Pengaruh Sampling Method dan Feature Extraction untuk Meningkatkan Detection Rate pada Minority Class pada Intrusion Detection System yang Disusun dari Support Vector Machine, Decision Tree, dan Naïve Bayes

Janthake Decuellar, Henry N. Palit, Justinus Andjarwirawan
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-Mail: janthakede@gmail.com, hnpalit@petra.ac.id, justin@petra.ac.id

ABSTRAK

Intrusion detection system (IDS) sudah mulai mengandalkan *machine learning* untuk melakukan *misuse detection* atau *anomaly detection*. Sebagai *misuse detection, machine learning* harus dapat mendeteksi berbagai jenis intrusi termasuk yang jarang terjadi. Tetapi *machine learning*, memiliki kelemahan khususnya jika dihadapkan dengan *dataset* yang *imbalanced*. Berbagai cara dilakukan untuk membuat *machine learning* dapat melakukan klasifikasi dengan tepat meskipun data yang diberikan bersifat *imbalanced*.

Salah satunya pada penelitian ini mencoba mengimplementasikan *Principal Component Analysis* sebagai *feature extraction*, *Tomek Links* sebagai *under-sampling* dan ADASYN sebagai *over-sampling* pada beberapa *datasets. Dataset* yang digunakan pada penelitian ini ada dua jenis, yaitu KDD-99 dan UNSW-NB15.

Hasil yang diperoleh dari penelitian pada *dataset* KDD'99, model *Support Vector Machine* dapat mengidentifikasi intrusi lebih banyak dan model *Decision Tree* memperoleh peningkatan pada *True Positive Rate* untuk *minority class* sebesar antara 0.03% hingga 4.762%. Hasil yang diperoleh dari penelitian pada *dataset* UNSW-NB15, pada model *Support Vector Machine* dan *Naïve Bayes* terjadi peningkatan pada akurasi antara 0.045% hingga 1.513%.

Kata Kunci: *intrusion detection system, machine learning, over-sampling, under-sampling, feature extraction*

ABSTRACT

Intrusion detection system (IDS) has started to rely on machine learning to perform misuse detection or anomaly detection. As misuse detection, machine learning must be able to detect various types of intrusions, including those that are rare. However, machine learning has weaknesses, especially when faced with imbalanced datasets. Various methods are used to make machine learning able to perform the classification correctly even though the data provided is imbalanced.

One of them in this study tries to implement Principal Component Analysis as feature extraction, Tomek Links as under-sampling and ADASYN as over-sampling on datasets. There are two types of datasets used in this research, namely KDD-99 and UNSW-NB15.

The results obtained from research on the KDD'99 dataset are, Support Vector Machine can identify more intrusions than before and True Positive Rate of Decision Tree model for minority classes is increased between 0.03% to 4.762%. The results obtained from research on UNSW-NB15 dataset, accuracies for Support Vector Machine and Naïve Bayes models are increased between 0.045% to 1.513%.

Keywords: *intrusion detection system, machine learning, over-sampling, under-sampling, feature extraction*

1. PENDAHULUAN

Intrusion detection system (IDS) adalah sistem keamanan yang memonitor sistem komputer dan jaringan, serta menganalisa jaringan yang berkemungkinan merupakan intrusi baik dari luar atau dari dalam jaringan dan mengirimkan pemberitahuan kepada *administrator* jika terjadi intrusi.[18] IDS dibutuhkan oleh organisasi untuk melindungi sistem dari ancaman yang muncul bersamaan dengan meluasnya jaringan dan kebergantungan terhadap sistem informasi.[18] Berdasarkan *detection model*, IDS dibagi menjadi tiga yaitu *anomaly detection model, misuse detection model,* dan *hybrid anomaly/misuse detection model* yang menggabungkan kedua metode yang sudah disebutkan sebelumnya.[18]

Pada penelitian ini, akan menggunakan model *Anomaly-based Detection* yang dibuat oleh Kathleen Goeschel tahun 2016.[3] Hal ini karena pada model yang dibuat oleh Kathleen memiliki akurasi yang tinggi dan *false positive* yang rendah dibanding dengan *hybrid machine learning model* lainnya.[14] Selain itu, model ini tidak perlu melibatkan *machine learning model* yang cukup rumit seperti *Deep Learning* dan *Random Forest* untuk mencapai akurasi di atas 99%. *Detection* ini disusun dari *Support Vector Machine*, *Decision Tree*, dan *Naïve Bayes*. Namun masalah pada model ini adalah, tidak dicantumkan mengenai *data preprocessing* yang digunakan. Masalahnya, pada *dataset* yang digunakan oleh Kathleen Goeschel yaitu KDD'99 memiliki beberapa kelemahan, seperti memiliki banyak data yang redundan dan tidak semua jenis intrusi yang dijadikan sebagai *class* merupakan benar-benar serangan.[16] Pada KDD'99 terdapat proporsi *class* yang tidak seimbang, di mana *class Denial of Service* memiliki proporsi 79.27% dari *dataset*, sedangkan untuk *class Root-to-Local* (R2L) memiliki proporsi 00.02% dari *dataset*, bahkan untuk *class User-to-Root* (U2R) hanya memiliki proporsi 00.001%.[2]

Ada beberapa penelitian yang berusaha untuk memperbaiki model *machine learning* agar bisa memprediksi *minority class*. Ada yang menempuh jalan mengubah algoritma agar bisa menangani *imbalanced dataset* seperti XGBoost [15] dan juga ada yang

menggunakan *machine learning* untuk mencari parameter cocok untuk *over-sampling method* [13], selain itu juga ada yang mengusulkan model kombinasi dari beberapa *machine learning* dan *sampling method* [9]. Penelitian ini akan menggunakan dua jenis *sampling method* yaitu *under-sampling* dan *over-sampling* karena pada *dataset* akan dilakukan *under-sampling* untuk *majority class* dan *over-sampling* untuk *minority class*.

Metode *sampling* yang digunakan untuk *undersampling* adalah *Tomek links* karena berdasarkan analisa Kok-Chin Khor, Choo-Yee Ting, dan Somnuk Phon-Amnuaisuk pada dataset KDD'99 bahwa terjadi *overlapped class* yang menyebabkan *boundaries* untuk U2R dan R2L khususnya, sehingga sulit untuk dibedakan oleh *machine learning*.[8] Lalu untuk metode *oversampling* yang digunakan adalah *Adaptive Synthetic Sampling* (ADASYN) karena menurut pernyataan dari Haibo He, Yang Bai, Edwardo A. Garcia, dan Shutao Li, ADASYN bekerja dengan melakukan *generate synthetic samples* dari *minority class* yang sulit untuk dipelajari.[4]

Penelitian ini juga akan menggunakan *feature extraction* yaitu *Principal Component Analysis* (PCA) untuk mengambil rangkuman dari *features* yang asli menjadi *features* baru.[6] Hal ini karena dengan PCA dapat memberikan *Detection Rate* yang mendekati 90% untuk *minority class*.[5] Selain itu, pada *dataset* yang lebih baru seperti CICIDS2017, PCA juga membantu mengurangi dimensi menjadi 10 *features* dari sebelumnya 81 features dengan akurasi yang tinggi yaitu 99.6%. [1]

Penelitian ini akan menggunakan sebuah dataset lain selain KDD'99 seperti yang digunakan oleh Kathleen Goeschel. Hal ini karena selain untuk menguji *sampling method* dan *feature extraction*, penelitian ini juga akan melakukan komparasi hasil model dari Kathleen Goeschel ini terhadap dataset lain yang berbeda.[3] Penelitian ini menggunakan dataset KDD'99 dan UNSW-NB15 karena untuk UNSW-NB15 cukup sering digunakan sebagai dataset pada penelitian sejenis.

2. DASAR TEORI

2.1 Model *Detection* oleh Kathleen Goeschel yang disusun dari *Support Vector Machine, Decision Tree* (J48), dan *Naive Bayes*

Model *detection* ini pertama kali diusulkan oleh Kathleen Goeschel pada tahun 2016. Tujuan utama dari model ini adalah untuk mengurangi *false positive* yang terjadi agar beban komputasi yang diperlukan untuk menganalisa data, mengirimkan sinyal kepada *analyst* tidak terbuang untuk data-data yang tidak relevan.[3] Model ini disusun terdiri dari tiga tahap. Tahap pertama menggunakan *Support Vector Machine* untuk melakukan prediksi apakah data yang masuk termasuk *class Attack* atau *Normal*. Jika merupakan *Normal*, maka *traffic* akan dibiarkan lewat, tidak perlu melalui tahap-tahap selanjutnya. Namun, jika data yang masuk merupakan *Attack*, maka data tersebut masuk ke tahap dua. Pada tahap dua, data-data yang merupakan *Attack* akan diklasifikasikan termasuk kategori apa dengan menggunakan *Decision Tree* yang dibuat menggunakan J48 *Algorithm*. Jika pada tahap kedua, data yang masuk tidak dapat diklasifikasikan maka akan menggunakan *Naïve Bayes* untuk menampilkan semua nilai kemungkinan dari suatu data merupakan suatu kategori serangan. Jika ditemukan probabilitas cukup tinggi maka, kemungkinan data tersebut merupakan jenis serangan baru yang merupakan turunan atau satu golongan dengan suatu jenis serangan.

2.2 *Tomek Links*

Tomek Link merupakan metode *under-sampling* dibuat pertama kali oleh Ivan Tomek tahun 1976.[17] Pada awalnya, metode ini digunakan untuk membuat *Condensed Nearest Neighbor* (CNN) yang lebih bagus dalam klasifikasi dengan komputasi yang lebih ringan karena berfokus pada hanya mengambil sebagian dari *dataset.* Algoritma yang digunakan oleh Tomek adalah dengan mengambil sepasang *records* yang berasal dari *class* yang berbeda, lalu menghitung titik tengah dari kedua *records* tersebut dengan menggunakan rumus jarak. Setelah itu, mencari apakah ada *record* lain yang memiliki jarak lebih dekat terhadap nilai tengah dibanding sepasang *records* yang telah dipilih diawal. Jika tidak ada *record* lain, maka sepasang *records* (disebut *Tomek Link*) tersebut digunakan sebagai *dataset* baru pembentuk CNN. Cara *under-sampling* dijelaskan oleh Marek Pawlicki, Micha Choraś, Rafal Kozik, dan Witold Holubowicz adalah dengan menghapus yang disebut *Tomek Link*.[11] *Tomek Link* adalah dua *records* yang berasal dari *class* yang berbeda, namun memiliki posisi yang sangat berdekatan, dimana tidak ada *record* lain yang berada diantara kedua *records* tersebut.[11,17]

2.3 ADASYN

ADASYN dikembang pada tahun 2008 oleh Haibo He, Yang Bai, Edwardo A. Garcia, dan Shutao Li.[4] ADASYN ini ide dasarnya adalah menggunakan *weighted distribution* untuk setiap *minority class* berdasarkan *level* dari *difficulty in learning*, yang di mana ADASYN akan men-*generate synthetic* data lebih untuk *minority class* yang lebih sulit untuk dipelajari.[4] Tujuan dari ADASYN ini adalah mengurangi bias pada *class imbalance* dan mengubah batasan antar *class* menuju *minority class*. [4]

2.4 *Principal Component Analysis*

PCA adalah metode untuk mengurangi dimensi dari *dataset* dengan mempertahankan *statistical information*.[7] Literatur mengenai PCA paling lama adalah dari Pearson dan Hotelling.[7] Langkah-langkah pokok dari PCA adalah sebagai berikut [5]:

1) Melakukan standarisasi data.
2) Meng-ekstrak *eigenvectors* dan *eigenvalues* dari *covariance matrix*:

$$CM = \frac{1}{n-1}\left((X - x')^T(X - x')\right) \quad (1)$$

Dimana x' adalah *mean vector* dengan rumus

$$x' = \left(\frac{1}{n}\right)\sum_{k=1}^{n}(x_i) \quad (2)$$

Covariance diantara dua *features* dihitung dengan rumus berikut:

$$Cv_{jk} = \left(\frac{1}{n-1}\right)\sum_{i=1}^{n}(x_{ij} - x'_j)(x_{ik} - x'_k) \quad (3)$$

3) Mengurutkan *eigenvalues* secara *descending* dan memilih *k eigenvector* yang berhubungan dengan *k eigenvalue* dimana *k* merupakan jumlah dimensi untuk *feature subspace* yang baru.
4) Membuat *projection matrix W* dari *k eigenvectors* yang sudah dipilih.
5) Transformasi *dataset* yang asli *X* melalui *W* untuk mendapatkan *k-dimensional feature subspace* $Y = X * W$.

3. DESAIN SISTEM

3.1 Analisis Sistem

Hal pertama yang dilakukan adalah membuat model *machine learning* yang akan digunakan sebagai percobaan. Model *machine learning* dibuat dengan menggunakan *library* python sklearn. Model *machine learning* yang pertama merupakan *support vector machine* yang akan digunakan sebagai *binary classification* untuk menentukan apakah sebuah *packet* merupakan intrusi atau bukan. Lalu pada model *machine learning* yang kedua adalah *Decision Tree* yang digunakan untuk melakukan klasifikasi jenis intrusi pada packet yang sudah diklasifikasi sebagai intrusi di model sebelumnya. Jika pada *Decision Tree*, sebuah *packet* tidak dapat diklasifikasikan jenis intrusinya, maka akan masuk ke model *machine learning* ketiga yang menggunakan *Naïve Bayes*. Pada model ketiga ini, sebuah *packet* akan dilihat probabilitas terhadap *class* yang ada, karena bisa jadi *packet* itu merupakan sebuah intrusi jenis baru yang merupakan turunan dari jenis intrusi yang sudah ada.

Analisis yang dilakukan adalah melihat *accuracy*, *false positive rate* (FPR), *false negative rate* (FNR), dan *detection rate* (DR) dari masing-masing model *machine learning*. Keempat hal tersebut akan dibandingkan pada saat melakukan *preprocessing* awal (*encoding*), saat melakukan PCA, setelah melakukan proses *under-sampling*, dan setelah melakukan proses *over-sampling*. Hal-hal ini menjadi penentu seberapa bagus *machine learning* yang sudah dihasilkan dengan menggunakan *dataset* yang telah dilakukan metode *sampling* tersebut. Hal lain yang perlu diperhatikan tetapi tidak menjadi faktor penentu pada penelitian ini adalah durasi waktu yang diperlukan untuk melakukan metode sampling untuk melihat dampak dari PCA terhadap metode *sampling*. Selain itu, juga diperlukan untuk melihat berapa banyak *records* yang berkurang dari setiap tahap.

3.2 Desain Sistem

3.2.1 Cleaning Dataset

Langkah awal yang dilakukan adalah *dataset* awal adalah membenahi *class* yang digunakan. Pada KDD'99 *dataset*, *class* yang digunakan masih merupakan bentuk intrusi bukan merupakan kategori dari intrusi tersebut, sehingga perlu dilakukan perubahan isi dari *class* menjadi bentuk kategori intrusi karena untuk mempermudah untuk melakukan perbandingan hasil pengujian terhadap penelitian-penelitian lainnya. Pada UNSW-NB15 *dataset*, *dataset* yang digunakan adalah *dataset* yang sudah dibersihkan dan dibagi menjadi *training* dan *testing* oleh Nour Moustafa. Pada *dataset* ini mengandung 175,341 *records* untuk *training set* dan 82,332 untuk *testing set*. Total dari *training* dan *testing* ada 257,673 *records* yang merupakan sekitar 10% (10.145%) dari total *dataset* asli yang berukuran 2,539,861 *records*.

Langkah selanjutnya dilakukan pengecekan apakah terdapat Not a Number (NaN) *value* atau terdapat *empty string* ("") *value* pada kolom atau ada *value* yang tidak sesuai dengan yang dideskripsikan. Pada KDD'99 *dataset* tidak ditemukan NaN *value* ataupun *empty string* sehingga langkah ini dilewatkan untuk *dataset* ini. Pada UNSW-NB15 *dataset*, meskipun sudah dibersihkan masih ditemukan pada kolom '*is_ftp_login*', yang seharusnya hanya mengandung *value* 0 atau 1, ternyata juga terdapat *value* 2 atau 4. Karena jumlah *records* yang mengandung *value* 2 atau 4 adalah 0.000101% (26 *records*) dari total *records* yang ada pada *training set* dan *testing set*, sehingga pada penelitian ini, *records* yang mengandung *value* 2 atau 4 dihapus. Selain itu, pada UNSW-NB15 *dataset* masih terdapat kolom *id* yang juga dihapus karena tidak dapat digunakan pada penelitian ini.

3.2.2 Memecah dataset menjadi training set dan testing set

Pada bagian, ini dilewati untuk *dataset* UNSW-NB15 karena *dataset* yang digunakan adalah *dataset* yang sudah dibagi menjadi *training set* dan *testing set*. Lalu pada *dataset* KDD'99, yang digunakan pada penelitian ini adalah 75% *dataset* dengan tambahan *minority classes* dari 25% sisa *dataset* untuk *testing set*. Penggunaan 75% *dataset* dikarenakan keterbatasan pada *hardware* untuk pengujian model pada saat setelah memasuki tahap standardisasi *dataset*. Pembagian 75% *dataset* menjadi *training set* dan *testing set* dilakukan dengan rasio 80% untuk *training set* dan 20% untuk *testing set* dengan tujuan membuat jumlah *testing set* khususnya pada *minority classes* tidak terlalu sedikit yang menyebabkan ketika pengujian dapat memberikan hasil yang jarak nilainya terlalu jauh.

3.2.3 Standardisasi Dataset

Setelah dilakukan *encoding* terhadap *dataset*, perlu dilakukan standardisasi *training set* dengan menggunakan *StandardScaler* yang mengimplementasikan *z-score* dengan tujuan membuat setiap *feature* dari *dataset* memiliki nilai *mean* 0 dan standar deviasi 1. Hal ini dilakukan agar hasil dari *Principal Component Analysis* (PCA) tidak condong ke kolom yang memiliki nilai yang besar.[10] Rumus dari *z-score* adalah sebagai berikut,

$$Z = \frac{x - \mu}{\sigma} \tag{4}$$

dimana Z merupakan nilai baru dari suatu *record* di suatu *feature*, x merupakan nilai asli dari *record* di suatu *feature*, μ merupakan *mean* dari *feature* tersebut, dan σ merupakan standar deviasi dari *feature* tersebut. Setelah dilakukan standarisasi pada *training set*, *test set* juga dilakukan standarisasi dengan menggunakan nilai *mean* dan standar deviasi dari *training set*. Hasil dari *dataset* yang sudah di standardisasi untuk ada tiga yaitu *Standard* yang berarti semua *features* diimplementasikan *StandardScaler*, *StandardPartial* yang berarti hanya *numerical features* yang diimplementasikan *StandardScaler*, dan *StandardPartialAlt* yang berarti semua *numerical features* dan *categorical features* yang **bukan** *binary* diimplementasikan *StandardScaler*. Hal ini menyebabkan pada *dataset* yang diimplementasikan *OneHotEncode* hanya memiliki *Standard* dan *StandardPartial*, karena hasil dari *OneHotEncode* sudah berbentuk *binary*.

3.2.4 Implementasi Principal Component Analysis

Langkah selanjutnya melakukan *feature extraction* pada *training set*. *Feature extraction* yang digunakan adalah *Principal Component Analysis* (PCA). Jumlah *feature* yang akan digunakan dari hasil *extraction* PCA akan ditentukan berdasarkan hasil percobaan. PCA dilakukan untuk mengurangi dimensi dari *dataset*. Hal ini dikarenakan pada metode *sampling* yang akan digunakan baik *Adaptive synthetic sampling* (ADASYN) dan *Tomek links*, keduanya menerapkan konsep *K-nearest neighbor* (KNN) pada algoritmanya, sehingga dengan meggunakan PCA, diharapkan dapat mengurangi beban komputasi dari metode-metode *sampling* tersebut. *Testing set* juga dilakukan PCA yang sudah dibentuk berdasarkan *training set*. Pada penelitian ini, jumlah *components* yang diambil untuk metode *sampling* ditentukan berdasarkan kumulatif dari *explained_variance_ratio* dari setiap *components* yang sudah diurutkan secara *descending* berdasarkan *explained_variance_ratio*. *explained_variance_ratio*

adalah *attribute* yang ada pada PCA yang disediakan oleh *sklearn* yang berguna untuk menampilkan persentase dari varians dari *dataset* awal yang dimiliki oleh suatu *component*. Nilai batas kumulatif yang digunakan adalah 0.95, yang berarti *component* diambil sampai nilai kumulatif dari *explained_variance_ratio* lebih dari 0.95 pertama.

3.2.5 Implementasi Undersampling

Selanjutnya pada *training set* dilakukan *undersampling* yaitu menghapus *record-record* tertentu. Metode *undersampling* yang digunakan adalah *Tomek Links*. Tujuan dari menggunakan metode ini adalah untuk menghilangkan *record-record* khususnya bagian dari *class* yang memiliki proporsi terbanyak yang berdekatan dengan *record* lain yang merupakan bagian dari *class* yang memiliki proporsi lebih sedikit. Metode ini juga membantu menghapus beberapa *record* yang merupakan *outlier* yang berada di antara *record-record* lain yang merupakan bagian dari *class* yang lain. Hal ini karena *Tomek Links* bekerja dengan menghapus salah satu *record* dari kedua *record* yang merupakan anggota *class* yang berbeda tetapi berdekatan, jika tidak ada *record* lain yang berada di antara kedua *record* tersebut.

3.2.6 Implementasi Oversampling

Metode *sampling* terakhir yang dilakukan pada *training set* adalah *oversampling* yaitu membuat *record* sintetis untuk *class-class* tertentu khususnya pada *class* yang memiliki proporsi yang paling kecil. Metode *oversampling* yang digunakan adalah *Adaptive Synthetic* (ADASYN). Metode ini digunakan untuk membuat *record* sintetis untuk *class* yang memiliki proporsi lebih sedikit dibanding *class* lainnya dengan melihat *density distribution* dari setiap *class*. Cara yang digunakan oleh metode ini untuk membuat *record* sintetis adalah dengan memilih satu *random record* yang merupakan *minority class* dari *K-nearest neighbors* dari sebuah *record* dari *minority class*, lalu mengimplementasikan rumus berikut,

$$s_i = x_i + (x_{zi} - x_i) \times \lambda \quad (5)$$

dimana s_i merupakan *record* sintetis yang akan dihasilkan, x_i merupakan sebuah *record* dari suatu *class*, x_{zi} merupakan *record* yang dipilih dengan melihat *K-nearest neighbor* dari x_i, dan λ merupakan sebuah angka acak yang berada di antara 0 sampai 1.

3.2.7 Training Machine Learning Model

Pada kedua proses *sampling* tersebut, parameter yang digunakan berdasarkan hasil percobaan. Setelah dilakukan kedua proses *sampling* tersebut, *training set* digunakan untuk *training* model *machine learning* yang sudah dibuat. Model *machine learning* tersebut terdiri dari *Support Vector Machine*, *Decision Tree*, dan *Naïve Bayes*. *Training set* ketika digunakan untuk *Support Vector Machine*, menggunakan *class* "1" untuk intrusi dan "0" untuk normal, sedangkan untuk *Decision Tree* dan *Naïve Bayes* menggunakan *class* yang merupakan intrusi saja. Selain itu, waktu yang diperlukan oleh setiap *machine learning* juga disimpan untuk perbandingan khususnya dengan setelah dilakukan *Principal Component Analysis*.

3.2.8 Testing Machine Learning Model

Setelah model *machine learning* tersebut disimpan, model *machine learning* tersebut digunakan ke dalam dua jenis *testing* yaitu *model performance testing*, dan *IDS performance testing*. Pada *model performance testing*, semua *records* yang ada, baik pada *training set* atau *testing set* akan digunakan sebagai pengujian pada setiap *machine learning model* secara terpisah.

Berbeda dengan *testing* menggunakan *model* performance, pada *IDS performance testing*, *records* yang digunakan akan berkurang secara bertahap berdasarkan hasil klasifikasi dari *machine learning model*. Tahap pengurangan pertama di mulai dari hasil klasifikasi *support vector machine*. Jika hasil klasifikasi dari *support vector machine* adalah 1 ("*intrusion*"), maka *record* tersebut diambil dan digunakan untuk *testing* pada proses *machine learning* selanjutnya, sebaliknya jika hasil klasifikasinya adalah 0 akan diabaikan. Selanjutnya pada tahap *Decision Tree*, untuk menentukan *records* yang akan digunakan lagi untuk *Naïve Bayes*, adalah dengan melihat menggunakan fungsi *predict_proba* yang ada pada *Decision Tree* pada *library* sklearn. *predict_proba* bekerja dengan melihat jumlah *records* untuk *class* yang sama yang ada pada suatu *leaf*.[12] Jika *predict_proba* untuk suatu *class* bernilai 1, berarti pada *leaf* tersebut, semua *records* yang digunakan untuk *training* memiliki *class* yang sama. Pada penelitian ini, *records* yang digunakan untuk *testing* pada N*aïve Bayes* adalah *records* yang tidak memiliki *predict_proba* bernilai 1 pada salah satu *class*, sebaliknya jika mengandung *predict_proba* bernilai 1, tidak diteruskan.

4. PENGUJIAN SISTEM

4.1 Pengujian pada *dataset* KDD'99

Pada *dataset* ini, setelah diimplementasikan *over-sampling*, model *Support Vector Machine* yang dihasilkan dengan melakukan *training* pada *dataset* yang sudah dimodifikasi dapat menjadi lebih sensitif terhadap *minority classes* yang merupakan *Probe*, R2L, dan U2R pada saat diujikan terhadap *testing set*. Meskipun terjadi pengurangan jumlah *records* yang merupakan *minority class* pada saat implementasi PCA, dikarenakan berkurangnya jumlah *components* yang diambil, sehingga Sebagian informasi juga hilang, pada saat diimplementasikan *under-sampling* dan *over-sampling*, jumlah penambahan lebih banyak daripada pengurangan yang terjadi.

Pengujian ini dilihat dengan cara proses dari *IDS Process*, di mana setelah melalui *Support Vector Machine*, *records* yang diklasifikasi sebagai "Normal" (0), akan diabaikan. Maka, pada saat data akan masuk ke *Decision Tree*, itu berarti data-data tersebut sudah dianggap oleh *Support Vector Machine* sebagai intrusi. Melalui semua data intrusi tersebut, dilihat berdasarkan *ground truth* yang sudah dibuat ada berapa banyak *records* yang sebenarnya merupakan *minority class*. Semakin banyak *records* yang aslinya merupakan *minority class*, berarti *Support Vector Machine* dapat dianggap semakin bagus dalam mendeteksi *minority class*.

Tabel 1. Perubahan Jumlah *Class Output* untuk *Minority Class* setelah melalui SVM

Probe	Testing							
Encoding	Standardize Data	AfterADASYNOri			AfterADASYNBagi5			PCs Used in PCA
		105%	110%	120%	105%	110%	120%	
Label	Standard	156	**215**	213	158	**215**	213	21
	StandardPartialAlt	185	219	219	184	219	**220**	17
OneHot	Standard	137	139	139	137	140	**141**	86
	StandardPartial	194	228	242	193	229	**243**	17

R2L	Testing							
Encoding	Standardize Data	AfterADASYNOri			AfterADASYNBagi5			PCs Used in PCA
		105%	110%	120%	105%	110%	120%	
Label	Standard	52	60	**101**	53	71	**101**	21
	StandardPartialAlt	65	71	88	65	71	**108**	17
OneHot	Standard	18	59	59	18	58	**60**	86
	StandardPartial	42	49	**87**	42	49	**87**	17

U2R	Testing							
Encoding	Standardize Data	AfterADASYNOri			AfterADASYNBagi5			PCs Used in PCA
		105%	110%	120%	105%	110%	120%	
Label	Standard	6	9	**10**	9	**10**	**10**	21
	StandardPartialAlt	9	9	**10**	9	9	**10**	17
OneHot	Standard	2	3	3	2	**4**	**4**	86
	StandardPartial	7	7	**9**	7	7	**9**	17

Berdasarkan pada Tabel 1, dapat dilihat perubahan kemampuan SVM dalam mendeteksi intrusi yang merupakan bagian dari *minority classes* pada setiap hal yang dilakukan pada *dataset* terhadap *dataset* awal yang hanya mengimplementasikan *encoding* dan *StandardScaler*. Melihat pada hasil akhir dari proses yaitu setelah diimplementasikan ADASYN, SVM dapat mendeteksi lebih banyak dibandingkan *dataset* awal, paling banyak 220 untuk *Probe*, 108 untuk R2L, dan 10 untuk U2R.

Pada proses *training* pada *Decision Tree*, *Tree* yang dihasilkan dengan menggunakan *dataset* ini, menghasilkan TPR 100% untuk setiap kali pengujian dengan data *training*, sehingga pada penelitian tidak dibahas. Mengenai *Detection Rate* (DR) atau *True Positive Rate* (TPR), hasil paling bagus terjadi pada *class* R2L, dapat dilihat pada Tabel 2, di mana pada *Decision Tree*, terdapat peningkatan paling baik sebesar 0,665% dan pada *Naïve Bayes* terhadap *data testing*, peningkatan paling besar adalah sebesar 47,894%.

Tabel 2. Perubahan TPR dari *Class* R2L terhadap *Dataset* Awal (dalam Persentase)

Decision Tree Test

Encoding	Standardize Data	AfterADASYNOri			AfterADASYNBagi5		
		105%	110%	120%	105%	110%	120%
Label	Standard	-0,443	-1,330	-2,439	**0,443**	-1,774	-1,330
	StandardPartialAlt	**0,665**	0,222	0,222	-0,443	-0,222	0,443
OneHot	Standard	**0,222**	-0,887	-1,330	-0,443	-1,109	-1,330
	StandardPartial	-0,665	-0,665	-0,887	-0,887	-1,774	**-0,443**

Naive Bayes Train

Encoding	Standardize Data	AfterADASYNOri			AfterADASYNBagi5		
		105%	110%	120%	105%	110%	120%
Label	Standard	49,007	49,180	49,421	49,420	49,349	**49,579**
	StandardPartialAlt	34,232	35,450	**36,173**	29,987	14,550	29,566
OneHot	Standard	-31,978	-33,521	-34,185	-32,239	-33,465	-33,987
	StandardPartial	40,196	42,026	42,931	24,895	42,038	**43,014**

Naive Bayes Test

Encoding	Standardize Data	AfterADASYNOri			AfterADASYNBagi5		
		105%	110%	120%	105%	110%	120%
Label	Standard	**47,894**	**47,894**	**47,894**	**47,894**	**47,894**	**47,894**
	StandardPartialAlt	16,851	**17,295**	**17,295**	15,743	5,543	15,965
OneHot	Standard	-19,290	-19,290	-19,290	-19,290	-19,290	-19,290
	StandardPartial	16,630	**17,073**	**17,073**	4,878	**17,073**	**17,073**

Class yang mengalami penurunan pada TPR adalah pada *Probe* dan U2R. Penurunan khususnya terjadi pada saat *testing* menggunakan model *Naïve Bayes*. Hal ini dapat dilihat pada Tabel 3 untuk perubahan pada *Probe* dan Tabel 4 untuk U2R. Dapat dilihat pada tabel-tabel tersebut, terjadi penurunan semua pada bagian *Naïve Bayes* kecuali untuk U2R bagian *Testing* dengan *One-Hot Encode* dan menggunakan *Standard* dengan penuh jika setelah mengimplementasikan ADASYN. Saat melakukan pengujian dengan menggunakan *data testing*, penurunan paling kecil adalah 19,24% untuk *Probe* dan 33,33% untuk U2R. Lalu pada *Decision Tree*, terjadi peningkatan paling bagus 0.03% untuk *class Probe* dan 4.762% untuk *class* U2R..

Ada dua hal yang mungkin menyebabkan hal ini terjadi. Pertama adalah karena algoritma *over-sampling* yang digunakan. Algoritma dari ADASYN adalah membuat data sintetis baru di antara dua *record* dengan *class* yang sama. Yang menjadi masalah pada algoritma ini adalah pada saat menentukan *record* mana yang akan dibuatkan data sintetis baru lebih banyak. ADASYN memanfaatkan yang namanya *density distribution* untuk menentukan setiap *record* memerlukan berapa data sintetis baru. *Density distribution* ini dihitung melihat berapa banyak *records* yang berasal dari *majority class* yang ada di sekitar dari sebuah *records minority class*. Semakin banyak *records* yang dari *majority class*, semakin banyak pula data sintetis yang perlu dibuat oleh ADASYN. Hal ini terlihat bagus apabila *record* dari *minority class* tersebut bukan *outlier*.

Kenyataannya, ADASYN tidak dapat melakukan pengecekan apakah sebuah *records* merupakan *outlier* atau bukan, sehingga jika sebuah *record* dari *minority class* merupakan *outlier*, secara otomatis, *record* tersebut akan dibuatkan data sintetis yang banyak, karena sekitarnya pasti merupakan *records* dari *class* yang lainnya. Ditambah dengan jumlah *records* dari *dataset* ini terdapat 2,939,015. Ketika *oversampling* dengan menambah sekitar 10% dari total *dataset* sudah sama dengan menambah sekitar 293,901 *synthetic records*, sedangkan total dari *records* milik *Probe*, R2L, dan U2R hanya 25,368 *records*, tidak sampai sepersepuluh dari total *synthetic records*.

Tabel 3. Perubahan TPR dari *Class Probe* terhadap *Dataset* Awal (dalam Persentase)

Decision Tree Test

Encoding	Standardize Data	AfterADASYNOri			AfterADASYNBagi5		
		105%	110%	120%	105%	110%	120%
Label	Standard	0,024	**0,030**	0,006	0,024	0,024	0,024
	StandardPartialAlt	-0,043	-0,055	**-0,036**	**-0,036**	-0,079	-0,049
OneHot	Standard	-0,091	-0,073	**-0,043**	-0,085	-0,085	-0,091
	StandardPartial	-0,067	-0,067	-0,061	-0,055	**-0,030**	-0,103

Naive Bayes Train

Encoding	Standardize Data	AfterADASYNOri			AfterADASYNBagi5		
		105%	110%	120%	105%	110%	120%
Label	Standard	-43,561	-47,215	-48,076	-44,361	-46,970	-48,098
	StandardPartialAlt	-50,619	-54,639	-56,310	-51,256	-54,205	-56,424
OneHot	Standard	-41,940	-38,165	**-36,078**	-41,987	-38,177	-36,093
	StandardPartial	-54,628	-59,986	-62,515	-55,008	-59,021	-62,242

Naive Bayes Test

Encoding	Standardize Data	AfterADASYNOri			AfterADASYNBagi5		
		105%	110%	120%	105%	110%	120%
Label	Standard	-19,240	-19,726	-19,678	-19,446	-19,665	-19,690
	StandardPartialAlt	-21,119	-21,302	-21,350	-21,144	-21,235	-21,344
OneHot	Standard	-89,897	-89,897	-89,891	-89,897	-89,897	-89,897
	StandardPartial	-22,646	-22,950	-27,032	-22,676	-22,835	-26,989

Tabel 4. Perubahan TPR dari *Class* U2R terhadap *Dataset* Awal (dalam Persentase)

Decision Tree Test

Encoding	Standardize Data	AfterADASYNOri			AfterADASYNBagi5		
		105%	110%	120%	105%	110%	120%
Label	Standard	-19,048	-9,524	-9,524	-19,048	-9,524	**-4,762**
	StandardPartialAlt	**0,000**	-14,286	-9,524	-14,286	-19,048	-23,810
OneHot	Standard	**4,762**	0,000	0,000	**4,762**	0,000	0,000
	StandardPartial	-19,048	-19,048	-14,286	-34,292	**-9,524**	**-9,524**

Naive Bayes Train

Encoding	Standardize Data	AfterADASYNOri			AfterADASYNBagi5		
		105%	110%	120%	105%	110%	120%
Label	Standard	-59,333	-60,270	-62,337	-60,506	-62,462	-63,056
	StandardPartialAlt	-92,035	-91,940	-92,990	-65,946	**-49,864**	-58,996
OneHot	Standard	-3,333	-3,061	-2,793	**-2,567**	-2,779	-2,841
	StandardPartial	-97,664	-97,710	-97,361	**-42,857**	-85,463	-93,179

Naive Bayes Test

Encoding	Standardize Data	AfterADASYNOri			AfterADASYNBagi5		
		105%	110%	120%	105%	110%	120%
Label	Standard	-33,333	-33,333	-33,333	-33,333	-33,333	-33,333
	StandardPartialAlt	-85,714	-85,714	-85,714	-57,143	**-52,381**	-57,143
OneHot	Standard	9,524	9,524	9,524	9,524	9,524	9,524
	StandardPartial	-90,476	-90,476	-90,476	**-61,905**	-76,190	-90,476

4.2 Pengujian pada *dataset* UNSW-NB15

Pada *dataset* ini, yang menjadi pengujian utama adalah melihat apakah ada peningkatan terhadap akurasi dari IDS *system* yang diusulkan oleh Goeschel jika ditambahkan *feature extraction* dan metode *sampling* pada *dataset* yang lebih baru. Kenyataannya, akurasi yang dihasilkan oleh IDS *System* ini lebih rendah dibandingkan dengan *dataset* KDD'99. Meskipun telah diberikan

feature extraction, *under-sampling*, dan *over-sampling*, justru akurasi yang dihasilkan cenderung menurun khususnya pada *Decision Tree*, dan pada *Gaussian Naïve Bayes*. ada model SVM yang dihasilkan menggunakan *training set* yang sudah di-*over-sampling*, terjadi peningkatan akurasi pada saat *testing*, namun peningkatan yang terjadi tidak lebih tinggi daripada saat *training* menggunakan *training set* yang tidak di-*over-sampling*.

Perubahan akurasi yang dimiliki oleh SVM yang ada pada Tabel 5, terjadi peningkatan paling banyak sebesar 0.045%, namun jumlah itu masih tidak sebesar jika tidak menggunakan ADASYN yaitu 0.068%. Pada metode *Decision Tree*, terjadi penurunan akurasi dengan penurunan paling kecil sebesar 4.395% dan penurunan akurasi terbesar yaitu 9.339%, saat setelah menggunakan metode ADASYN. Sedangkan saat menggunakan *Naïve Bayes*, juga paling sering terjadi penurunan di mana paling besar, akurasi menurun sebanyak 43.913%, untuk kenaikan akurasi setelah menggunakan ADASYN paling besar adalah 1.513%, namun angka tersebut lebih kecil dibandingkan jika tidak menggunakan ADASYN yaitu sebesar 2.813% kenaikan akurasi jika hanya menggunakan PCA.

Tabel 5. Perubahan Akurasi dari *Testing* yang dilakukan pada *Machine Learning* terhadap *Dataset Pure* (dalam Persentase)

Support Vector Machine

Encoding	Standardize Data	AfterPCA	AfterTomek	AfterADASYNOri			AfterADASYNBagi5		
				105%	110%	120%	105%	110%	120%
Label	Standard	0,041	**0,068**	0,045	0,040	0,017	0,053	0,044	0,029
	StandardPartialAlt	0,026	**0,063**	0,029	0,024	0,004	0,030	0,024	0,016
OneHot	Standard	-0,030	**-0,019**	-0,029	-0,030	-0,029	-0,021	-0,023	-0,024
	StandardPartial	0,030	**0,040**	0,029	-0,035	-0,043	0,029	0,027	-0,043

Decision Tree (Except Normal)

Encoding	Standardize Data	AfterPCA	AfterTomek	AfterADASYNOri			AfterADASYNBagi5		
				105%	110%	120%	105%	110%	120%
Label	Standard	-4,048	**-3,587**	-5,030	-5,835	-5,432	-5,270	-5,471	-5,979
	StandardPartialAlt	-3,717	**-3,711**	-4,569	-5,094	-6,076	-5,030	-5,769	-6,188
OneHot	Standard	-10,792	**-4,432**	-5,546	-8,763	-8,300	-5,926	-9,339	-9,175
	StandardPartial	-6,230	-5,330	**-4,395**	-5,079	-6,735	-5,862	-6,652	-5,875

Naïve Bayes (Except Normal)

Encoding	Standardize Data	AfterPCA	AfterTomek	AfterADASYNOri			AfterADASYNBagi5		
				105%	110%	120%	105%	110%	120%
Label	Standard	**1,194**	0,902	-1,231	-2,978	-4,167	-2,755	-3,993	-5,151
	StandardPartialAlt	**2,396**	2,336	1,513	1,114	0,516	0,545	-0,141	-0,896
OneHot	Standard	-45,427	-43,975	-43,909	-43,913	**-43,891**	-43,924	-43,898	-43,896
	StandardPartial	**2,813**	2,301	-0,827	-2,264	-3,455	-1,624	-2,308	-4,119

4.3 Pengaruh antara PCA dengan metode *sampling*

Tujuan awal dari penggunaan PCA pada penelitian ini adalah untuk membantu metode-metode sampling yang digunakan dapat berjalan lebih cepat. Asumsi ini berasal dari dasar algoritma dari ADASYN dan Tomek Links. Kedua algoritma tersebut sama-sama ada memanfaatkan *K-Nearest Neighbor* (K-NN) sebagai inti dari algoritmanya. Pada *Tomek Links*, *1-Nearest Neighbor* digunakan untuk mencari yang disebut *Tomek Links* sebelum akhirnya salah satu dari dua *records* dihapus untuk menghilangkan *records* yang berada di *border*. Lalu pada ADASYN, K-NN digunakan untuk menghitung *density distribution* dari sebuah *record* yang merupakan *minority class*.

Berdasarkan hasil observasi yang ada pada Tabel 6, PCA membantu mengurangi waktu yang diperlukan untuk Tomek Links paling banyak berkurang 48.444% dari jika tidak menggunakan PCA untuk dataset KDD'99 dan pada dataset UNSW-NB15 waktu menjadi berkurang paling banyak 98% dari waktu awal yang tidak menggunakan PCA. Lalu pada Tabel 7, merupakan waktu yang diperlukan untuk mengimplementasikan ADASYN. Berdasarkan Tabel 7, waktu yang diperlukan oleh ADASYN untuk melakukan over-sampling berkurang paling banyak 19.192% untuk dataset KDD'99 dan 99.276% untuk dataset UNSW-NB15 dibandingkan dengan jika tidak menggunakan PCA.

Tabel 6. Waktu yang Diperlukan untuk *Under-Sampling*

KDD'99

Tomek Links

Encoding	Standardize Data	Without PCA (in seconds)	With PCA 95% explained_variances (in seconds)	Time Reduced (in percentage)
Label	Standard	163000,603	127353,111	21,86954586
	StandardPartialAlt	190906,172	134996,799	29,28630985
OneHot	Standard	231895,326	179193,591	22,72651901
	StandardPartial	193915,612	99975,636	48,44374071

UNSW-NB15

Tomek Links

Encoding	Standardize Data	Without PCA (in seconds)	With PCA 95% explained_variances (in seconds)	Time Reduced (in percentage)
Label	Standard	516,942	18,889	96,34596625
	StandardPartialAlt	413,131	24,646	94,03436797
OneHot	Standard	3938,854	78,594	98,00464598
	StandardPartial	943,300	24,018	97,45385519

Tabel 7. Jumlah Pengurangan Waktu yang Diperlukan untuk *Over-Sampling*

KDD'99

ADASYN

Encoding	Standardize Data	Time Reduced From Without PCA to With PCA (in percentage)					
		Ori 105%	Ori 110%	Ori 120%	Bagi5 105%	Bagi5 110%	Bagi5 120%
Label	Standard	7,204	14,194	8,291	6,602	16,353	8,186
	StandardPartialAlt	4,392	11,122	8,669	11,804	17,453	17,290
OneHot	Standard	-6,366	5,183	-1,381	2,743	11,127	-1,927
	StandardPartial	9,038	19,192	17,691	6,666	15,936	12,846

UNSW-NB15

ADASYN

Encoding	Standardize Data	Time Reduced From Without PCA to With PCA (in percentage)					
		Ori 105%	Ori 110%	Ori 120%	Bagi5 105%	Bagi5 110%	Bagi5 120%
Label	Standard	99,093	99,084	99,131	99,212	99,076	98,652
	StandardPartialAlt	99,104	98,997	99,193	99,199	98,987	98,452
OneHot	Standard	97,056	97,297	97,615	96,955	97,248	96,584
	StandardPartial	99,207	98,978	99,244	99,276	99,195	98,665

5. KESIMPULAN DAN SARAN

Penelitian yang telah dilakukan dapat disimpulkan bahwa dengan hasil penelitian ini yang kurang memuaskan pada beberapa seperti berikut:

- Pada model *Naïve Bayes*, TPR pada *dataset* KDD'99 yang sudah dimodifikasi (menggunakan kombinasi metode PCA, *Tomek Links*, dan ADASYN) ternyata lebih rendah daripada TPR pada *dataset* KDD'99 asli. Penurunan TPR terjadi pada *Probe class* (sebesar 19.44%) dan U2R *class* (sebesar 33.33%). Sebaliknya, peningkatan TPR terjadi pada R2L *class* (sebesar 47.894%).
- Pada *Decision Tree*, akurasi pada *dataset* UNSW-NB15 yang sudah dimodifikasi (menggunakan kombinasi metode PCA, *Tomek Links*, dan ADASYN) menjadi lebih kecil daripada tidak dimodifikasi. Penurunan yang terjadi sebesar 4.395% paling kecilnya.
- Pada SVM dan *Naïve Bayes*, akurasi pada *dataset* UNSW-NB15 yang sudah dimodifikasi (menggunakan kombinasi metode PCA, *Tomek Links*, dan ADASYN) meningkat. Peningkatan akurasi terjadi sebesar 0.045% pada SVM dan 1.513% pada *Naïve Bayes*. Namun, peningkatan yang terjadi masih lebih kecil daripada menggunakan UNSW-NB15 yang

sudah di-PCA saja (akurasi SVM meningkat sebesar 0.068% dan akurasi *Naïve Bayes* meningkat sebesar 2.813%).

Selain hasil yang kurang memuaskan, pada penelitian ini juga ada beberapa hasil yang cukup memuaskan seperti berikut:

- *Support Vector Machine* dapat mengenali lebih banyak *records* milik *class* intrusi khususnya dari *minority class* pada *dataset* KDD'99 yang sudah dimodifikasi (menggunakan kombinasi metode PCA, *Tomek Links*, dan ADASYN) daripada tanpa modifikasi. Penambahan terjadi sebanyak 220 *records* untuk *Probe class*, 108 *records* untuk R2L *class*, dan 10 *records* untuk U2R *class*.
- Pada model *Decision Tree*, TPR pada *dataset* KDD'99 yang sudah dimodifikasi (menggunakan kombinasi metode PCA, *Tomek Links*, dan ADASYN) mengalami peningkatan pada *Probe class* (sebesar 0.03%), R2L *class* (sebesar 0.665%), dan U2R *class* (sebesar 4.762%).
- Waktu yang diperlukan untuk *under-sampling* dan *over-sampling* pada *dataset* yang sudah dimodifikasi (menggunakan PCA) lebih sedikit daripada jika tidak dimodifikasi.

6. DAFTAR PUSTAKA

[1] Abdulhammed, Razan, Musafer, Hassan, Alessa, Ali, Faezipour, Miad, and Abuzneid, Abdelshakour. 2019. Features dimensionality reduction approaches for machine learning based network intrusion detection. *Electron.* 8, 3. DOI:https://doi.org/10.3390/electronics8030322

[2] Atilla, Ozgur and Hamit, Erdem. 2016. A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015. *PeerJ*, 0–21. DOI:https://doi.org/10.7287/peerj.preprints.1954v1

[3] Goeschel, Kathleen. 2016. Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees, and naive Bayes for off-line analysis. In *Conference Proceedings - IEEE SOUTHEASTCON*. DOI:https://doi.org/10.1109/SECON.2016.7506774

[4] He, Haibo, Bai, Yang, Garcia, Edwardo A., and Li, Shutao. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. *Proc. Int. Jt. Conf. Neural Networks* 3 (2008), 1322–1328. DOI:https://doi.org/10.1109/IJCNN.2008.4633969

[5] Ibrahimi, Khalil and Ouaddane, Mostafa. 2017. Management of intrusion detection systems based-KDD99: Analysis with LDA and PCA. *Proc. - 2017 Int. Conf. Wirel. Networks Mob. Commun. WINCOM 2017*. DOI:https://doi.org/10.1109/WINCOM.2017.8238171

[6] Ippolito, Pier Paolo. 2019. Feature Extraction Techniques. An end to end guide on how to reduce a… | by Pier Paolo Ippolito | Towards Data Science. URI= https://towardsdatascience.com/feature-extraction-techniques-d619b56e31be

[7] Jollife, Ian T. and Cadima, Jorge. 2016. Principal component analysis: A review and recent developments. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* 374, 2065. DOI:https://doi.org/10.1098/rsta.2015.0202

[8] Khor, Kok-Chin, Ting, Choo-Yee, and Phon-Amnuaisuk, Somnuk. 2014. The Effectiveness of Sampling Methods for the Imbalanced Network Intrusion Detection Data Set. In *Advances in Intelligent Systems and Computing*. 613–622. DOI:https://doi.org/10.1007/978-3-319-07692-8_58

[9] Miah, Md Ochiuddin, Khan, Sakib Shahriar, Shatabda, Swakkhar, and Farid, Dewan Md. 2019. Improving Detection Accuracy for Imbalanced Network Intrusion Classification using Cluster-based Under-sampling with Random Forests. *1st Int. Conf. Adv. Sci. Eng. Robot. Technol. 2019, ICASERT 2019*, Icasert, 1–5. DOI:https://doi.org/10.1109/ICASERT.2019.8934495

[10] Patil, Aniket. Principal Component Analysis(PCA) | by Aniket Patil | Analytics Vidhya | Medium. URI= https://medium.com/analytics-vidhya/principal-component-analysis-pca-8a0fcba2e30c

[11] Pawlicki, Marek, Choraś, Michał, Kozik, Rafał, and Hołubowicz, Witold. 2020. On the Impact of Network Data Balancing in Cybersecurity Applications. In Krzhizhanovskaya, Valeria V., Závodszky, Gábor, Lees, Michael H., Dongarra, Jack J., Sloot, Peter M. A., Brissos, Sérgio and Teixeira, João (eds.). Springer International Publishing, Cham, 196–210. DOI:https://doi.org/10.1007/978-3-030-50423-6_15

[12] Scikit-learn. sklearn.tree.DecisionTreeClassifier — scikit-learn 0.24.2 documentation. URI= https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.predict_proba

[13] Seo, Jae Hyun and Kim, Yong Hyuk. 2018. Machine-learning approach to optimize smote ratio in class imbalance dataset for intrusion detection. *Comput. Intell. Neurosci.* 2018, (2018). DOI:https://doi.org/10.1155/2018/9704672

[14] Singh, Rohit, Kalra, Mala, and Solanki, Shano. 2019. A hybrid approach for intrusion detection based on machine learning. *Proc. Int. Conf. Intell. Sustain. Syst. ICISS 2019* Iciss, 187–192. DOI:https://doi.org/10.1109/ISS1.2019.8908116

[15] Su, Peihuang, Liu, Yanhua, and Song, Xiang. 2018. Research on intrusion detection method based on improved SMOTE and XGBoost. *ACM Int. Conf. Proceeding Ser.*, 42–49. DOI:https://doi.org/10.1145/3290480.3290505

[16] Tavallaee, Mahbod, Bagheri, Ebrahim, Lu, Wei, and Ghorbani, Ali A. 2009. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, IEEE, 1–6. DOI:https://doi.org/10.1109/CISDA.2009.5356528

[17] Tomek, Ivan. 1976. Two Modifications of Cnn. *IEEE Trans. Syst. Man Cybern.* SMC-6, 11, 769–772. DOI:https://doi.org/10.1109/TSMC.1976.4309452

[18] Wu, Junqi and Hu, Zhengbing. 2008. Study of Intrusion Detection Systems (IDSs) in Network Security. In *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, IEEE, 1–4. DOI:https://doi.org/10.1109/WiCom.2008.1085