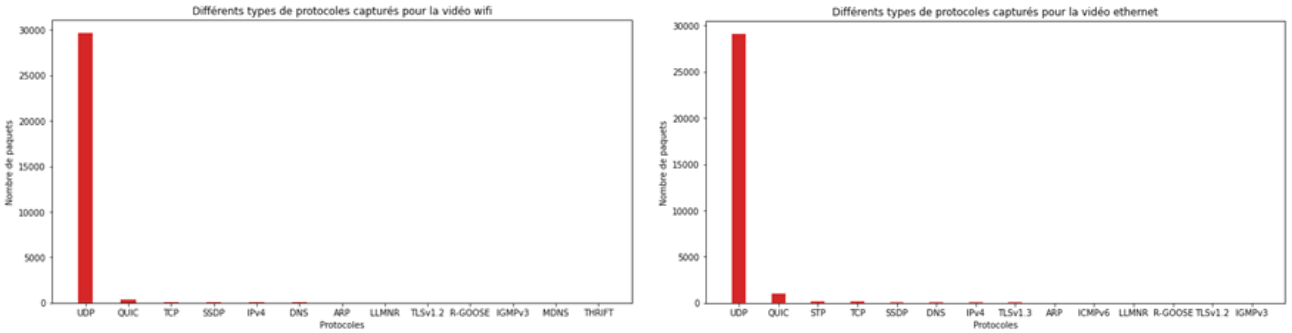
FIGURE 4.2 – Graphique du nombre de paquets par protocole pour le téléchargement de la librairie Lapack

On constate que le protocole TCP est prédominant. Le protocole TLSv1.2 représente approximativement un tier des données. C'est un protocole qui utilise TCP en couche transport, cela confirme donc notre hypothèse.

4.1.3 Vidéos

Protocoles : principalement UDP.

Par un raisonnement analogue à celui pour l'appel nous supposons que lors du visionnement d'une vidéo il est préférable de prioriser la fluidité du visionnage à la fiabilité des données.



(a) Vidéo Wi-Fi (b) Vidéo Ethernet

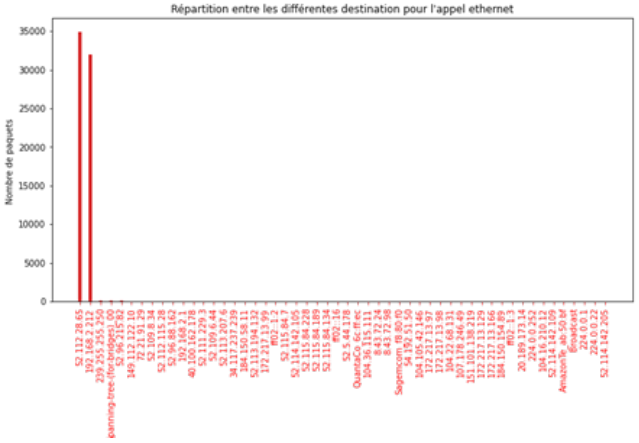
FIGURE 4.3 – Graphique du nombre de paquets par protocole pour le visionnage d'une vidéo

La majorité du trafic est effectivement en UDP. Nous pouvons nous apercevoir de la présence du protocole QUIC, lequel est un protocole de couche de transport (expérimental) développé par Google. Son objectif est de réduire la latence par rapport à TCP tout en garantissant une plus grande fiabilité.

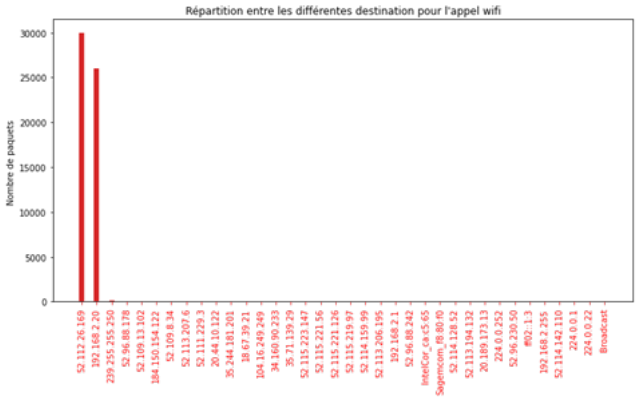
4.1.4 Sources et les destinations

- Adresse source de notre machine : 192.168.2.212 en Ethernet, 192.168.2.20 en Wi-Fi
- Adresse destination de l'appel : 52.112.28.65

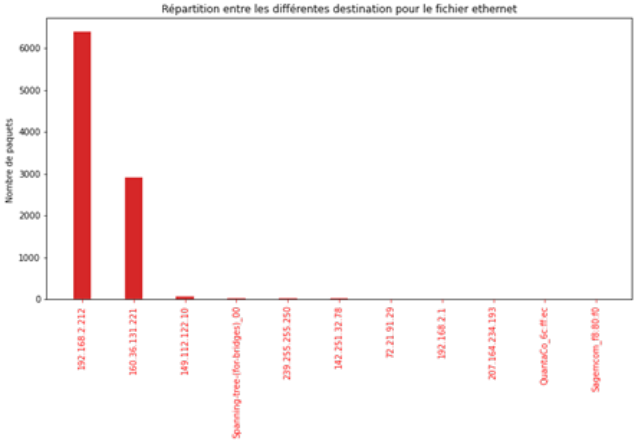
Les graphiques suivants montrent que ce sont les deux adresses les plus courantes lors de la prise des mesures.



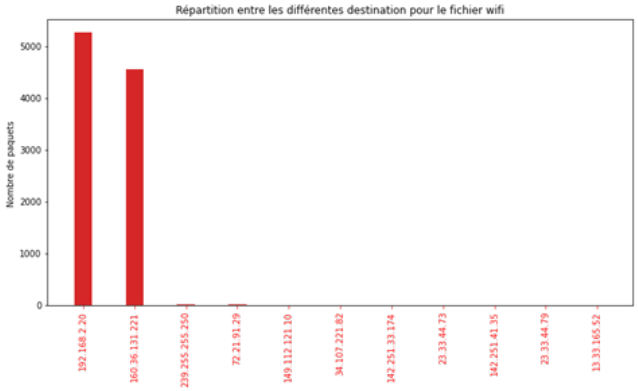
(a) Appel Ethernet



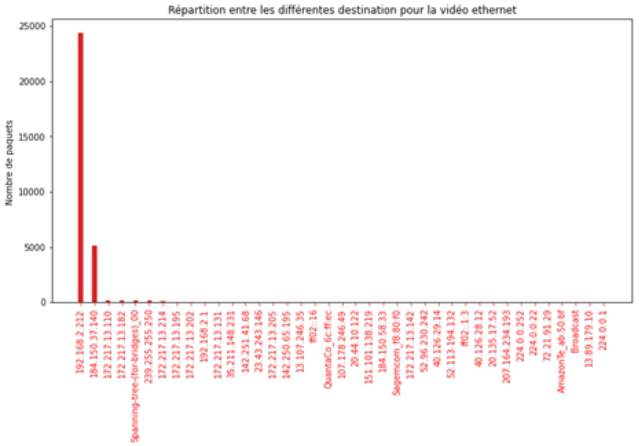
(b) Appel Wi-Fi



(c) Téléchargement Ethernet



(d) Téléchargement Wi-Fi



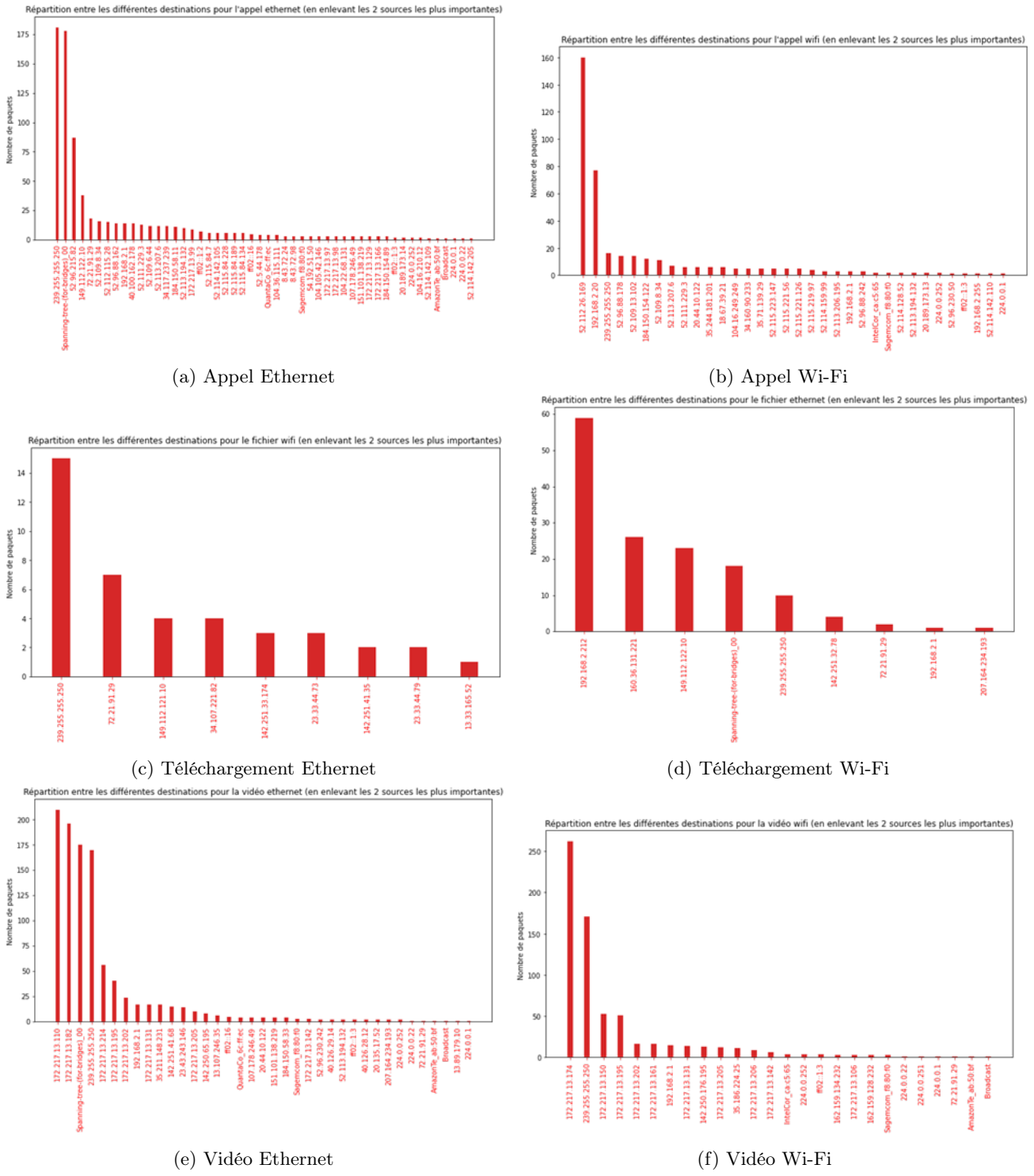
(e) Vidéo Ethernet

(f) Vidéo Wi-Fi

FIGURE 4.4 – Graphiques de répartition des différentes destinations pour chaque expérience

Les graphiques pour les adresses sources sont similaires, mais inversent le premier et le deuxième contributeur, qui logiquement reçoivent au lieu d'envoyer les paquets. Ils sont mis en annexe.
On note que lors de l'appel, la proportion de paquet reçu et envoyé est relativement similaire, ce qui est attendu, car en plus de recevoir l'information de la personne avec qui on passe l'appel on transmet nos propres données.
Pour le téléchargement, le protocole majoritaire est TCP. On s'attend donc à ce qu'il y ait plus de messages vers l'hôte et des messages d'acknowledgement en réponse vers le serveur. On observe qu'il y a plus de messages à destinations du serveur pour le téléchargement en Wi-Fi. On l'on attribue aux demandes de correction de renvoi de paquet plus fréquentes avec une connexion sans-fil, plus propice aux erreurs de transmission.
Enfin, lors du visionnement de la vidéo on constate comme attendu une proportion bien supérieure de paquets à destination de l'hôte.

Par la suite, on enlève les deux sources les plus importantes, avec l'objectif de visualiser les sources et destinations secondaires. L'existence de ces adresses secondaires est due au fait que Wireshark capture non seulement les paquets destinés au réseau à l'étude, mais également les autres paquets sur le réseau. Ce phénomène peut également être dû à la présence d'applications qui fonctionnent en arrière-plan et qui ne peuvent pas nécessairement être fermées.



(a) Appel Ethernet

(b) Appel Wi-Fi

(c) Téléchargement Ethernet

(d) Téléchargement Wi-Fi

(e) Vidéo Ethernet

(f) Vidéo Wi-Fi

FIGURE 4.5 – Graphique de répartition des différentes destinations pour chaque expérience sans les 2 destinations les plus communes

Dans la suite de notre étude, nous considérerons comme des paquets de contrôle les paquets n'utilisant pas les protocoles UDP et TCP en fonction du cas, ainsi que les paquets dont la source et la destination ne correspondent pas aux deux contributeurs majoritaires à ces classes.

4.2 Taille des paquets

Dans cette section, la taille des paquets sera observée pour chaque configuration des expérimentations. Nous utiliserons uniquement les paquets transmis, en fonction du cas, en UDP ou TCP entre le serveur qui héberge l'application et notre machine.

La moyenne de la taille des paquets et l'écart type sont calculés en utilisant python. On obtient les valeurs suivantes :

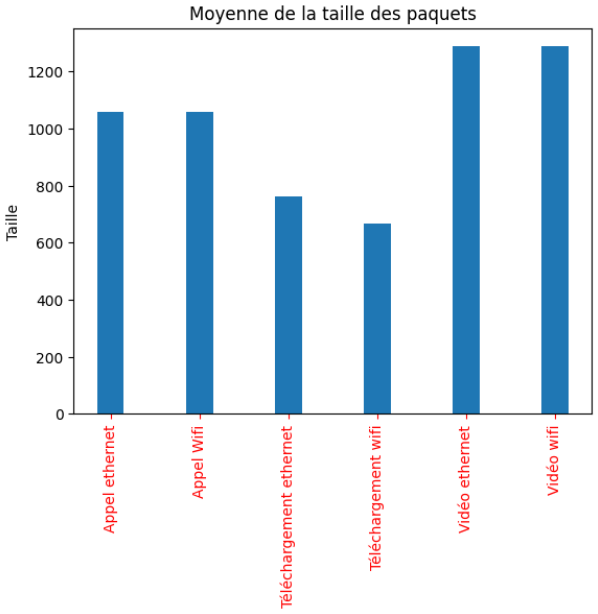


FIGURE 4.6 – Moyenne de la taille des paquets selon l'expérience

En observant le graphique ci-dessus, il est possible de voir que la moyenne de la taille des paquets est plus importante lors d'un visionnage numérique comparé au téléchargement d'un fichier ou bien un appel VOIP. On constate donc expérimentalement qu'en moyenne les paquets UDP sont plus longs que les paquets TCP. Ce fait ne change pas selon la couche physique utilisée, qu'elle soit filaire ou non. Cela peut s'expliquer par le fait que diminuer la taille des paquets diminue la probabilité qu'il y ait une erreur dans un paquet, limitant le nombre de retransmissions en TCP. Cependant, il est nécessaire de faire un compromis afin de ne pas subdiviser l'information en un trop grand nombre de paquets. Nous vérifierons cette hypothèse en regardant la répartition de la taille des paquets.

Concernant les appels VOIP, il s'agit de l'expérience qui possède la deuxième plus grande taille moyenne de paquets. ils font en moyenne 1120 octets. Il y a autant de paquets dans la connexion Ethernet que la connexion Wi-Fi. Puisque les appels audio utilisent le protocole UDP, il est normal de dire que la taille moyenne des paquets soit inférieure à celle de l'expérience avec le streaming vidéo, puisque la quantité d'information est moins grande à transférer, puisque la qualité vidéo est inférieur.

Du côté du téléchargement, c'est la méthode qui possède la taille moyenne de paquet la plus faible des trois. Comparée aux autres méthodes, c'est la seule qui utilise le protocole TCP, puisque lors du téléchargement, il faut

être sûr que les paquets arrivent à destination. On peut observer que c'est la seule expérience qui possède une différence entre le transfert d'information à travers une connexion Wi-Fi et une connexion Ethernet. Ce dernier possède une taille moyenne plus grande que cette sans-fil. Nous attribuons la taille moyenne inférieure des paquets en Wi-Fi à un plus grand nombre de demandes de retransmission, qui sont de courts messages.
Dans le cas de l'écart-type, il est possible d'observer ces différentes valeurs pour chaque expérience :

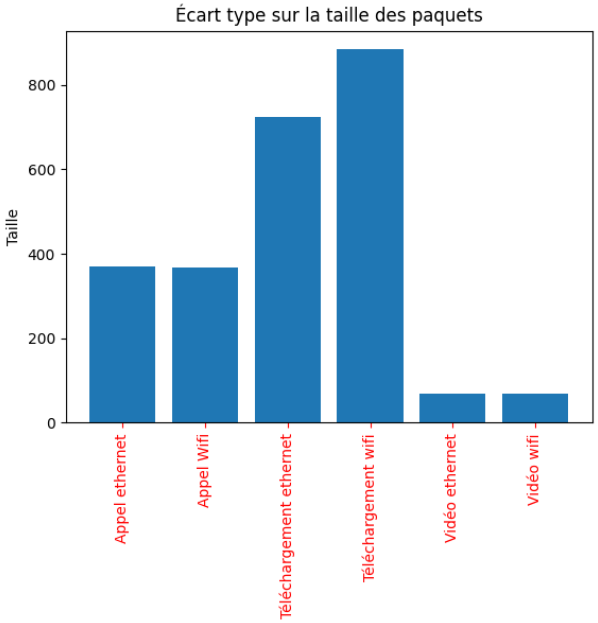
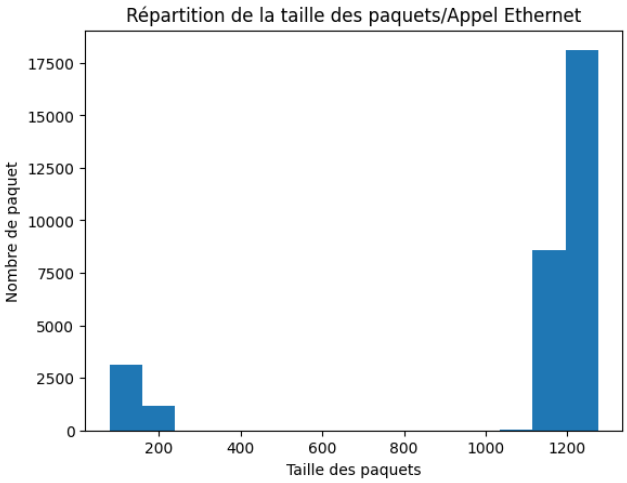
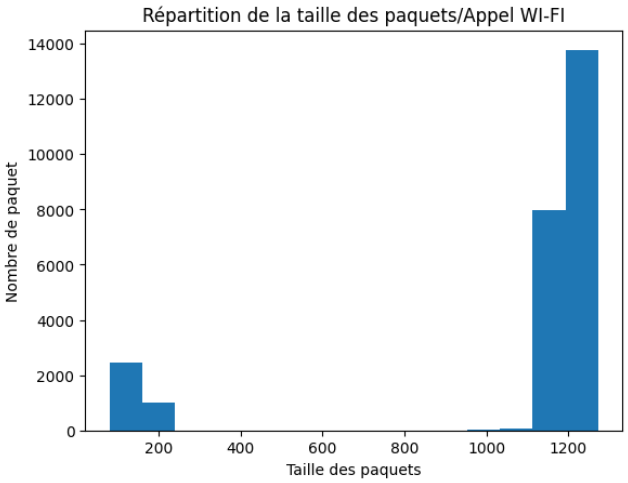


FIGURE 4.7 – Écart-type de la taille des paquets selon l'expérience

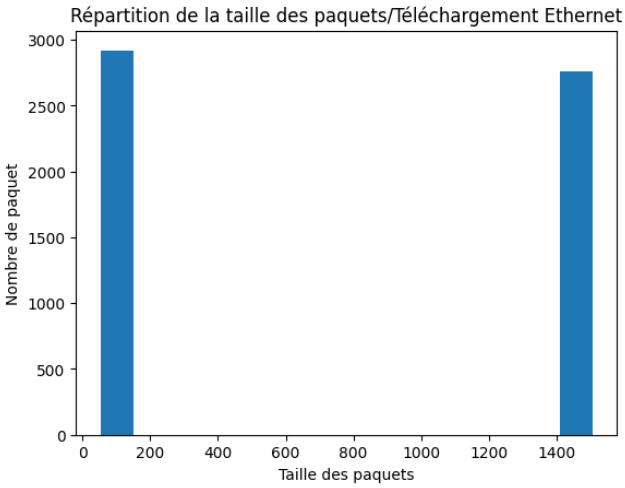
Dans la figure au-dessus, nous pouvons remarquer que l'écart-type pour le téléchargement est le plus grand, suivi par l'appel et finalement par le visionnement vidéo. Nous pouvons dire que l'écart-type pour le téléchargement est supérieur du côté sans-fil que du côté avec fil. En utilisant le protocole TCP, il est possible de dire que la taille des paquets est très variable lors d'un téléchargement. Les autres expériences montrent un écart-type équivalant, autant du côté Wi-Fi qu'Ethernet. Cela montre que lors de l'utilisation du protocole TCP, l'écart-type de taille des paquets est supérieur à ceux du protocole UDP.
Nous expliquons cette variabilité bien supérieure par la présence de très courts messages d'acknowledgement, de demande de retransmission ou de handshake en TCP, tandis qu'en UDP, il y a quasi uniquement des paquets contenants les datas. Ainsi, malgré un entête plus petit en UDP (8 octets vs 20 octets), la taille moyenne des paquets est plus importante, tandis que l'écart type est beaucoup plus important en TCP.
On observe la répartition de la taille des paquets :



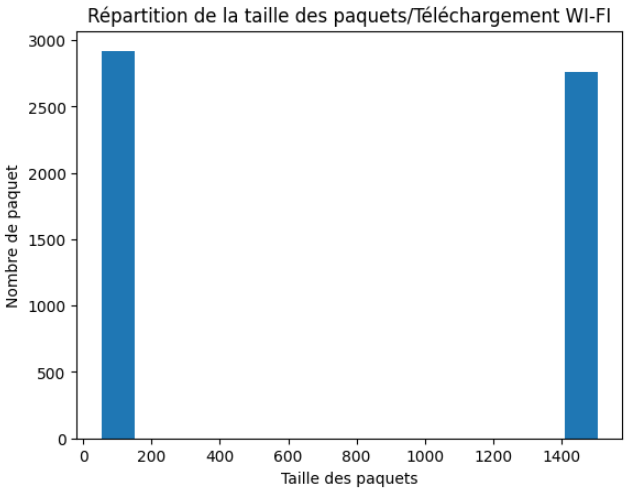
(a) Appel Ethernet



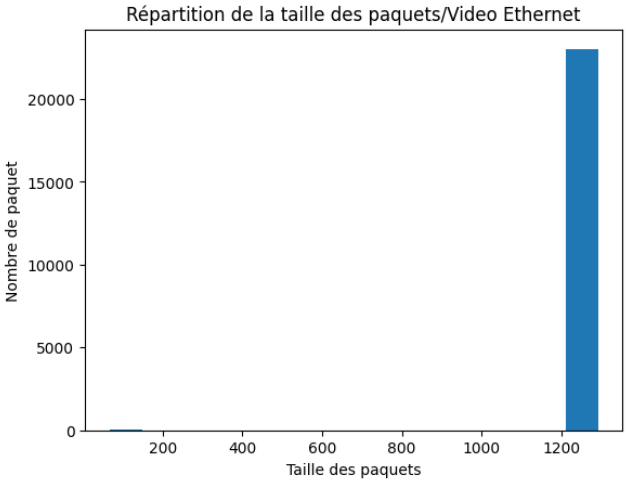
(b) Appel WI-FI



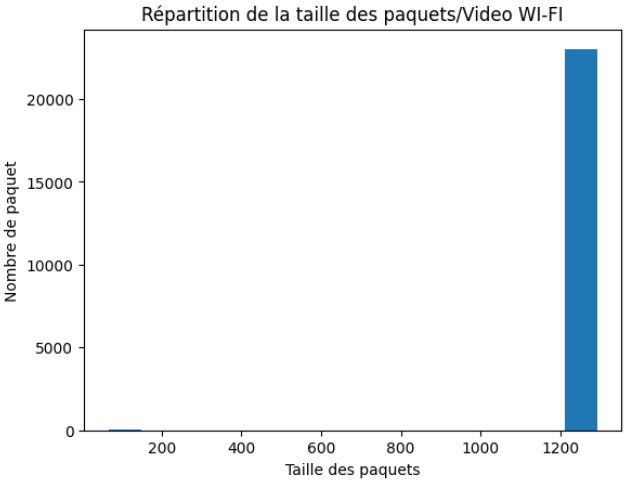
(c) Téléchargement Ethernet



(d) Téléchargement WI-FI



(e) Vidéo Ethernet



(f) Vidéo Wi-Fi

FIGURE 4.8 – Graphique des différentes tailles des paquets pour chaque expérience

On constate que les paquets contenant de la data en UDP font entre 1100 et 1300 octets, contre plus de 1400 en TCP. Cela réfute notre hypothèse quant à une plus petite longueur des données en TCP pour limiter le

nombre de retransmissions. La moyenne de la taille des paquets en TCP est effectivement baissée par des paquets de petites tailles.

4.3 Gigue

Dans cette section, nous allons analyser les valeurs de la gigue pour chaque expérience. Le graphique ci-dessous montre le délai en seconde :

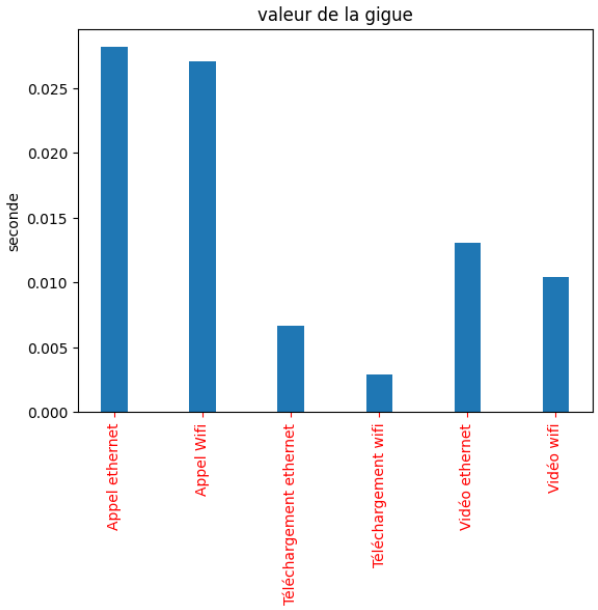


FIGURE 4.9 – Valeur de la gigue selon les différentes expériences

Afin d'obtenir une expérience de navigation internet optimale, il est nécessaire de minimiser le temps de transmission de données. C'est là où le concept de gigue intervient. Nous pouvons remarquer la valeur de la gigue est petite pour le téléchargement Ethernet et Wi-Fi. L'appel en connexion Ethernet et en connexion Wi-Fi possède la plus grande valeur de gigue, comparée au visionnement télévisuel. Cela peut s'expliquer grâce à l'utilisation du protocole TCP qui privilégie un envoi des paquets qui sont précis comparé à l'UDP. C'est donc dans le cas de l'appel et le visionnement de vidéo que nous allons observer le phénomène de 'Jittering'. Dans le cas présent, il est possible de remarquer que l'appel possède plus de latence que le visionnage vidéo. De plus, la latence est plus prononcée du côté de la communication sans fil qu'avec fil. Il est normal d'observer ce phénomène : il y a plus de bruit et d'interférence dans un lien de communication sans fil qu'avec fil. Ceci explique une légère augmentation de 'Jittering'. Du côté de la vidéo, cette dernière a moins de latence que l'appel grâce au protocole QUIC développé par Google. Le protocole QUIC en résumé, selon Wikipédia, permet de réduire la latence et la bande passante afin de limiter la congestion dans les flux de données WIKIPEDIA s. d. C'est donc pour cette raison que la valeur de la gigue est inférieure dans l'expérience de visionnement comparée à l'appel VOIP.

5 Apprentissage Machine

5.1 Preprocessing

Nos données contiennent 7 features : "No.","Time","Source","Destination","Protocol","Length","Info". Nous n'utiliserons pas les features "No." et "Time" pour notre classification, car elles n'apportent pas d'informations utiles. On drop donc ces 2 colonnes au début du preprocessing. Comme analysé précédemment, les sources et destinations pour chaque cas d'usage sont majoritairement les mêmes. Dans le cas de l'appel en Ethernet par exemple, les adresses IP de l'ordinateur utilisé et du serveur représentent plus de 95% des paquets envoyés. Un raisonnement similaire s'applique pour les protocoles, dans le cas de l'appel en Ethernet les protocoles UDP et STUN représentent également plus de 95% des paquets. Il existe une plus grande variabilité sur la longueur des paquets et les infos associées.

Nous avons fait le choix de garder tous les paquets que nous avons collectés pour la classification, afin d'assurer une plus grande variabilité dans notre dataset.

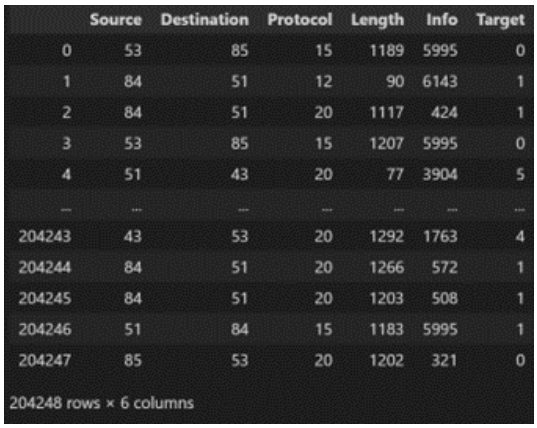
Nous réalisons un nouveau Dataframe contenant les features "Source", "Destination", "Protocol", "Length", "Info" et une nouvelle feature "Target" qui permet de différencier le cas d'utilisation, allant de 0 à 5.

	Source	Destination	Protocol	Length	Info	Target
0	184.150.37.140	192.168.2.20	UDP	1292	443 > 59673 Len=1250	5
1	192.168.2.212	52.112.28.65	STUN	1220	ChannelData TURN Message	0
2	184.150.37.140	192.168.2.20	UDP	1292	443 > 59673 Len=1250	5
3	192.168.2.212	52.112.28.65	STUN	1242	ChannelData TURN Message	0
4	184.150.37.140	192.168.2.212	UDP	1292	443 > 59814 Len=1250	4
...	...	...	...	...	...	...
204243	52.112.28.65	192.168.2.212	UDP	1225	3480 > 50023 Len=1183	0
204244	192.168.2.212	52.112.28.65	STUN	1191	ChannelData TURN Message	0
204245	192.168.2.20	160.36.131.221	TCP	54	60227 > 443 [ACK] Seq=1172 Ack=4395687 Win=2...	3
204246	192.168.2.212	52.112.28.65	STUN	127	ChannelData TURN Message	0
204247	184.150.37.140	192.168.2.212	UDP	1292	443 > 59814 Len=1250	4

204248 rows × 6 columns

FIGURE 5.1 – Dataframe de travail

Nous réalisons ensuite l'encodage de nos données, afin de convertir les chaînes de caractères en valeurs numériques. Nous utilisons la fonction LabelEncoder() de la bibliothèque sklearn.preprocessing pour assigner à chaque chaîne de caractère une valeur unique. On mélange les données et obtient le dataset suivant :



	Source	Destination	Protocol	Length	Info	Target
0	53	85	15	1189	5995	0
1	84	51	12	90	6143	1
2	84	51	20	1117	424	1
3	53	85	15	1207	5995	0
4	51	43	20	77	3904	5
...	...	...	...	...	...	...
204243	43	53	20	1292	1763	4
204244	84	51	20	1266	572	1
204245	84	51	20	1203	508	1
204246	51	84	15	1183	5995	1
204247	85	53	20	1202	321	0

204248 rows × 6 columns

FIGURE 5.2 – Dataframe après l'encodage

5.2 Division des statistiques entre entrainement, test et validation

Nous séparons notre jeu de données en 3 sous parties, notre set d'entraînement, sur lequel on réalisera l'entraînement de nos algorithmes d'apprentissage machine, notre set de test, sur lequel on viendra tester les modifications apportées à notre algorithme, et notre set de validation sur lequel on calculera la précision. On place 80% de nos données dans le set d'entraînement et 10% dans les sets de test et de validation.

5.3 Modèles d'apprentissage machine

Nous étudierons deux cas, avec 2 features différentes à classifier : "Protocol" et "Target". Dans chacun de ces cas, nous retirerons de notre dataset la feature en question. Il est à noter que ces deux features sont catégoriques.

5.3.1 Modèle basé sur la régression linéaire

Nous cherchons dans un premier temps à classifier la feature "Protocol". Nous utilisons le modèle LinearRegression de sklearn.linear_model. Après la phase d'entraînement, nous cherchons à prédire de nouvelles données et générons les prédictions pour le set d'entraînement. Nous nous attendons à ce que les valeurs numériques prédites soient proches des valeurs entières associées à chaque protocole :

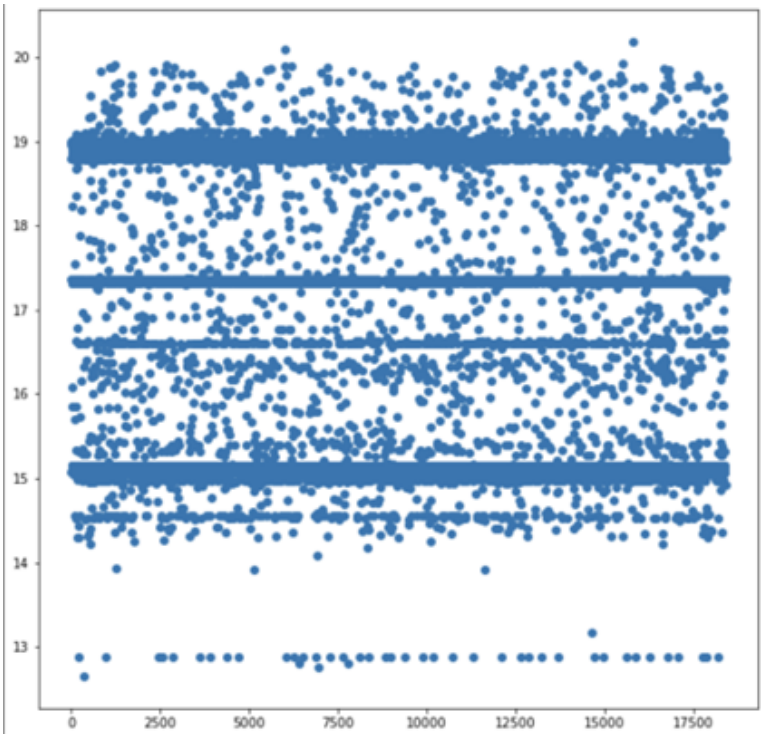


FIGURE 5.3 – Valeurs numériques prédites par la régression linéaire pour les protocoles

On note que pour les valeurs 15 et 19, qui correspondent à l'encodage numérique des protocoles UDP et TCP, les valeurs numériques prédites par la régression linéaire sont relativement proches des valeurs entières. Ce sont les classes les plus peuplées, il est donc attendu que les poids de la régression linéaire aient été calculés afin de les prédire précisément.

Nous réalisons la prédiction sur le jeu de données de validation. Nous obtenons dans un premier temps le coefficient de détermination R2 de notre modèle : 0.780. Ce coefficient nous indique que les données n'ont pas un caractère parfaitement linéaire, auquel cas, il serait plus proche de 1. Nous calculons ensuite l'erreur quadratique moyenne à l'aide de la fonction $mean\_squared\_error$ de sklearn.metrics et obtenons une erreur quadratique moyenne de 0.922.

Nous souhaitons obtenir la précision de notre algorithme puisque la feature sur laquelle nous avons réalisé la régression est catégorique. Pour ce faire, nous arrondissons à l'entier le plus proche chaque valeur numérique associée à un paquet dans notre set de validation. Nous obtenons une précision sur la classification de 83%.

Nous réalisons les mêmes étapes pour la classification de la feature "Target". On réalise les prédictions sur le dataset de test et obtient les prédictions suivantes :

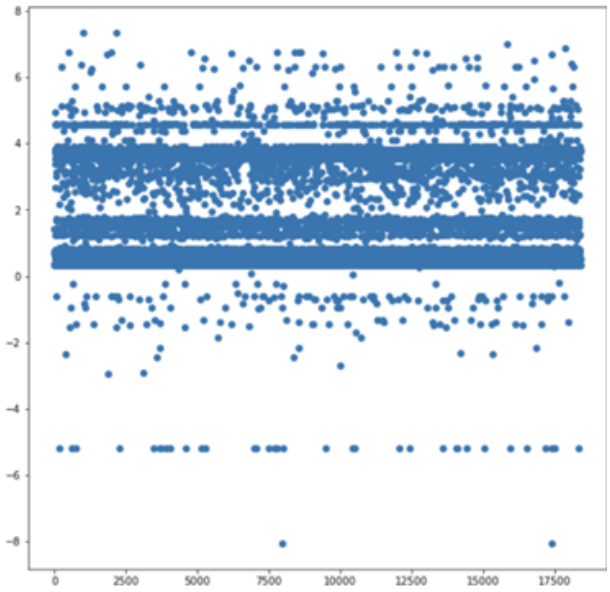


FIGURE 5.4 – Valeurs numériques prédites par l'apprentissage machine pour la classification des "Target"

La structure que l'on observait pour la classification de la feature "Protocol" qui émergeait autour des valeurs entières est ici plus dur à distinguer. Les valeurs ne se concentrent pas principalement vers des valeurs entières. Par ailleurs, il est à noter que les valeurs du champ Target sont [0,1,2,3,4,5], tandis qu'on observe ici que les valeurs numériques retournées par la régression linéaire sont comprises entre 7,5 et -8.

Nous réalisons les prédictions sur le set de validation et obtenons un coefficient de détermination R2 de 0.729. Nous calculons l'erreur quadratique moyenne à l'aide la fonction mean_squared_error de sklearn.metrics et obtenons la valeur suivante : 0.960.

Nous transformons les valeurs numériques obtenues après la régression linéaire en l'entier le plus proche, en contraignant que si la valeur est inférieure à 0 elle deviendra 0 et si elle est supérieure à 5 elle deviendra 5. Nous obtenons ainsi une précision de 55.8 %.

5.3.2 Modèles basé sur la classification

Nous allons tenter dans cette section d'appliquer un algorithme d'apprentissage machine basée sur la classification, car nous supposons qu'il pourrait avoir des performances de classification supérieure à un algorithme de régression appliqué à des données catégoriques.

Nous utilisons le modèle RandomForest, de la librairie sklearn.ensemble. L'algorithme de Random Forest va créer des arbres de décision qui à partir d'un d'une sélection aléatoire de features et d'observations vont définir des nœuds auxquels des décisions seront prises de manière à obtenir une classification. La classification finale est celle qui aura recueilli le vote majoritaire auprès des arbres de la forêt. Nous cherchons à classifier la feature "Protocol". Nous appliquons la fonction de recherche des hyperparamètres GridSearchCV() de la librairie sklearn.model_selection à notre dataset en entier, replié 10 fois en utilisant la fonction RepeatedKFold(). Après optimisation sur le nombre d'arbres dans la forêt, nous obtenons sur les sets de cross validation une précision maximale de 1.00 pour 50

arbres, nous gardons donc cette valeur et n'optimisons pas les autres paramètres.

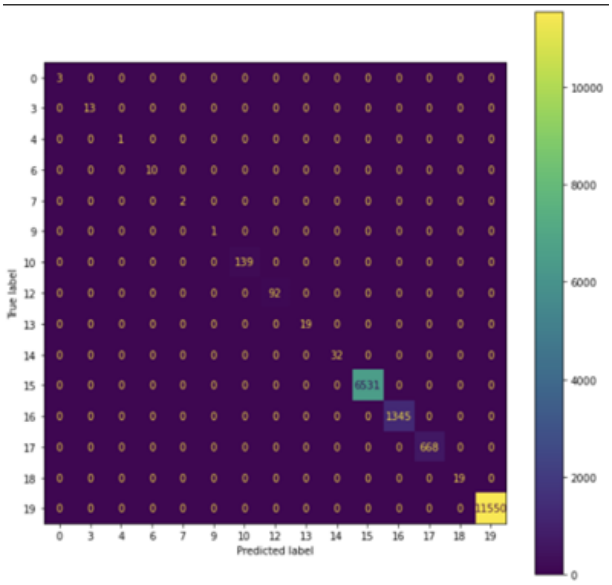


FIGURE 5.5 – Matrice de confusion pour le feature "Protocol"

On applique le même protocole pour la classification de la feature "Target". Nous recherchons dans un premier temps à optimiser l'hyperparamètre du nombre d'arbres à utiliser. En réutilisant un set 10-fold combiné à plusieurs valeurs pour $n_{estimators}$ correspondant au nombre d'arbres. On a de nouveau un optimal pour 50, donnant une précision avec ce paramètre de 1.00. Nous réalisons les prédictions sur un set de validation correspondant à 10% de nos données et obtenons la matrice de confusion suivante :

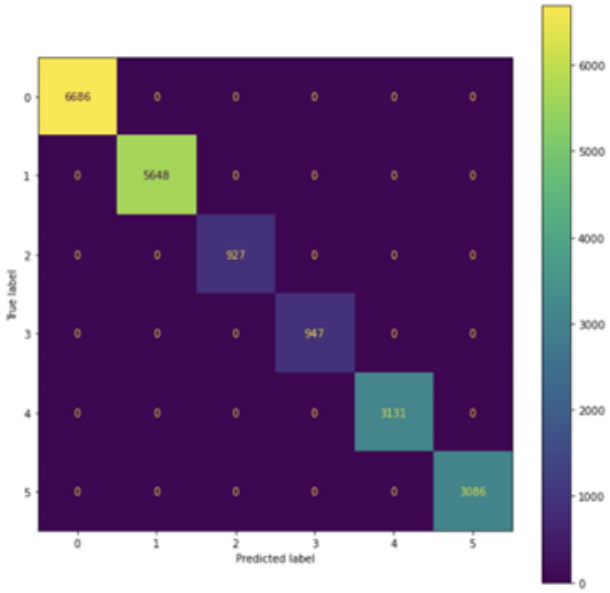


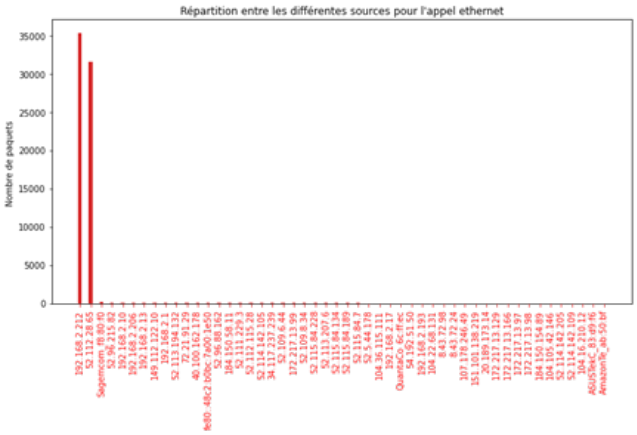
FIGURE 5.6 – Matrice de confusion pour le feature "Target"

6 Conclusion

Au cours de ce projet de fin de session, nous avons appliqué toutes les notions vues dans le cours ELE8704 - Transmission de données et réseaux internet. En utilisant le logiciel WireShark, il a été possible de récupérer les différents paquets et d'en faire un dataset afin d'obtenir des statistiques pertinentes dessus. Ensuite, un fichier Jupyter a été crée et le logiciel Visual Code a été utilisé avec le langage Python afin d'afficher les données et de faire de l'apprentissage machine. Nous avons constaté que la majorité des paquets étaient soit à destination ou en provenance de l'adresse IP du serveur hébergeant l'application ou de l'hôte, comme attendu. Dans le cas d'un appel, il y a presque autant de paquets envoyés que reçus, ce qui correspond au fait qu'on reçoit le flux audio et vidéo de la personne qu'on appelle et on transmet le nôtre. Dans le reste des cas, la majorité des paquets sont envoyés par le serveur hébergeant l'application. La taille moyenne des paquets en TCP est inférieure à celle en UDP puisqu'un grand nombre de paquets de petite taille sont transmis pour assurer une transmission plus robuste des données. La gigue est inférieure dans le cas du téléchargement utilisant le TCP. L'apprentissage machine a été effectué ensuite sur le dataset à l'aide de la librairie SKLearn. Nous avons classifié la feature Protocol et Target, correspondant au cas d'utilisation. Deux modèles ont été utilisés : la régression linaire et le modèle Random Forest. Après plusieurs étapes, nous obtenons une précision plus élevée de 17% et 40% respectivement en utilisant Rando Forest. L'algorithme de classification s'applique mieux dans notre cas puisque les données sont catégoriques.

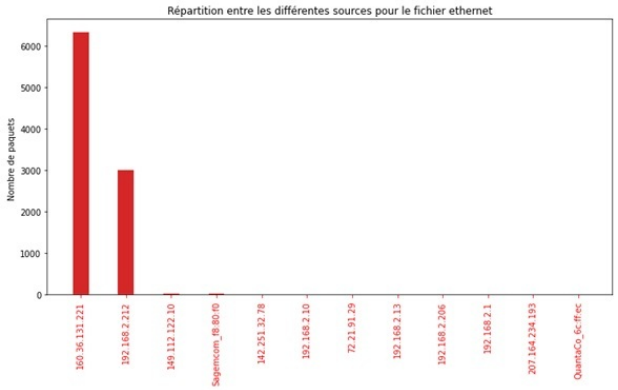
7 Annexe

Les images ci-dessous représentent le nombre de paquets en fonction des adresses des sources.

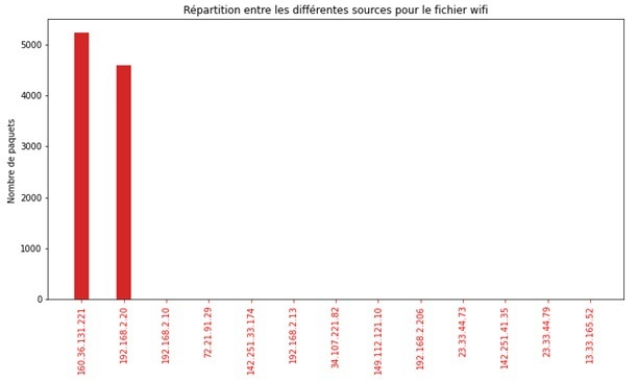


(a) Appel Ethernet

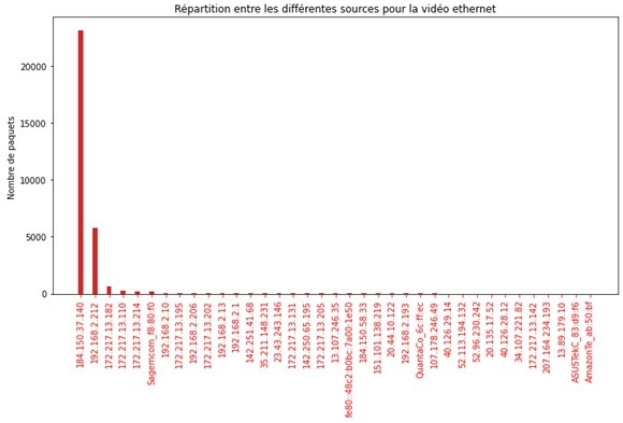
(b) Appel Wi-Fi



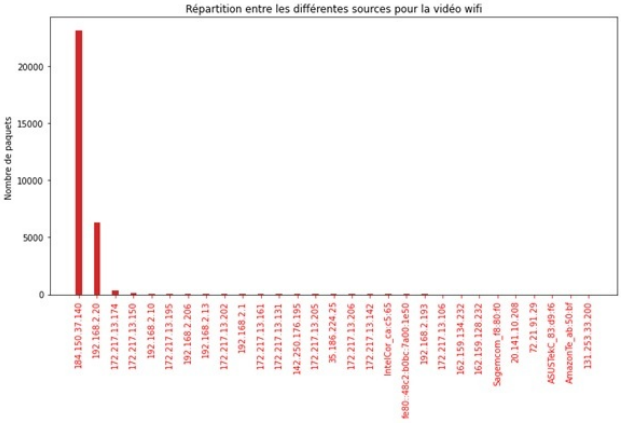
(c) Téléchargement Ethernet



(d) Téléchargement Wi-Fi



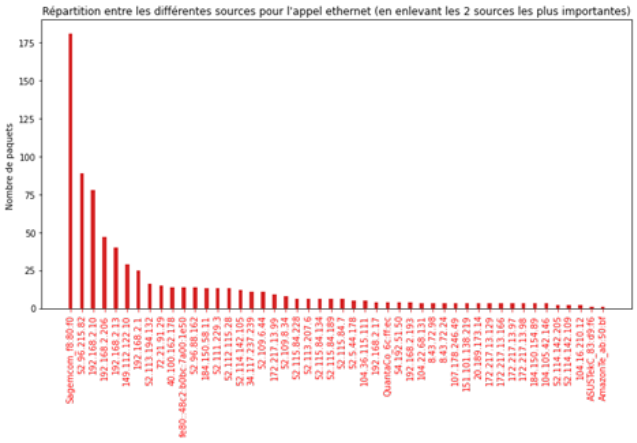
(e) Vidéo Ethernet



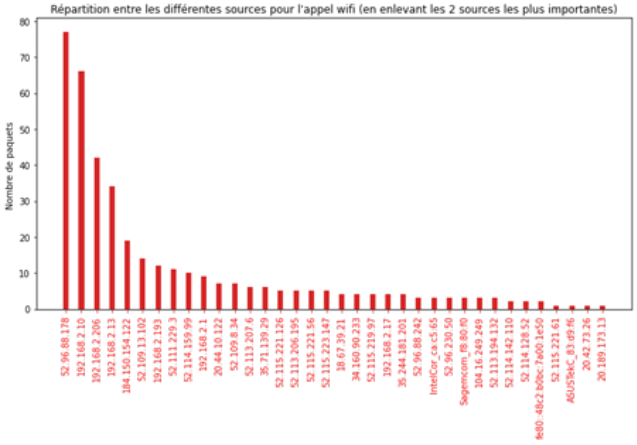
(f) Vidéo Wi-Fi

FIGURE 7.1 – Graphique de répartition des différentes sources pour chaque expérience

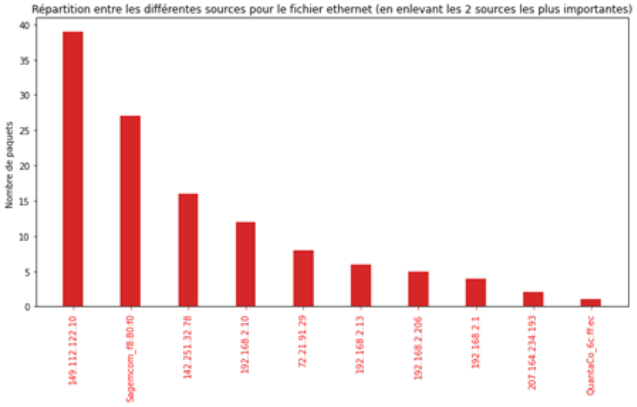
Les images ci-dessous représentent le nombre de paquets en fonction des adresses des sources lorsque l'on ne considère pas les deux sources les plus communes.



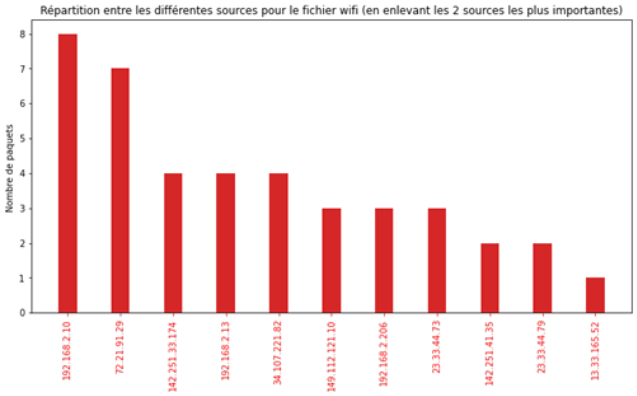
(a) Appel Ethernet



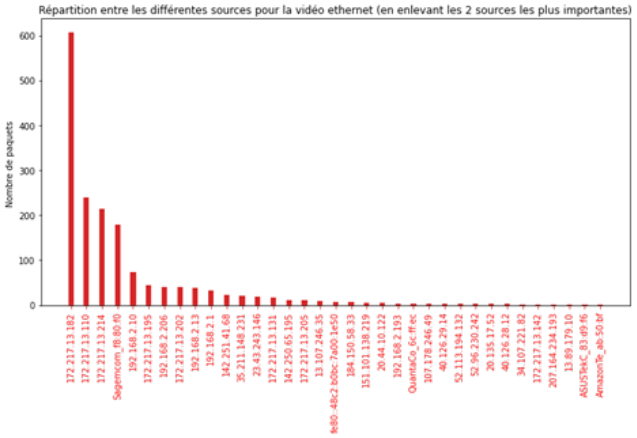
(b) Appel Wi-Fi



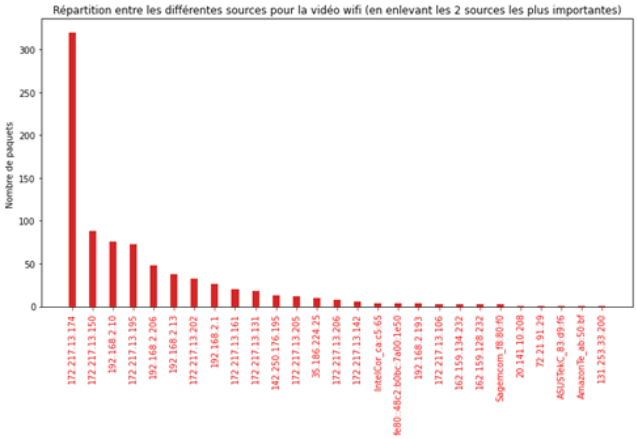
(c) Téléchargement Ethernet



(d) Téléchargement Wi-Fi



(e) Vidéo Ethernet



(f) Vidéo Wi-Fi

FIGURE 7.2 – Graphique de répartition des différentes sources pour chaque expérience sans les 2 sources les plus communes

Bibliographie

[net] NETLIB. *LAPACK, version 3.7.1*. URL : https://netlib.org/lapack/#_lapack_version_3_7_1. *(accessed : 04.12.2022).

[Sem] Sky SEMIJON. *Montreal city in 5K by Sky Semijon*. URL : https://www.youtube.com/watch?v=MveOhOvhV6U. (accessed : 01.12.2022).

[Wik] In WIKIPEDIA. *QUIC, Communication Protocol*. URL : https://en.wikipedia.org/wiki/QUIC. (accessed : 03.12.2022).