UQAC
Université du Québec
à Chicoutimi

TELECOM Nancy
Ingénieurs du numérique • Inspiring your digital future

Cours 8INF950 : SUJET SPÉCIAL EN MAITRISE INFORMATIQUE

Rapport final

Yohann Jardin

Session : Été 2015
Professeur : Abdenour BOUZOUANE

Sommaire

Introduction

Le nombre de données généré chaque année augmente exponentiellement. Dans le domaine de la classification, être capable de suivre cette évolution est un des enjeux à relever.
Pour cela, Apache propose le projet Hadoop, un système de fichier accompagné d'une méthode de traitement qui a pour but d'être scalable et performant.

Après avoir présenté les lectures qui m'ont permis de survoler l'état de l'art et comprendre le fonctionnement d'Hadoop, je présenterais mon implémentation de l'algorithme C4.5 basé sur Hadoop. S'ensuivra les expériences qui ont été effectuées et une conclusion.

Lectures

Deux articles ont été lus pour connaître ce qui est envisagé pour paralléliser ou utiliser Hadoop avec l'algorithme C4.5.

- Wei Dai et coll. [1] présentent le fonctionnement du système de fichiers d'Hadoop (HDFS) et de MapReduce, ainsi que leur implémentation de l'algorithme de C4.5 utilisant Hadoop pour faire de la parallélisation de données verticales.
- Nuno Amano et al. [1] présentent les différentes formes de parallélismes qui peuvent être utilisées suivi d'une implémentation de C4.5 basé sur la parallélisation hydride.

Le livre de Tom White [3] présente comment Hadoop fonctionne, comment travailler avec, ainsi que les projets qui y sont liés.

Des résumés de mes lectures se trouvent en annexes.

Implémentation de C4.5

Algorithme

L'algorithme se base sur une parallélisation horizontale des données, le but étant de montrer qu'il est possible de traiter n'importe quel volume de données venant de plusieurs ordinateurs.
Pour fonctionner, il est découpé en « Job », chacun ayant une opération Map et une opération Reduce.

Le code complet est disponible sur github à l'adresse https://github.com/yohannj/C4.5_Hadoop, sous licence GPL.

Job

Trois jobs différents sont utilisés :

1) « summarizeData » récupère les données de fichiers se trouvant sur HDFS et regroupe les lignes identiques en leur ajoutant un compteur. À la fin du job, les données sont sous le format :
   Clé : valeur de l'attribut 1, valeur de l'attribut 2 (…) valeur de l'attribut n, classification
   Valeur : nombre d'instances

2) « calcAttributesInfo » sélectionne les données du nœud actuel de C4.5 et va retravailler les données pour les séparer par attribut et les mettre dans un format plus adapté pour la suite de l'algorithme. À la fin du job, les données sont sous le format :
   Clé : Text
   Valeur : MapWritable<Text, MapWritable<Text, IntWritable>>

   La clé est le nom d'un attribut. La valeur permet de parcourir les différentes valeurs possibles de cet attribut, pour chacune d'aller de parcourir les différentes classifications possibles et d'avoir le nombre d'instances qui correspondent.

3) « findBestAttribute » calcule le ratio du gain d'information de chaque attribut puis ne garde que le meilleur attribut pour séparer les données. À la fin du job, les données sont sous le format :
   Clé : NullWritable (Revient à dire qu'il n'y a pas de clés)
   Valeur : Colonne du meilleur attribut, valeur 1 de l'attribut, nombre de classes dans le nœud valeur 1, classe majoritaire dans le nœud valeur 1, (…), valeur n de l'attribut, nombre de classes dans le nœud valeur n, classe majoritaire dans le nœud valeur n, nombre d'attributs restant pour de futures séparations de données.

Fonctionnement

Le premier job est effectué une seule fois au début de l'algorithme. Ensuite, pour chaque nœud, les jobs 2 et 3 sont effectués l'un après l'autre.
Lorsque la construction de l'arbre est finie, toutes les règles trouvées sont envoyées à l'utilisateur.

Utilisation

Sur un ordinateur où Hadoop est installé et configuré, mettez un jeu de données sur Hadoop. Si ce jeu de données est découpé en plusieurs fichiers, mettez-le dans un même dossier.
Créez un .jar du code source https://github.com/yohannj/C4.5_Hadoop. Mettez son chemin dans la variable d'environnement « HADOOP_CLASSPATH »

En ligne de commande, écrivez :

Hadoop main/C4_5 <chemin du fichier ou dossier du jeu de données> <un chemin pour stocker les calculs temporaires> <un chemin pour noter les calculs finaux>

Une fois que l'exécution de l'algorithme sera terminée, son résultat sera affiché dans l'invite de commande.

Limites

C4.5 est une évolution de l'algorithme ID3 permettant entre autres de travailler avec des attributs dont les données sont continues.
Le travail avec de tels attributs demande de déterminer des valeurs pour séparer ces attributs de manière optimale. Ce calcul n'a pas été envisagé, les attributs des jeux de données utilisés sont donc considérés comme discrets.

Expériences

Conditions

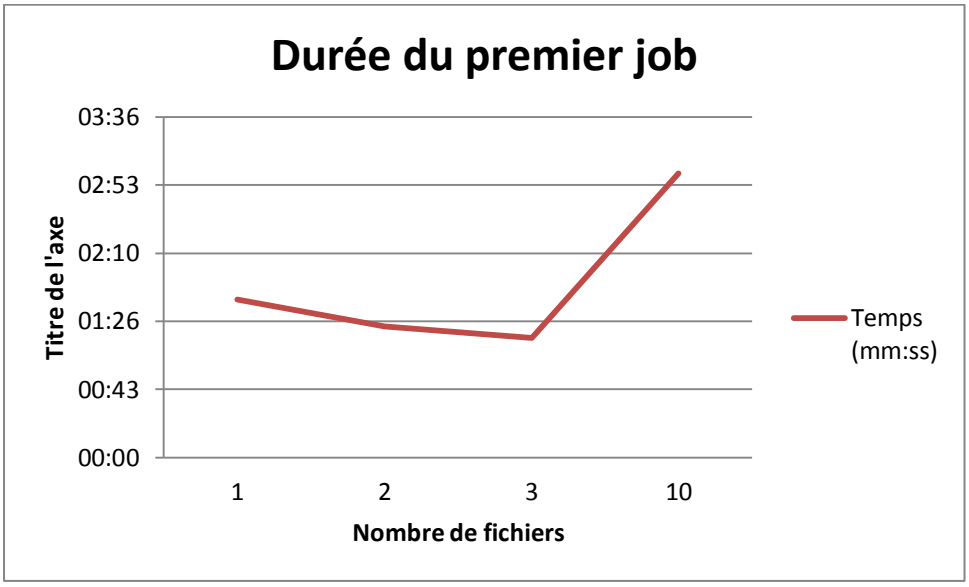
Toutes les expériences ont été effectuées avec un seul ordinateur, au sein d'un environnement virtuel de Cloudera [4] où Hadoop est installé et configuré. 3 cœurs du processeur AMD FX-6100 3.3GHz ont été utilisés.

Jeu de données

Le jeu de donnée utilisé est basé sur « KDD Cup 1999 » [5] où seuls les attributs discrets ont été gardés : protocol_type, service, flag, land, logged_in, is_host_login, is_guest_login.

Le premier ne contient que 14 lignes, 4 attributs et 2 classifications sont possibles. Le deuxième contient 4 898 431 lignes et 23 classifications sont possibles.

Résultats



Dans l'intérêt d'être scalable, l'influence du nombre de fichiers pour représenter le jeu de données a été testée. Le seul impact de cette modification est sur le premier job qui n'est lancé qu'une seule fois au début.

Hadoop fonctionne avec des morceaux de fichiers plutôt que le fichier lui-même. Les morceaux peuvent faire jusqu'à 128 Mo. Dans le cas où l'on a un fichier, celui-ci pèse 134Mo, il sera donc dans deux morceaux de 128 Mo et 6 Mo.
Dans les autres cas, les différents fichiers pèsent moins de 128 Mo, et le poids est réparti équitablement.

Avec 1, 2 et 3 fichiers, on utilise au plus 3 morceaux de fichiers, la machine utilisée pour les tests ayant 3 cœurs, c'est le morceau le plus gros qui impose la durée du calcul. Ceci explique le gain de performance observé.

Dans le cas avec 10 morceaux, Hadoop perd du temps à gérer l'ordonnancement des calculs sur le processeur. Dans un cas où les fichiers pèsent bien moins que 128 Mo et qu'il y en a plus qu'il n'y a de cœurs, il est préférable de les regrouper ensemble pour limiter les calculs d'ordonnancement.

Par la suite, les fichiers ont toujours fait moins de 128 Mo, MapReduce ne travaillait donc qu'avec 1 morceau et n'exploitait pas toute la puissance disponible.
Le temps total que l'algorithme génère les règles pour ce jeu de données est de 5 heures et 42 minutes. Il n'a pas été possible de faire de comparaison avec le logiciel Weka, ce dernier ne travaille qu'avec un seul fichier et a besoin de le mettre en mémoire vive, ce qui n'était, ici, pas possible.

Pour se rendre compte du travail effectué en 5 heures et 42 minutes, les règles trouvées par l'algorithme sont disponibles en annexe.

Remarque

Le but de ce cours est d'implémenter C4.5 en utilisant MapReduce et mon implémentation a été effectuée en ce sens. Cependant, on se rend compte que le dernier job, qui calcule le ratio du gain d'information, ne travaille toujours qu'avec un seul cœur, ce qui est particulièrement dérangeant pour un calcul qui est très gourmand en temps processeur.
En ce sens, en n'utilisant pas MapReduce pour calculer le ratio du gain d'information, il aurait été possible d'effectuer ce calcul en parallèle pour chaque attribut sur les 3 processeurs. On peut ainsi s'attendre à de meilleures performances.

Conclusion

Au sein de ce cours, j'ai montré qu'un algorithme tel que C4.5 est adaptable au modèle MapReduce et permet de travailler avec des volumes importants de données en les répartissant sur un cluster. Pour de futurs travaux, il serait intéressant de limiter l'usage de MapReduce à certaines parties de l'algorithme pour s'assurer d'utiliser le parallélisme lors des parties des parties calculatoires. Je tenterais aussi d'utiliser de Spark, semblable à MapReduce, qui semble plus adapté à ce type de problème et devraient lui aussi apporter un gain de performance important.

Références

[1] Wei Dai, Wei Ji. *A MapReduce implementation of C4.5 decision tree algorithm*. International Journal of Database Theory and Application, 2014, 7(1):49-60.Z

[2] Nuno Amano, Joao Gama, and Fernando Silva. *Exploiting Parallelism in Decision Tree Induction*. In Proceedings from the ECML/PKDD Workshop on Parallel and Distributed computing for Machine Learning, pages 13–22, September 2003.

[3] Tom White, *Hadoop: The Definitive Guide*, Fourth Edition, O'Reilly.

[4] Environnement virtuel *QuickStart VMs for CDH 5.4.x*, version sortie le 27 avril 2015, http://www.cloudera.com/content/cloudera/en/downloads/quickstart_vms/cdh-5-4-x.html, Cloudera. Site internet accessible le 24 juin 2015.

[5] Jeu de donnée KDD Cup 1999, *kddcup.data.gz*, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Site internet accessible le 24 juin 2015.

Annexes

Résumé de l'article « A MapReduce Implementation of C4.5 Decision Tree Algorithm »

Introduction

Les arbres de décisions sont des méthodes de forages de données dont le but est d'aider la prise de décisions en classifiant des tuples. L'un des plus connus est C4.5.

Leurs créations sont généralement extrêmement longues à exécuter avec de gros volumes de données, et bien que cela puisse être parallélisé, il faut réussir à minimiser les coûts de communication.

Pour traiter les gros volumes de données tout en minimisant les coûts de communication, nous utilisons le modèle MapReduce.

Arbres de décisions

Un arbre de décisions est composé de nœuds de décisions pour tester un attribut, et de liens indiquant une valeur, ou intervalle de valeurs, de l'attribut tester. Après une série de tests, on arrive à un bout de l'arbre appelé feuille, indiquant le résultat de la classification.

C4.5

Pour créer l'arbre de décision, C4.5 cherche à trouver le meilleur attribut pour séparer les données à chaque étape, en s'appuyant sur le ratio du gain d'information.
Ainsi l'entropie est d'abord calculée ; elle permet de calculer l'information, et donc le gain. Ce gain est ensuite est divisé par la fonction SplitInfo, donnant ainsi le ratio du gain d'information.

Hadoop MapReduce

Nous avons utilisé une implémentation open source de MapReduce, Hadoop, proposant en plus un système de fichier distribué.
MapReduce travaille avec des données de type clé-valeur. L'opération « Map » permet de travailler sur des données de type clé-valeur pour générer des couples clé-valeur. Ces couples sont ensuite triés et distribués aux opérations « Reduce » pour en ressortir un résultat.

Algorithme proposé

Notre algorithme utilise 3 structures de données :

1. Une table d'attributs contenant les informations sur les attributs
2. Une table de comptage indiquant le nombre d'instances d'une classe si les données sont séparées par un attribut a
3. Une table de hachage pour lier les informations entre nœuds de l'arbre, ainsi que sa hiérarchie

Les données sont partitionnées équitablement pour optimiser l'efficacité de la parallélisation.

L'algorithme en lui-même est décomposé en 4 phases :

1. Préparation des données : Un Map et un Reduce permettent de créer respectivement la table d'attributs et la table de comptage.

2. Sélection du meilleur attribut : Un premier Reduce part des données de la table de comptage pour créer des couples clé-valeur dont la clé est un attribut et la valeur le nombre d'instances possédant cet attribut. Ensuite un Map calcule l'information et le SplitInfo pour chaque attribut. Un second Reduce s'occupe de calculer le ratio du gain d'information associé à chaque attribut.
3. Mise à jour : La table de comptage et la table de hachage sont mises à jour par le biais de deux Map.
4. Croissance de l'arbre : Création des nœuds enfants. Soit une feuille, soit un nouveau choix d'attribut.

Expérience

Nous nous sommes focalisés sur la rapidité d'exécution de l'algorithme.

Quatre ordinateurs identiques ont été utilisés, ils ont chacun 2 cœurs dans leurs processeurs. Ils se sont réparti un jeu de données de 6 attributs. Pour partager les données, trois ordinateurs ont reçu un attribut, et le dernier a reçu trois attributs.

Un premier test en faisant tous les calculs sur un cœur montre que plus il y a de tuples, plus le temps d'exécution dure longtemps. Cependant, ce temps augmente plus faiblement avec notre implémentation.

En rajoutant de cœurs pour effectuer le calcul, le temps d'exécution diminue. De plus, avec un nombre suffisant de cœurs, le temps d'exécution varie peu avec une augmentation du nombre de tuples.

Conclusion

Les arbres de décisions séquentiels ne permettent pas de gérer les grandes masses de données, notre version parallélisée de C4.5 résout ce problème, est scalable et permet d'avoir de bonnes performances concernant le temps d'exécution.

Résumé de l'article « Exploiting Parallelism in Decision Tree Induction »

Introduction

Parmi les méthodes de classifications, les arbres de décisions sont rapides à construire et apporte des résultats aussi bons voire meilleurs que les autres méthodes. Les données à traiter étant de plus en plus importantes, il est nécessaire que les algorithmes de classifications soient efficaces et scalable. La parallélisation peut répondre à ce besoin, mais il faut être capable de gérer les données qui vont devoir être distribuées pour que le travail soit réparti uniformément, sans que les coûts de communications ne dégradent les performances de l'algorithme.

Nous avons implémenté une version parallélisée de C4.5, en effectuant une construction en largeur, qui gère les valeurs manquantes.

Approches de parallélisation

Parallélisme de tâches

Fonctionnement : Initialement un seul processeur crée l'arbre. Dès qu'il y a autant de nœuds de décisions que de processeur, chacun s'occupe d'un nœud.
Problèmes : Mauvaise répartition de la charge de travail. Soit un coût de communications élevé, soit toutes les données sont en mémoire.

Parallélisme de données vertical

Fonctionnement : Partage les données entre différents processeurs en séparant les attributs.
Problèmes : En séparant par les attributs, une mauvaise répartition de la charge de travail est présente, pour gérer les attributs continus. De plus, cette approche est peu scalable vu qu'il n'y a pas plus de processeurs que d'attributs.

Parallélisme de données horizontal

Fonctionnement : Partage les données entre différents processeurs en séparant les tuples.
Problèmes : Coûts de communication important pour déterminer le meilleur attribut pour séparer les données, ainsi que pour gérer les attributs continus.

Parallélisme hybride

Fonctionnement : Utilise d'abord un parallélisme de données et fini la construction de l'arbre avec un parallélisme de tâche. Cela permet de limiter les problèmes dû à la mauvaise répartition de charges tout en limitant les coûts de communications.

Implémentation parallélisée de C4.5

Nous avons suivi l'approche hybride, avec un parallélisme de données horizontal.

Les tuples sont répartis équitablement entre les processeurs pour avoir une bonne répartition du travail. Lorsque le nombre de tuples dans chaque nœud est plus faible qu'une valeur fixée, les processeurs se partagent les nœuds pour finir l'arbre. Si un processeur a fini de s'occuper des nœuds qui lui sont attribués, il contacte les autres processeurs pour équilibrer la charge de travail.

Dans la première étape, chaque processeur construit une liste d'attributs et une liste de classes, basé sur ses données. Les processeurs communiquent la distribution de leurs données entre eux pour déterminer le meilleur attribut.
Pour les attributs continus, étant donnés qu'ils étaient préalablement triés, chaque processeur va

tester toutes les séparations possibles en ayant en mémoire les distributions des processeurs avant lui qui sont pour les valeurs les plus faibles de l'attribut, et les distributions des processeurs après lui. Une fois le meilleur attribut trouvé, la séparation des données est effectuée, chaque processeur doit alors mettre à jour leurs listes d'attributs et de classes.

Passer à la deuxième étape au bon moment est crucial : c'est l'un des facteurs les plus importants impactant les performances de l'algorithme. Nous faisons le changement lorsque les coûts totaux de communications sont plus importants que les coups pour construire un nœud localement.

Résultats expérimentaux

Nous nous sommes focalisés sur la rapidité d'exécution de l'algorithme.

Quatre ordinateurs identiques ont été utilisés. Le jeu de données choisi est « Synthetic » d'Agrawal et ses collaborateurs ; il comprend 9 attributs parmi lesquels 5 sont continus.

Un premier test fait varier le nombre de processeurs utilisés entre 1 et 4, et fait varier le nombre de tuples entre 100k, 200k et 400k.
Notre algorithme montre un gain de vitesse, tel que prévu. Le ralentissement observé avec 4 processeurs est dû aux coûts de communications important durant la phase de séparation des données.

Un second test utilise cette fois-ci 50k, 75k et 100k tuples et montre une bonne scalabilité. Encore une fois, les coûts de communications impactent nos résultats.

Conclusion

Après avoir énoncé des généralités sur le parallélisme pour la construction d'arbre de décisions, nous avons implémenté une version parallélisée de C4.5 plus rapide et scalable tout en ayant les mêmes résultats, mais sujet à des problèmes de performances dû aux coûts de communications.

Résumé de chapitres du livre « Hadoop The Definitive Guide »

Chapitre 1 – Meet Hadoop

Sujet	Concepts clés
Histoire	Comment Hadoop a été créé, l'usage pour lequel il est fait
HDFS	Système de fichier distribué
MapReduce	Modèle de programmation basé sur l'utilisation de couples clé-valeur
Projets	Des projets complémentaires permettent d'enrichir Hadoop, Apache en héberge plusieurs. Notamment Spark améliorant les performances des algorithmes itératifs (type machine learning)

Chapitre 2 – MapReduce

Sujet	Concepts clés
Map	Processus qui sélectionne les données qui vont être utilisées
Shuffle	Processus qui trie et regroupe les valeurs selon leur clé
Reduce	Processus qui effectue un calcul à partir de couples clé-valeur
Combiner	Optimisation optionnelle pour minimiser la taille des données envoyée entre Map et Reduce
org.apache.hadoop	API permettant de travailler avec Hadoop en Java
Hadoop Streaming	API permettant d'effectuer des opérations MapReduce en lisant des données en Streaming, utilisable avec tout langage pouvant lire et écrire sur l'entrée standard
Job	Unité de travail, spécifiant ce qu'il faut accomplir et comment

Chapitre 3 – The Hadoop Distributed Filesystem

Sujet	Concepts clés
Block	Taille de donnée minimum écrite/lue : 128MB dans HDFS
Namenode	Gère le système de fichier et les références, ainsi que les datanodes (maître/esclaves)
Datanode	Stocke et lis des blocks.
High Availability	Évite les interruptions de services lorsqu'une panne a lieu
Accessibilité	Possibilité de parcourir les fichiers en ligne de commandes, en utilisant une interface graphique ou encore une API Java
Redondance	Les données sont sauvegardées à plusieurs endroits évitant les pertes s'il y a un problème avec l'une des copies

Chapitre 4 – YARN

Sujet	Concepts clés
YARN	Gère les ressources d'un cluster et offre une couche d'abstraction d'HDFS, utilisable par MapReduce et d'autres applications
But	Simplifie et rend plus ergonomique l'utilisation d'Hadoop. Permets aussi une meilleure scalabilité que Mapreduce 1
Ordonnancement	Gère la répartition des tâches à effectuer en FIFO, selon les capacités ou en étant équitable

Chapitre 5 – Hadoop I/O

Sujet	Concepts clés
Intégrité	Les données sont contrôlées pour vérifier qu'elles ne sont pas corrompues
Compression	Possibilité de compresser les données pour le stockage et pour l'envoi à l'intérieur d'un cluster Différent codec sont disponibles selon le besoin de vitesse, de taux de compression ou la possibilité décompresser un morceau d'un fichier qui a été morcelé
Writable	Interface offrant une abstraction sur la sérialisation/dé-sérialisation des données Redéfinis les types de base de Java
WritableComparable	Interface ajoutant la comparaison à l'interface Writable
SequenceFile	Structure de données compressible permettant de stocker des données de type clé-valeur ou des blocks

Chapitre 19 – Spark

Sujet	Concepts clés
Spark	Framework pour traiter de grandes masses de données n'utilisant pas Mapreduce bien que similaire, compatible avec YARN et utilisant HDFS
RDD	« Resilient Distributed Dataset » : Abstraction de la lecture de donnée au cœur de Spark Un RDD peut soit subir une transformation créant un autre RDD, mais n'effectuant pas encore le calcul de la transformation, soit une action qui va alors lancer tous les calculs à faire
Job	Ensemble d'étapes de calcul à effectuer, elles-mêmes séparées en tâches Le driver gère l'organisation du job, tandis que des exécuteurs s'occupent d'appliquer les tâches (maître/esclaves)
Variable partagée	Variable envoyée à chaque exécuteur et mise en cache
Persistance	Les données peuvent être mises en cache pour être plus rapidement accessibles à un autre moment dans le job
Accessibilité	Utilisable en ligne de commande ou via une API (Java/Scala/Python)

Résultat de l'algorithme sur le jeu de données KDD Cup 99

395 règles ont été déterminées par l'algorithme en 5 heures et 42 minutes :

0=icmp, 1=eco_i, 2=SF, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: ipsweep
0=icmp, 1=red_i, CLASSIFICATION: normal
0=icmp, 1=tim_i, 2=SF, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=icmp, 1=urh_i, CLASSIFICATION: normal
0=icmp, 1=urp_i, 2=SF, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=IRC, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=IRC, 2=RSTO, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=IRC, 2=RSTO, 4=1, CLASSIFICATION: normal
0=tcp, 1=IRC, 2=RSTR, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=IRC, 2=RSTR, 4=1, CLASSIFICATION: normal
0=tcp, 1=IRC, 2=S1, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=IRC, 2=S1, 4=1, CLASSIFICATION: normal
0=tcp, 1=IRC, 2=S3, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=IRC, 2=S3, 4=1, CLASSIFICATION: normal
0=tcp, 1=IRC, 2=SF, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=IRC, 2=SF, 4=1, CLASSIFICATION: normal
0=tcp, 1=X11, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=X11, 2=RSTR, 4=1, CLASSIFICATION: normal
0=tcp, 1=X11, 2=S0, 3=0, 4=0, CLASSIFICATION: satan
0=tcp, 1=X11, 2=S1, 4=1, CLASSIFICATION: normal
0=tcp, 1=X11, 2=SF, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=X11, 2=SF, 4=1, CLASSIFICATION: normal
0=tcp, 1=Z39_50, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=Z39_50, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=Z39_50, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=Z39_50, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=aol, 3=0, 4=0, CLASSIFICATION: satan
0=tcp, 1=auth, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=auth, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=auth, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=auth, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=auth, 2=SF, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=auth, 2=SF, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=auth, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=bgp, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=bgp, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=bgp, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=bgp, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=courier, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=courier, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=courier, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=courier, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=csnet_ns, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=csnet_ns, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=csnet_ns, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=csnet_ns, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=ctf, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=ctf, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=ctf, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=ctf, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=ctf, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=daytime, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: satan
0=tcp, 1=daytime, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=daytime, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=daytime, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=daytime, 2=RSTR, 4=1, CLASSIFICATION: satan
0=tcp, 1=daytime, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune

0=tcp, 1=daytime, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=discard, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: satan
0=tcp, 1=discard, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=discard, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=discard, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=discard, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=discard, 2=SF, 4=1, CLASSIFICATION: satan
0=tcp, 1=discard, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=domain, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=domain, 2=RSTO, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=domain, 2=RSTO, 4=1, CLASSIFICATION: satan
0=tcp, 1=domain, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=domain, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=domain, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=domain, 2=SF, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: ipsweep
0=tcp, 1=domain, 2=SF, 4=1, CLASSIFICATION: normal
0=tcp, 1=domain, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=echo, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: satan
0=tcp, 1=echo, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=echo, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=echo, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=echo, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=echo, 2=SF, 4=1, CLASSIFICATION: satan
0=tcp, 1=echo, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=efs, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=efs, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=efs, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=efs, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=exec, 2=REJ, 3=0, 4=0, CLASSIFICATION: satan
0=tcp, 1=exec, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=exec, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=exec, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=exec, 2=SF, 3=0, 4=0, CLASSIFICATION: satan
0=tcp, 1=exec, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=finger, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=finger, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=finger, 2=RSTR, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: portsweep
0=tcp, 1=finger, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=finger, 2=S0, 3=1, 4=0, 5=0, 6=0, CLASSIFICATION: land
0=tcp, 1=finger, 2=S1, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=finger, 2=S3, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=finger, 2=SF, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=finger, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=ftp, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=ftp, 2=RSTO, 3=0, 4=1, 5=0, 6=1, CLASSIFICATION: normal
0=tcp, 1=ftp, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=ftp, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=ftp, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=ftp, 2=S1, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=ftp, 2=S1, 4=1, CLASSIFICATION: normal
0=tcp, 1=ftp, 2=S2, 4=1, CLASSIFICATION: normal
0=tcp, 1=ftp, 2=SF, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=ftp, 2=SF, 3=0, 4=0, 5=0, 6=1, CLASSIFICATION: normal
0=tcp, 1=ftp, 2=SF, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=ftp, 2=SF, 3=0, 4=1, 5=0, 6=1, CLASSIFICATION: normal
0=tcp, 1=ftp, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=ftp_data, 2=OTH, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=ftp_data, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=ftp_data, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=ftp_data, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=ftp_data, 2=RSTR, 4=1, CLASSIFICATION: warezclient
0=tcp, 1=ftp_data, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune

0=tcp, 1=ftp_data, 2=S1, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=ftp_data, 2=S1, 4=1, CLASSIFICATION: normal
0=tcp, 1=ftp_data, 2=S2, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=ftp_data, 2=S2, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=ftp_data, 2=S3, 4=1, CLASSIFICATION: normal
0=tcp, 1=ftp_data, 2=SF, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=ftp_data, 2=SF, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=ftp_data, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=gopher, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=gopher, 2=RSTO, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=gopher, 2=RSTO, 4=1, CLASSIFICATION: ipsweep
0=tcp, 1=gopher, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=gopher, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=gopher, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=gopher, 2=SF, 3=0, 4=0, CLASSIFICATION: ipsweep
0=tcp, 1=gopher, 2=SF, 4=1, CLASSIFICATION: satan
0=tcp, 1=gopher, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=harvest, 3=0, 4=0, CLASSIFICATION: satan
0=tcp, 1=hostnames, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=hostnames, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=hostnames, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=hostnames, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=http, 2=OTH, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=http, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=http, 2=RSTO, 4=1, CLASSIFICATION: normal
0=tcp, 1=http, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=http, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=http, 2=RSTR, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: back
0=tcp, 1=http, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=http, 2=S1, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=http, 2=S2, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=http, 2=S2, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=http, 2=S3, 4=1, CLASSIFICATION: normal
0=tcp, 1=http, 2=SF, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=http, 2=SF, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=http, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=http_2784, 3=0, 4=0, CLASSIFICATION: satan
0=tcp, 1=http_443, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=http_443, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=http_443, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=http_443, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=http_8001, 3=0, 4=0, CLASSIFICATION: satan
0=tcp, 1=imap4, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=imap4, 2=RSTO, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=imap4, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=imap4, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=imap4, 2=S1, 3=0, 4=0, CLASSIFICATION: imap
0=tcp, 1=imap4, 2=SF, 3=0, 4=0, CLASSIFICATION: imap
0=tcp, 1=imap4, 2=SF, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=imap4, 2=SH, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: imap
0=tcp, 1=iso_tsap, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=iso_tsap, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=iso_tsap, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=iso_tsap, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=klogin, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=klogin, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=klogin, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=klogin, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=kshell, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=kshell, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=kshell, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=kshell, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap

0=tcp, 1=ldap, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=ldap, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=ldap, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=ldap, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=link, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=link, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=link, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=link, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=login, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=login, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=login, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=login, 2=SF, 3=0, 4=0, CLASSIFICATION: satan
0=tcp, 1=login, 2=SF, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: ftp_write
0=tcp, 1=login, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=mtp, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=mtp, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=mtp, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=mtp, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=mtp, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=name, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=name, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=name, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=name, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=name, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=netbios_dgm, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=netbios_dgm, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=netbios_dgm, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=netbios_dgm, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=netbios_ns, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=netbios_ns, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=netbios_ns, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=netbios_ns, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=netbios_ssn, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=netbios_ssn, 2=RSTO, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=netbios_ssn, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=netbios_ssn, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=netbios_ssn, 2=SF, 3=0, 4=0, CLASSIFICATION: satan
0=tcp, 1=netbios_ssn, 2=SF, 4=1, CLASSIFICATION: satan
0=tcp, 1=netbios_ssn, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=netstat, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=netstat, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=netstat, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=netstat, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=netstat, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=nnsp, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=nnsp, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=nnsp, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=nnsp, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=nntp, 2=OTH, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=nntp, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: portsweep
0=tcp, 1=nntp, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=nntp, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=nntp, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=nntp, 2=SF, 4=1, CLASSIFICATION: satan
0=tcp, 1=nntp, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=other, 2=OTH, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=other, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: satan
0=tcp, 1=other, 2=RSTO, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=other, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=other, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: satan
0=tcp, 1=other, 2=S1, 4=1, CLASSIFICATION: normal
0=tcp, 1=other, 2=S2, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal

0=tcp, 1=other, 2=S2, 4=1, CLASSIFICATION: normal
0=tcp, 1=other, 2=S3, 4=1, CLASSIFICATION: warezclient
0=tcp, 1=other, 2=SF, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=other, 2=SF, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=other, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=pm_dump, 3=0, 4=0, CLASSIFICATION: satan
0=tcp, 1=pop_2, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=pop_2, 2=RSTO, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=pop_2, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=pop_2, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=pop_2, 2=SF, 3=0, 4=0, CLASSIFICATION: satan
0=tcp, 1=pop_2, 2=SF, 4=1, CLASSIFICATION: satan
0=tcp, 1=pop_2, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=pop_3, 2=REJ, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=pop_3, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=pop_3, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=pop_3, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=pop_3, 2=S1, 4=1, CLASSIFICATION: normal
0=tcp, 1=pop_3, 2=SF, 3=0, 4=0, CLASSIFICATION: satan
0=tcp, 1=pop_3, 2=SF, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=pop_3, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=printer, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=printer, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=printer, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=printer, 2=S1, 4=1, CLASSIFICATION: satan
0=tcp, 1=printer, 2=SF, 3=0, 4=0, CLASSIFICATION: satan
0=tcp, 1=printer, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=private, 2=OTH, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=private, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=private, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=private, 2=RSTO, 4=1, CLASSIFICATION: satan
0=tcp, 1=private, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=private, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=private, 2=S1, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=private, 2=SF, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=private, 2=SF, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: satan
0=tcp, 1=private, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=remote_job, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=remote_job, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=remote_job, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=remote_job, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=remote_job, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=rje, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=rje, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=rje, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=rje, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=shell, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=shell, 2=RSTO, 4=1, CLASSIFICATION: normal
0=tcp, 1=shell, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=shell, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=shell, 2=S1, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=shell, 2=SF, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=shell, 2=SF, 4=1, CLASSIFICATION: satan
0=tcp, 1=shell, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=smtp, 2=OTH, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=smtp, 2=REJ, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=smtp, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=smtp, 2=RSTO, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=smtp, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=smtp, 2=RSTR, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: portsweep
0=tcp, 1=smtp, 2=RSTR, 4=1, CLASSIFICATION: normal
0=tcp, 1=smtp, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune

0=tcp, 1=smtp, 2=S1, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=smtp, 2=S1, 4=1, CLASSIFICATION: normal
0=tcp, 1=smtp, 2=S2, 4=1, CLASSIFICATION: normal
0=tcp, 1=smtp, 2=S3, 4=1, CLASSIFICATION: normal
0=tcp, 1=smtp, 2=SF, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=smtp, 2=SF, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=smtp, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=sql_net, 2=OTH, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=sql_net, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=sql_net, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=sql_net, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=sql_net, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=ssh, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=ssh, 2=RSTO, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=ssh, 2=RSTO, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: ipsweep
0=tcp, 1=ssh, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=ssh, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=ssh, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=ssh, 2=SF, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=ssh, 2=SF, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=ssh, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=sunrpc, 2=OTH, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=sunrpc, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=sunrpc, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=sunrpc, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=sunrpc, 2=SF, 3=0, 4=0, CLASSIFICATION: satan
0=tcp, 1=sunrpc, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=supdup, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=supdup, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=supdup, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=supdup, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=systat, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=systat, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=systat, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=systat, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=systat, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=telnet, 2=OTH, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=telnet, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=telnet, 2=RSTO, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=telnet, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=telnet, 2=RSTR, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: portsweep
0=tcp, 1=telnet, 2=RSTR, 4=1, CLASSIFICATION: normal
0=tcp, 1=telnet, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=telnet, 2=S1, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=telnet, 2=S1, 4=1, CLASSIFICATION: normal
0=tcp, 1=telnet, 2=S3, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: guess_passwd
0=tcp, 1=telnet, 2=S3, 4=1, CLASSIFICATION: normal
0=tcp, 1=telnet, 2=SF, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=telnet, 2=SF, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=telnet, 2=SF, 3=0, 4=1, 5=0, 6=1, CLASSIFICATION: normal
0=tcp, 1=telnet, 2=SF, 3=0, 4=1, 5=1, 6=0, CLASSIFICATION: normal
0=tcp, 1=telnet, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=telnet, 3=1, 4=0, CLASSIFICATION: land
0=tcp, 1=time, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=time, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=time, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=time, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=time, 2=S2, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=time, 2=S3, 3=0, 4=0, CLASSIFICATION: normal
0=tcp, 1=time, 2=SF, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=time, 2=SF, 3=0, 4=1, 5=0, 6=0, CLASSIFICATION: normal
0=tcp, 1=time, 2=SH, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal

0=tcp, 1=uucp, 2=REJ, 3=0, 4=0, CLASSIFICATION: satan
0=tcp, 1=uucp, 2=RSTO, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=uucp, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=uucp, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=uucp, 2=SF, 4=1, CLASSIFICATION: satan
0=tcp, 1=uucp, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=uucp_path, 2=OTH, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=uucp_path, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=uucp_path, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=uucp_path, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=uucp_path, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=vmnet, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=vmnet, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=vmnet, 2=S0, 3=0, 4=0, CLASSIFICATION: neptune
0=tcp, 1=vmnet, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 1=whois, 2=REJ, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=whois, 2=RSTOS0, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=whois, 2=RSTR, 3=0, 4=0, CLASSIFICATION: portsweep
0=tcp, 1=whois, 2=S0, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: neptune
0=tcp, 1=whois, 2=SH, 3=0, 4=0, CLASSIFICATION: nmap
0=tcp, 2=OTH, 4=1, CLASSIFICATION: normal
0=tcp, 2=REJ, 4=1, CLASSIFICATION: normal
0=tcp, 2=S0, 4=1, CLASSIFICATION: portsweep
0=udp, 1=domain_u, 2=SF, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=udp, 1=ntp_u, CLASSIFICATION: normal
0=udp, 1=other, 2=SF, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=udp, 1=private, 2=SF, 3=0, 4=0, 5=0, 6=0, CLASSIFICATION: normal
0=udp, 1=tftp_u, CLASSIFICATION: normal