Individual Practical Project

Student ID Number: [redacted]

Student Name: Farah Adam

Word Count (excluding appendices and bibliography): 2, 083

Module assignment type: Individual Practical Project

Assignment title: Cyber Threat Detection Competition using Machine Learning

School: BEI

Department: Computer Science and Information System

Module Title: Applied Machine Learning

Module SITS Code: BUCI077H7

Abstract

With the rise of the internet, it is vital to use Intrusion Detection System (IDS) and protect networks. IDS can be used to analyse network data and detect anomalies in real-time. For this, we will look at intrusion detection as a threat to a network using the Aegean WiFi Intrusion/threat Dataset (AWID).

Before the creation of the AWID dataset, other datasets were used for intrusion detection in IDS: KDD99 [1], KDDCU99 [2] and NSL-KDD [3]. Nevertheless, some of these datasets, if not all, would no longer be relevant today, are considered outdated and inadequate [2].

The challenges faced by current IDS can be resolved with machine learning (ML) and deep learning (DL) methods [4]. This is because ML-based IDS do not require pre-determined signatures for attacks but rather deduce them via the use of classification or clustering algorithm [5]. This project will construct ML-based models that can distinguish between 'intrusive/ anomaly' and 'good/ normal' traffic. This project will be achieved using the benchmark AWID dataset built by Kolias et al [5]. The dataset that we will use has 152 input features and one target feature.

Introduction

Cyber threat detection is a binary classification problem that categories between anomalies and regular traffic. Our best model would form the basis of an IDS that could effectively detect anomalies in our dataset but could also potentially be used with other datasets. Our dataset should be carefully analysed to create a classification model for anomaly detection. As part of that analysis, we will use a representation learning algorithm, a sparse autoencoder, for feature extraction to improve our models' accuracy. The autoencoder would reduce the dimensionality of the dataset and created 20 new features from there. This dataset would then passed through a series of feature selection that reduced the dataset to 5 features that mostly explained the results. ML models were then built over those five features using Logistic Regression as a base model that gave us an accuracy rate of 93%.

Furthermore, Support Vector Machine algorithm (SVM) was used to separate the regular network traffic from the anomalies. Various SVM kernels were used, which lead to varying results. Those SVM kernels produced varying accuracy rate between 91% and 93%.

Our aim for this project is to develop models that can be relied on to detect anomaly on network traffic. This ML IDS can be used in real-time systems like IoT, self-driving car and more.

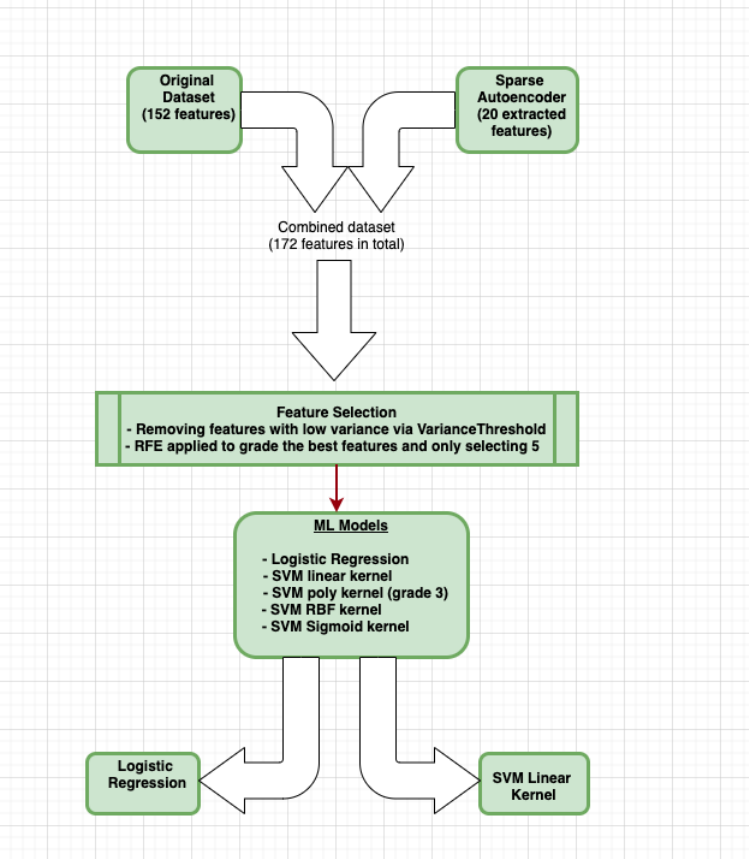
Experimental Evaluation

1- Methodology

Dataset size might cause multiple issues when building ML models, and among those issues, underfitting might be caused by a lack of datapoints in the dataset, among other things. However, having too many features on the dataset might lead to overfitting [6]. In this project, hundreds of features were available, so condensing them into a smaller number of significant features with important dimensions might help the models perform better [6]. So, as a first step, feature extraction was used to extract features that the original dataset did not provide.

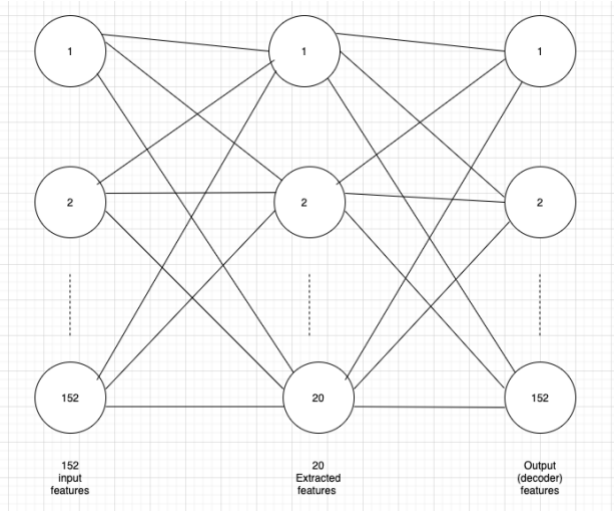
The total number of usable features was then reduced using feature selection by removing features that are not useful to the models. The feature selection block consists of 2 feature selection methods applied one after the other to help reduce the dataset to just the most significant features, thus allowing us to get the optimal number of features, which, in our case, is 5 features as proved by Seo et al (2020) [7]and Parker et al (2019) [8], their best models had only 5 features. Feature extraction and selection are important methods for optimising the learning process of ML models. Feature extraction creates new and more meaningful features, while feature selection removed redundant features [9]. The reasoning behind feature extraction and selection is that some of the dataset's attributes may play a less important role. In contrast, in the worst-case scenario, many of those attributes may behave as noise and thus misdirect the ML models. This demonstrates the importance of feature extraction and selection in creating better features for the ML model. (See appendix for further information on feature selection and extraction)

Figure 1: Individual Practical Project steps



The 20 extracted features were then added to the original dataset. The dataset after feature extraction has 172 features in total.

Figure 2: Sparse Autoencoder used for Feature Extraction



Using feature selection, we can reduce the number of input variables to consider in the classification models to just the most relevant ones. So, our first step is to remove all variables where 80% of the samples contain the same value thus leading to low variance using VarianceThreshold from sklearn. Since a low variance indicates that all the variables are close to each other, so those features have been removed. The variables were then ranked in order of important, thus variables explaining the results variable the most were ranked higher using Recursive Feature Elimination (RFE).

In the final 5 features selected, a number of features came from the feature engineering step using the sparse autoencoder. Thus, proving that feature engineering is an essential step in solving this classification problem.

Normalising the original features before feature extraction using a sparse autoencoder was necessary. The attributes in the dataset have a large difference between the minimum and maximum values. It would be difficult to estimate their variance or even use the dataset like autoencoder to extract feature. So MinMaxScaler from sklearn was used. Once the dataset was normalised, the sparse autoencoder was used to create new 20 features derived from the original dataset. Those newly extracted features have been added to the original training and testing dataset using the concatenate function from numpy. The merged features (total number 172) were then used to select the best features for our problem statement during our feature selection process. Then, those features were fed into our ML models for training and evaluation purposes.

To reduce both the technical and cost complexity of the models and increase accuracy, this selection process locates the most significant features and eliminates redundant ones [7], [9]. The reduction in cost complexity comes from the fact that as an unsupervised algorithm, AE does not require costly labels [9].

This project used Logistic Regression (LR) and various kernel version of SVM to deploy machine learning algorithms that use a reduced number of features that have gone through feature extraction and selection via sparse autoencoder, variance threshold selection, RFE. LR can be applied to classify data into discrete outcomes and as such, can be used in our classification problem as our base model.

We then used various SVM kernels. These were:

SVM kernel 1: Linear

SVM kernel 2: Polynomial (degree: 3)

SVM kernel 3: rbf

SVM kernel 4: Sigmoid

SVM can be used in classification problems by creating a hyperplane between 2 or more classes, in such a way that the distance between the hyperplane and the most adjacent datapoint of each class are as far apart as possible [10]. The polynomial kernel of degree 3 was selected via trial and error, and degree 3 gave the best accuracy result.

Figure 3: Models Summary

Common steps:

- Sparse Autoencoder (20 features extracted)
- Combining original dataset and the 20 newly extracted features (new dataset has 172 features)
- VarianceThreshold used to remove features with low variance
- RFE used to select the 5 most useful features

Models used:

- Logistic Regression
- SVM with Linear kernel
- SVM with Polynomial kernel (degree: 3)
- SVM with Radial Basis Function (RBF)
- SVM with Sigmoid Kernel

2- Results

The dataset has been left with the unbalance between the number of anomalies and normal traffic to mirror real-world case. Having trained and tested the SVM kernel models, they were then assessed by comparing them to the base model Logistic regression as per the table below, which shows that SVM with rbf and sigmoid kernel performed the worse.

Table 1: SVM Kernels VS Logistic Regression results

	Macro Accuracy Average
Logistic Regression	93%
SVM Linear	91%
SVM Poly	90%
SVM RBF	89%
SVM Sigmoid	83%

Other metrics that can be used are: AUC and F1 score, FAR (False Alarm rate), precision, recall, TPR and FPR.

The models were all fitted on the training dataset, and then tested on the test dataset. The results show that SVM with linear and Logistic Regression models were our best models for this classification problem. This shows that the simpler models work best for our intrusion detection problem.

Unfortunately, the speed at which each model run was very different with Logistic Regression running the fastest. Thus, making our 2 best models: Logistic Regression and SVM with Linear kernel.

3- Discussion

Even with similar accuracy among our 2 best models, their confusion matrix displays very different results. The confusion matrix is widely used to assess the efficiency and effectiveness of binary classification ML models [7].

Table 2: Confusion matrix

	Actual values predicted positive	Actual values predicted negative
Predicted values predicted positive	True Positive (TP)	False Positive (FP)
Predicted values predicted negative	False Negative (FN)	True Negative (TN)

Table 3: ML model 1: Logistic Regression confusion matrix

```
[[19603   476]
 [ 2428 17651]]
```

Table 4: ML model 2: SVM with Linear Kernel confusion matrix

```
[[19675   404]
 [ 3886 16193]]
```

Table 5: ML model 3: SVM with poly kernel confusion matrix

```
[[19640   439]
 [ 3997 16082]]
```

Table 6: ML model 4: SVM with RBF kernel confusion matrix

```
[[19684   395]
 [ 4668 15411]]
```

Table 7: ML model 5: SVM with Sigmoid kernel confusion matrix

```
[[19684   395]
 [ 4668 15411]]
```

If, we were to apply our ML IDS to autonomous vehicle to test practical applications, the most significant measures to take into consideration would be the speed of the ML IDS, accuracy and its ability to reduce False positive so a low Precision. Due to security concerns of such vehicle, minimizing false positives (precision) would be more important than minimizing false negatives (recall). This shows that, the selected model should aim to consider those features.

Further measuring metrics to be used are: accuracy (acc), detection rate (DR), false alarm rate (FAR), F-measure (F_1) and Matthew's correlation coefficient (Mcc) with the below calculations [7], [8] and [9].

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

$$DR\,(recall) = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$FAR = \frac{FP}{TN + FP}$$

$$FNR = \frac{FN}{FN + TP}$$

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

$$Mcc = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP+FP)\,(TP+FN)\,(TN+FP)\,(TN+FN)}}$$ [7]

Models with high acc, Mcc and F_1 while FAR is lowest, are the best models [7].

Table 8: Evaluation metrics results

Models	Acc (%)	DR	FAR	F_1	Mcc
Logistic Regression	93	89	2.62	93	86
SVM Linear	89	84	2.43	90	80
SVM poly	89	83	2.66	90	79
SVM rbf	87	81	2.49	87	77
SVM sigmoid	82	89	23	80	65

The further metrics in table 8 are confirming that our best models are Logistic Regression and SVM linear kernel.

Table 9: Evaluation results for LR

```
[[19603   476]
 [ 2428 17651]]
              precision    recall  f1-score   support

           0       0.89      0.98      0.93     20079
           1       0.97      0.88      0.92     20079

    accuracy                           0.93     40158
   macro avg       0.93      0.93      0.93     40158
weighted avg       0.93      0.93      0.93     40158
```

Table 10: Evaluation results for SVM Linear Kernel

```
[[19675   404]
 [ 3886 16193]]
              precision    recall  f1-score   support

           0       0.84      0.98      0.90     20079
           1       0.98      0.81      0.88     20079

    accuracy                           0.89     40158
   macro avg       0.91      0.89      0.89     40158
weighted avg       0.91      0.89      0.89     40158
```

Edge computing allows computation to happen near or closer to the edge of the network [7]. By combining 5G and edge computing, ML models that are able to reduce data dimensionality and efficiently select the most significant features would be essential in IDS when applied to autonomous vehicles. Through the extraction and selection of functions, the data dimensionality needed for training and testing of the model is minimised, thus increasing the performance of the model [7]. This means that feature extraction and selection are an essential part of the ML models.

Binary classification IDS for autonomous vehicle could potentially be implemented using our methodology. In this instance the purpose of the model would be to verify whether images seen on the in-vehicle communications system, Control Area Network (CAN), are real or anomalies [11].

Further research on the suitability of SVM kernel in IDS could done by creating an efficient SVM ensemble as demonstrated on the paper by Liu and Pi, 2017 [12] with the creation of their GTNID-SVM model that combines the polynomial and Radial Basis Function (RBF) kernel with game theory. Further options on our current models would be to include more parameters like C and gamma. (See SVM Appendix for further details on SVM).

Conclusion

SVM with linear kernel and Logistic Regression ML models achieve the best performance in this project compared to our other models. Also, in papers [7] and [8], SVM performed better as well. So, here various SVM kernels were used to identify which one would provide the best result.

In this problem, feature engineering takes on a significant role since a higher percentage of extracted features had been selected during the feature selection process. When using Variance Threshold, a larger number of extracted features were selected than from the original dataset. The same occurred when using RFE as part of the second step of feature selection.

In cybersecurity, there is a growing interest in using ML and DL algorithms to solve anomaly classification problems as can be seen by multiple publications like papers [7], [8] and [2], among others. Those publications show that they have been using different architectures, but they do share the common interest of using large dataset and automatic feature engineering. In this project, it has been shown that LR and SVM kernels have provided a respectable accuracy. Feature engineering is significant [7] so further feature engineering to improve the dataset's quality could help achieve better accuracy and results.

Bibliography

[1] N. Araújo, R. de Oliveira, E. Ferreira, A. A. Shinoda and B. Bhargava, "Identifying important characteristics in the KDD99 intrusion detection dataset by feature selection using a hybrid approach," *2010 17th International Conference on Telecommunications*, Doha, 2010, pp. 552-558. doi: 10.1109/ICTEL.2010.5478852

[2] K. Siddique, Z. Akhtar, F. Aslam Khan and Y. Kim, "KDD Cup 99 Data Sets: A Perspective on the Role of Data Sets in Network Intrusion Detection Research," in *Computer*, vol. 52, no. 2, pp. 41-51, Feb. 2019.
doi: 10.1109/MC.2018.2888764

[3] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," *2015 International Conference on Signal Processing and Communication Engineering Systems*, Guntur, 2015, pp. 92-96.
doi: 10.1109/SPACES.2015.7058223

[4] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah and F. Ahmad, ``Network intrusion detection system: A systematic study of machine learning and deep learning approaches," Transactions on Emerging Telecommunications Technologies, vol. 32, no. 1, Oct. 2020

[5] 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., vol. 18, no. 1, pp. 184_208, 1st Quart., 2016.

[6] J. Grus, “Getting Data,” “Machine Learning,” “Logistic Regression,” *Data science from scratch: First principles with python*. M. Beaugureau, 1st ed., O’Reilly Media, 2015, chap. 9, 11 and 16.

[7] S. J. Lee et al., "IMPACT: Impersonation Attack Detection via Edge Computing Using Deep Autoencoder and Feature Abstraction," in IEEE Access, vol. 8, pp. 65520-65529, 2020, doi: 10.1109/ACCESS.2020.2985089.

[8] L. R. Parker, P. D. Yoo, T. A. Asyhari, L. Chermak, Y. Jhi, and K. Taha, ``DEMISe: Interpretable deep extraction and mutual information selection techniques for IoT intrusion detection," in Proc. 14th Int. Conf. Availability, Rel. Secur., 2019, pp. 1_10.

[9] M. E. Aminanto, R. Choi, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, ``Deep abstraction and weighted feature selection forWi-Fi impersonation detection," IEEE Trans. Inf. Forensics Security, vol. 13, no. 3, pp. 621_636, Mar. 2018.

[10] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali and M. Guizani, "A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security," in *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646-1685, thirdquarter 2020, doi: 10.1109/COMST.2020.2988293.

[11] T. Srivastava, P. Arora, C. Wang and S. Chattopadhyay, "Work-in-Progress: Road Context-Aware Intrusion Detection System for Autonomous Cars," 2018 International Conference on Embedded Software (EMSOFT), Turin, 2018, pp. 1-3, doi: 10.1109/EMSOFT.2018.8537210.

[12] Y. Liu and D. Pi, "A Novel Kernel SVM Algorithm with Game Theory for Network Intrusion Detection," KSII Transactions on Internet and Information Systems, vol. 11, no. 8, pp. 4043-4060, 2017. DOI: 10.3837/tiis.2017.08.016.

[13] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[14] G. James, D. Witten, and T. Hastie, “Support Vector Machine,” *An introduction to statistical learning: With applications in R*. New York, NY: Springer, 2013, ch. 9, pp.337-353.

[15] A. Géron and O’Reilly Media., “Representation Learning and Generative Learning Using autoencoders,” *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems*, R. Roumeliotis and N. Tache, 2th ed., CA 95472, Canada: O’Reilly, 2019, ch. 17, p. 568.

[16] https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/

[17] https://scikit-learn.org/stable/modules/feature_selection.html

APPENDIX A: Logistic Regression (LR)

Logistic Regression (LR) is a supervised classification algorithm.

A binomial LR model (with binary target variables) can be built to forecast the probability of belonging to one class over another for any datapoint, which has been used in this project.

In Scikit-learn, the parameter 'solver' can be setup for LR models. The default value is: 'lbfgs' but this solver parameter can be set to 'liblinear', 'newton-cg', 'sag', and 'saga' [13].

Using Logistic Regression as our base model saves us the computational time and expense.

An alternative approach to classification problem is to use SVM since it uses a hyperplane to separate the data distinctly.

APPENDIX B: SVM Kernels

A Support Vector Machine (SVM) is a supervised Machine Learning algorithm used for regression and classification problems depending on the dataset and problem to be solved.
SVM finds a hyperplane that separates datapoints while sometimes being lenient on the violation of this distinct separation. The created hyperplane separates 2 or more classes so that the distance between the hyperplane and the most adjacent datapoint of each class is as far apart as possible [10].

Some of the benefits of SVM are their capacity and scalability to perform real-time intrusion detection. They also have efficient memory storage since they can create a hyperplane in time complexity O(n2). One of the disadvantages of SVM would be the selection of kernels [10].

Considering that SVM is one of the best 'out of the box' classifiers [14] and it was the chosen classifier model by Seo et al, 2020 [7] and advised as a good IDS ML model by Al-Garadi et all, 2020 [10], we use SVM in our models.

For this individual personal project's problem statement, SVM Has been chosen as the classifier and how it works on binary classification problems is what matters. The main objective of SVM is to find the hyperplane that separate data points that belong to different classes. So, data points that are located to one side of the hyperplane would belong to one class and the data points to the other side would belong to another class which is what is required for this binary classification problem. Support Vectors are the datapoints that would be the closest to the hyperplane, so the orientation and the position of the hyperplane depend on those support vectors [14].

The number of support vectors chosen for the SVM algorithm can be chosen arbitrarily.
In this project, the interest is classification SVM, which can be divided into two main groups:
Group 1 with hard margin where there is no tolerance for misclassification. Group 2 with soft margin where there is a tolerance for misclassification can be set to select a hyperplane that could potentially better generalise to unseen data.

When datapoints are non-linearly separable, SVM kernel tricks solve this issue by transforming the data into higher dimensional space. This trick uses a kernel function to map the datapoints into a higher-dimensional space that should allow a hyperplane to be found [6].

SVM with kernel leads to better classification for the model and better robustness to individual datapoints [6]. For non-linearly separable datapoints, a strong line cannot be found so a kernel trick is necessary. Also, an additional parameter can be set, when using SVM kernel models, called gamma.

This gamma parameter would allow specifying how much the kernel would influence the model. Some of the various kernel that SVM can use are:
Linear

Polynomial (with degree)

Radial Basis Function (RBF)

Sigmoid

Pre-computed

SVM kernels can also be used as part of an ensemble ML model. The kernel trick further develops the feature space for a non-linearly separable dataset, which would be valuable in this project. The kernel trick's limitation is that the more complex the kernel is, the longer it would take for the model to run. So, we will need to keep this limitation to mind when selecting our models.

APPENDIX C: Autoencoder

The Autoencoder (AE) is an unsupervised greedy learning algorithm from the artificial neural networks (ANN) family. Structurally, the autoencoder is composed of an encoder-decoder architecture that aims to learn from the input data. AE learn new valuable insight from the compressed dataset. Thus, confirming the AE value as a feature extraction algorithm [8].

In our project, a sparse autoencoder was used to learn features for our classification task. L1 regularization had been used to make the autoencoder sparse; this helped the sparse AE recognize the distinctive statistical features of our dataset. This allowed our feature extraction step to generate newly learned features with useful insights. Thus, our AE's sparsity element allows it to learn how to represent data efficiently [15].

APPENDIX D: Feature Selection

Feature Selection can be divided into multiple methods that can be grouped into 4 categories:

1- Filter Methods
2- Wrapper Methods
3- Embedded Methods
4- Hybrid Methods

1- Filter Methods:

Filter methods could be:
Information Gain
Chi-Square test
Fisher's Score
Correlation Coefficient
Variance Threshold
Mean Absolute Difference (MAD)
Dispersion Ratio

Here, we will concentrate on variance threshold, the 1st feature selection method used in our project. This was chosen as of 1st step because the variance threshold is a good baseline approach. It allows us to remove features that do not meet our threshold, like low variance since those features would not contain any useful or insightful information. This method's restriction is inherent to filter methods; those methods do not consider any possible relationship between the input and target variables [16]. For our report, we selected a variance threshold of
0.8. * (1 – 0.8), which allowed us features that are either 0 or 1 in 80% of the feature column [17].

2- Wrapper Methods:

Wrapper methods include:
Forward Feature Selection
Backward Feature Elimination
Exhaustive Feature Selection
Recursive Feature Elimination (RFE) [16]

Here, we will concentrate on RFE since this is the methods used in our project. Recursive Feature Elimination (RFE) feature selection is built on the idea that a model is constructed. The best performing features are chosen, or the worst performing features are dropped, repeating that process

over each feature. By recursively considering a smaller set of features, RFE is to be able to select the most significant features for the problem being considered [16].

RFE was used to measure the usefulness of features based on the classifier performance (classifier used here: Logistic Regression). Unfortunately, this is computationally expensive, so the dataset was reduced by using variance threshold first.

3- Embedded Methods:

Embedded methods include:

LASSO Regularization (L1)

Random Forest Importance [16]