SIT720 Machine Learning

Task 8.3

Michael Rideout
Student Id: s225065259

Introduction

This report details the investigation of network traffic in relation to IoT devices. These devices are a target for malicious attacks, and this work is an analysis of network attack types and supervised models that can classify such attacks.

Task 1A

Model Selection

In Shmuel et al. (2024) they investigate the comparative performance of supervised learning methods on common tabular datasets. Based on their results (Table 4) I selected the top 3 non ensemble, non SVM models that had usable implementations (gplearn was excluded due to this). The models chosen were, Resnet (a deep learning architecture), KNN (K-Nearest Neighbour, an instance based learner) and Logistic Regression (a linear classification model)

Preprocessing Strategies

Several preprocessing steps were applied to the dataset. Those being:

- Removal of missing or 'na' values in the target, we cant process instances with missing targets
- Imputation with mean for numerical missing values, independent variables need to be populated for most models.
- Categorical features had one-hot encoding applied. Most algorithms cannot handle categorical features represented as strings so we transform them into new columns represented by a binary representation for each category class.
- The target class itself was represented as a string so it was encoded to a scalar
- Scaling was applied to numeric features as many linear models require this
- Feature selection was performed to reduce dimensionality and improve likely performance
- The target class was highly imbalanced so SMOTE resampling was performed. After this the dataset was excessively large so a reductive resampling was applied.
- The dataset was split into a train / test set with the ratio of 80 / 20.

Task 1B

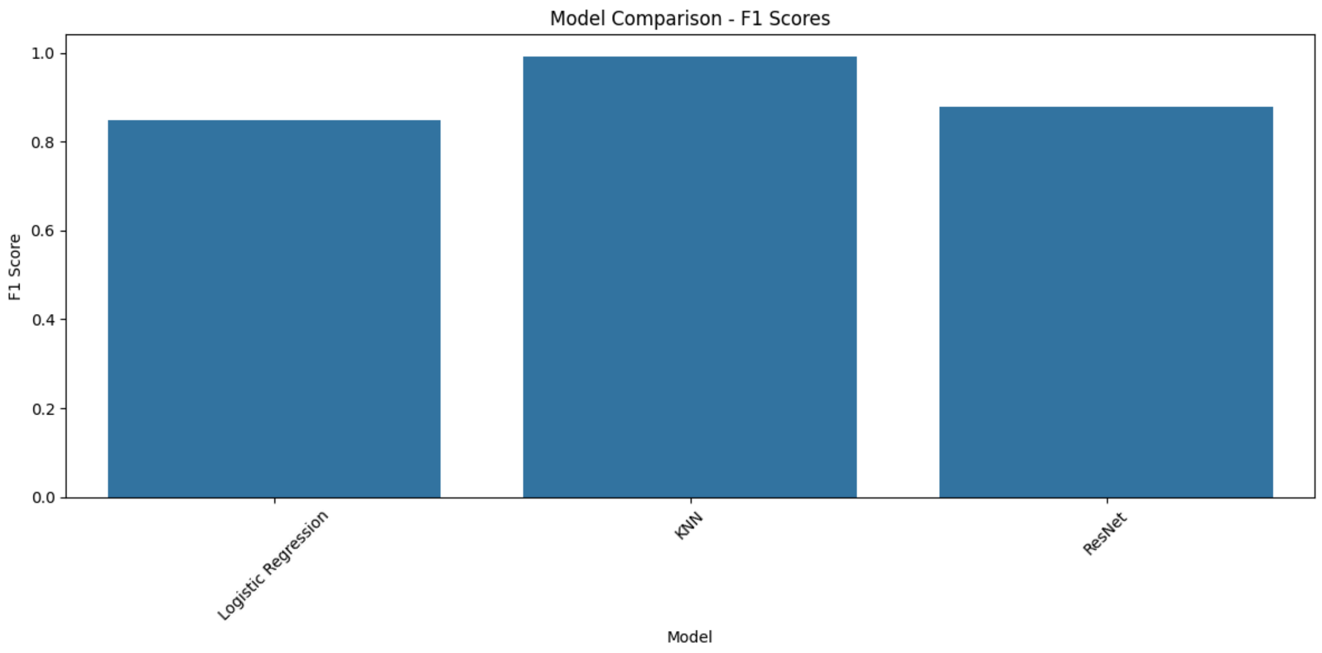
As this was a classification task, the two evaluation metrics used for all models in this investigation were:

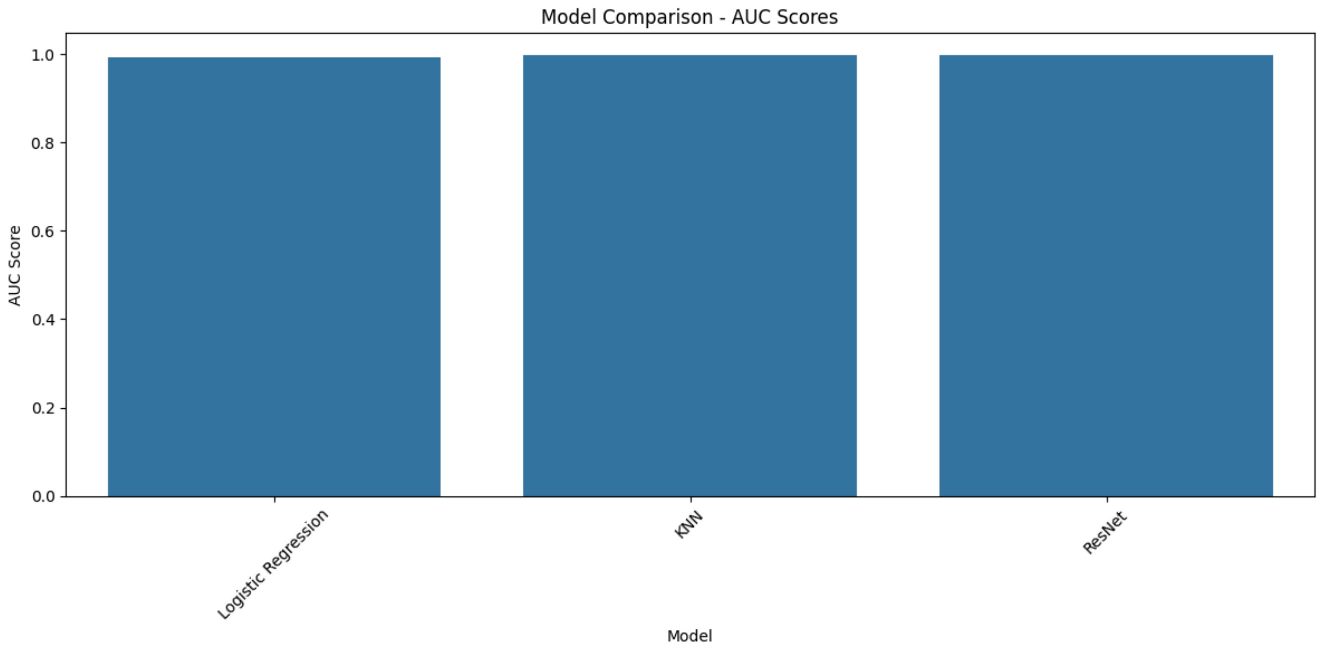
- F1-Score - is a combined metric of both precision (accuracy of true positives) and recall (true positive rate)
- AUC - Area Under the ROC Curve, is a measure of the model's ability to correctly distinguish between classes. A score of 0.5 is not better than a random guess. 1 means the model distinguishes between classes perfectly

The three models were run in two iterations, one using commonly used hyperparameters, the other, optimisation was performed using a grid search for the best hyperparameters. The results were:

Model	**F1-score**	**F1-score Optimised**	**AUC score**	**AUC score Optimised**
Logistic Regression	0.8476	0.8481	0.9930	0.9932
K-Nearest Neighbors	0.9880	0.9911	0.9972	0.9973
ResNet	0.8637	0.8674	0.9934	0.9981

Barcharts for F1 and AUC Scores:





From this it can be seen that all three models performed extremely well on the AUC score both optimised and not optimised. Their F1-scores differ though with LR and ResNet in the mid to high eighties, whilst KNN achieved the highest F1-score of 0.99. KNN is clearly the most performant model.

Task 1C

Each model had their available hyperparameters optimised using a grid search approach. Listed below for each model are the parameters that were optimised, along with their original and new values and an explanation as to the effect of the change

Logistic Regression

Parameter	Original Value	Optimised Value	Reason for Change	Impact on Performance
C	1.0	10	Weaker regularization was slightly more beneficial, suggesting the model might have been underfitting with stronger regularization.	Small uplift in performance

KNN

Parameter	Original Value	Optimised Value	Reason for Change	Impact on Performance
n_neighbors	5	3	Considering fewer, closer neighbors led to better classification accuracy, indicating data points within a closer radius are more important for accuracy.	Improved F1-score
weights	uniform	distance	Giving closer neighbors more influence on the prediction improved performance by prioritizing more relevant data points.	Further improved performance
metric	euclidean	manhattan	Manhattan distance proved to be a better measure of similarity between data points in this specific context.	Better results

ResNet

Parameter	Original Value	Optimised Value	Reason for Change	Impact on Performance
dropout_rate	0.2	0.1	Reducing the level of regularization allowed the model to learn slightly more complex patterns.	Better performance
learning_rate	0.001	0.0005	More refined updates to the model's weights helped the model converge to a slightly better solution and avoid overshooting the local minima.	Better solution
batch_size	1	32	More stable gradient estimates during training and potentially better utilization of hardware resulted in a small uplift in performance.	Small uplift in performance

Task 1D

Logistic regression's performance was good with an F1-score of 0.848 and AUC 0.993, representing a highly accurate (but not the most accurate) model. It isn't a complex model as the linear weights given to features is easily interpretable. Computationally it is efficient to compute. In general LR generalises well due to the simplicity of its algorithm.

KNN's performance in the task was outstanding with a f1-score of 0.988 and an AUC of 0.997. This was the most performant model out of the three classifiers on both metrics. KNN is not an overly complex model as it is simply finding neighbours in a n-dimensional space. There is however computational complexity as an instance has to compute its distance to every other instance in the training set. It is intuitive to understand that data points that are similar are represented closely in the n-dimensional space. Generalisation may be a problem if the features are not scaled correctly.

ResNet's performance was on par to that of LR, with a F1-score of 0.867 and an AUC score of 0.998. Resnet is an ANN (artificial neural network) so it is inherently one of the most complex models in the machine learning toolkit, and due to this, is also one of the least interpretable models. Generalisation can be an issue if the appropriate generalisation parameters are not configured correctly (drop out rate, learning rate and batch size)

Task 2A

The methods chosen to determine the importance of features were:

Random Forest Feature Importance

Random forest works by creating many random decision trees. It derives the importance of features through the decrease in impurity calculated during each tree splitting. It is computationally efficient and can handle nonlinear relationships between variables.

Permutation Feature Importance

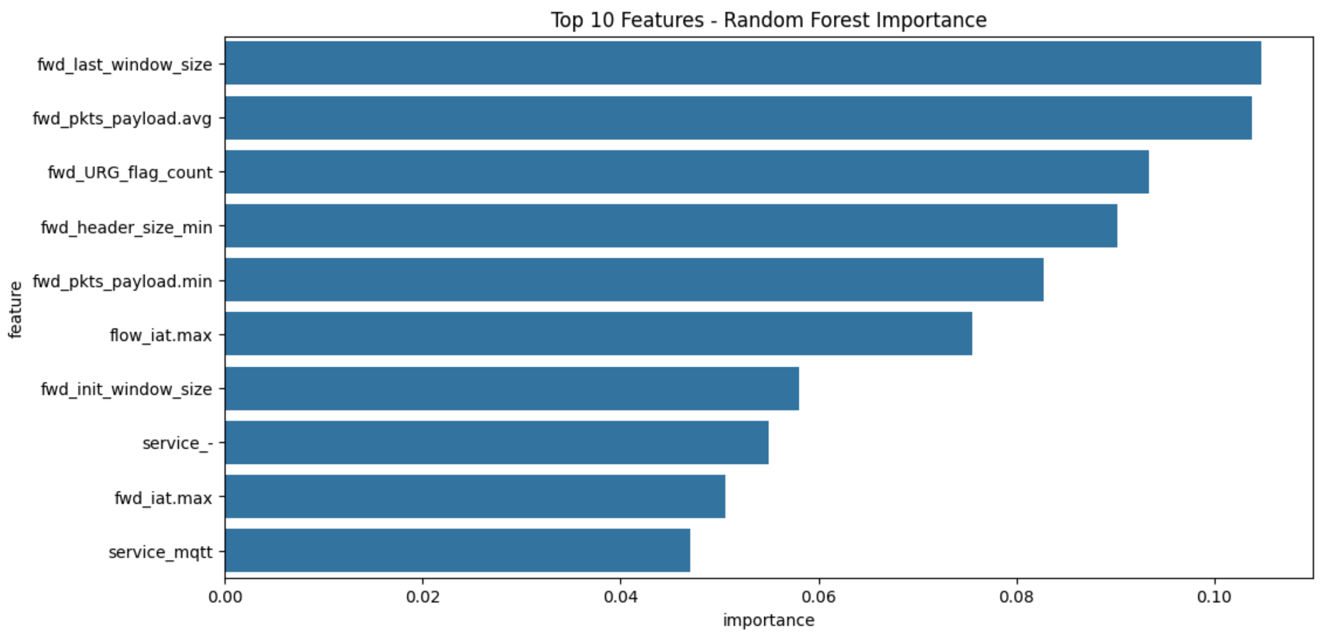
Permutation feature selection works by measuring the performance drop of a model after it has mutated values of a feature and reevaluated the model's performance. It is a simple and intuitive method to determine feature importance, however it is computational intensive, especially if the dataset is highly dimensional.

Consensus Feature Importance

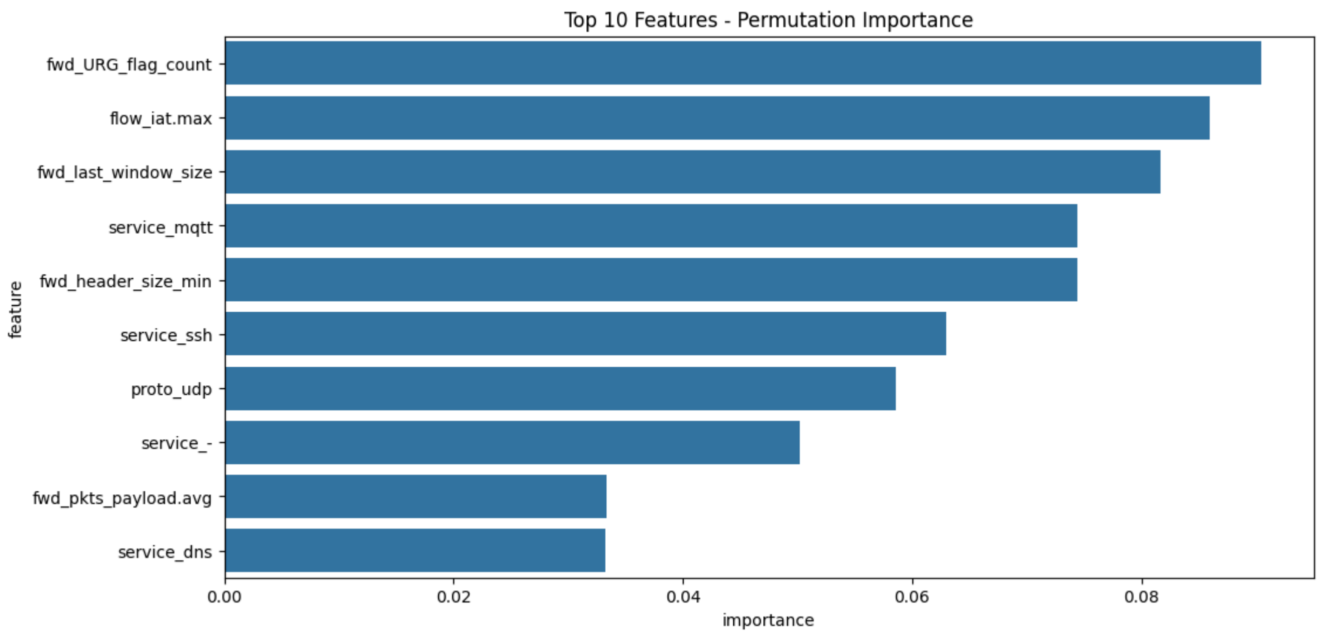
This method combines the results from multiple feature importance ranks into a final rank. It averages the rank scores over all methods. Features with lower ranks are considered to be more important.

Task 2B

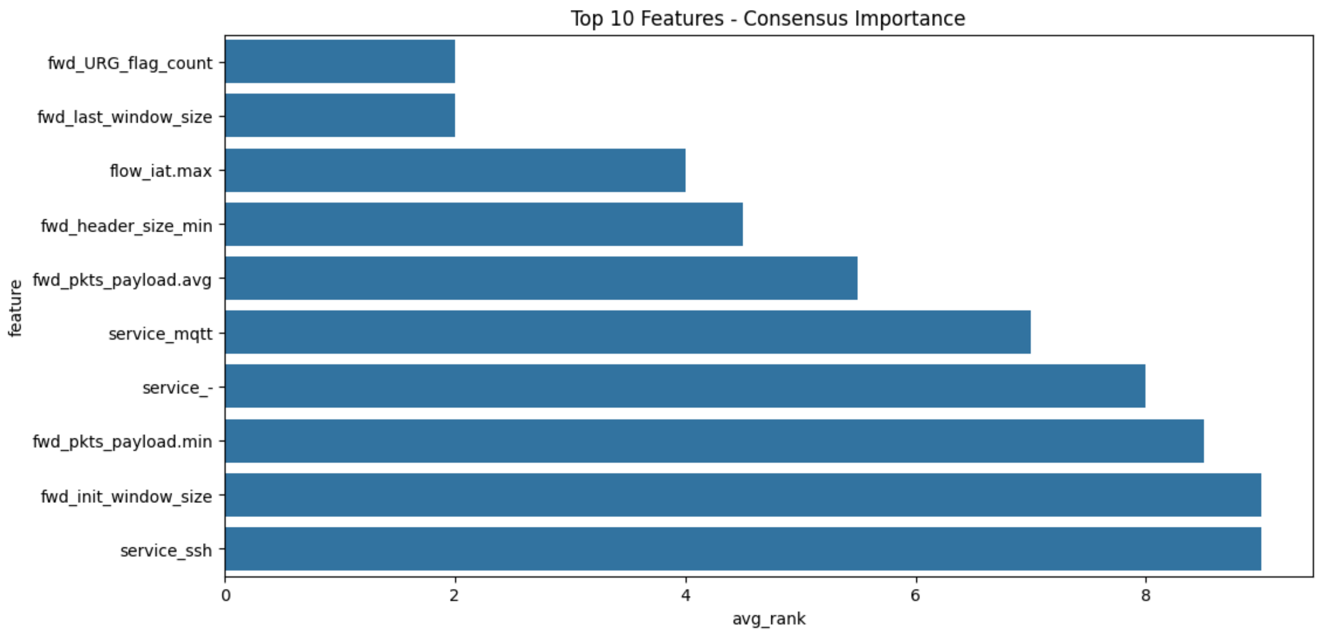
The feature importance results for the random forest method were:



The feature importance results for the permutation method were:



The consensus feature importance method ranking the features as follows:



Consistent Results

The following features consistently appeared in the top 10 important features for both importance methods. The results being:

Feature	Random Forest Rank	Permutation Importance Rank	Consensus (Average Rank)
fwd_URG_flag_count	3	1	2
fwd_last_window_size	1	3	2
flow_iat.max	6	2	4
fwd_header_size_min	4	5	4.5
service_ssh	8	8	8
service_mqtt	10	4	7

The features ‘fwd_URG_flag_count’ and ‘fwd_last_window_size’ were shown to be the two most important items in the dataset according to these importance methods. The remaining features also showed high agreement with their relative rankings between methods.

Divergent Results

The following table shown the features in which there were a stark difference in the results between the two importance methods:

Feature	Random Forest Rank	Permutation Importance Rank	Consensus (Average Rank)
fwd_pkts_payload.avg	2	9	5.5
fwd_pkts_payload.min	5	>10	8.5
service_ssh	>10	6	9
proto_udp	>10	7	>10
fwd_init_window_size	7	>10	9

From this it can be seen that some features are in the top 10 list of important features for one method but not the other. ‘Fwd_pkts_payload.avg’ is ranked the second highest important feature for random forest but is only ranked as 9th for the permutation method.

Task 2C

The process of determining feature importance is a useful tool for guiding feature engineering by allowing investigators to focus on features that offer high predictive value. The use of multiple importance methods offers confidence that features truly do have influence and value to ML models. It can also be a useful tool to identify redundant features or features that may have interactions. It can also allow users to make more informed decisions on dimensionality reduction versus feature enrichment. This process may also influence the model algorithm selection process as it might be prudent to select models that have an advantage over other models for the top features in question. Deployment may also benefit as the effort to acquire all features may be onerous, so focusing on a reduced set may reduce the complexity of the system.

Task 3A

Two ensemble methods were implemented:

Stacking of Existing Base Models

A stacking ensemble which combined the KNN and logistic regression models from Task 1. A stacking classifier uses the outputs from the base models to train a new final classifier (in this case another logistic regression model) which then makes the final predictions. As the KNN base model already had a high f1-score (0.988) it was interesting to see if the ensemble could match this. The f1-score for the ensemble was 0.987 which was very close to that of the KNN classifier.

New Models with Voting Classification

An ensemble model was implemented which trained 2 new models (random forest and SVM) and used the existing LR model. These are very diverse models, one being decision tree based and the others linear. This increases the ability of the model as a whole to generalise. Soft voting is then employed to aggregate and average the class probabilities from the base models.

Task 3B

The F1-score performance of the stacking ensemble method with existing models is:

Model	F1 Score
Logistic Regression	0.848054538473794
KNN	0.9910593371882499
Ensemble 1 (Stacking)	0.9907485045438676

The performance of the ensemble was just as good as the best base model (KNN) but with an added advantage of increasing generalisability. The robustness of the model is also increase as algorithm specific deficiencies are negated

The F1-score performance for the soft voting ensemble method with mostly new models is:

Model	F1 Score
Random Forest	0.9015317972087113
Logistic Regression	0.848054538473794
SVM	0.8415986568578278
Ensemble 2 (New Models)	0.8766077161801293

Although the f1-score of the ensemble (0.877) isnt as high as the random forest base model (0.902) it is higher than the other two base models LR and SVM. As stated above, the generalisability and robustness of an ensemble model is in itself a big advantage. It may be worthwhile taking a slight hit to performance to increase robustness and generalisability.

Task 3C

The decision to use a difficult one, and one that usually comes down to the outcome of a cost benefit analysis of the use of ensemble methods. The use case for developing the model should indicate what are the important aspects for a successful deployment of a model into production.

The following advantages and disadvantages of ensemble methods may help model designers calculate the cost / benefits of such models.

Ensemble Pros

- **Improved Accuracy and Performance** - Ensemble methods usually perform better than individual constituent base models
- **Improve Generalisation -** by employing diverse methods to model data, generalisation is vastly improved and the risk of overfitting reduced
- **Improved Robustness** - Specific issues affecting one model will most likely not affect others that are different in their design. This increases the stability of the entire system.

Ensemble Cons

- **Increased Complexity -** Individual models are complex, but multiple models expand the level of complexity dramatically. This in turn makes it harder to interpret the model's behaviour.
- **Increased Computational Cost -** Having to train multiple models is computationally expensive and may not even be possible on large datasets.

Task 4A

Support Vector Machines (SVMs) are binary classifiers meaning that they can classify only between two classes. In order to handle multiclass classification problems, two methods have been used to train multiple SVMs, then combine the results to produce a single prediction for a multiclass target. The two methods are:

One vs Rest
Here as many classifiers are trained as there are classes in the target. Then at inference time, the class with the highest predictive score is chosen as the target class.

One vs One
This trains a classifier for each possible pair of target classes. At inference time, each class votes for the class it believes is correct, and then the final class is the class with the majority vote.

The scikit learn SVM classifier (SVC) uses the one vs one approach by default.

Task 4B

The predictive performance results of the SVM was for a F1-score of 0.8416 and A AUC score of 0.9810. Whilst this is deemed to be a good performance in general for a machine learning model, it is very much on par with the lesser performing models trained in this investigation (LR, ResNet and Random Forest)
One issue with using SVM for multiclass targets, is there have to be multiple classifiers trained, based on the number of classes in the target. If the target class count is large, using SVM would not scale well.

Task 4C

One limitation with the implementation was that hyperparameter optimisation was not performed. This may have unlocked better performance from the SVM classifier. One limitation observed was the high computational cost of training the model in comparison to other more performant models (sometimes 10x larger). Another limitation of SVM is it lacks interpretability in how it defines boundary separations in n-dimensional space.

In summary, with this dataset, it is advisable to use a simpler algorithm that has achieved a better performance (such as KNN) or logistic regression which achieves almost identical performance with improved interpretability and lower computational cost.

Conclusion

Through this investigation, multiple machine learning models were trained that were able to achieve good performance in classifying the category of network traffic to IoT devices. The comparative performance of all models is as follows:

Model	F1 Score	AUC Score
KNN	0.9911	0.9973
Ensemble 1 (Existing)	0.9907	0.9997
ResNet	0.8773	0.9969
Ensemble 2 (New)	0.8766	0.9986
Logistic Regression	0.8481	0.9932
SVM	0.8416	0.9810

Reference

Shmuel, A., Glickman, O., & Lazebnik, T. (2024). *A comprehensive benchmark of machine and deep learning across diverse tabular datasets*. arXiv. https://arxiv.org/abs/2408.14817