Intrusion Detection System in IoT Devices

Descrição e motivação do projeto

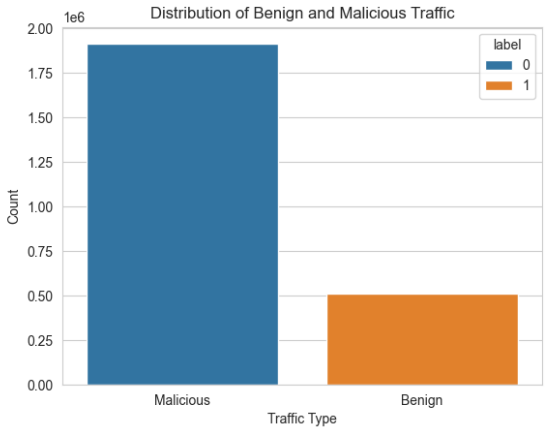
Este projeto foi concebido como parte integrante da unidade curricular de Aprendizagem Aplicada à Segurança do Mestrado em Cibersegurança. O seu foco abrange a implementação de um Sistema de Deteção de Intrusões (IDS) utilizando técnicas de *machine learning*, representando uma oportunidade valiosa para a concretização dos conceitos e habilidades adquiridos ao longo do curso. Com uma abordagem supervisionada, o projeto abraça **duas vertentes de classificação**:

- **Classificação binária**, destinada a discernir entre tráfego considerado normal e atividades suspeitas.
- **Classificação multiclasse**, direcionada à identificação e categorização de diversos tipos de ataques.

Esta abordagem não só reforça a base teórica adquirida, mas também evidencia a aplicação prática desses conhecimentos, conferindo ao sistema a capacidade adaptativa necessária para uma deteção eficaz de intrusões cibernéticas.

Processo de coleta de dados

O conjunto de dados usado é denominado de "*IoT Dataset for Intrusion Detection Systems (IDS)*". Foi criado por Azal Hawas e está disponível no *Kaggle*. Este conjunto de dados visa auxiliar os investigadores e profissionais no domínio da cibersegurança a desenvolver e melhorar os sistemas de deteção de intrusões (IDS) para redes IoT.
Contém dados de tráfego de rede recolhidos de uma rede IoT, incluindo tráfego normal e anómalo. O conjunto de dados inclui mais de dois milhões de registos, com 27 características para cada registo.
O *dataset* encontra-se *labeled*, o que significa que cada registo está associado a uma *label* que indica se o tráfego de rede é normal ou anómalo. O tráfego anómalo inclui dois ataques *botnet* (Mirai e Gafgyt).



A sua distribuição é desequilibrada, com mais registos maliciosos em comparação com registos normais (aproximadamente 3,75x mais).
É fornecido num formato CSV e pode ser facilmente carregado em várias ferramentas de análise de dados e de aprendizagem automática para análise posterior. Também inclui uma descrição pormenorizada de cada característica, bem como algumas análises preliminares e a visualização dos dados.

Aplicação de ML nos dados

Assim como foi referido anteriormente, foram desenvolvidas duas implementações usando *machine learning* (com classificação binária e outra multiclasse). Em ambas as situações, o projeto encontra-se dividido nas seguintes fases:

- Recolha de dados e pré-processamento;
- Construção e ajuste dos modelos;

Para a classificação binária, foi feita uma normalização usando Min-Max, ou escala linear, ajustando os valores para um intervalo [0,1]. A fórmula é dada por:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Isto é feito de modo a obter uma convergência mais rápida nos modelos, a melhorar a interpretação e generalização dos dados e evitando possíveis problemas numéricos.
Depois, foi feita a construção do modelo usando a técnica *Grid Search CV*, de modo a encontrar os melhores hiperparâmetros para um conjunto de modelos. Esta técnica recebe uma tabela de modelos com os seus possíveis parâmetros e constrói um modelo para cada combinação de valores, para treinar o modelo esta usa *cross-validation* (dividindo o treino e test em subconjuntos).

Os modelos usados para esta construção foram:

1. *Logistic Regression*: Porque é simples, interpretável e computacionalmente eficiente, e funciona bem quando a relação entre as *features* e a variável alvo é aproximadamente linear.
2. *Random Forest*: Porque é resistente ao *overfitting* (ajuste em excesso aos dados), lida bem com a não-linearidade e pode capturar relações complexas nos dados.
3. *XGBoost*: Porque é altamente preciso, lida bem com a não-linearidade e com as interações entre *features*.

Depois disso, foram comparados os modelos usando a técnica de *McNemar*.

Para a classificação multiclasse, o objetivo é atribuir uma classe a uma instância de dados entre três ou mais classes possíveis. Na normalização é usado o Standart Scaler do scikit-learn, esta consiste em transformar os dados de cada variável tenha uma média zero ou um desvio padrão unitário.

$$z = \frac{X - \text{média}(X)}{\text{desvio padrão}(X)}$$

Este processo ajuda a remover a escala dos dados tornando-os comparáveis e facilitando o treino.
Deste modo, foi realizado o treino nos modelos:

- *Logistic Regression*: É relativamente simples e eficiente, especialmente quando a relação entre as variáveis é linear. Além disto fornece uma probabilidade associada a cada classe;
- *Random Forest*: É resistente a *overfitting* e lida bem com características complexas e não lineares. Não requerendo também um grande ajuste nos parâmetros;
- *Decision Tree*: É um algoritmo que é bom na sua interpretabilidade, facilidade de implementação e capacidade de lidar naturalmente com variáveis categóricas.

Foi também necessário binarizar as classes, para tentar lidar com problemas de classificação multiclasse, utilizando o *LabelBinarizer* ou o *OneHotEncoder*. Ou seja, transforma informação labeled numa matriz binária indicando a presença ou a ausência de cada classe.

Resultados e Discussão

Nesta fase, foi avaliada o desempenho dos modelos em ambas as implementações (binária e multiclasse). Para isso, foram utilizadas as seguintes métricas:

A **matriz de confusão** fornece uma visão detalhada sobre o comportamento do modelo, mostrando os erros cometidos pelo mesmo. Através da mesma podemos obter **precisão**, **sensibilidade**, **F1 score**.

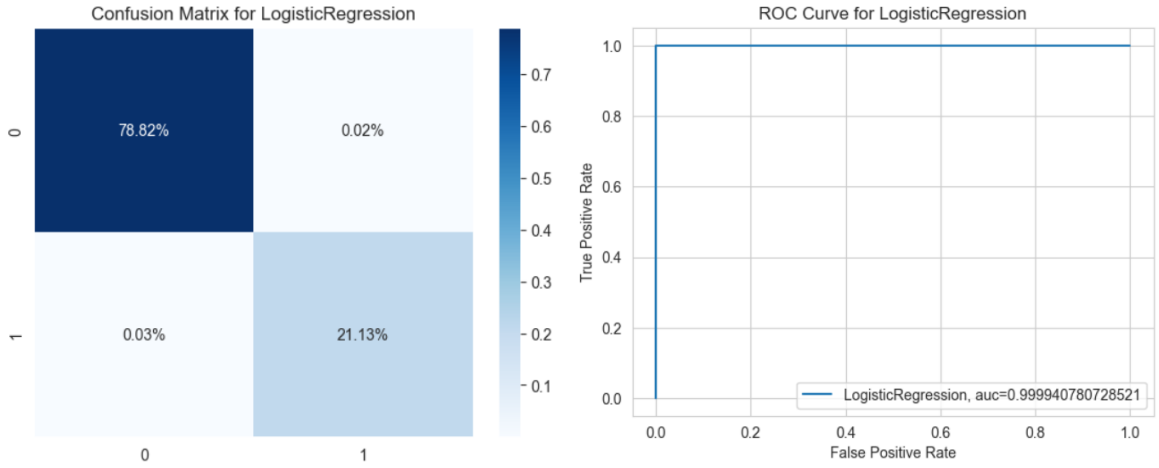
A **Curva ROC** permite avaliar a capacidade de decisão do modelo. É um gráfico da taxa de verdadeiros positivos (TPR) contra a taxa de falsos positivos (FPR). É uma boa métrica quando o custo dos falsos positivos e falsos negativos é semelhante.

A área sob a curva (**AUC**) é uma medida de quão bem um parâmetro pode distinguir entre dois grupos.

Feature importance, permite-nos quais são as *features* mais úteis para a previsão do resultado.

Estes foram os resultados obtidos na **classificação binária usando *Grid-Search***:

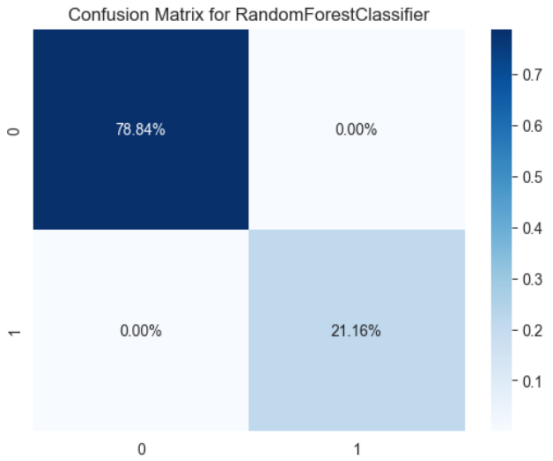
- **Logistic Regression (C: 100, max_iter: 1000):**

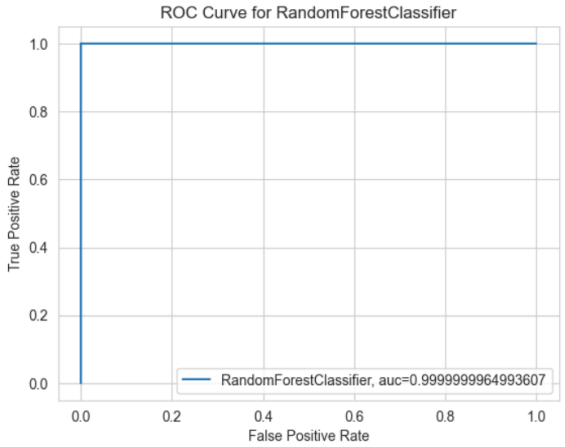


Classification report for LogisticRegression:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	573924
1	1.00	1.00	1.00	154049
accuracy			1.00	727973
macro avg	1.00	1.00	1.00	727973
weighted avg	1.00	1.00	1.00	727973

- **Random Forest ('max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200):**

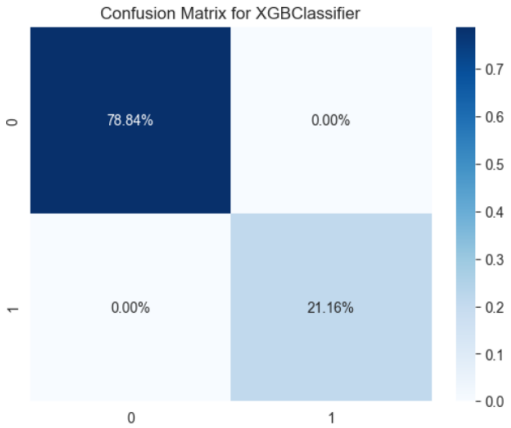


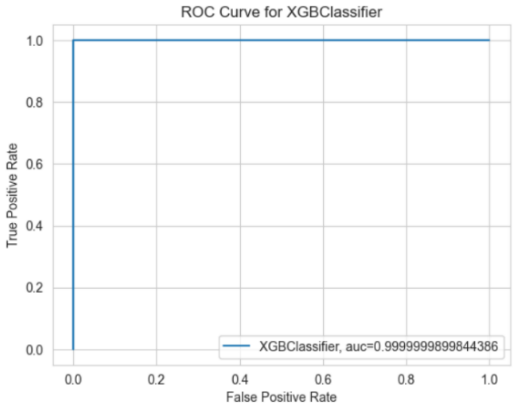


Classification report for RandomForestClassifier:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	573924
1	1.00	1.00	1.00	154049
accuracy			1.00	727973
macro avg	1.00	1.00	1.00	727973
weighted avg	1.00	1.00	1.00	727973

- **XGBoost ('gamma': 0, 'learning_rate': 0.1, 'max_depth': None, 'n_estimators': 200, 'reg_lambda': 0):**





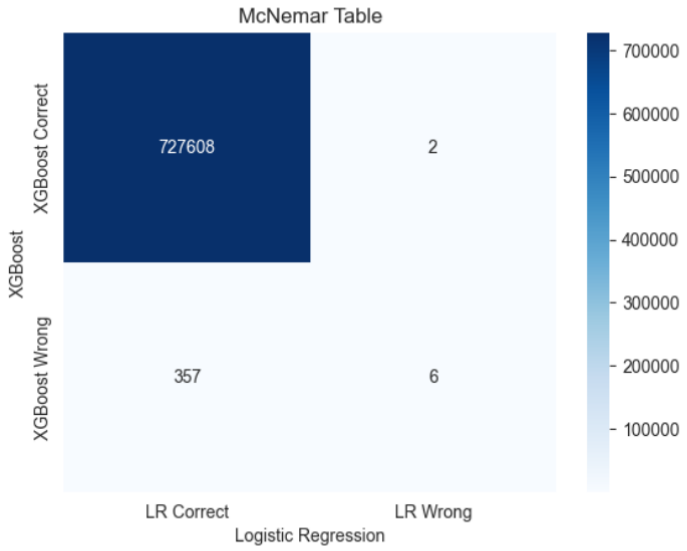
Classification report for XGBClassifier:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	573924
1	1.00	1.00	1.00	154049
accuracy			1.00	727973
macro avg	1.00	1.00	1.00	727973
weighted avg	1.00	1.00	1.00	727973

Como podemos ver, todos os modelos tiveram um bom desempenho, com valores entre 99 e 100 por cento. Devido ao tamanho do conjunto de dados, quase todas as estatísticas são muito elevadas e a curva ROC é quase perfeita. No entanto, o classificador *Random Forest* levou muito mais tempo para ser treinado.

Portanto, nesse caso, comparámos os modelos de *Logistic Regression* e *XGBoost* (com os parâmetros encontrados anteriormente como os melhores) para verificar se há uma diferença significativa entre os dois modelos. Obtendo a seguinte tabela *McNemar*:

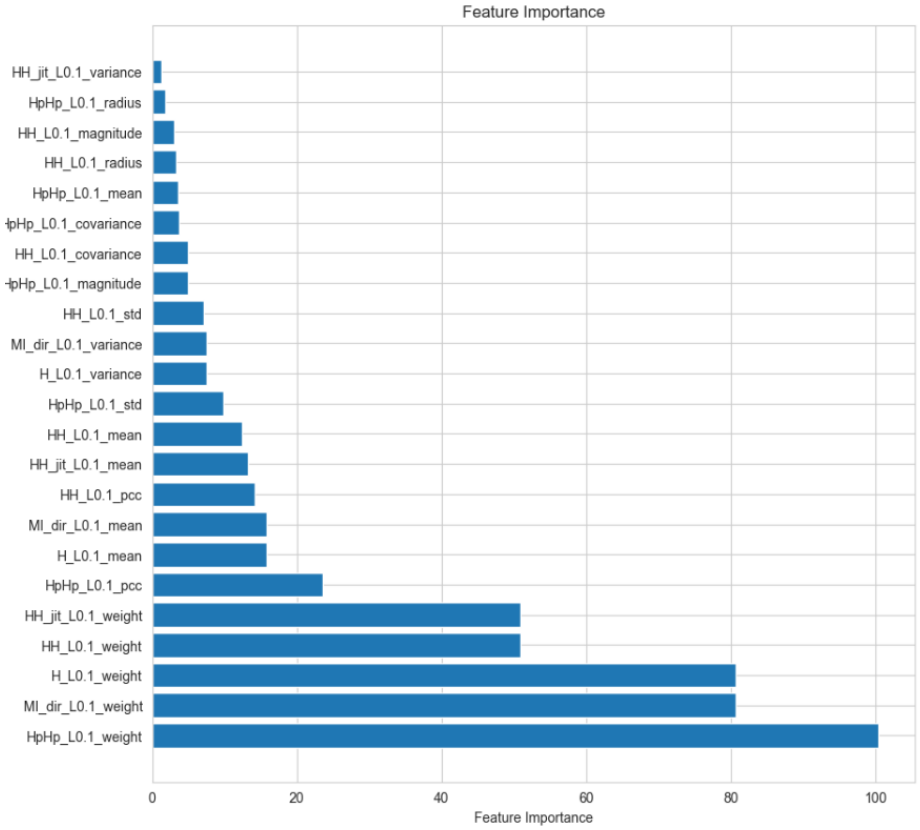
```
Logistic Regression train loss: 0.0025261803204122287
XGBoost train loss: 7.628906473595571e-06
```




```
McNemar's Test Statistic: 150.0
p-value: 0.0021612733742205708
There is a significant difference in the performance of the two models.
```

Como podemos ver, a perda do treino do modelo de *Logistic Regression* (0.25%) é superior à perda do treino do modelo *XGBoost* (0.00076289%), significa que o modelo *XGBoost* é **mais preciso** do que o modelo de *Logistic Regression*.

O valor de p é **inferior** ao nível de significância (0.05), pelo que podemos rejeitar a hipótese nula e concluir que existe uma diferença significativa entre os dois modelos. E a partir da tabela de contingência, o modelo de *Logistic Regression* tem mais **previsões corretas** do que o modelo *XGBoost*.



Após determinar que o *Logistic Regression* era o modelo mais adequado para a implementação binária, foi calculado a f***eature importance*** do mesmo, obtendo os seguintes valores.

Sendo as **cinco melhores** *features*:

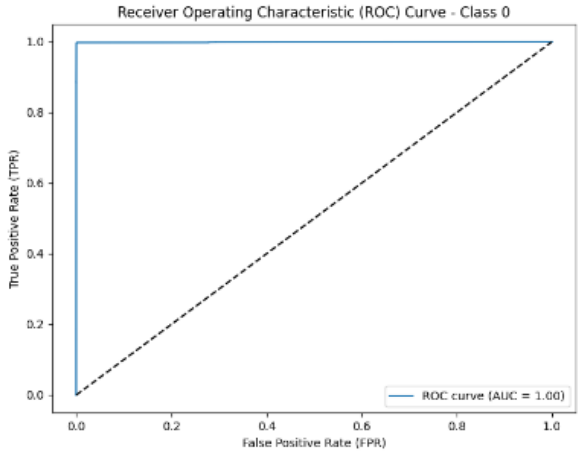
- **HpHp_L0.1_weight**: Entropia condicional para uma ligação direta com um *lag* de 0.1 (peso).
- **MI_dir_L0.1_weight**: Informação mútua para uma ligação direta com um *lag* de 0.1 (peso).
- **H_L0.1_weight**: Entropia para uma ligação direta com um *lag* de 0.1 (peso).
- **HH_L0.1_weight**: Entropia conjunta para uma ligação direta com um *lag* de 0.1 (peso).
- **HH_jit_L0.1_weight**: Entropia conjunta para uma ligação direta com um *lag* de 0.1 (peso).

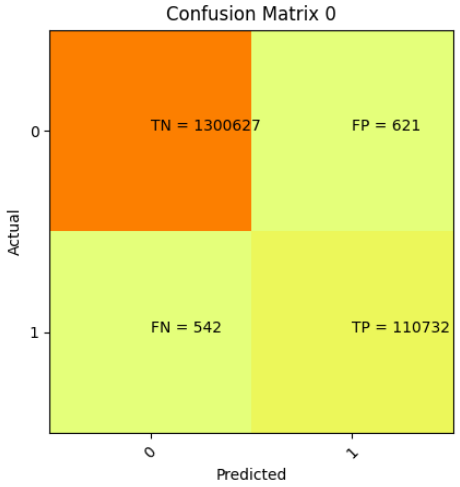
Estes foram os resultados obtidos na classificação multiclasse:

- # Logistic Regression(max_iter=400):

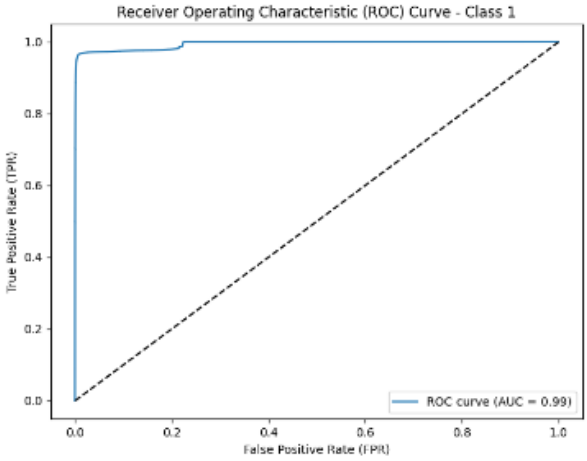
	precision	recall	f1-score	support
Normal	0.99	1.00	0.99	110884
ack	0.95	0.95	0.95	129528
combo	0.83	0.81	0.82	103035
junk	0.64	0.67	0.66	52352
scan	0.99	0.99	0.99	158083
syn	1.00	1.00	1.00	146279
tcp	0.03	0.00	0.00	171995
udp	0.67	0.95	0.79	435721
udpplain	0.78	0.71	0.75	104645
accuracy			0.81	1412522
macro avg	0.77	0.79	0.77	1412522
weighted avg	0.73	0.81	0.76	1412522

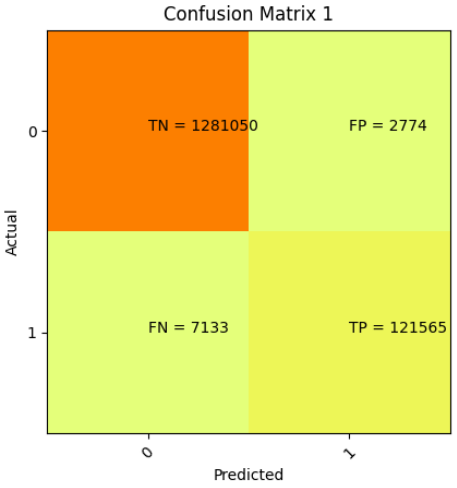
- **Normal (Não houve ataques)**



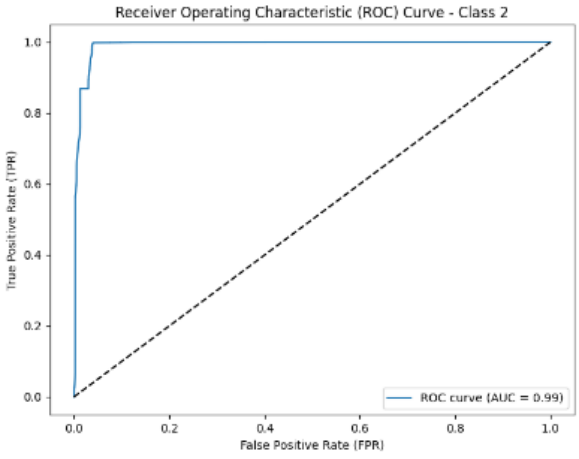


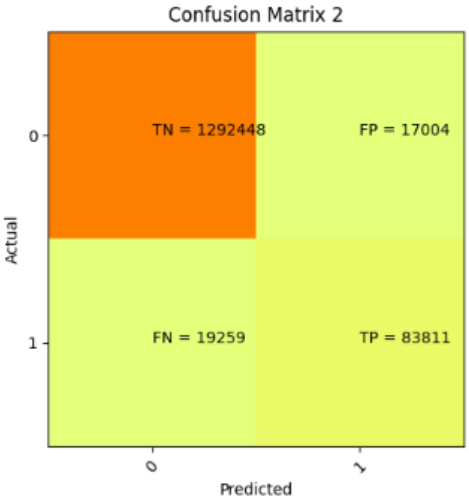
- **Ack**



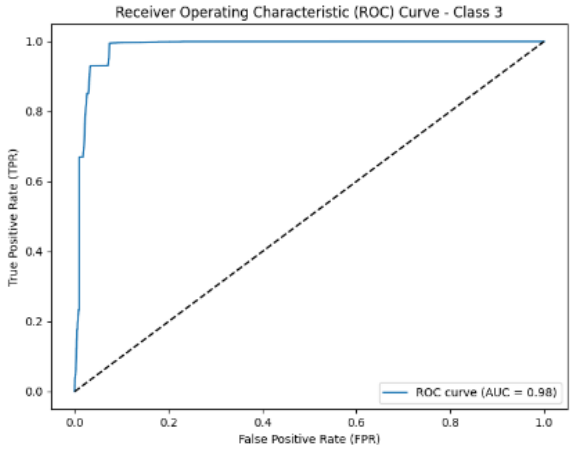


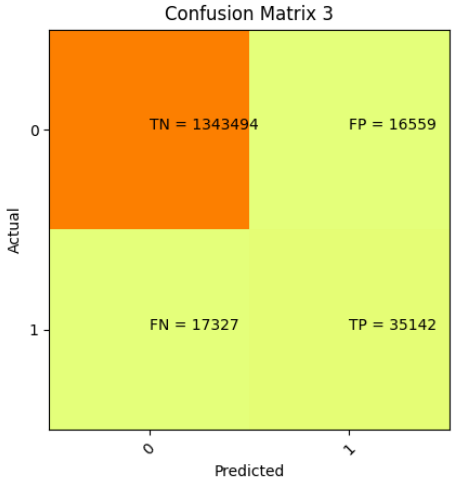
- **Combo**



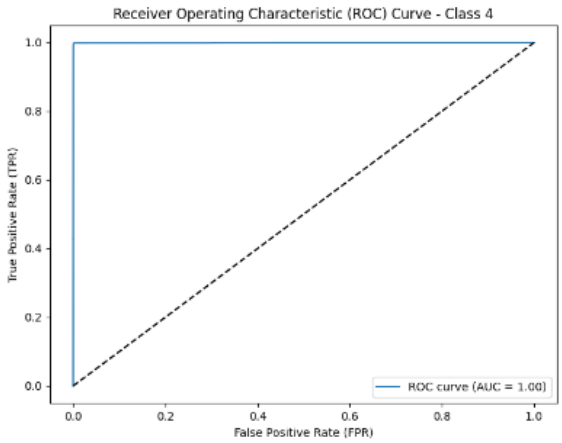


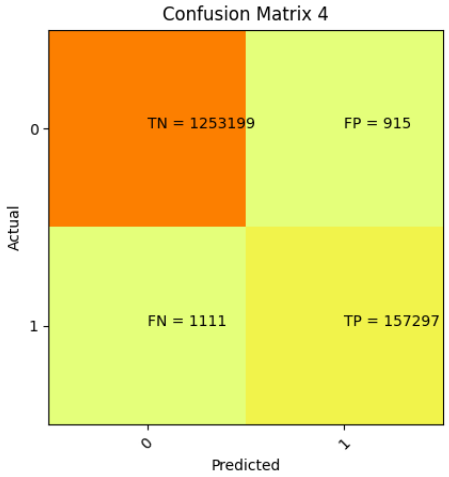
- **Junk**



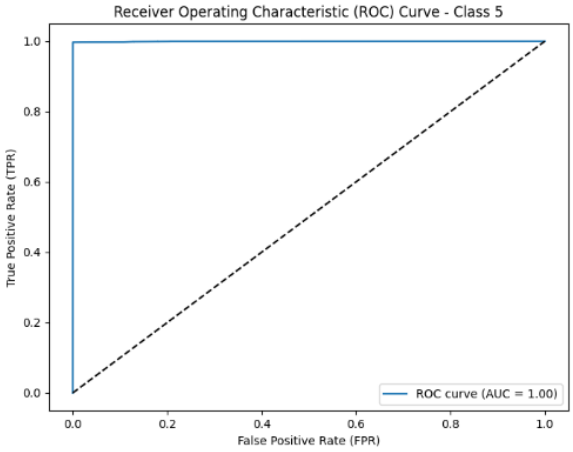


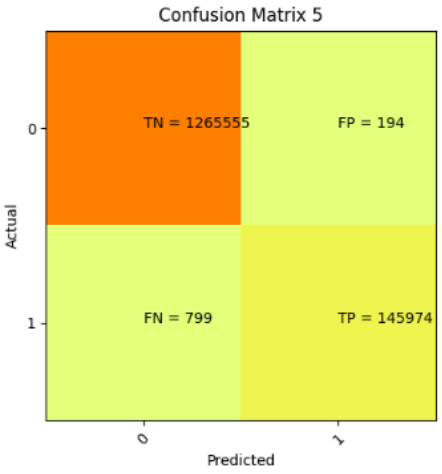
- **Scan**



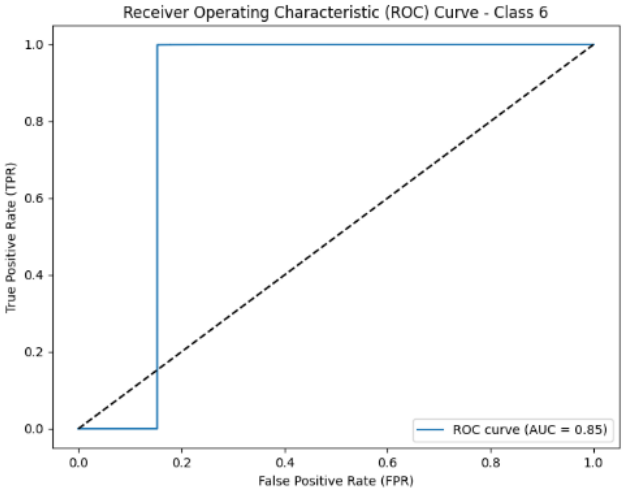


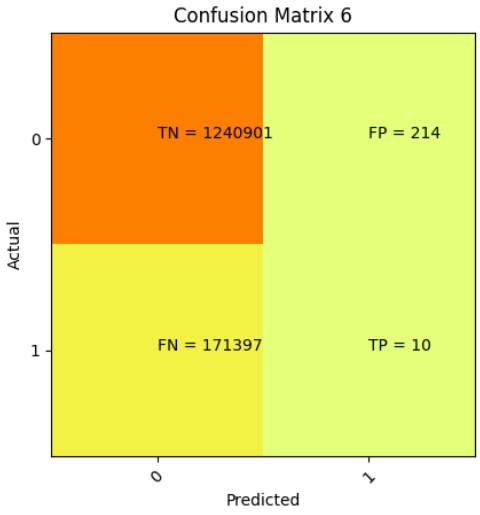
- **Syn**



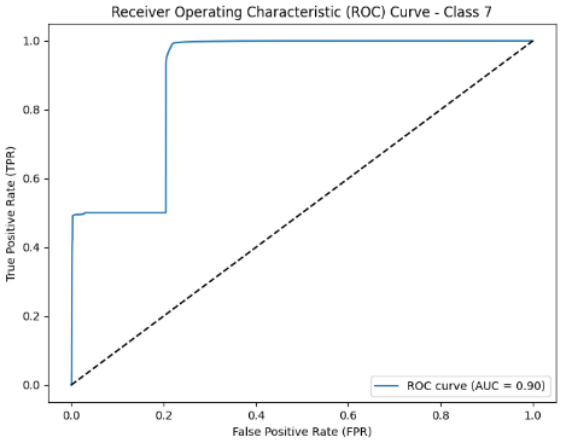


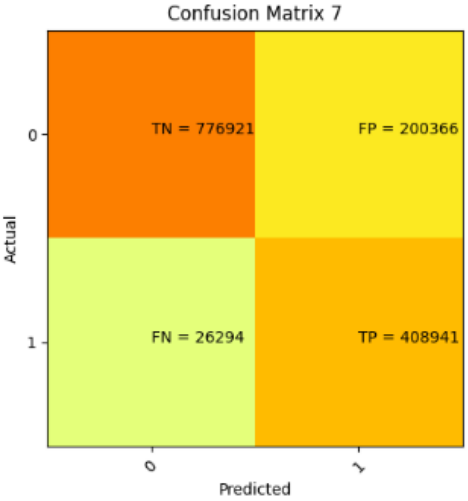
- **Tcp**



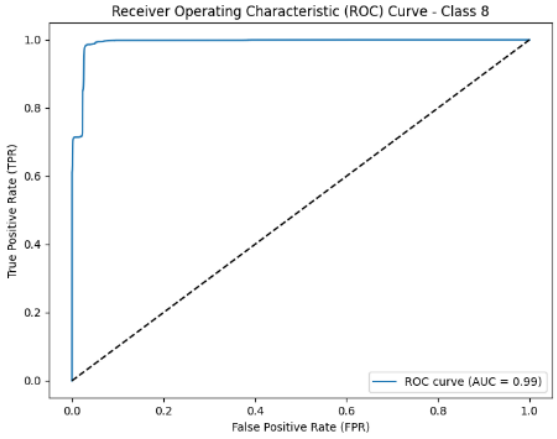


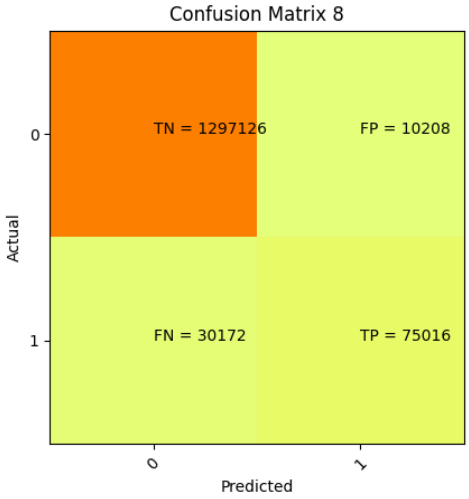
- **Udp**





- **Udpplaine**





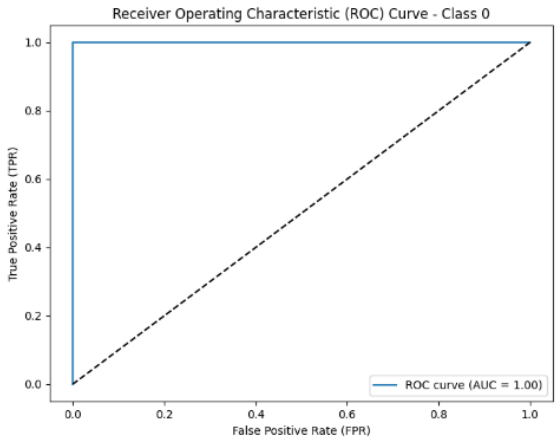
Logistic Regression

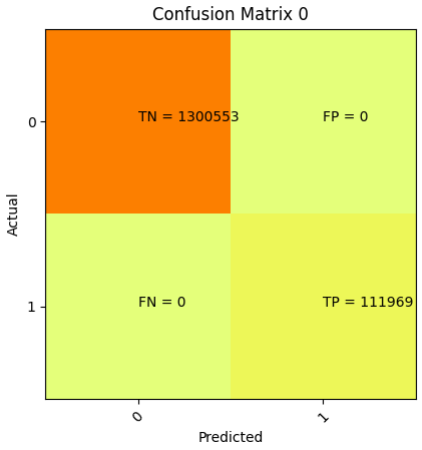
- **Precision**: O modelo Logistic Regression apresenta uma precisão relativamente alta para a maioria das classes. No entanto, para a classe "tcp", a precisão é baixa (0.03), indicando muitos falsos positivos;
- **Recall**: O recall varia, sendo 1.00 para algumas classes e mais baixo para outras. Para a classe "tcp", o recall é 0.00, sugerindo que o modelo não identifica corretamente as instâncias dessa classe;
- **F1-Score**: A pontuação F1 é afetada pelas diferenças entre precision e recall. Em geral, o modelo parece ter desempenho razoável, mas pode haver desafios com certas classes.

- ## Random Forest (n-estimators=5, random_state=42)

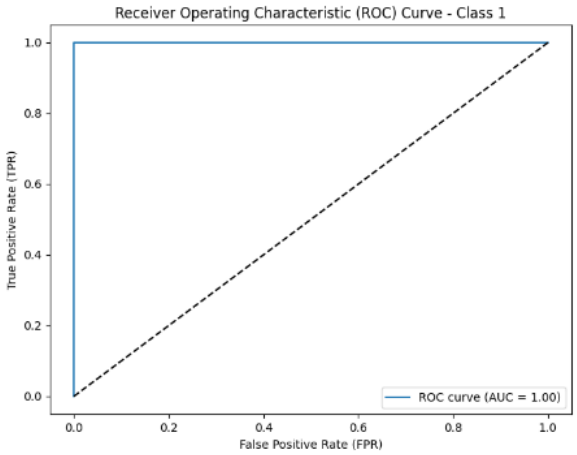
	precision	recall	f1-score	support
Normal	1.00	1.00	1.00	111969
ack	1.00	1.00	1.00	128575
combo	1.00	1.00	1.00	103141
junk	1.00	1.00	1.00	52320
scan	1.00	1.00	1.00	158940
syn	1.00	1.00	1.00	145843
tcp	1.00	0.00	0.00	171980
udp	0.72	1.00	0.84	435087
udpplain	1.00	1.00	1.00	104667

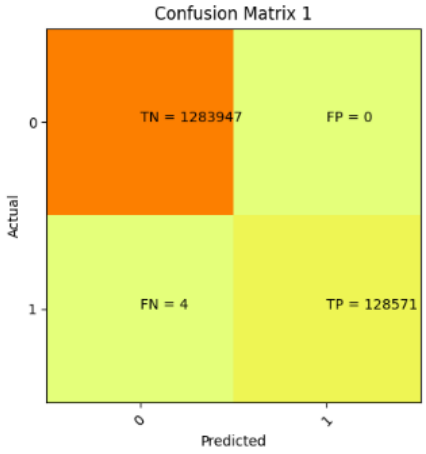
  - Normal (Não houve ataques)



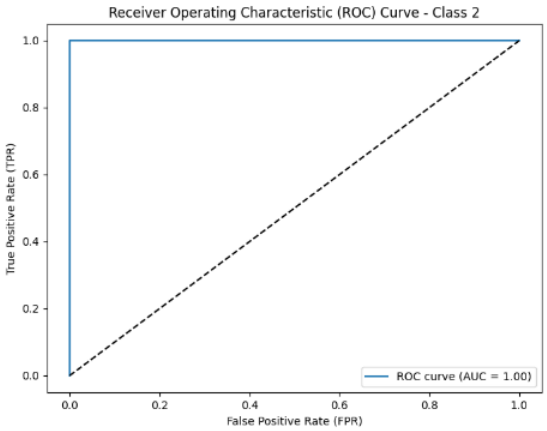


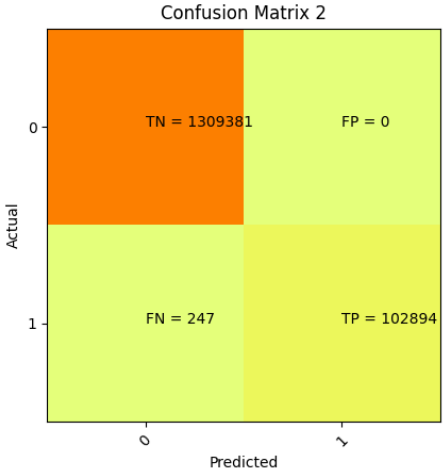
- Ack



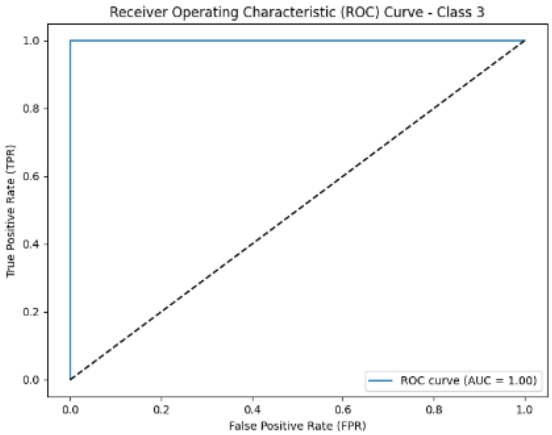


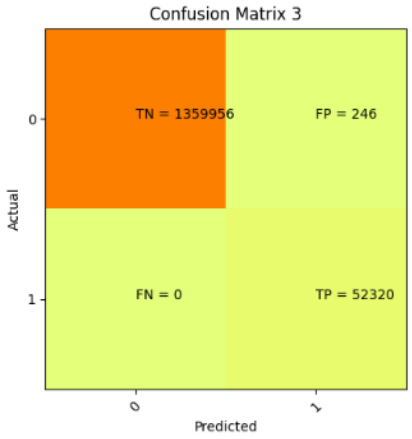
- Combo



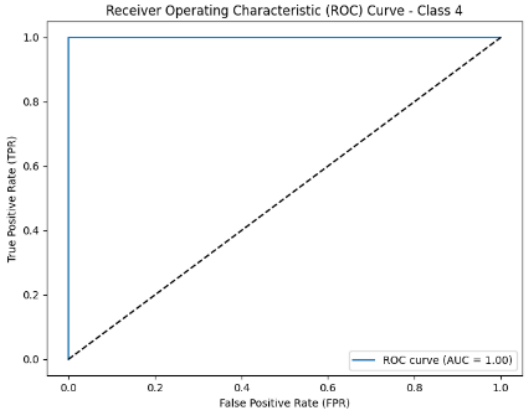


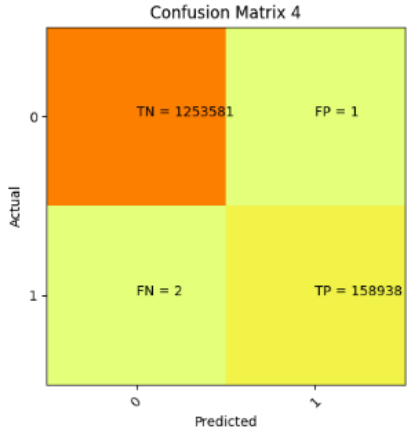
- Junk



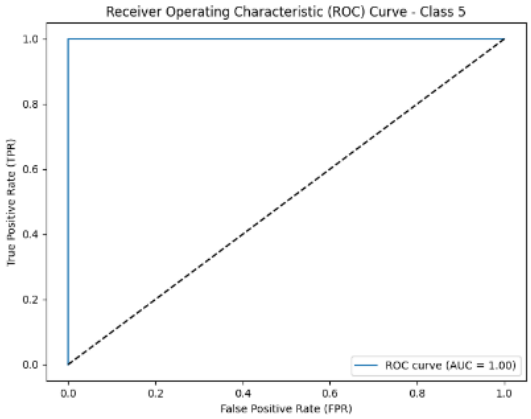


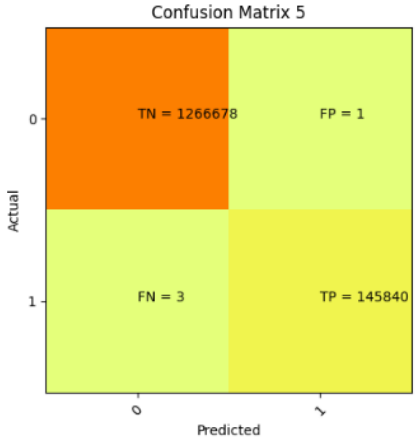
- Scan


Receiver Operating Characteristic (ROC) Curve - Class 4
True Positive Rate (TPR)
False Positive Rate (FPR)
ROC curve (AUC = 1.00)

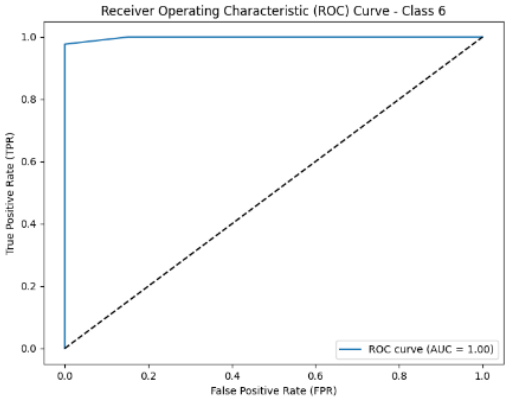

Confusion Matrix 4
TN = 1253581
FP = 1
FN = 2
TP = 158938
Actual
Predicted

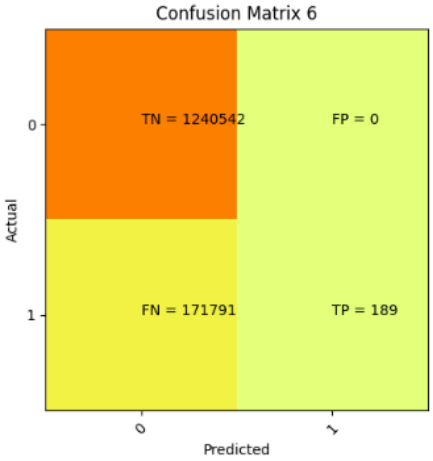
- Syn


Receiver Operating Characteristic (ROC) Curve - Class 5
True Positive Rate (TPR)
False Positive Rate (FPR)
ROC curve (AUC = 1.00)

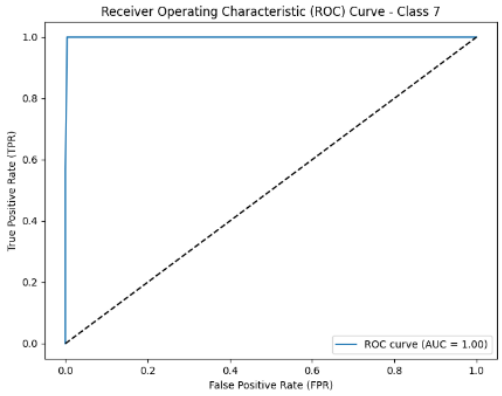

Confusion Matrix 5
TN = 1266678
FP = 1
FN = 3
TP = 145840
Actual
Predicted

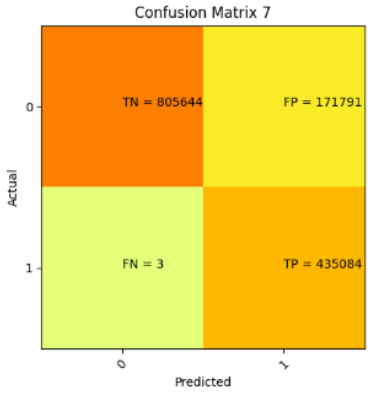
- Tcp


Receiver Operating Characteristic (ROC) Curve - Class 6
True Positive Rate (TPR)
False Positive Rate (FPR)
ROC curve (AUC = 1.00)

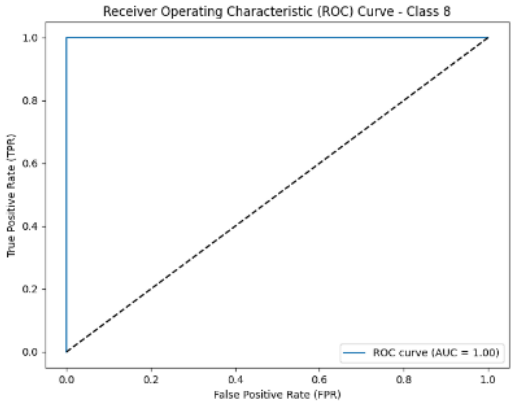

Confusion Matrix 6
TN = 1240542
FP = 0
FN = 171791
TP = 189
Actual
Predicted

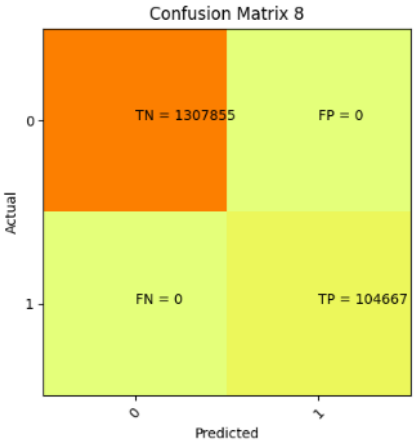
- Udp





- Udpplain





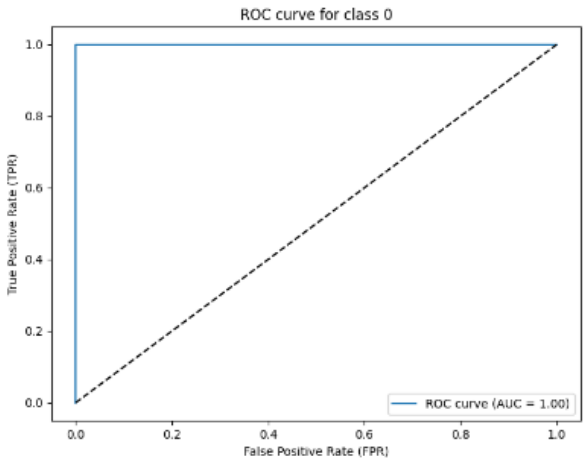
Random Forest:

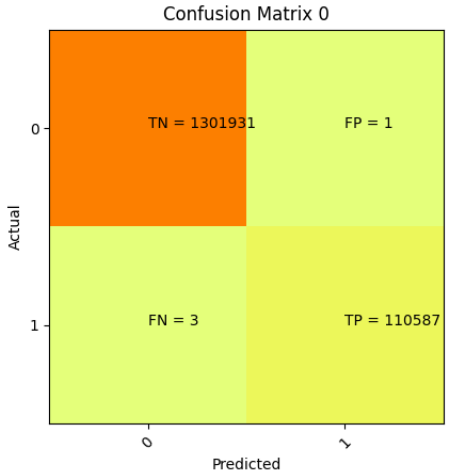
- **Precision**: O modelo Random Forest apresenta uma precisão de 1.00 para muitas classes, indicando uma boa capacidade de evitar falsos positivos. No entanto, para a classe "tcp", a precisão é 1.00, mas o recall é 0.00, sugerindo que o modelo não está a identificar corretamente as instâncias dessa classe;
- **Recall**: O recall de 1.00 para a maioria das classes sugere que o modelo é capaz de identificar a grande maioria das instâncias verdadeiramente positivas;
- **F1-Score**: A pontuação F1 é geralmente alta para a maioria das classes, indicando um bom equilíbrio entre precision e recall.

- Decision Tree

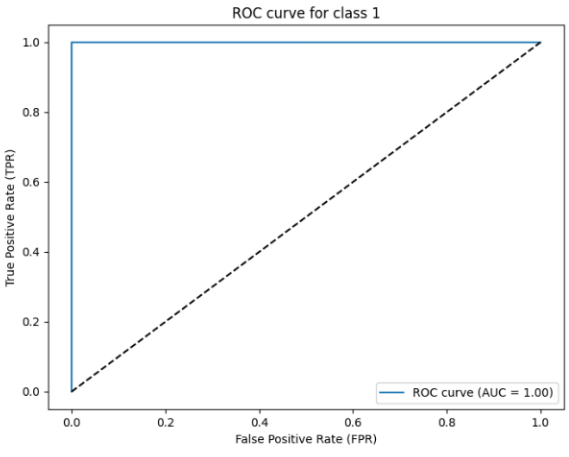
	precision	recall	f1-score	support
Normal	1.00	1.00	1.00	110590
ack	1.00	1.00	1.00	128589
combo	1.00	1.00	1.00	102976
junk	1.00	1.00	1.00	52443
scan	1.00	1.00	1.00	158631
syn	1.00	1.00	1.00	146594
tcp	1.00	0.00	0.00	172591
udp	0.72	1.00	0.83	435497
udpplain	1.00	1.00	1.00	104611
accuracy			0.88	1412522
macro avg	0.97	0.89	0.87	1412522
weighted avg	0.91	0.88	0.83	1412522

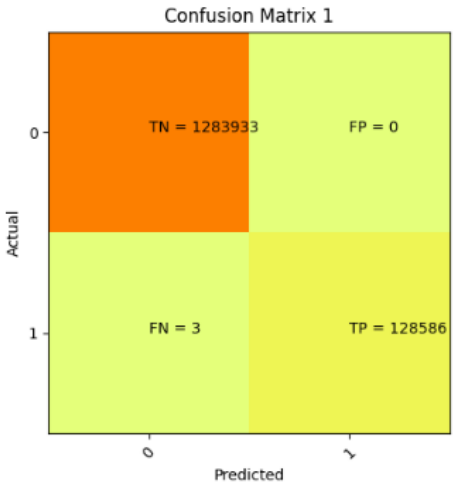
  - Normal (Não houve ataques)



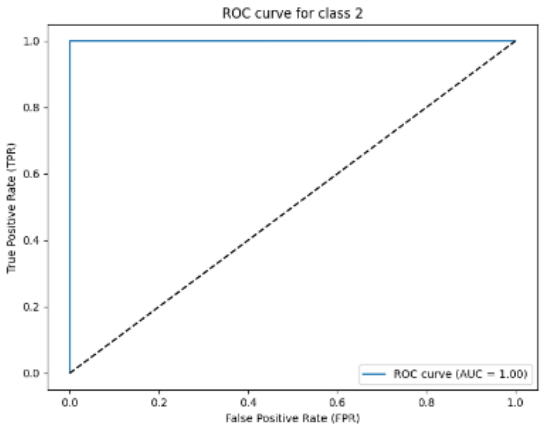


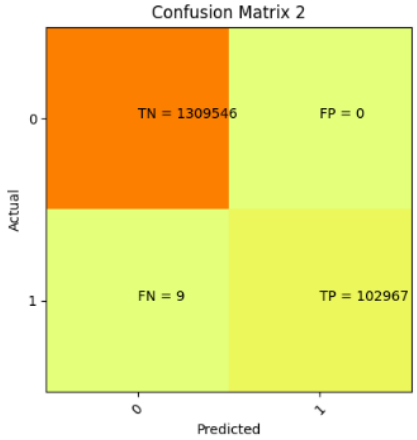
  - Ack



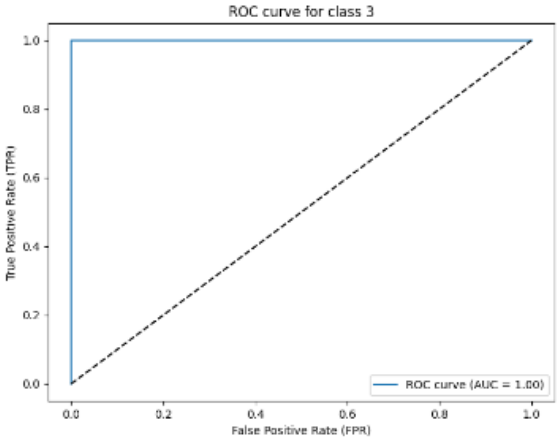


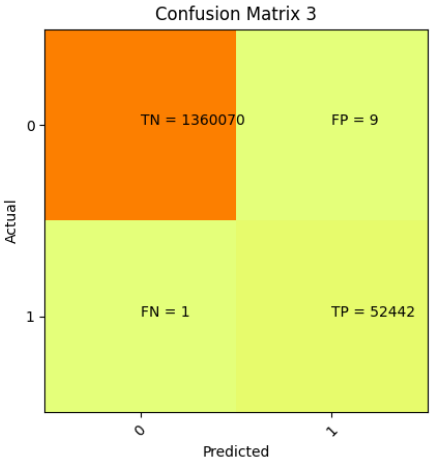
- Combo



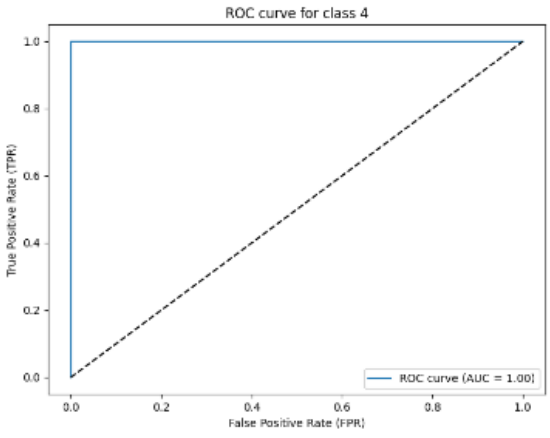


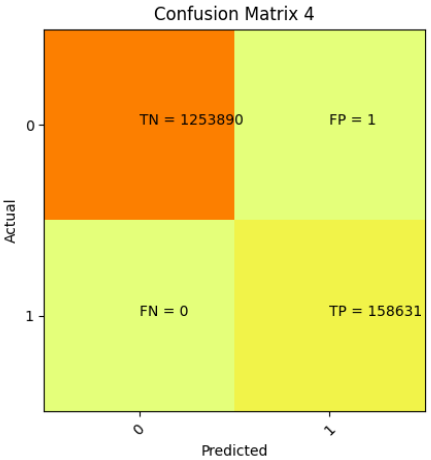
- Junk



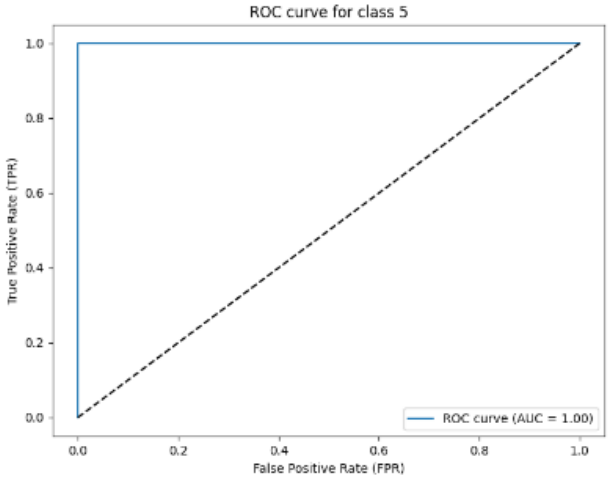


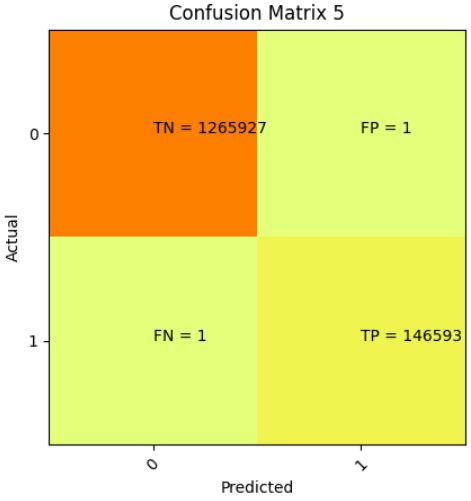
- Scan



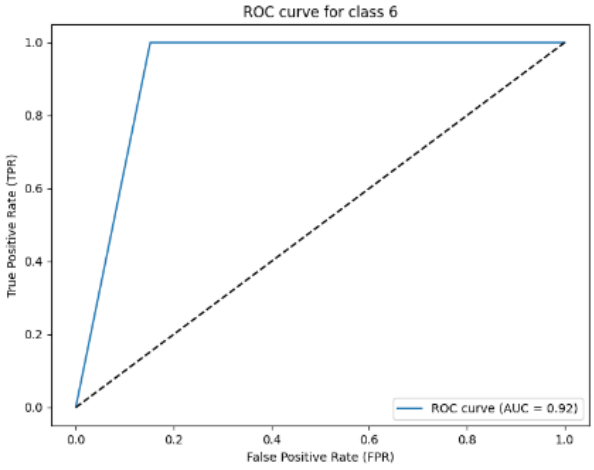


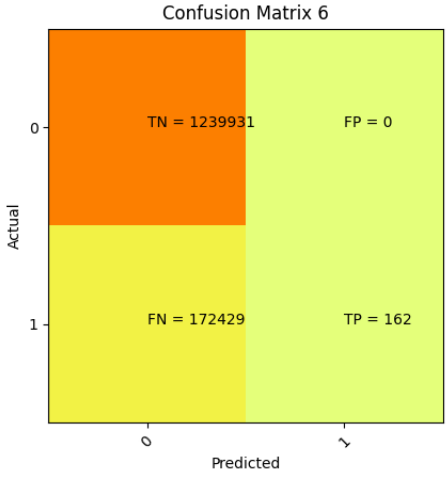
- Syn


ROC curve for class 5
True Positive Rate (TPR)
False Positive Rate (FPR)
ROC curve (AUC = 1.00)

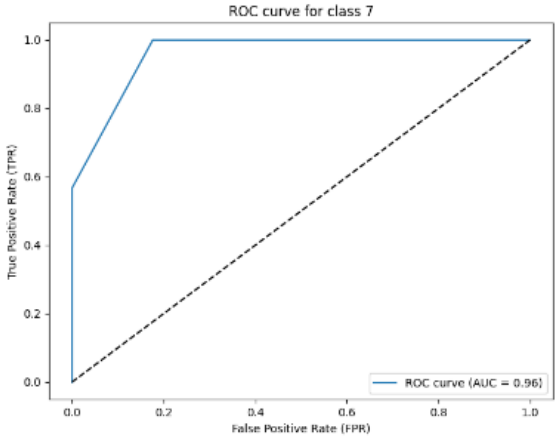

Confusion Matrix 5
TN = 1265927
FP = 1
FN = 1
TP = 146593
Actual
Predicted

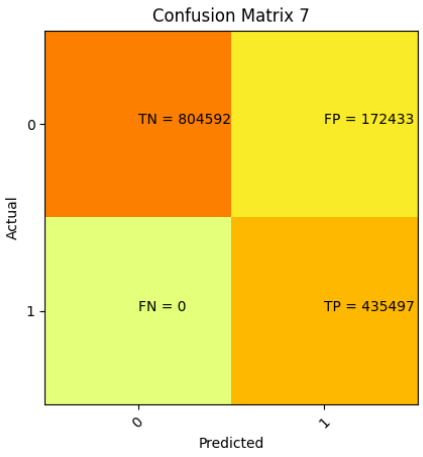
- Tcp


ROC curve for class 6
True Positive Rate (TPR)
False Positive Rate (FPR)
ROC curve (AUC = 0.92)

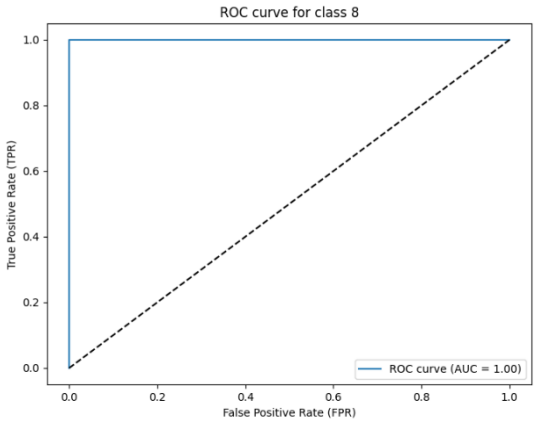

Confusion Matrix 6
TN = 1239931
FP = 0
FN = 172429
TP = 162
Actual
Predicted

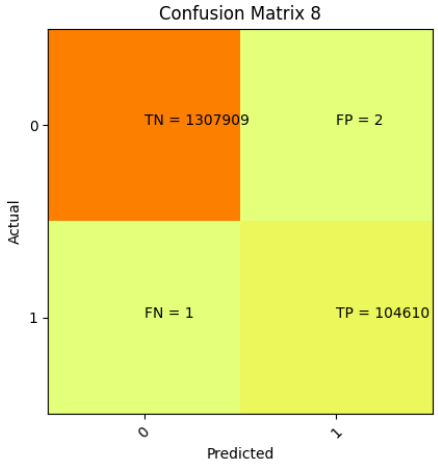
- Udp


ROC curve for class 7
True Positive Rate (TPR)
False Positive Rate (FPR)
ROC curve (AUC = 0.96)


Confusion Matrix 7
TN = 804592
FP = 172433
FN = 0
TP = 435497
Actual
Predicted

- Udpplain





Decision Tree:

- **Precision**: Assim como o Random Forest, o Decision Tree apresenta uma precisão de 1.00 para muitas classes. Entretanto, para a classe "tcp", a precisão é 1.00, mas o recall é 0.00, indicando desafios na identificação dessa classe;
- **Recall**: O recall de 1.00 para a maioria das classes sugere que o modelo é capaz de identificar a grande maioria das instâncias verdadeiramente positivas;
- **F1-Score**: A pontuação F1 é geralmente alta para a maioria das classes, mas a classe "tcp" novamente mostra um desequilíbrio entre precision e recall.

Conclusão

Ao longo deste relatório, exploram as várias aplicações de modelos de *machine learning* para a classificação binária e multiclasse do tipo de ataque.
Ao nível da classificação binária, foi feito uma *Grid-Search* com três modelos diferentes, comparando os hiperparâmetros em cada um deles. A avaliação destes modelos mostrou ir de acordo ao resultado esperado, com ótima precisão e performance.
Ao nível da classificação multiclasse, foram avaliados três modelos *Random Forest*, *Logistic Regression* e *Decision Tree*, nestes foi notório o desafio apresentado na classe tcp. Assim, dois motivos para a dificuldade dos três modelos seria algum tipo de ruído nos dados ou problemas inerentes á classe.
Este relatório fornece uma visão abrangente do desempenho dos modelos de classificação multiclasse e binário.