**Feature Selection – Alex Bates**

There are various techniques used for selecting features for machine learning models. ‘Filter’ methods select features based on their intrinsic properties. ‘Wrapper’ methods measure the worth of a feature using a classifier performance. Embedded methods are like wrapper methods but where the feature selection is embedded with the learning algorithm. Sometimes, just domain knowledge can be used to pick relevant features [1].

The AWID dataset contains 152 features to consider. Initially, features with zero variance were removed, as they can have no predictive power on the target. This left 78 features for consideration.

Due to the way the pipeline was organised, with different individuals working remotely on different stages, embedded feature selection methods were deemed to be too difficult to implement. Instead, various wrapper methods were investigated to consolidate the features to a reasonable number for the algorithmic stage from the 78 candidates.

One common technique from the academic literature is the use of an auto-encoder to reduce dimensionality in the dataset and generate new features [1]. This technique was not considered initially due to its complexity and the time constraints on the project, but could be investigated in future. Simpler methods, with greater interpretability, were first considered instead.

One key question is to decide how many features to select to train the model. Aminanto et. al [2] found, when training an artificial neural network on the AWID dataset, that there was little improvement in the F1 score from increasing the number of features beyond 10. In light of this, 10 features were selected in the first iteration.

On this first iteration, recursive feature elimination (‘RFE’) with a logistic regression classifier was used as the primary feature selection method. This gave a satisfactory result on several algorithms. The same method with just 5 features gave worse performance, suggesting that the model is not over-fitting due to the number of features. Using a different classifier with RFE resulted in broadly similar features being selected.

The first alternative to RFE considered was to use a randomised decision tree classifier and pick the top 10 features by feature importance. This approach yielded a very different set of features to RFE with only two common features between the approaches. Furthermore, the top two features by feature importance in the decision tree approach were not selected by any of the RFE algorithms tested. The decision tree approach yielded good results when used with a KNN algorithm in the learning phase. However, this set of features did not generally do as well in the learning phase as the features selected through RFE.

The final method considered was Principal Component Analysis. This method has the advantage of specifically capturing the variance in the *entire* dataset. However, by reducing dimensionality and combining features, some interpretability is lost. In our initial analysis (on the dataset using the Robust Scaler approach) the first 5 principal components explained over 98% of the variance in the dataset. On this basis, the top 5 components were selected for the learning phase but the results were not very encouraging and the method was discarded.

The best model evaluated in the project used RFE with 10 features. The final feature selection process can be summarised in the following flow chart:

Filter out zero variance features

152 AWID features

72 features with variance

10 features for learning

Recursive feature elimination with logistic classifier

The 10 features selected in this method were:

* radiotap.datarate
* wlan.fc.retry
* wlan.fc.pwrmgt
* wlan.fc.protected
* wlan\_mgt.fixed.capabilities.preamble
* wlan\_mgt.fixed.capabilities.short\_slot\_time
* wlan\_mgt.fixed.timestamp
* wlan\_mgt.fixed.auth\_seq
* wlan\_mgt.rsn.akms.type
* wlan.wep.key

The feature selection methods adopted have the advantage of simplicity and speed of computation (in both the feature selection and learning phase). A possible next step would be to increase the number of features and complexity of method and assess the impact this has during learning.

Please see the relevant appendix for relevant snippets of python code and a brief explanation of the different feature selection algorithms discussed above.

**References**

[1] Kolias et. al, “Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset”, IEEE Communication Surveys & Tutorials, Vol. 18, No. 1, First Quarter 2016

[2] Parker et. al, “DEMISe: Interpretable Deep Extraction and Mutual Information Selection Techniques for IoT Intrusion Detection” in ACM International Conference on Availability, Reliability and Security (ARES), 26-29 Aug. 2019, U.K

[3] Aminanto et. al, “Wi-Fi Intrusion Detection Using Weighted-Feature Selection for Neural Networks Classifier” in 2017 International Workshop on Big Data and Information Security (IWBIS)

**Appendix – Feature Selection – Alex Bates**

1. **Recursive feature elimination**

Recursive feature elimination is an algorithm that starts by building a model using all attributes to predict the target. It then removes features (removing the least useful features first) and re-builds a model on the remaining attributes. This is repeated until the specified number of attributes is reached.

Python code – recursive feature elimination using a logistic regression classifier with 10 features

# Initiate model

model = LogisticRegression()

# Try 10 features

rfe\_log = RFE(model, 10)

fit\_rfe\_log = rfe\_log.fit(X, Y)

# Create list with names of features

rfe\_log\_features = X.columns[fit\_rfe\_log.get\_support()]

for feature in rfe\_log\_features:

print(feature)

radiotap.datarate

wlan.fc.retry

wlan.fc.pwrmgt

wlan.fc.protected

wlan\_mgt.fixed.capabilities.preamble

wlan\_mgt.fixed.capabilities.short\_slot\_time

wlan\_mgt.fixed.timestamp

wlan\_mgt.fixed.auth\_seq

wlan\_mgt.rsn.akms.type

wlan.wep.key

1. **Feature importance and decision trees**

In this method, a random set of decision trees are fitted to subsets of the data with the results averaged across the trees. The top 10 features by feature importance are selected for the learning phase. Feature importance is calculated as the decrease in “node-impurity” weighted by the probability of reaching that node. Node impurity is a measure of the homogeneity of labels at a node. Nodes of a decision tree that greatly increase the homogeneity of labels imply that that feature has strong predictive power on the target – it is therefore a more important feature.

Python code – extra trees classifier with top 10 most important feature selected

# Initiate model and fit to data

model\_tree = ExtraTreesClassifier()

fit\_tree = model\_tree.fit(X,Y)

# Ensure reproducibility

np.random.seed(999)

# Get feature importances

feature\_importance = fit\_tree.feature\_importances\_

# Join feature importances to feature names and rank by importance

feature\_ranking\_data = {'Feature':X.columns.values, 'Importance': feature\_importance}

tree\_features\_imp = pd.DataFrame(data=feature\_ranking\_data).nlargest(10,'Importance')

# Get list object of the top 10 features

tree\_features = list(tree\_features\_imp['Feature'])

print(tree\_features\_imp)

22 wlan.fc.subtype 0.181461

23 wlan.fc.ds 0.120586

13 radiotap.datarate 0.113596

15 radiotap.channel.type.cck 0.070449

27 wlan.fc.protected 0.064409

28 wlan.duration 0.058927

16 radiotap.channel.type.ofdm 0.056004

2 frame.len 0.047525

11 radiotap.mactime 0.040032

3 frame.cap\_len 0.025655

1. **Principal Component Analysis**

Principal Component Analysis compresses a dataset statistically into its principal components. It captures as much of the variance in the dataset as possible at each principal component. These principal components can be used as features in the learning phase.

Python code – PCA with top 5 principal components selected

pca = PCA(n\_components=5)

fit\_pca = pca.fit(X)

# summarize components

print("Explained Variance: %s" % fit\_pca.explained\_variance\_ratio\_)

print("Total variance explained by first 5 components: %s" % sum(fit\_pca.explained\_variance\_ratio\_))

Explained Variance: [0.80629725 0.12093112 0.02414355 0.0183569 0.01252285]

Total variance explained by first 5 components: 0.9822516621351488