Sieci Społeczne i Eksploracja Danych

Sprawozdanie z projektu

Skład grupy:
Wojciech Bożek
Korneliusz Caputa
Grzegorz Cerowski
Michał Dydak
Paweł Garbacik
Michał Głowacki
Agnieszka Gondek
Grzegorz Holewa
Marek Kojder
Michał Kowal
Dawid Mazur
Krystian Pypłacz
Anna Romik
Dominik Samociuk
Paweł Szopa

1. Wstęp

1.1. Opis projektu

Projekt z przedmiotu Sieci Społeczne i Eksploracja Danych zakłada analizę zadanego zbioru danych pod kątem klasyfikacji za pomocą dostarczonych algorytmów. Dane wejściowe zostały przygotowane przez firmę zewnętrzną i zostały wykorzystane w pracy naukowej, o którą projekt został oparty. Zestaw algorytmów zaawansowanej alalizy danych składa się m.in. z algorytmów undersamplingu i oversamplingu, metody SMOTE (Synthetic Minority Oversampling Technique), metody WRF (Weighted Random Forest) oraz metody BRF (Balanced Random Forest).

1.2. Geneza projektu

Dane wejściowe wykorzystane w projekcie zostały zebrane przez firmę Lincoln Labs. Przed dziewięć tygodni gromadzono surowe dane TCP z typowej sieci LAN, jednocześnie przeprowadzając na sieć szereg ataków. Następnie na zebranych danych zostały zagregowane odpowiednie połączenia. Wykorzystując wiedzę na temat ataków sieciowych, do każdego rekordu dołączone zostały dodatkowe informacje, co pozwoliło podzielić ataki na 38 różnych typów i 4 główne kategorie.

Oprogramowanie do wykrywania włamań sieciowych chroni sieć komputerową przed nieautoryzowanymi użytkownikami. Zadaniem algorytmu detekcji jest zbudowanie modelu prognostycznego (tj. klasyfikatora) zdolnego do rozróżniania „złych" połączeń, zwanych włamaniami lub atakami, od „dobrych", normalnych połączeń.

W 1998 roku forma Lincoln Labs przygotowała program rządowy pod nazwą Intrusion Detection Evaluation, którego celem było badanie oraz ocena wykrywania włamań sieciowych. Specjalnie dla tego programu dostarczono dane symulujące ruch w wojskowym otoczeniu sieciowym. Zbiór treningowy to ok. cztery GB skompresowanych danych oraz siedem tygodni ruchu w sieci, co w sumie dało ok. pięć milionów rekordów. Z kolei dane testowe reprezentowane są przez zrzut rekordów z dwutygodniowej pracy sieci.

Wartym odnotowania jest fakt, że dane testowe nie mają tego samego rozkładu prawdopodobieństwa co dane treningowe i zawierają pewne specyficzne typy ataków, które nie występują w zbiorze treningowym. To sprawia, że zadanie klasyfikacji zyskuje na realistyczności. Niektórzy eksperci potwierdzają, że większość nowych typów ataków jest jedynie modyfikacją istniejących, co pozwala na ich rozpoznanie za pomocą analizy starszych danych. Zestaw danych zawiera w sumie 24 rodzaje ataków treningowych i 14 rodzajów występujących tylko w danych testowych.

2. Spotkania projektowe

2.1. 10 grudnia 2013 r.

Pierwsze spotkanie projektowe dotyczyło podstawowych aspektów przygotowania do wykonania projektu, tj. opisu danych, sposobu wykonania własnych operatorów dla oprogramowania RapidMiner oraz opisu metod klasyfikacji. Sekcja nr 1 w składzie Agnieszka Gondek oraz Michał Dydak przygotowała krótkie wyjaśnienie czego otrzymane dane dotyczą, jakich wyników powinno się oczekiwać oraz dlaczego nie są one zalecane do stosowania przez osoby zajmujące się analizą danych. Zaznaczono również, że plik z danymi w obecnej formie jest niemożliwy do przetworzenia i na jego podstawie należy spreparować nowe dane wejściowe, co wymaga określonych ram czasowych.

Sekcja nr 2 w składzie Paweł Garbacik i Marek Kojder przedstawiła metodologie udostępnioną przez twórców oprogramowania RapidMiner, które w zamyśle miały umożliwić przygotowanie własnych operatorów. Sekcja położyła nacisk na zamknięty dostęp do kodu obecnej wersji programu, wysoce ograniczone wsparcie dla wersji OpenSource oraz na nieaktualne i niekompletne rozwiązania proponowane przez użytkowników.

2.2. 17 grudnia 2013 r.

Kolejne spotkanie projektowe dotyczyło ponownego zapoznania się z tworzeniem operatorów dla oprogramowania RapodMiner oraz opisu metod klasyfikacji danych. Sekcja nr 3 w składzie Anna Romik, Michał Głowacki oraz Grzegorz Holewa przedstawiła podstawowe pojęcia stosowane przy klasyfikacji, takie jak zbiór danych niezbalansowanych, oversamling oraz undersampling. Wyjaśniono, że dane niezbalansowane to takie, w których liczebność jednej klasy jest znacznie większa od liczebności pozostałych klas, a oversampling i undersampling to odpowiednio: zwiększenie liczebności klasy zdominowanej oraz zmniejszenie liczebności klasy dominującej.

Sekcja nr 4 w składzie Wojciech Bożek, Dawid Mazur, Krystian Pypłacz i Paweł Szopa przygotowała prezentację na temat metody SMOTE (Synthetic Minority Oversampling Technique). Odniesiono się do wcześniejszych informacji dotyczących oversamplingu oraz danych niezbalansowanych oraz wyjaśniono, że metoda polega na syntetycznym generowaniu dodatkowych przykładów z klasy zdominowanej na podstawie odległości euklidesowej od k najbliższych sąsiadów rozpatrywanego obiektu oraz czynnika losowego.

Sekcja nr 2 w składzie Marek Kojder i Paweł Garbacik przedstawiła wszystkie możliwe sposoby tworzenia operatorów w RapidMinerze. Niestety żadna z nich nie okazała się trafna i nie generowała poprawnego rezultatu. Ustalono z prowadzącym, że należy rozważyć wykonanie projektu w innym środowisku (Weka, Matlab).

Sekcje nr 5 i 6 wyjaśniły, że zaproponowane metody wykorzystujące lasy losowe są teoretycznymi rozważaniami autorów pracy naukowej, na której jest oparty projekt i sposób implementacji zależy wyłącznie on grupy. Zaznaczono, że teoria została podparta przykładami, których dobór i sposób badania nie zostały w pracy wyjaśnione i należy brać pod uwagę, że autorska implementacja grupy może nie dawać tych samych rezultatów.

2.3. 22 stycznia 2014 r.

Na spotkaniu projektowym przedstawiona została prezentacja dotycząca metody BRF (Balanced Random Forest), która wyjaśniła genezę jej powstania oraz podstawowe założenia. Przedstawiono obszar zastosowań metody (dane niezrównoważone) oraz przykład jej implementacji w oprogramowaniu RapidMiner.

Pokazano również wyniki działania operatorów undersamplingu i oversamplingu na danych wejściowych. Zarówno sekcja nr 1 jak i sekcja nr 3 wskazały na problem dużej liczby danych, która nie każda maszyna jest w stanie sobie poradzić. Zaznaczono, że konieczna jest odpowiednia konfiguracja maszyny wirtualnej Java w celu przyporządkowania odpowiedniej liczby pamięci RAM tak, aby obliczenia wykonywały się w skończonym czasie.

2.4. 29 stycznia 2014 r.

Ostatnie spotkanie projektowe zostało przeznaczone na prezentację wyników działania implementacji algorytmu SMOTE na testowych danych wejściowych. Wyniki działania algorytmu były poprawne, algorytm wykonywał oversampling na klasie zdominowanej korzystając z metody generacji sztucznych przykładów. Zaprezentowane zostały również źródła w języku Java, które w dużej mierze były oparte o istniejące rozwiązania napisane wcześniej przez sekcję nr 3.

Przygotowana została również prezentacja wyjaśniająca podstawowe zagadnienia dot. metody WRF.
Wyjaśniono pojęcie Gini Impurity oraz przedstawiono problem doboru odpowiednich wag do klas.

3. Dane wejściowe

3.1. Przedstawienie danych

Dane wejściowe pochodzą z konkursu KDD Mining Cup 1999. KDD Cup jest światowym konkursem Eksploracji Danych i Odkrywania Wiedzy organizowanym przez ACM SIGKDD - Special Interest Group on Knowledge Discovery and Data Mining. Zadaniem konkursu było zbudowanie modelu prognostycznego (tj. klasyfikatora) zdolnego do rozróżniania połączeń „złych", zwanych atakami lub włamaniami, oraz „dobrych" (normalnych) połączeń. Zestaw obejmuje różnego rodzaju ataki symulowane na wojskowym środowisku sieciowym. Są to między innymi:

- Denial of Service Attack (DoS - odmowa usługi)
- Remote to Local (R2L)
- User to Root (U2R)
- Probe

Zbiór danych wejściowych w liczbach:

- 24 treningowe rodzaje ataków
- 14 rodzajów ataków w danych testowych
- 8 050 290 rekordów (1 173 MB)
- 4 940 000 rekordów treningowych (734 MB)
- 3 110 290 rekordów testowych (430 MB)
- 42 zmienne opisu pojedynczego rekordu
- 5 zmiennych dotyczących tego samego hosta
- 13 zmiennych dotyczących tej samej usługi

3.2. Opis rekordu danych

3.2.1. 42 zmienne opisujące rekord

duration	czas połączenia w sekundach
protocol_type	rodzaj protokołu
service	rodzaj usługi
flag	połączenie normalne lub błędne
src_bytes	liczba bajtów przesłanych od źródła do celu
dst_bytes	liczba bajtów przesłanych od celu do źródła
land	1 jeżeli połączenie jest z tego samego hosta lub portu, 0 w przeciwnym wypadku
wrong_fragment	liczba błędnych fragmentów
urgens	liczba pakietów z flagą „urgent”
hot	liczba wskaźników „hot”
num_failed_logins	liczba logowań zakończonych niepowodzeniem
logged_in	1 jeżeli użytkownik zalogowany, 0 w przeciwnym wypadku
num_compromised	liczba skompromitowanych warunków
root_shell	1 jeżeli uzyskano dostęp do powłoki Root, 0 w przeciwnym wypadku
su_attempted	1 jeżeli użyto komendy “su root”, 0 w przeciwnym wypadku
num_root	liczba dostępów do konta “Root”
num_file_creations	liczba operacji tworzenia pliku
num_shells	liczba monitów powłoki
num_access_files	liczba operacji kontroli dostępu na plikach
num_outbound_cmds	liczba komend wychodzących w sesji ftp
is_host_login	1 jeżeli zalogowany na konto “hot”, 0 w przeciwnym wypadku
is_guest_login	1 jeżeli zalogowano na koncie gościa, 0 w przeciwnym wypadku
count	liczba połączeń do tego samego hosta w przeciągu ostatnich 2 sekund
class	klasa ruchu

3.2.2. 5 zmiennych dotyczących tego samego hosta

serror_rate	procent połączeń z błędem “SYN”
rerror_rate	procent połączeń z błędem “REJ”
same_srv_rate	procent połączeń do tego samego serwisu
diff_srv_rate	procent połączeń do innego serwisu
srv_count	liczba połączeń do tej samej usługi w przeciągu ostatnich 2 sekund

3.2.3. 13 zmiennych dotyczących tego samego hosta

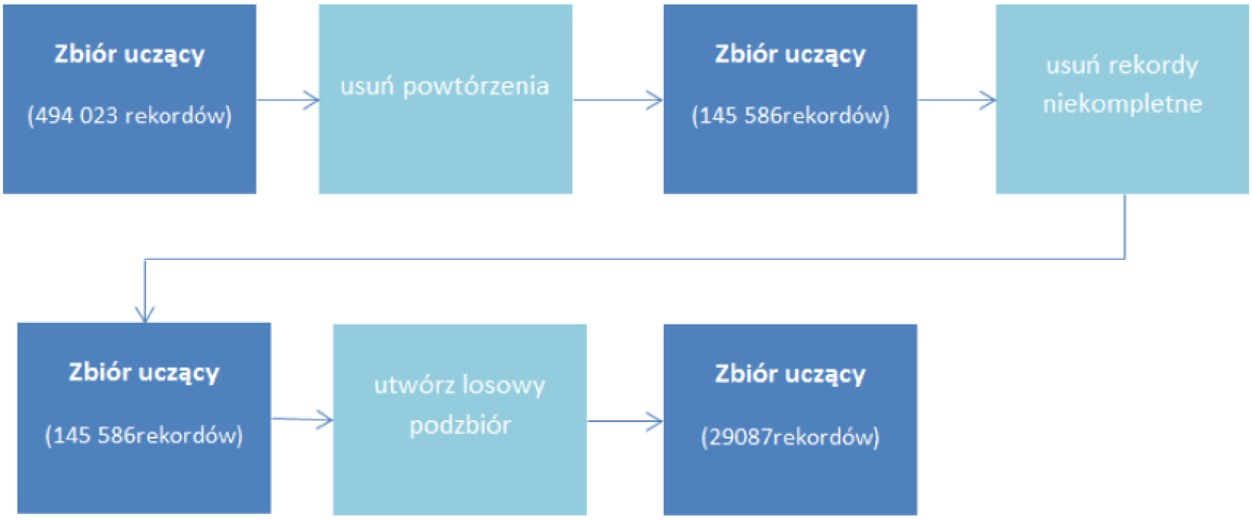
srv_serror_rate	procent połączeń z błędem “SYN”
srv_rerror_rate	procent połączeń z błędem “REJ”
srv_diff_host_rate	procent połączeń do innego hosta
dst_host_rate	procent połączeń do tego samego hosta docelowego
dst_host_count	liczba połączeń do tego samego hosta docelowego
dst_host_srv_count	liczba połączeń do tego samego hosta docelowego i tej samej usługi
dst_host_same_srv_rate	procent połączeń do tego samego hosta docelowego i do tej samej usługi
dst_host_diff_srv_rate	procent różnych usług połączonych do danego hosta
dst_host_same_src_port_rate	procent połączeń do hosta z tym samym portem źródłowym
dst_host_srv_diff_host_rate	procent połączeń do tej samej usługi od różnych hostów
dst_host_serror_rate	procent błędnych połączeń do hosta z błędem „SYN”
dst_host_srv_serror_rate	procent błędnych połączeń do hosta I tej samej usługi z błedem „SYN”
dst_host_rerror_rate	procent błędnych połączeń do hosta z błędem „REJ”

3.3. Przygotowanie danych

Dane dostarczone z projektem okazały się zbyt obszerne dla dalszej analizy. Aby wykonać zadanie klasyfikacji, należało przeprowadzić redukcję oraz eksplorację danych, w czym pomogło oprogramowanie STATISTICA 10. Za jego pomocą wykonano:

- odczyt danych
- przypisanie danych właściwych etykiet
- usunięcie z arkusza powtarzających się rekordów
- minimalizację ryzyka wystąpienia problemu, w którym model nauczy się dobrze rozpoznawać jedynie obiekt najliczniej występujący w zbiorze uczącym
- usunięcie rekordów niekompletnych

W ramach dodatkowej redukcji danych przeprowadzono losowanie podzbiorów przypadków, odpowiednio 20% rekordów ze zbioru uczącego oraz 5% rekordów ze zbioru testowego.



Rys. 1. Proces eksploracji danych zbioru uczącego.

Po przeprowadzeniu powyższych operacji zbiór uczący zawiera 29087, a testowy 7655 rekordów.

3.4. Opis danych

3.4.1. Przygotowania

Do wczytania danych wykorzystano środowisko RapidMiner działające w otoczeniu maszyny wirtualnej Java w wersji 64-bitowej. Aby zapewnić skończoność wykonywania operacji skonfigurowano maszynę tak, aby jej zajętość pamięciowa wynosiła 3GB. W tym celu posłużono się opcją:

```
-Xmx3072m
```

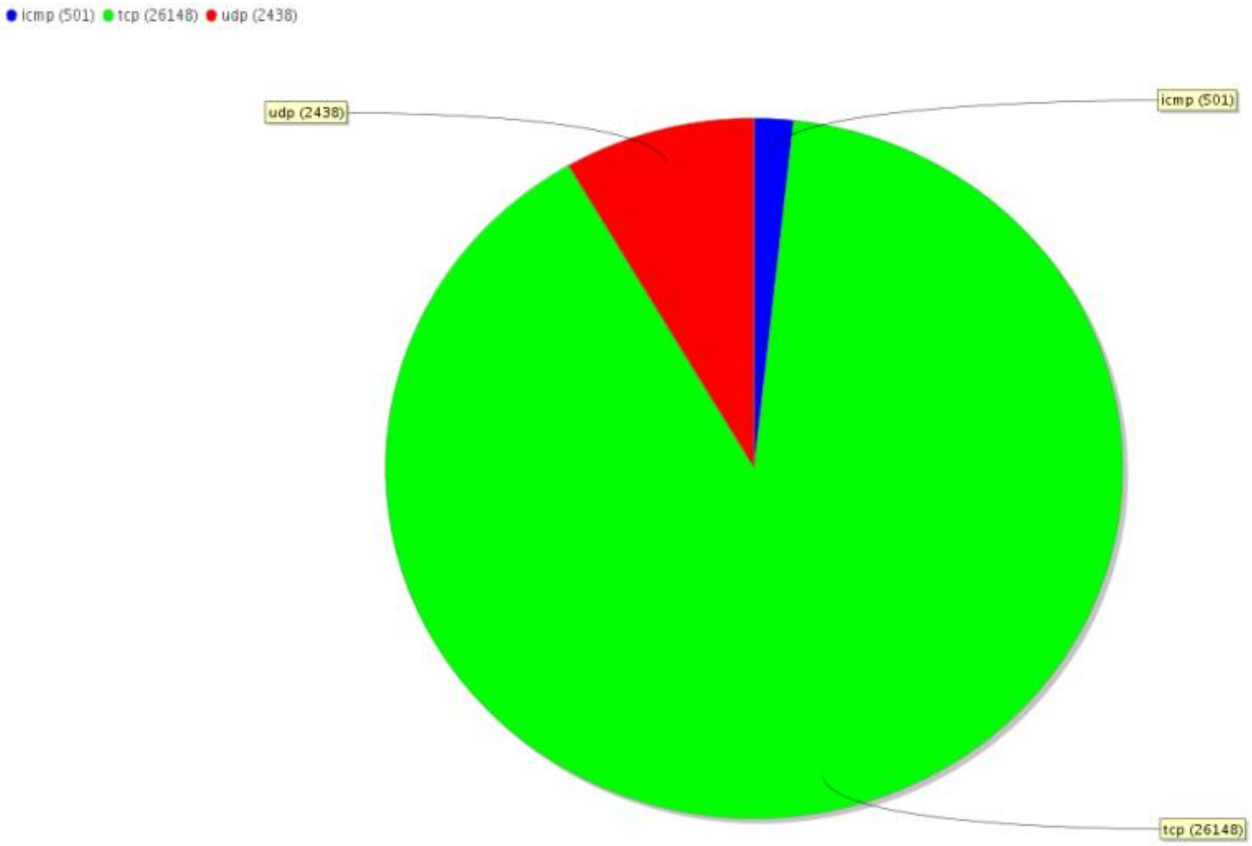
Również w oprogramowaniu RapidMiner (plik RapidMinerGUI.bat) należało wprowadzić odpowiednią opcję:

```
if "%MAX_JAVA_MEMORY%"=="" set MAX_JAVA_MEMORY=3072
```

Dane zostały załadowane za pomocą bloku Read Excel oraz zapewnionego kreatora. Proces ten po powyższej konfiguracji zajął 32 sekundy.

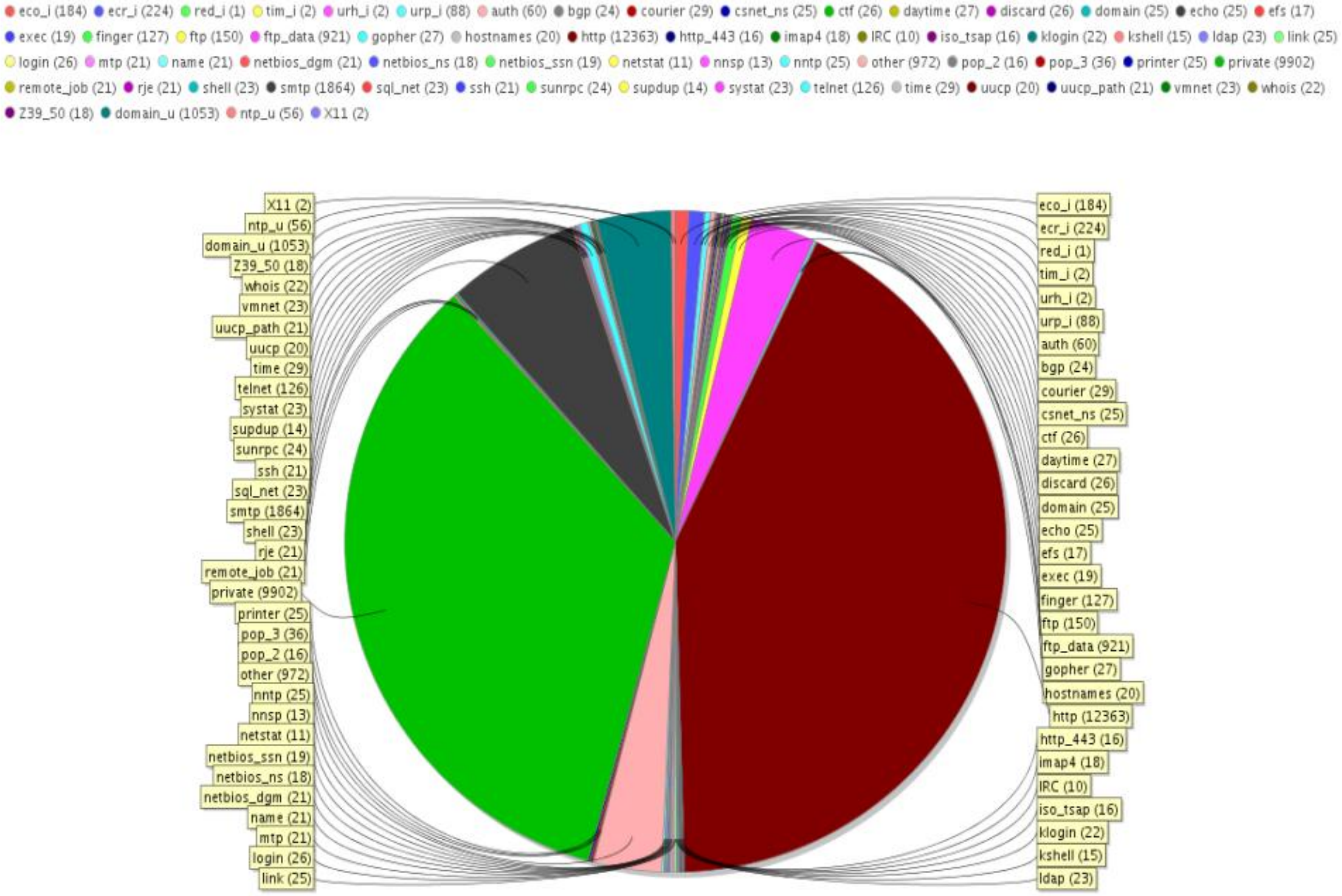
3.4.2. Prezentacja danych

W zbiorze uczącym o liczności 29087 rekordów wyróżniono trzy protokoły sieciowe:



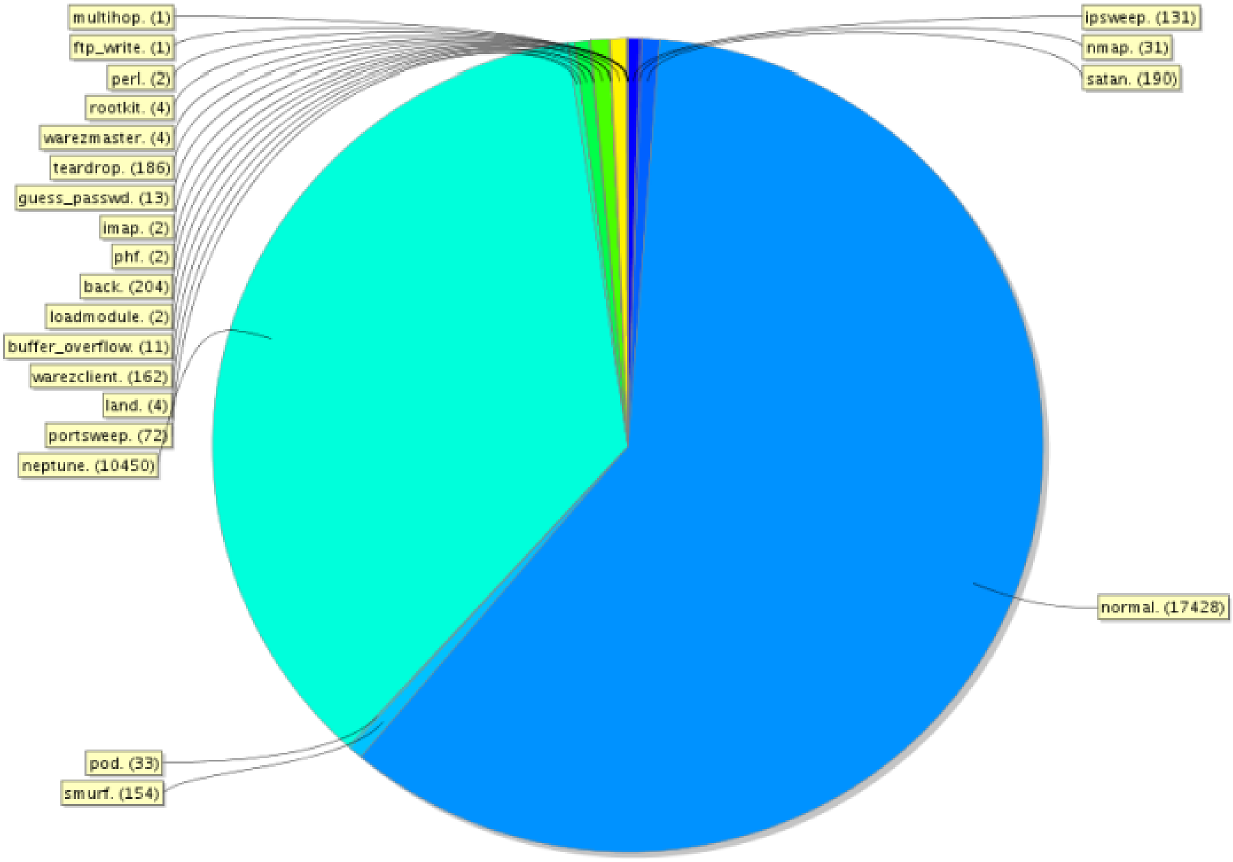
Rys. 2. Protokoły sieciowe w zbiorze uczącym.

Odnaleziono również 64 rodzaje usług, z jakich korzystali użytkownicy sieci:



Rys. 3. Rodzaje usług sieciowych.

Dokonano również podziału pakietów na zwyczajne oraz ataki wraz z wyszczególnieniem rodzajów ataków:



Rys. 4. Podział pakietów na zwykłe i poszczególne ataki.

4. Operatory klasyfikacyjne w środowisku RapidMiner

Oprogramowanie RapidMiner nie dostarczało odpowiednich operatorów, które pozwoliłyby na wykonanie projektu. Stąd niezbędne było wykonanie dodatkowych operatorów poprzez ingerencję w kod źródłowy. Aby to zrobić, należy do katalogu roboczego skopiować dwa projekty Eclipse:

- *RapidMiner_Extension_Template_Unuk*
- *Rapidminer_Unuk*

Powyższe projekty są wymagane do skompilowania rozszerzenia dla oprogramowania RapidMiner. Nowe operatory definiowane są w projekcie *RapidMiner_Extension_Template_Unuk* i można je wykonać w następujący sposób:

1) W katalogu *src/com/rapidminer/* definiujemy nową klasę dla operatora.
2) Nowa klasa powinna rozszerzać klasę `Operator` i przeciążać następujące metody:
    a. *doWork* - funkcje realizowane w trakcie przetwarzania danych
    b. *getParameterTypes* - parametry operatora

Wejścia i wyjścia operatora definiowane są jako pola klasy:

```
private final InputPort exampleSetInput = getInputPorts().createPort("nazwa wejscia");
private final OutputPort exampleSetOutput = getOutputPorts().createPort("nazwa wyjscia");
```

Konstruktor ma postać:

```
public NowyOperator(OperatorDescription description) { super(description); }
```

W ciele konstruktora typowe jest nadawanie warunków początkowych dla wejść / wyjść, ale tylko w przypadku, jeżeli oczekujemy ostrzeżenia gdy wejścia lub wyjścia nie są podłączone. Przykładowe odbieranie danych z portów / wysyłanie na port:

```
ExampleSet inputExampleSet = exampleSetInput.getData(ExampleSet.class);
exampleSetOutput.deliver(result);
```

Możemy posiłkować się istniejącym operatorem.

3) Po zdefiniowaniu klasy należy przejść do katalogu `resources/com/rapidminer/resources/` i w pliku `BalancingOperators.xml` dodać informacje o swoim operatorze:

```
<group key="">
  <group key="data_transformation">
    <group key="data_balancing">
      <operator>
        <key>unikalna_nazwa</key>
        <class>com.rapidminer.NowyOperator</class>
      </operator>
    </group>
  </group>
</group>
```

Znaczniki `<group key="nazwa">` informują w jakiej kategorii zostanie umieszczony nowy operator (panel `Operators` w rapidminerze).

4) Nazwa nadana w tagach `<key></key>` może zostać użyta do tłumaczenia nazwy operatora w pliku `OperatorsDocTemplate.xml` w katalogu `resources/com/rapidminer/resources/i18n/`.

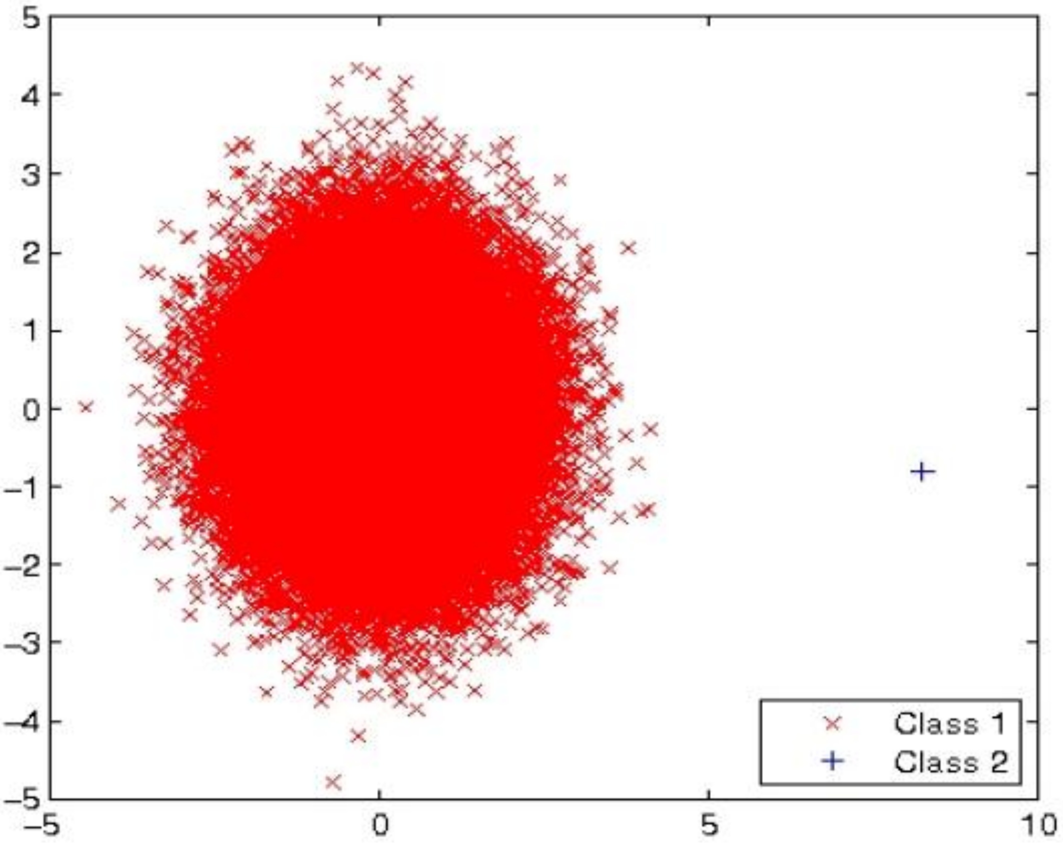
```
<operator>
  <key>unikalny_nazwa</key>
    <name>Nazwa widoczna w rapidmierze</name>
    <synopsis>Krótki opis</synopsis>
    <help>Długi opis</help>
</operator>
```

5) Budujemy projekt narzędziem Ant poprzez plik `build.xml` (PPM -> Run as -> Ant build) w głównym katalogu. Zbudowane rozszerzenie znajdzie się w projekcie `Rapidminer_Unuk` w folderze `lib/plugins`. Od teraz wystarczy uruchomić `lib/rapidminer.jar` i nowe operatory powinny działać. Rozszerzenie można też skopiować do analogicznego folderu w RapidMinerze 6.

5. Oversampling i undersampling

5.1. Problem danych niezrównoważonych

Problem danych niezbalansowanych pojawia się wtedy, gdy liczność jednej klasy (klasy dominującej, przyjmuje się że jest to klasa negatywna) jest istotnie wyższa niż liczność drugiej klasy (klasy zdominowanej, pozytywnej). Istotą problemu niezbalansowania jest fakt, że zastosowanie klasycznych mechanizmów uczenia na niezrównoważonym zbiorze danych może prowadzić do faworyzowania przez wyuczony klasyfikator klasy dominującej kosztem klasy zdominowanej.



Rys. 5. Prezentacja problemu danych niezbalansowanych.

Do rozwiązania problemu niezbalansowania stosuje metody przetwarzania danych, takie jak oversampling i undersampling.

5.2. Undersampling

Undersampling to technika analizy danych wykorzystywana do regulacji podziału klas zbioru danych. Odrzuca ona część próbek z klasy większościowej. Istnieją cztery główne metody undersamplingu:

- **Random Undersampling**
  Usuwanie losowych próbek.
- **Tomek Links**
  Polega na usuwaniu przykładów granicznych i szumu z klasy większościowej. Dla każdego przykładu z klasy większościowej i klasy mniejszościowej sprawdzany jest następujący warunek: jeżeli nie istnieje przykład El z pary (Ei, Ej) nazywanej „Tomek Link”, dla którego spełniona jest zależność:

  `d(Ei, El) < d(Ei, Ej) lub d(Ej, El) < d(Ei, Ej)`

  to przykłady będące w Tomek Link są usuwane ze zbioru.
- **Edited Nearest Neighborhood**
  Dla każdego przykładu z klasy większościowej znajdowanych jest trzech najbliższych sąsiadów. Jeżeli klasy dwóch sąsiadów są różne od klasy przykładu to przykład jest usuwany.
- **Nearest Cleaning Rule**
  Dla każdego przykładu ze zbioru znajdowanych jest trzech najbliższych sąsiadów. Jeżeli przykład należy do klasy większościowej i sąsiedzi klasyfikują go inaczej to jest usuwany. Jeżeli przykład należy do klasy mniejszościowej i sąsiedzi klasyfikują go inaczej to usuwani są sąsiedzi z klasy większościowej

Problemy, jakie może rodzić metoda undersamplingu, to usunięcie zbyt wielu przykładów klasy większościowej oraz doprowadzenie do gwałtownego pogorszenia rozpoznawania klas większościowych.

5.3. Oversampling

Oversampling to technika analizy danych wykorzystywana do regulacji podziału klas zbioru danych. Wybiera więcej próbek z klasy mniejszościowej (duplikuje dane). Replikacja zwiększa wagę próbek populacji mniejszościowej ale nie zwiększa ilości informacji. Główną metodą oversampling jest metoda SMOTE, której zostanie poświęcony osobny rozdział.

5.4. Praca z danymi

5.4.1. Pakiety zwykłe i ataki

Dane wejściowe reprezentowane przez rysunek nr 4 (podział na ataki i pakiety zwykłe) zostały poddane klasyfikacji za pomocą drzew decyzyjnych i lasów losowych (dobierając odpowiednio oversampling oraz undersampling) ze względu na występowanie i niewystępowanie ataku. Wyniki przedstawiają się następująco:

5.4.1.1. Drzewa decyzyjne (dane niezrównoważone)

		TRUE		**Class precision**
		Normal	**Attack**	
PREDICTABLE	**Normal**	1481	125	92.22%
	Attack	10	5581	99.82%
Class recall		99.33%	97.81%	

Acc	0,981242184
Acc+	0,993293092
Acc-	0,978093235
G-mean	0,985663864
Weighted Accuracy	0,985693164
Precision	0,922166874
Recall	0,993293092
F-measure	0,956409428

5.4.1.2. Drzewa decyzyjne (oversampling)

		TRUE		**Class precision**
		Normal	**Attack**	
PREDICTABLE	**Normal**	1480	125	92.21%
	Attack	11	5581	99.80%
Class recall		99.26%	97.81%	

Acc	0,981103237
Acc+	0,992622401
Acc-	0,978093235
G-mean	0,985331039
Weighted Accuracy	0,985357818
Precision	0,92211838
Recall	0,992622401
F-measure	0,956072351

5.4.1.3. Drzewa decyzyjne (undersampling)

		TRUE		**Class precision**
		Normal	**Attack**	
PREDICTABLE	**Normal**	1414	109	92.84%
	Attack	77	5597	98.64%
Class recall		94.84%	98.09%	

Acc	0,974155898
Acc+	0,948356808
Acc-	0,980897301

G-mean	0,96448983
Weighted Accuracy	0,964627054
Precision	0,928430729
Recall	0,948356808
F-measure	0,938287989

5.4.1.4. Lasy losowe (oversampling)

		TRUE		**Class precision**
		Normal	**Attack**	
PREDICTABLE	**Normal**	1486	4586	24.47%
	Attack	5	1120	99.56%
Class recall		99.66%	19.63%	

Acc	0,362095317
Acc+	0,996646546
Acc-	0,196284613
G-mean	0,442296712
Weighted Accuracy	0,596465579
Precision	0,244729908
Recall	0,996646546
F-measure	0,392965754

5.4.1.5. Lasy losowe (undersampling)

		TRUE		**Class precision**
		Normal	**Attack**	
PREDICTABLE	**Normal**	197	0	100.00%
	Attack	1294	5706	81.51%
Class recall		13.21%	100.00%	

Acc	0,820202862
Acc+	0,13212609
Acc-	1
G-mean	0,363491527
Weighted Accuracy	0,566063045
Precision	1
Recall	0,13212609
F-measure	0,233412322

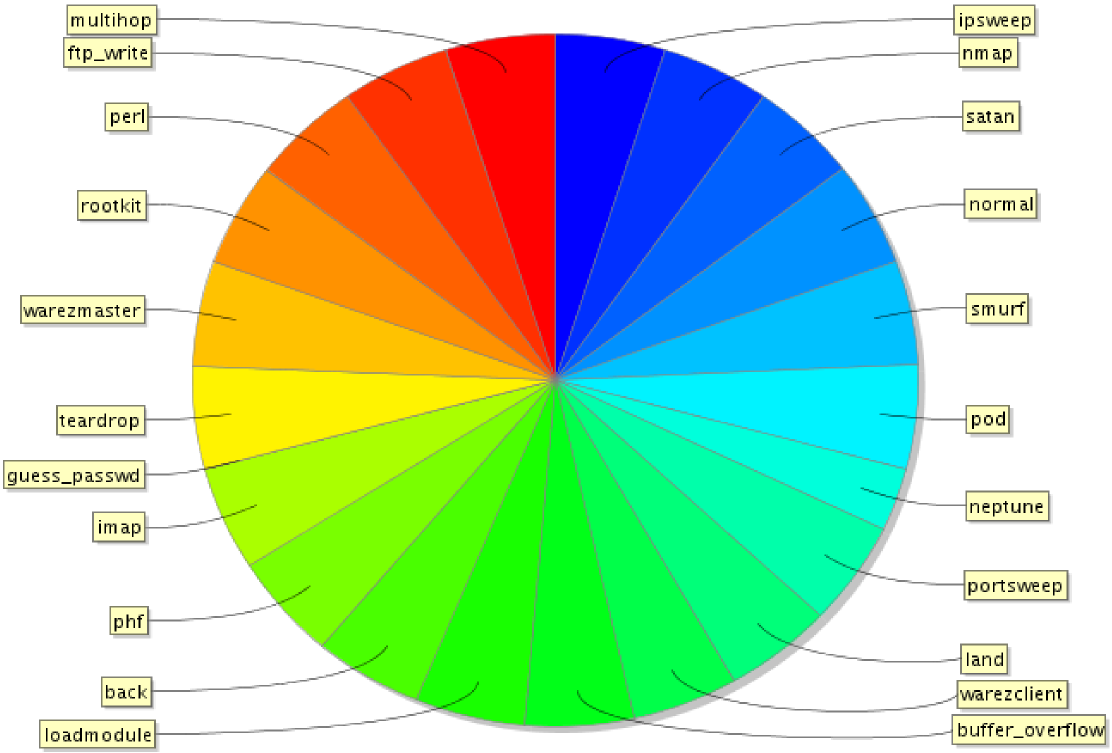
5.4.2. Rodzaje ataków

Dane wejściowe reprezentowane przez rysunek nr 4 (podział na ataki) zostały poddane klasyfikacji za pomocą drzew decyzyjnych i lasów losowych (dobierając odpowiednio oversampling oraz undersampling) ze względu na rodzaj ataku. Wyniki przedstawiają się następująco:

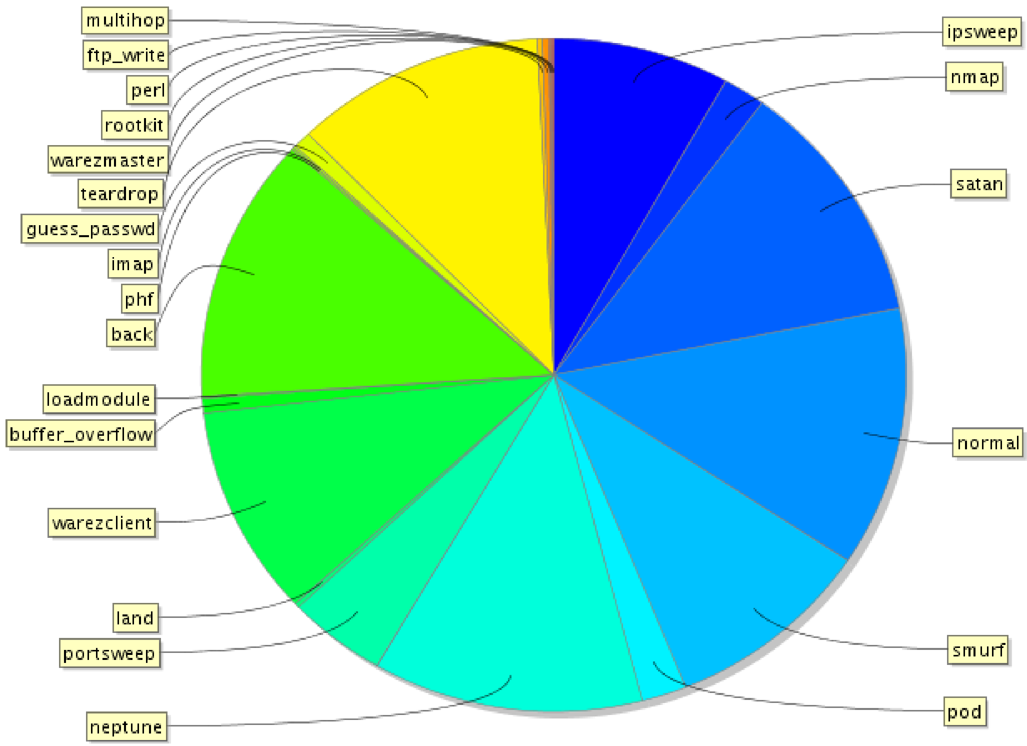
5.4.2.1. Oversampling i undersampling

Typ pakietu	Liczność	Oversampling mnożnik	Oversampling liczność	Undersampling dzielnik	Undersampling liczność
normal	17428	1	17428	87,14	200
ipsweep	131	133,0382	17428		131
nmap	31	562,1935	17428		31
satan	190	91,72632	17428		190

smurf	154	113,1688	17428		154
pod	33	528,1212	17428		33
neptune	10450	1,667751	17428	52,25	200
portsweep	72	242,0556	17428		72
land	4	4357	17428		4
warezclient	162	107,5802	17428		162
buffer_overflow	11	1584,364	17428		11
loadmodule	2	8714	17428		2
back	204	85,43137	17428		204
phf	2	8714	17428		2
imap	2	8714	17428		2
guess_passwd	13	1340,615	17428		13
teardrop	186	93,69892	17428		186
warezmaster	4	4357	17428		4
rootkit	4	4357	17428		4
perl	2	8714	17428		2
ftp_write	1	17428	17428		1
multihop	1	17428	17428		1



Rys. 6. Wykres liczności typów pakietów po operacji oversamplingu.



Rys. 6. Wykres liczności typów pakietów po operacji undersamplingu.

5.4.2.2. Drzewa decyzyjne (dane niezrównoważone)

		TRUE																						
		normal	smurf	ipsweep	portsweep	warezmaster	guess_passwd	buffer_overflow	satan	back	neptune	httptunnel.	pod	nmap	land	warezclient	loadmodule	phf	imap	teardrop	rootkit	perl	ftp_write	multihop	class precision
PREDICTABLE	normal	1480	0	0	0	30	87	1	0	0	2	4	0	0	0	0	0	0	0	0	0	0	0	0	92.27%
	smurf	0	4039	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
	ipsweep	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
	portsweep	4	0	0	9	0	0	0	10	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	37.50%
	warezmaster	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%

loadmodule	warezclient	land	nmap	pod	httptunnel.	neptune	back	satan	buffer_overflow	guess_passwd
0	0	0	0	0	0	1	0	4	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	8	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	20
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	30	0	0
0	0	0	0	0	0	0	27	0	0	0
0	0	0	0	0	0	1425	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	2	0	0	0	0	0	0
0	0	0	2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0.00%	0.00%	0.00%	100.00%	66.67%	0.00%	99.93%	100.00%	88.24%	0.00%	100.00%

	phf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	imap	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	teardrop	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	rootkit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	perl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	ftp_write	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	multihop	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
class recall		99.26%	100.00%	100.00%	100.00%	0.00%	18.69%	0.00%	75.00%	100.00%	99.79%	0.00%	100.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	

5.4.2.3. Drzewa decyzyjne (oversampling)

		TRUE																							
		normal	smurf	ipsweep	portsweep	warezmaster	guess_passwd	buffer_overflow	satan	back	neptune	httptunnel.	pod	nmap	land	warezclient	loadmodule	phf	imap	teardrop	rootkit	perl	ftp_write	multihop	class precision
PREDICTABLE	normal	1469	1	0	0	25	103	1	0	4	2	4	0	0	0	0	0	0	0	0	0	0	0	0	91.30%
	smurf	0	3858	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%
	ipsweep	0	180	8	0	0	4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	4.15%

land	nmap	pod	httptunnel.	neptune	back	satan	buffer_overflow	guess_passwd	warezmaster	portsweep
0	0	0	0	0	1	4	11	0	0	3
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	8
0	13	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	11	0	28	0	0	0	1
0	0	0	0	0	23	0	0	0	0	0
0	0	0	0	1403	0	13	0	0	0	9
0	0	0	0	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0	0	0
0	2	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0.00%	13.33%	100.00%	0.00%	99.22%	95.83%	59.57%	0.00%	0.00%	0.00%	38.10%

	warezclient	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	loadmodule	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	phf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	imap	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	teardrop	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	rootkit	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	perl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	ftp_write	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	multihop	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
class recall		98.52%	95.52%	100.00%	88.89%	0.00%	0.00%	0.00%	70.00%	85.19%	98.25%	0.00%	100.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	

5.4.2.3. Drzewa decyzyjne (undersampling)

	TRUE																							
	normal	smurf	ipsweep	portsweep	warezmaster	guess_passwd	buffer_overflow	satan	back	neptune	httptunnel.	pod	nmap	land	warezclient	loadmodule	phf	imap	teardrop	rootkit	perl	ftp_write	multihop	class precision

PREDICTABLE										
httptunnel.	neptune	back	satan	buffer_overflow	guess_passwd	warezmaster	portsweep	ipsweep	smurf	normal
0	0	5	7	0	0	0	6	0	33	1382
0	0	0	0	0	0	0	0	0	4039	0
0	0	0	0	0	0	0	0	8	0	0
0	0	0	1	0	0	0	8	0	0	0
0	0	14	2	0	0	0	0	0	0	0
0	0	0	33	0	16	0	0	2	0	50
0	0	0	0	0	0	0	0	0	0	0
0	0	0	30	0	0	0	10	0	0	0
0	0	27	0	0	0	0	0	0	0	0
0	1403	0	1	0	0	0	3	0	0	0
0	0	0	0	0	0	0	0	0	0	4
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0.00%	100.00%	58.70%	40.54%	0.00%	100.00%	0.00%	29.63%	80.00%	99.19%	96.24%

multihop	ftp_write	perl	rootkit	teardrop	imap	phf	loadmodule	warezclient	land	nmap	pod
0	0	0	0	2	6	0	0	38	0	7	5
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	10	0	13	0
0	0	0	0	0	0	0	0	6	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	21	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	2
0	0	0	0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	4.65%	28.57%

class recall	92.69%	100.00%	100.00%	88.89%	0.00%	14.95%	0.00%	75.00%	100.00%	98.25%	0.00%	100.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	

5.4.2.3. Lasy losowe (oversampling)

		TRUE																						
		normal	smurf	ipsweep	portsweep	warezmaster	guess_passwd	buffer_overflow	satan	back	neptune	httptunnel.	pod	nmap	land	warezclient	loadmodule	phf	imap	teardrop	rootkit	perl	ftp_write	multihop	class precision
PREDICTABLE	normal	3	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20.00%
	smurf	0	3	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50.00%
	ipsweep	431	1597	4	1	6	40	1	10	3	683	4	0	0	0	0	0	0	0	0	0	0	0	0	0.14%
	portsweep	6	0	0	8	0	0	0	13	0	242	0	0	0	0	0	0	0	0	0	0	0	0	0	2.97%
	warezmaster	84	0	0	0	3	26	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	2.61%
	guess_passwd	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	buffer_overflow	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	satan	4	667	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.15%

rootkit	teardrop	imap	phf	loadmodule	warezclient	land	nmap	pod	httptunnel.	neptune	back
0	26	28	0	42	1	202	0	622	0	0	38
0	157	0	0	0	0	839	0	776	0	0	0
0	0	0	0	0	0	3	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	2	1	7	0	6	0	0	0
0	0	0	0	3	0	25	0	12	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	3	0	0	0
0	0	2	0	0	0	0	0	0	0	0	18
0	221	0	0	0	0	153	0	89	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	32.14%

	perl	4	0	0	0	0	1	0	7	0	40	0	0	1	0	0	0	0	0	0	0	0	0	0	0.00%
	ftp_write	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	multihop	0	0	0	0	1	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
class recall		0.20%	0.07%	50.00%	88.89%	7.69%	0.00%	0.00%	2.50%	66.67%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	

5.4.2.3. Lasy losowe (undersampling)

		TRUE																						
		normal	smurf	ipsweep	portsweep	warezmaster	guess_passwd	buffer_overflow	satan	back	neptune	httptunnel.	pod	nmap	land	warezclient	loadmodule	phf	imap	teardrop	rootkit	perl	ftp_write	multihop	class precision
PREDICTABLE	normal	607	1	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	99.35%
	smurf	16	4038	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	99.61%
	ipsweep	3	0	8	0	18	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	26.67%
	portsweep	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	66.67%
	warezmaster	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.00%
	guess_passwd	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100.00%

imap	phf	loadmodule	warezclient	land	nmap	pod	httptunnel.	neptune	back	satan	buffer_overflow
0	0	0	0	0	0	0	0	3	851	4	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	6	0
0	0	0	3	0	0	0	0	0	16	0	0
0	0	0	0	0	0	0	0	0	85	0	0
0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	8	0	32	0
0	0	0	0	0	0	0	0	0	27	0	0
0	0	0	0	0	0	0	0	1427	0	1	0
0	0	0	0	0	0	0	0	0	0	4	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	2	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.03%	2.76%	68.09%	0.00%

class recall	multihop	ftp_write	perl	rootkit	teardrop
40.71%	0	0	0	0	6
99.98%	0	0	0	0	0
100.00%	0	0	0	0	0
22.22%	0	0	0	0	0
0.00%	0	0	0	0	2
1.87%	0	0	0	0	17
0.00%	0	0	0	0	0
80.00%	0	0	0	0	0
100.00%	0	0	0	0	0
99.93%	0	0	0	0	0
0.00%	0	0	0	0	0
0.00%	0	0	0	0	0
0.00%	0	0	0	0	0
0.00%	0	0	0	0	0
0.00%	0	0	0	0	0
0.00%	0	0	0	0	0
0.00%	0	0	0	0	0
0.00%	0	0	0	0	0
0.00%	0	0	0	0	0
0.00%	0	0	0	0	0
0.00%	0	0	0	0	0
0.00%	0	0	0	0	0
0.00%	0	0	0	0	0
	0.00%	0.00%	0.00%	0.00%	0.00%

6. Metoda SMOTE

7. Metoda BRF

8. Metoda WRF

9. Podsumowanie