CYBER THREAT DETECTION USING MACHINE LEARNING

INTRODUCTION

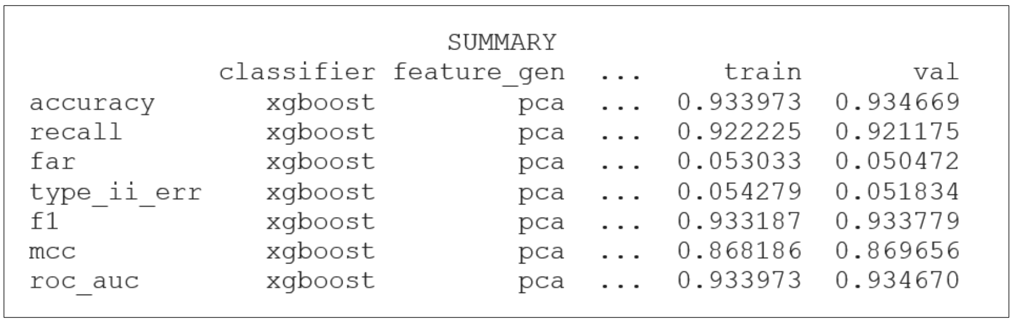
This study is about cyber threats in a Wi-Fi network. Although the original projects distinguished different types of cyber attacks, such injection, flooding, impersonation and passive attacks, this study is focused on impersonation attacks, classifying traffic “intrusive” or attacks and “good” normal traffic. It is based on the Aegean WiFi Intrusion/threat Dataset (AWID2) project, that was prepared and managed by George Mason University (U.S.) and University of the Aegean (Greece). The objective was to survey and evaluate research in cyber threat detection. This study has a reduced version of this famous dataset, with the idea of showing a comparison of the performance of several algorithms, using a baseline values as a reference.

STRATEGY

The strategy followed in this study starts with the definition of a baseline reference, from a basic setup of the chosen method, to gradually add different variants and methods, with the intention of beating the marks of the mentioned baseline. Extensive explanations have been included in the Appendixes of this assessment, including baseline reference and evaluation metrics justification, and the study of TB and TT.

EXPERIMENTS

Once the baseline has been stablished, it has been tried to add more complex characteristics to the model, starting by applying the baseline using only the features generated by the PCA technique as per figures 1 and 2.



```
                          SUMMARY
             classifier feature_gen  ...     train       val
accuracy        xgboost         pca  ...  0.933973  0.934669
recall          xgboost         pca  ...  0.922225  0.921175
far             xgboost         pca  ...  0.053033  0.050472
type_ii_err     xgboost         pca  ...  0.054279  0.051834
f1              xgboost         pca  ...  0.933187  0.933779
mcc             xgboost         pca  ...  0.868186  0.869656
roc_auc         xgboost         pca  ...  0.933973  0.934670
```

Fig. 1: Table for PCA generated features only combined with baseline.

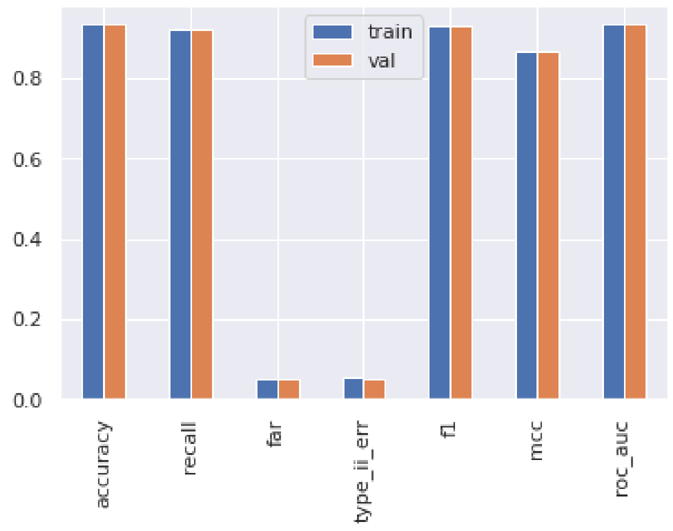
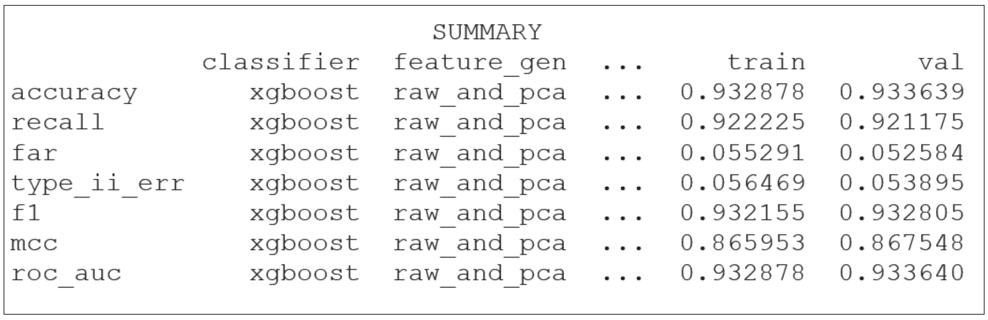


Fig. 2: Bar plot for PCA generated features only combined with baseline.

The application of PCA has generated 10 new features and the model results are virtually the same than using raw features. This can be interpreted as PCA is correctly gathering latent relations between original features and translating it to 10 new features. Metrics values in figures Adding these new features to the original group of features generates the results in figures 3 and 4.



```
                              SUMMARY
            classifier  feature_gen  ...     train       val
accuracy       xgboost  raw_and_pca  ...  0.932878  0.933639
recall         xgboost  raw_and_pca  ...  0.922225  0.921175
far            xgboost  raw_and_pca  ...  0.055291  0.052584
type_ii_err    xgboost  raw_and_pca  ...  0.056469  0.053895
f1             xgboost  raw_and_pca  ...  0.932155  0.932805
mcc            xgboost  raw_and_pca  ...  0.865953  0.867548
roc_auc        xgboost  raw_and_pca  ...  0.932878  0.933640
```

Fig. 3: Table for PCA generated + original features combined with baseline.

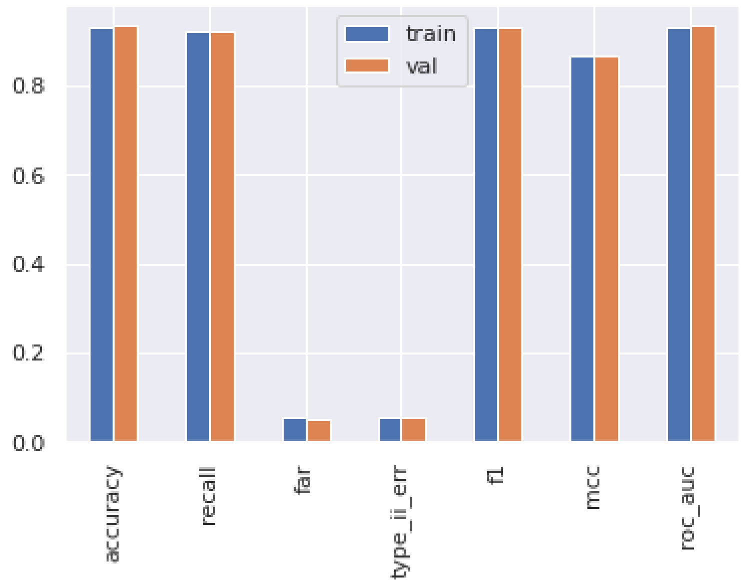
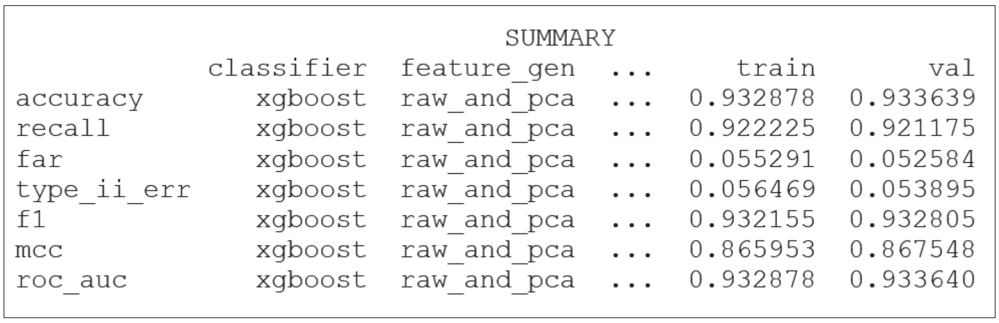


Fig. 4: Bar plot for PCA generated + original features combined with baseline.

No significant difference has made the last change, confirming that the dimensional reduction is very efficient, reducing 152 features to 10, and keeping almost all relevant statistical information of the dataset.

The next experiment uses SVM filter, providing a selection of the features from the earlier case. As recommended at [11], linear SVM models using weighted feature selection, are a good starting point as they deliver good results. Linear SVM deliver a best fit hyperplane, discriminating the classes, where that hyperplane is represented by the mentioned weights or coefficients. Other publications, as [12] and [13], supports the use of linear SVM weight based for classification problems as well.

Executing this method, a selection of 76 features, 66 from the original dataset and 10 from the PCA process, has been used, and 86 removed. The results in figures 5 and 6, not making significant difference with the results of the baseline.



```
                              SUMMARY
             classifier  feature_gen  ...     train       val
accuracy        xgboost  raw_and_pca  ...  0.932878  0.933639
recall          xgboost  raw_and_pca  ...  0.922225  0.921175
far             xgboost  raw_and_pca  ...  0.055291  0.052584
type_ii_err     xgboost  raw_and_pca  ...  0.056469  0.053895
f1              xgboost  raw_and_pca  ...  0.932155  0.932805
mcc             xgboost  raw_and_pca  ...  0.865953  0.867548
roc_auc         xgboost  raw_and_pca  ...  0.932878  0.933640
```

Fig. 5: Metrics values for SVM selected features from PCA + original features, combined with baseline.

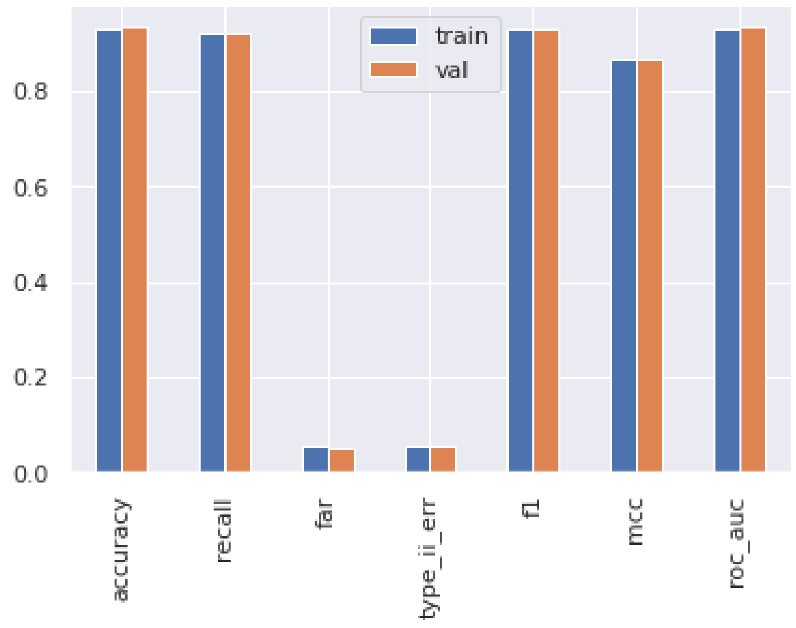


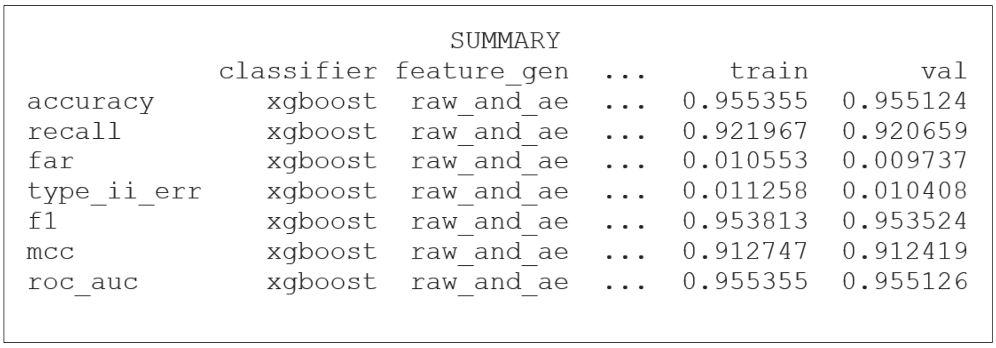
Fig. 6: Bar plot of metric values for SVM selected features from PCA + original features, combined with baseline.

In the next experiment, the number of features will be limited to the most significant ones, based on the weights applied by SVM method. Different number of features has been used, as 50, 25, 15 and

10, comparing their perform, seeing that using the top 10 features the perform decreased dramatically, showing low values for Accuracy, recall, F1 Score and Mcc, and higher for FAR and Type II error. With the top 50 features, obtained the same results than no limiting the number of features selected by default by the SVM technique. When using the top 25 and 15 features, as the results kept similar to the top 50 case, and showing that, between top 10 and top 15 features, there are significant features that provides value, although including them, don't outreach the baseline performance, as in the figure 12.

This shows that, once the PCA generates features, it improves around 1% in relation with the baseline in all metrics, but feature selection doesn't add value, being the best result at this point the use of the top 15 weighted features selected by SVM, but still very similar than applying just PCA. This remarks the good performance of PCA.

After this, has been implemented a different approach, using Autoencoder, as shown in best classification pipelines in [1], [9] and [10]. Autoencoders are capable of finding latent space that are non-linear, otherwise than PCA. Therefore, if the features present some internal non-linear relation, new encoded features generated should improve the performance of the model. The results of using autoencoder to generate features, combined with XGBoost, are very similar than the obtained at this point, what means that they encapsulate the most important implicit relations of the features. Following this path, an experiment has been done using SVM for feature selection from encoder and original features, applying to the top 15 weighted features. This method has produced a certain performance increment according to the evaluation markers, that have improved their values, as shown in figures 7 and 8. MCC metrics jump up by roughly 5% and AUC ROC by 4%.



```
                         SUMMARY
           classifier feature_gen  ...     train       val
accuracy      xgboost  raw_and_ae  ...  0.955355  0.955124
recall        xgboost  raw_and_ae  ...  0.921967  0.920659
far           xgboost  raw_and_ae  ...  0.010553  0.009737
type_ii_err   xgboost  raw_and_ae  ...  0.011258  0.010408
f1            xgboost  raw_and_ae  ...  0.953813  0.953524
mcc           xgboost  raw_and_ae  ...  0.912747  0.912419
roc_auc       xgboost  raw_and_ae  ...  0.955355  0.955126
```

Fig. 7: Metrics values for Top 15 weighted SVM selected features from encoder + original features, combined with baseline (XGBoot).

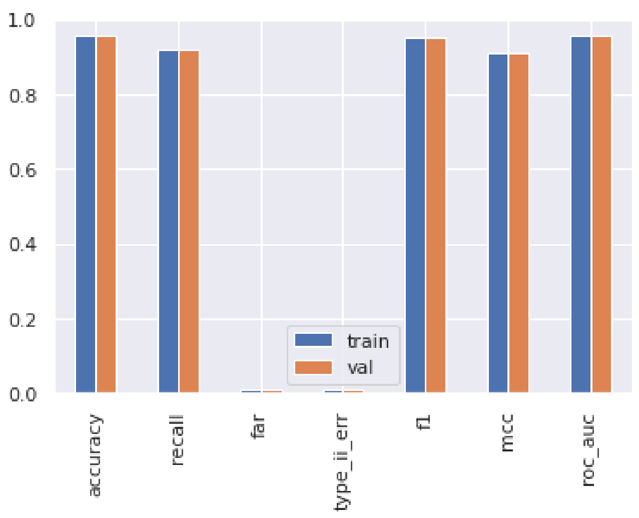


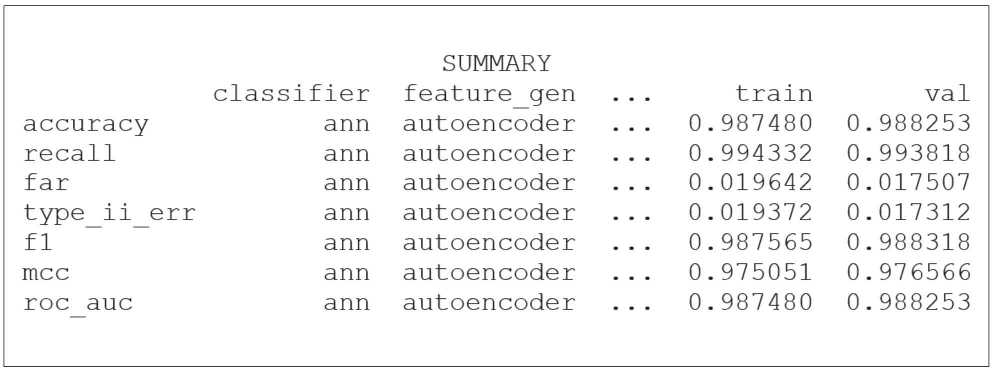
Fig. 8: Bar plot of values for Top 15 weighted SVM selected features from encoder + original features, combined with baseline (XGBoot).

It is remarkable that only 2 encoded features are inside the top 15 SMV selection, but it has been enough to boost the performance of the model. Another fact is that the Type II error and FAR have decreased considerably, what makes the model robust in terms of detection capabilities and false alerts.

The next step in this study, to give a general overview of different methods used for classification, will be to test two other models: Artificial Neural Network, following the deep learning path, and a Support Vector Machine, to tackle one of the most simplistic models available, just in opposition to ANN, in terms of complexity.

ARTIFICIAL NEURAL NETWORK

In this experiment will be used Autoencoder for feature generation, SVM for feature selection and Multi-layer Perceptron as classification method. The combination of encoded features, no feature selection and ANN classification, provides even better results than the previous candidates: MCC 97%, F1 98%, AUC ROC 98%. These results make sense, as the architecture is a simplified version of the most performant IDS architectures, as per [1]. It has been used a simple autoencoder instead of a stacked one, but still delivers impressive results, as can be seen in figures 9. Given these high values, should be taken into consideration that too good results may be caused by overfitting. The presence of some FAR and Type II error may suggest that overfitting may not be happening. In any case, it will be checked when applying to the test data set during the final comparison.

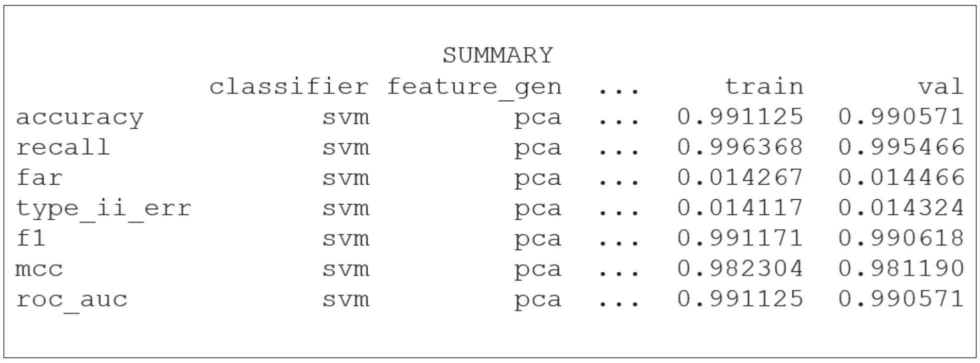


```
                          SUMMARY
            classifier  feature_gen  ...     train       val
accuracy           ann  autoencoder  ...  0.987480  0.988253
recall             ann  autoencoder  ...  0.994332  0.993818
far                ann  autoencoder  ...  0.019642  0.017507
type_ii_err        ann  autoencoder  ...  0.019372  0.017312
f1                 ann  autoencoder  ...  0.987565  0.988318
mcc                ann  autoencoder  ...  0.975051  0.976566
roc_auc            ann  autoencoder  ...  0.987480  0.988253
```

Fig. 9: Table of values for autoencoder + SVM + ANN evaluation markers.

PCA AND SVM CLASSIFIER.

An ensemble algorithm has been tested, performing robustly in almost any kind of domain (XGBoost), and an ANN classifier has been tested as well, showing that is capable of finding very complex relations in the data. Until now, the complexity of the algorithm used has been increased, in terms of architecture and hyperparameters. Hence, just to evaluate a full range of classification methods, instead of trying more complex algorithms, SVM as classifier will be tested. This algorithm, in its linear version creates linear hyperplanes that separate the data into decision regions. It presents very few hyperparameters to tune and very good explicability, therefore, it is a good candidate to always be considered. Along with it, will be used PCA generated features, to continue with the approach of been very simplistic in terms of models used. Results in figure 10.



```
                          SUMMARY
            classifier feature_gen  ...     train       val
accuracy           svm         pca  ...  0.991125  0.990571
recall             svm         pca  ...  0.996368  0.995466
far                svm         pca  ...  0.014267  0.014466
type_ii_err        svm         pca  ...  0.014117  0.014324
f1                 svm         pca  ...  0.991171  0.990618
mcc                svm         pca  ...  0.982304  0.981190
roc_auc            svm         pca  ...  0.991125  0.990571
```

Fig. 10: Table of values for PCA feature generator + SVM classification.

The reason of why SVM Classification method outperforms more complex models, relays in its nature. If SVM is able to set a hyperplane on the data, its performance will be very good. But to achieve this, data relation space should be linear. Otherwise, non-linear kernels shouldn't be used to map the hyperplane to the more complex data representation space. The correct selection of the kernel is a very hard task, and problem specific. In this case, with the data provided, it looks like just a linear kernel is enough. On the other hand, ensemble (XGBoost) and deep learning methods are much more robust in terms of adaptability to the faced problem. Therefore, if approaches using SVM, ANN or

XGBoost present very similar performance, it would be better to use models that do not need domain specific tunning (kernel selection), like SVM.

If False Alarm Ratio and type II error are analysed, the best approach would be XGBoost Classification, combined with encoded and raw features, and SVM as feature selection. This approach would prevail in the case of false alerts or not detected attacks cause significative damage. Usually, improving False Alarm Rate increases Type II error rate and vice versa, therefore could be noticed that, in all the approaches tested, both values are low, meaning that there is a very good robustness overall. Just to give an example of domain models very critical in terms of FAR and type II error, could be image-based cancer detection classification. It is expected higher FAR (more false positives) than Type II errors (cancer cases not detected).

CONCLUSION

Have been applied to all datasets, including the test one, the three final candidate architectures: SVM Classification over 10 PCA features, ANN Classification over 5 AE features, and XGBoost Classification over 5 AE + Raw features and SVM algorithm top 15 selected features. The three approaches present great performance in train and validation set, but when facing test set results, significantly differ. SVM and ANN approaches based in just PCA and Encoded features clearly overfit. They have good performance in train and validation set but very low performance in the test dataset.

On the other hand, XGBoost approach shows great performance over all no matter the data set, showing strong predicting behaviour (ROC AUC over 90 %) and good predictive capability for both positive and negative classes (F1 score over 90%, MCC over 80%). Moreover, FAR is roughly 1.4% and Type II error 2.3% in test dataset, showing robustness in keeping both false negatives and false positives at low level.

To finish, the use of new generated features by an Autoencoder along with the original ones, and a Support Vector Machine as feature selection mechanism, make classification method stronger against overfitting. In appendix 5, is shown the table of all methodologies applied with evaluation metrics.

BIBLIOGRAPHY

[1] M. E. Aminanto , R. Choi, H. C. Tanuwidjaja, P. D. Yoo , K. Kim, "Deep Abstraction and Weighted Feature Selection for Wi-Fi Impersonation Detection", IEEE Transactions on information forencsics and security, Vol. 13, March 2018.

[2] S. M. H. Mahmud, W. Chen, H. JAahan, Y. Liu, N. I. Sujan, S. Ahmed, "iDTi-CSsmoteB: Identification of Drug_Target Interaction Based on Drug Chemical Structure and Protein Sequence Using XGBoost With Over-Sampling Technique SMOTE", April 2019 doi: 10.1109/access 2019.2910277

[3] J. Parashar, Sumiti, M. Rai, "Breast cancer images classification by clustering of ROI and mapping of features by CNN with XGBOOST learning" Maharishi Markandeshwar, Ambala 133207, Haryana, India, September 2020.

[4] X. Y. Liew, N. Hameed, J. Clos, "An investigation of XGBoost-based algorithm for breast cancer classification", University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, United Kingdom, 2021.

[5] J. Wu, Y. Li, Y. Ma, "Comparison of XGBoost and the Neural Network model on the class-balanced datasets", 2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC), 978-1-6654-0605-5/21/$31.00 ©2021 IEEE | doi:10.1109 / ICFTIC 54370.2021.9647373

[6] S. Zhao, D. Zeng, W. Wanga, X. Chen, Z. Zhang, F. Xu, X. Maob, X. Liu, "Mutation grey wolf elite PSO balanced XGBoost for radar emitter individual identification based on measured signals", Journal homepage: www.elsevier.com/locate/ Measurement 159 (2020) 107777, March 2020

[7] Jolliffe IT, Cadima J. 2016 "Principal component analysis: a review and recent developments" Phil. Trans. R. Soc. A 374:20150202. http://dx.doi.org/10.1098/rsta.2015.0202

[8] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," IEEE Commun. Surveys Tuts., 2016.

[9] L. R. Parker, P. D Yoo, T. A. Asyhari, L. Chermak, Y. Jhi, K. Taha, "DEMISe: Interpretable Deep Extraction and Mutual Information Selection Techniques for IoT Intrusion Detection", The 14th ACM International Conference on Availability, Reliability and Security (ARES), 26-29 Aug. 2019, U.K.

[10] S. J. LEE, P. D. YOO, A. T. ASYHARI, Y. JHI, L. CHERMAK, C. Y. YEUN, K. TAHA, "IMPACT: Impersonation Attack Detection via Edge Computing Using Deep Autoencoder and Feature Abstraction", April 2020, Centre for Electronic Warfare, Information and Cyber (CEWIC), dio:10.1109/ACCESS.2020.2985089.

[11] I. Guyon, A. Elisseeff, "An Introduction to Variable and Feature Selection", Journal of Machine Learning Research 3 (2003) 1157-1182 March 2003.

[12] M. E. Aminanto, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, "Weighted feature selection techniques for detecting impersonation attack in Wi-Fi networks," in Proc. Symp. Cryptogr. Inf. Secur. (SCIS), Naha, Japan, 2017.

[13] J. B. K. P. Bennet, M. Embrechts, C.M.B.M.Song, "Dimensionality Reduction via Sparse Support Vector Machines", Journal of Machine Learning Research 3 (2003) 1229-1243, March 2003

[14] A. A. Reyes , F. D. Vaca, G. A. C. Aguayo, Q. Niyaz and V. Devabhaktuni, "A Machine Learning Based Two-Stage Wi-Fi Network Intrusion Detection System", October 2020, Electrical and Computer Engineering Department, College of Engineering and Sciences, Purdue University Northwest, Hammond, IN 46323, USA.

APPENDIX 1: BASELINE REFERENCE

The AWID dataset has been repeatedly studied in several publications, where is mentioned that, from the 156 features of the original AWID dataset, some of them add more information than others Moreover, some features are acting as noise, misleading the procedure to build the model. For example, at [8], is stated that only 20 out of 156 features add value to the model. As the table shown in the figure 5, all the cases except one, undertake dimension reduction. Therefore, seems reasonable to apply a feature generation method that provides dimension reduction as PCA. As explained at [7], this technique reduces the dimensionality of large datasets keeping as much statistical information as possible, and it is widely used for feature classification. At the same time, Autoencoders have been used in other several previous studies, as [9] and [10], delivering good results. Same case for SVM method, that is typically applied for binary classification cases.

The method used as baseline has been Extreme Gradient Boosting, which is a highly optimised ensemble algorithm, with strong feature extraction and classification ability, based on the good results it has provided in previous studies [2], [3], [4], [5] and [6] . This method uses gradient boosting decision tree, creating a linked group of trees from the features, adding them sequentially to generate the sample class, where every tree attempts to reduce the difference from the target and the previous predictions attempts. In [7] is described the application of this method to five different binary classification datasets, four of them balanced and one not, addressing this method as the best option for this kind of classification when compared to PCA and Multi-layer Perceptron Neural Network. Also, according to [2], XGBoost was addressed as the best solution when compared to three different classifiers as Random Forest, Logistic regression and Support Vector Machine. In the mentioned document, these classifiers were applied to four different datasets, which were previously balanced to the application of these methodologies. XGBoost was addressed as the best performer overall, based on the results obtained for AUC ROC metric for four datasets. Finally, XGBoost has been proved as a high-performance classifier when combined with other methodologies as deep learning, as it is mentioned at [4], for binary classification cases.

XGBoost has been trained with modified training data set (cut to 80% of the original file) and applied to the validation data set. It has been setup using no feature generation/selection with Simple XGBoost Classifier, with just 3 estimators/trees with a maximum depth of 3 nodes, having the default booster tree based. Figures 1 and 2 show the results obtained:

Classifier	feature_gen	train	val
Accuracy	xgboost	0.998686	0.998403
recall	xgboost	0.999614	0.999691
far	xgboost	0.002245	0.002893
type_ii_err	xgboost	0.002241	0.002885
f1	xgboost	0.998687	0.998405
mcc	xgboost	0.997374	0.996809
roc_auc	xgboost	0.998686	0.998403

Fig. 1: Evaluation markers values for non-linear XGBoost

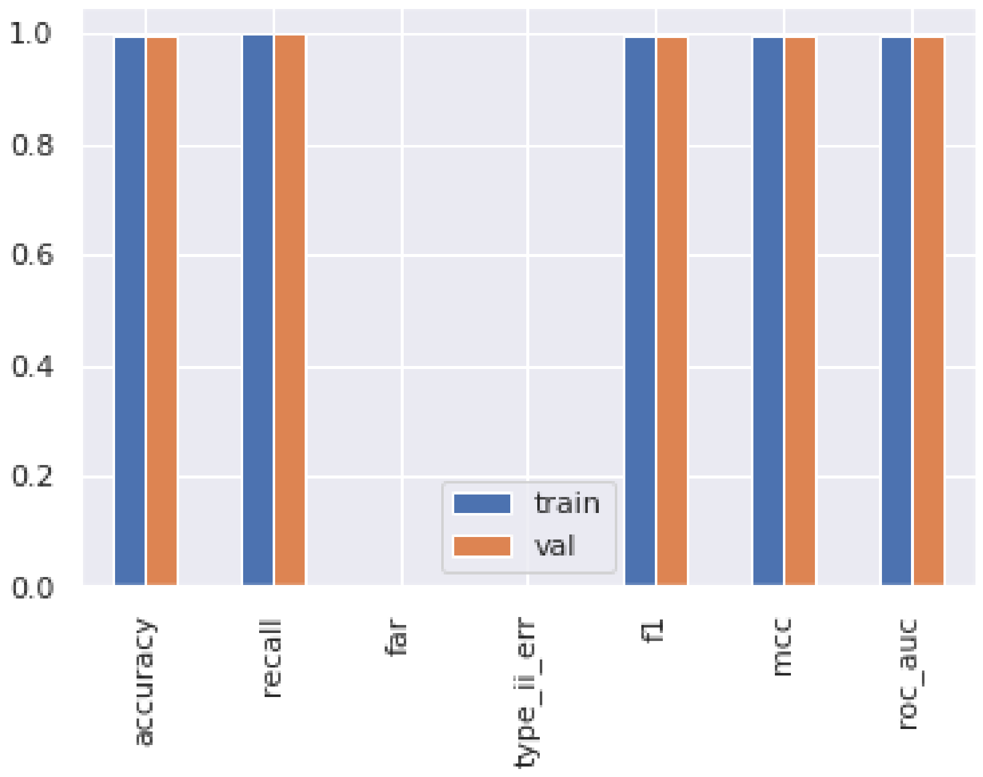


Fig. 2: Evaluation markers bar plot for non-linear XGBoost

According to these results, values obtained are extremely good, looking unrealistic, (F1 & MCC over 0.995). Furthermore, the evaluation markers values for the test dataset are very poor. Assuming that have the same characteristics, the model might be overfitting on the training dataset. As the chosen hyperparameters make the model very simple in terms of architecture, has been considered the use of a booster based in linear functions, obtaining the results at figures 3 and 4.

Classifier	feature_gen	train	val
accuracy	xgboost	0.928267	0.929363
recall	xgboost	0.925625	0.925090
far	xgboost	0.068729	0.065802
type_ii_err	xgboost	0.069092	0.066364
f1	xgboost	0.928076	0.929063
mcc	xgboost	0.856546	0.858757
roc_auc	xgboost	0.928267	0.929363

Fig. 3: Evaluation markers values for linear XGBoost

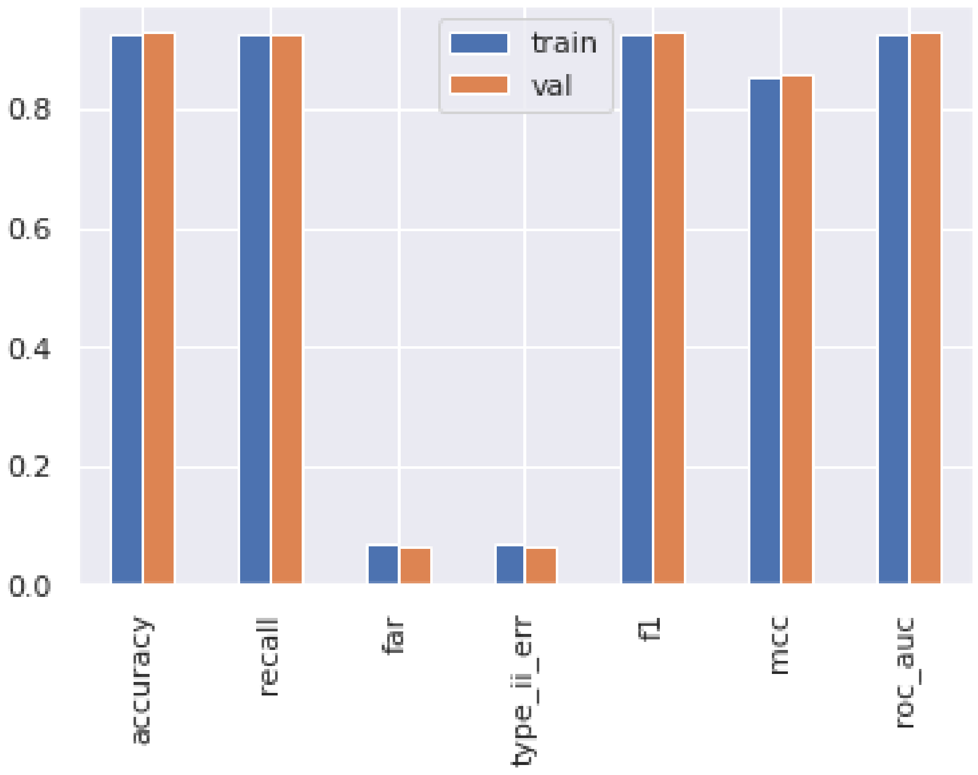


Fig. 4: Evaluation markers bar plot for linear XGBoost

These results are more coherent and seems that the overfitting issue has been resolved by changing from a very basic non-linear setup of XGBoost to a linear booster approach.

Although the results are decent, MCC marker is under 90% and accuracy values are considerably lower than earlier studies, as in figure 5. leaving room for improvement.

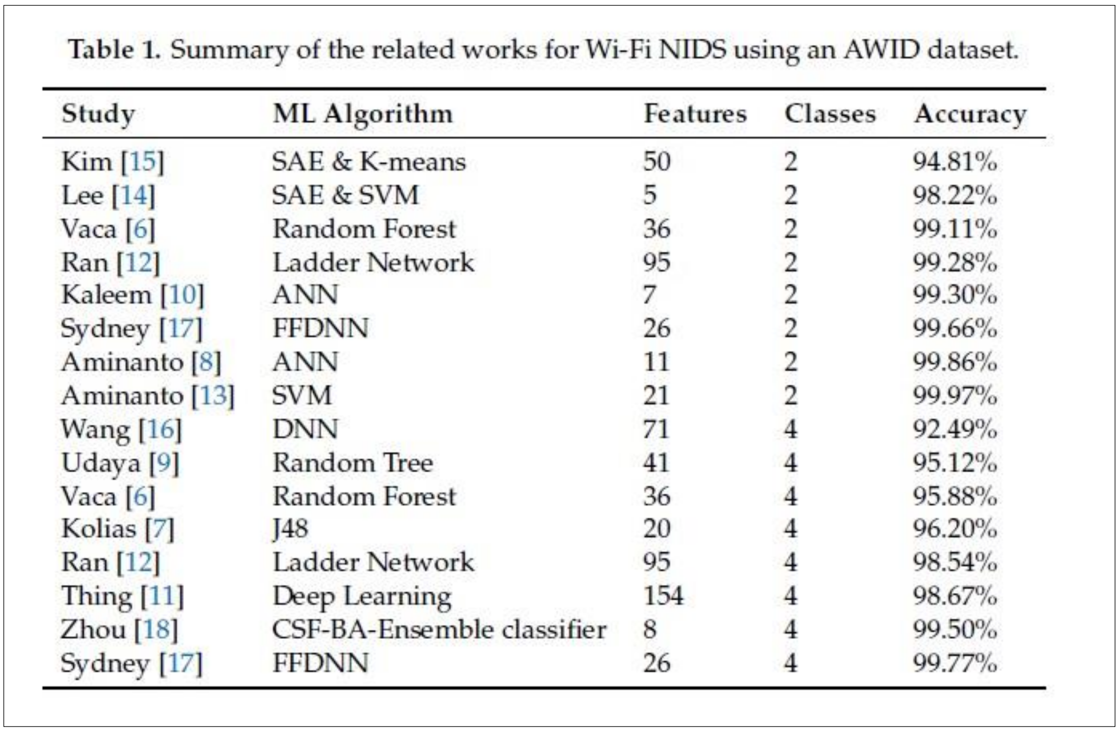


Table 1. Summary of the related works for Wi-Fi NIDS using an AWID dataset.

Study	ML Algorithm	Features	Classes	Accuracy
Kim [15]	SAE & K-means	50	2	94.81%
Lee [14]	SAE & SVM	5	2	98.22%
Vaca [6]	Random Forest	36	2	99.11%
Ran [12]	Ladder Network	95	2	99.28%
Kaleem [10]	ANN	7	2	99.30%
Sydney [17]	FFDNN	26	2	99.66%
Aminanto [8]	ANN	11	2	99.86%
Aminanto [13]	SVM	21	2	99.97%
Wang [16]	DNN	71	4	92.49%
Udaya [9]	Random Tree	41	4	95.12%
Vaca [6]	Random Forest	36	4	95.88%
Kolias [7]	J48	20	4	96.20%
Ran [12]	Ladder Network	95	4	98.54%
Thing [11]	Deep Learning	154	4	98.67%
Zhou [18]	CSF-BA-Ensemble classifier	8	4	99.50%
Sydney [17]	FFDNN	26	4	99.77%

Fig. 5: Comparison of results from several papers. [14]

APPENDIX 2: EVALUATION METRICS

Based on the datasets of this study, this is case of balanced binary classification. Previous studies, as [1], where the definition of the evaluation metrics used in this study can be found, stated that the aim is to reach high values of Accuracy, Recall, Precision, Matthews correlation coefficient (Mcc) and F1Score, while having low values for False Alarm Rate (FAR), CPU time to build the model (TBM), and CPU time to test the model (TT). Additionally, in this study has been applied the marker AUC ROC, that shows the relationship between the True Positives (TP) and True Negatives (TN), showing the capability of the model to predict a right result. It has been included as Type II Error or False Negative rate, to check the capability of the model to miss impersonation attacks.

APPENDIX 3: ANALISYS OF THE DATASETS

These datasets consist in 97044 observations for the training set, while testing set has 40158, with 153 columns where the last one (155) is the target. There are 152 features. The dataset has no null values and it is perfectly balanced. All variables are numeric, having 48 float and 105 integers. The correlation matrix shows that there are a number of features which correlations are undefined, having no variance and giving no value to the model, being expected to be refused by the feature selection methods. Once these features have been removed for this dataset analysis, there are 48 features with float values and 30 features with integer values. 78 features left show correlation between them, in some cases, high positive or negative correlation, as correlation matrix shows in figure 1. Regarding the correlation between features and target, 4 of them are very strong, with an absolute value higher than 0.65, being expected to be selected when performing the feature selection process.

Since the number of observations is high enough, the training dataset has been split into a training and validation datasets (80/20) and has been forced to be balanced, to keep the same characteristics of the original training dataset.

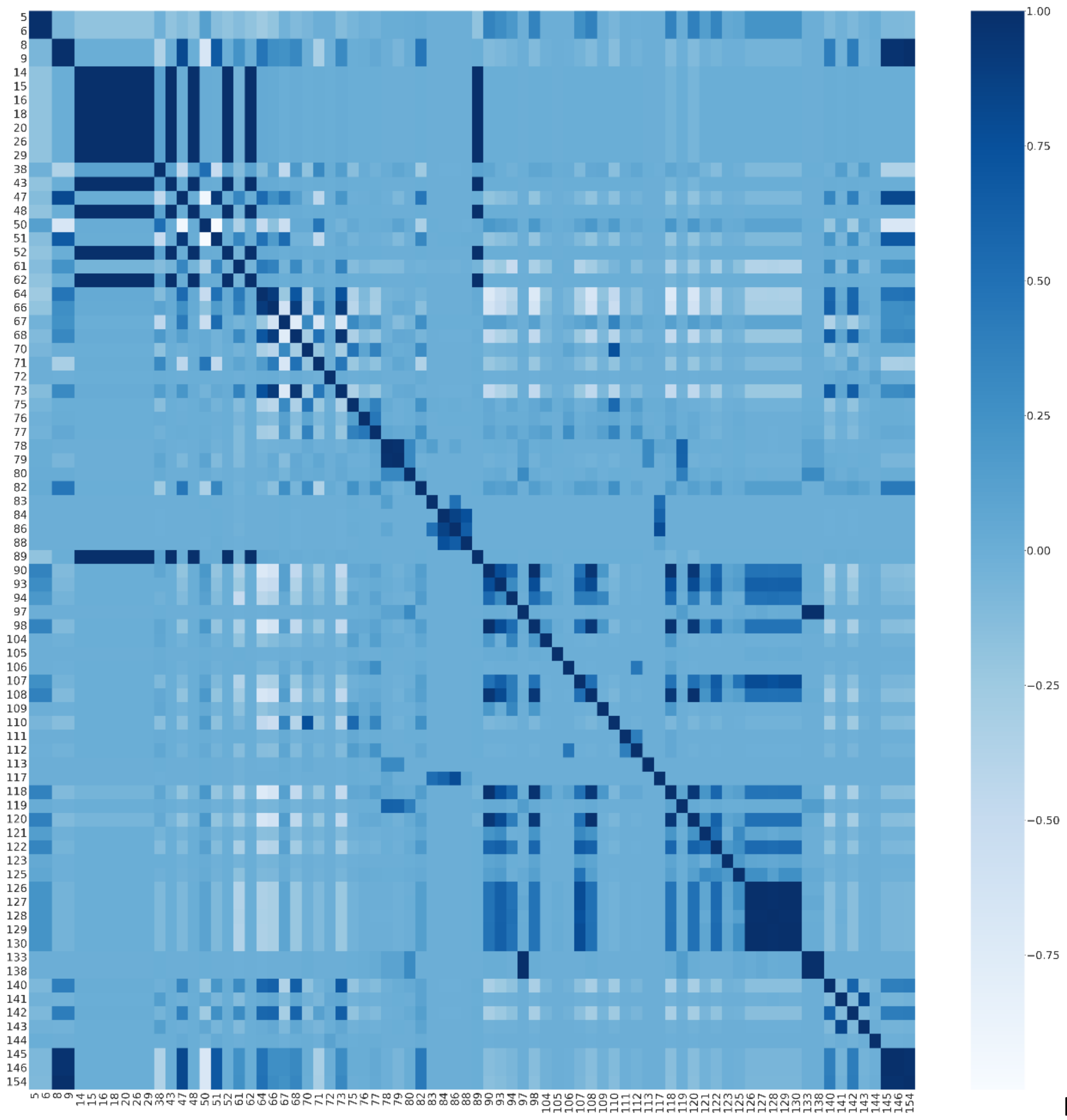


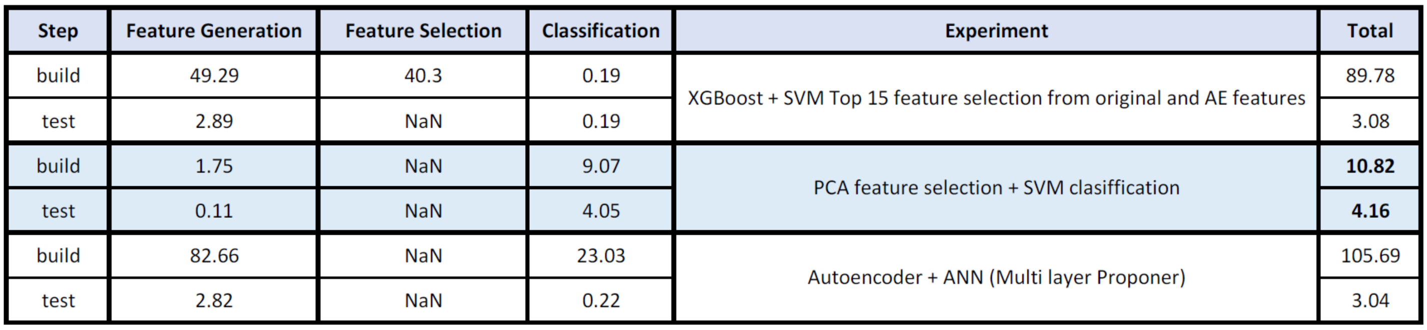
Fig. 1:

Correlation matrix of trained dataset once no variant features removed.

APPENDIX 4: TTB AND TT COMPARISON

As the selected architectures should work in a real environment, very often these scenarios are limited in terms of execution time. Models should be trained and be able to serve predictions as fast as possible. In this line, time to build the model and time to test are metrics to evaluate these aspects. They have been calculated for the candidates with the highest performance based in the evaluation markers, i.e., XGBoost Classifier + Raw & AE generated features + SVM selection method (top 15), SVM Classifier + PCA features, and ANN Classifier + AE features.

As per figure 1, could be told that the introduction of neural networks as feature generation algorithms increase drastically TTB. It was expected, as the process of training Neural Networks is computationally expensive compared to other techniques. It can also be seen that including feature selection phase impacts on building times, but now always is translated into performance improvements.

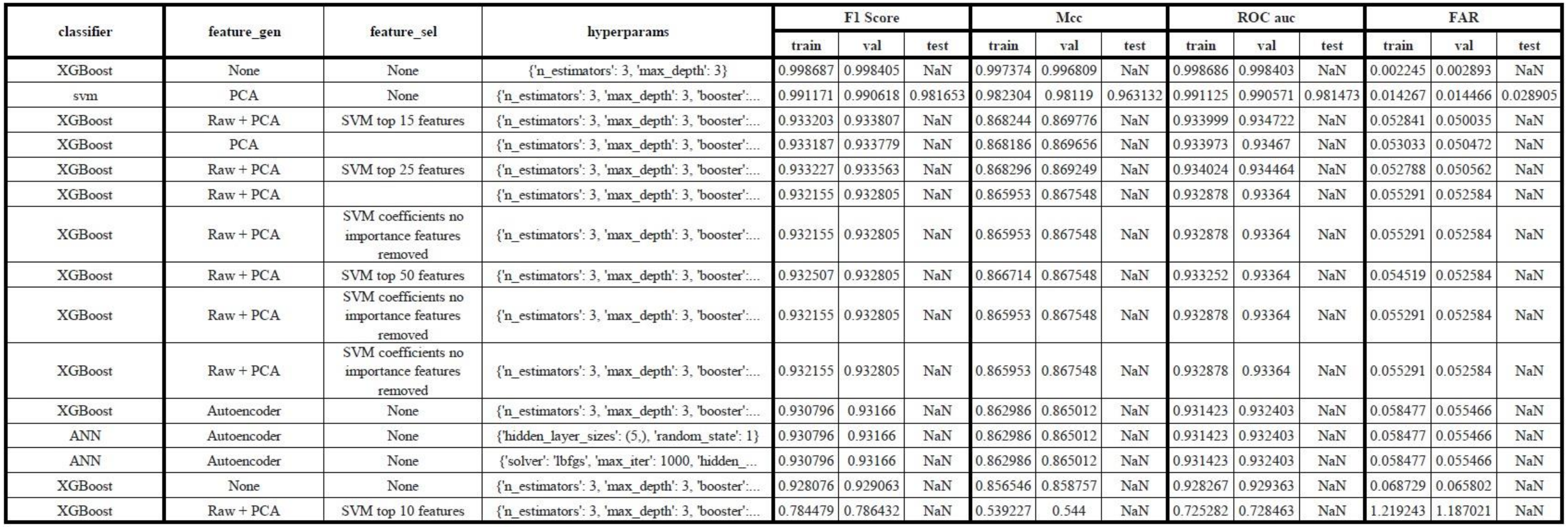


Step	Feature Generation	Feature Selection	Classification	Experiment	Total
build	49.29	40.3	0.19	XGBoost + SVM Top 15 feature selection from original and AE features	89.78
test	2.89	NaN	0.19		3.08
build	1.75	NaN	9.07	PCA feature selection + SVM clasiffication	**10.82**
test	0.11	NaN	4.05		**4.16**
build	82.66	NaN	23.03	Autoencoder + ANN (Multi layer Proponer)	105.69
test	2.82	NaN	0.22		3.04

Fig. 1: TTB and TT comparison for best performance experiments

Can be deducted that the introduction of neural networks, as feature generation algorithms, increases drastically TTB. This was expected as the process of training NNs is computationally expensive compared to other techniques. Can also be seen that including feature selection phase also impacts building times, but not always is translated into performance improvements.

APPENDIX 5: COMPARISON TAB



classifier	feature_gen	feature_sel	hyperparams	F1 Score			Mcc			ROC auc			FAR		
				train	val	test	train	val	test	train	val	test	train	val	test
XGBoost	None	None	{'n_estimators': 3, 'max_depth': 3}	0.998687	0.998405	NaN	0.997374	0.996809	NaN	0.998686	0.998403	NaN	0.002245	0.002893	NaN
svm	PCA	None	{'n_estimators': 3, 'max_depth': 3, 'booster':...	0.991171	0.990618	0.981653	0.982304	0.98119	0.963132	0.991125	0.990571	0.981473	0.014267	0.014466	0.028905
XGBoost	Raw + PCA	SVM top 15 features	{'n_estimators': 3, 'max_depth': 3, 'booster':...	0.933203	0.933807	NaN	0.868244	0.869776	NaN	0.933999	0.934722	NaN	0.052841	0.050035	NaN
XGBoost	PCA		{'n_estimators': 3, 'max_depth': 3, 'booster':...	0.933187	0.933779	NaN	0.868186	0.869656	NaN	0.933973	0.93467	NaN	0.053033	0.050472	NaN
XGBoost	Raw + PCA	SVM top 25 features	{'n_estimators': 3, 'max_depth': 3, 'booster':...	0.933227	0.933563	NaN	0.868296	0.869249	NaN	0.934024	0.934464	NaN	0.052788	0.050562	NaN
XGBoost	Raw + PCA		{'n_estimators': 3, 'max_depth': 3, 'booster':...	0.932155	0.932805	NaN	0.865953	0.867548	NaN	0.932878	0.93364	NaN	0.055291	0.052584	NaN
XGBoost	Raw + PCA	SVM coefficients no importance features removed	{'n_estimators': 3, 'max_depth': 3, 'booster':...	0.932155	0.932805	NaN	0.865953	0.867548	NaN	0.932878	0.93364	NaN	0.055291	0.052584	NaN
XGBoost	Raw + PCA	SVM top 50 features	{'n_estimators': 3, 'max_depth': 3, 'booster':...	0.932507	0.932805	NaN	0.866714	0.867548	NaN	0.933252	0.93364	NaN	0.054519	0.052584	NaN
XGBoost	Raw + PCA	SVM coefficients no importance features removed	{'n_estimators': 3, 'max_depth': 3, 'booster':...	0.932155	0.932805	NaN	0.865953	0.867548	NaN	0.932878	0.93364	NaN	0.055291	0.052584	NaN
XGBoost	Raw + PCA	SVM coefficients no importance features removed	{'n_estimators': 3, 'max_depth': 3, 'booster':...	0.932155	0.932805	NaN	0.865953	0.867548	NaN	0.932878	0.93364	NaN	0.055291	0.052584	NaN
XGBoost	Autoencoder	None	{'n_estimators': 3, 'max_depth': 3, 'booster':...	0.930796	0.93166	NaN	0.862986	0.865012	NaN	0.931423	0.932403	NaN	0.058477	0.055466	NaN
ANN	Autoencoder	None	{'hidden_layer_sizes': (5,), 'random_state': 1}	0.930796	0.93166	NaN	0.862986	0.865012	NaN	0.931423	0.932403	NaN	0.058477	0.055466	NaN
ANN	Autoencoder	None	{'solver': 'lbfgs', 'max_iter': 1000, 'hidden_...	0.930796	0.93166	NaN	0.862986	0.865012	NaN	0.931423	0.932403	NaN	0.058477	0.055466	NaN
XGBoost	None	None	{'n_estimators': 3, 'max_depth': 3, 'booster':...	0.928076	0.929063	NaN	0.856546	0.858757	NaN	0.928267	0.929363	NaN	0.068729	0.065802	NaN
XGBoost	Raw + PCA	SVM top 10 features	{'n_estimators': 3, 'max_depth': 3, 'booster':...	0.784479	0.786432	NaN	0.539227	0.544	NaN	0.725282	0.728463	NaN	1.219243	1.187021	NaN