Trabalho Final de Ciência de Dados para Segurança

Gabriel M. Segatti[1], Fernanda Cassemiro Pereira[1], Victor Rocha de Abreu[1]

[1]Universidade Federal do Paraná (UFPR) – Curitiba – PR– Brasil

gms18@inf.ufpr.br, fcp16@inf.ufpr.br, vra17@inf.ufpr.br

1. Dataset e Objetivo

O conjunto de dados escolhido para o trabalho final da disciplina trata-se de acessos a *Honeypots* do serviço AWS da companhia *Amazon.* Um *Honeypot* é definido como um mecanismo de segurança computacional com objetivo de detectar e desviar o uso não autorizado de sistemas computacionais (COLE, 2018). Geralmente consistem em dados, aparentemente legítimos, contendo informações que parecem ter valores àqueles que invadem, quando, na verdade, são parcelas do sistema isoladas, monitoradas e capazes de bloquear atacantes.

O *dataset* final consiste em aproximadamente 300 mil linhas, sem nenhum campo em branco, dividido entre 9 colunas:

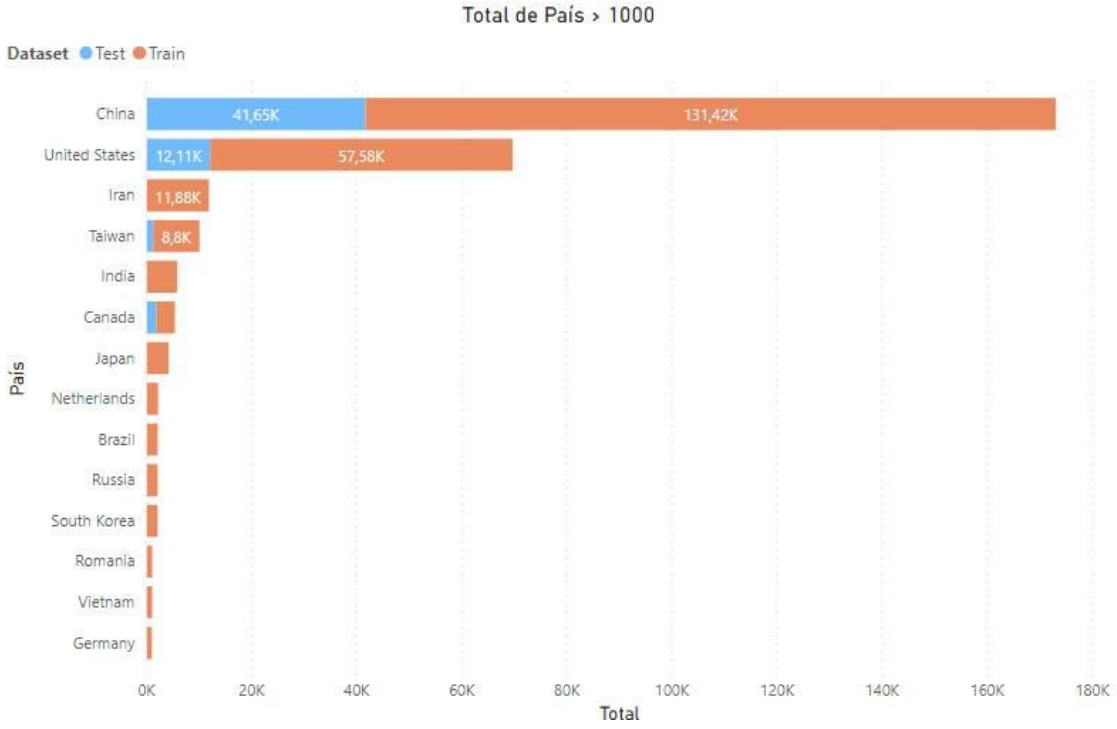
- Host (8 labels)
- Data (mm/dd/aaa)
- Hora (hh:mm:ss)
- Protocolo
- Porta Source
- Porta Destino
- IP
- País
- Local

1.1. Distribuição de Dados

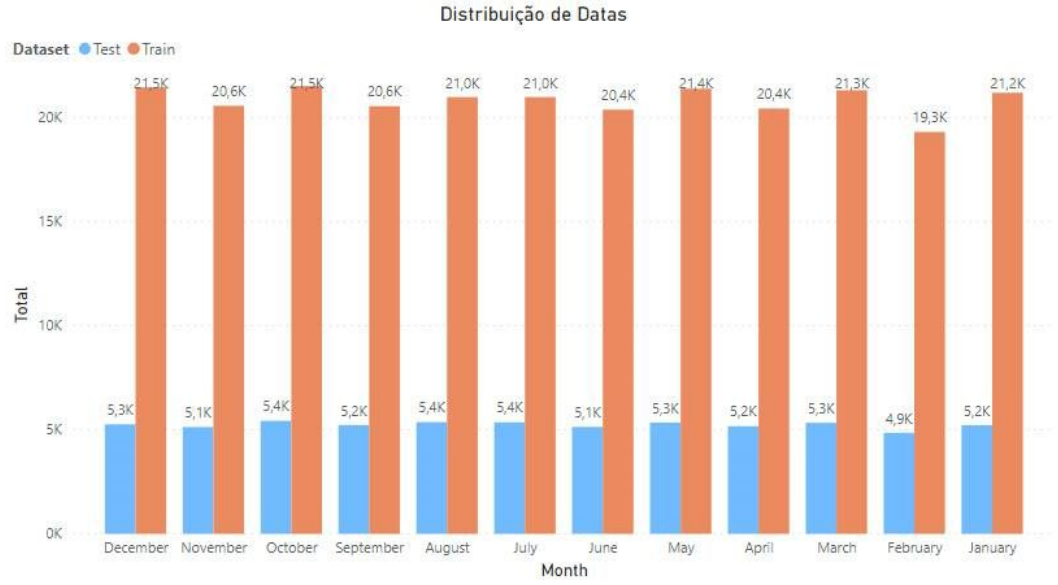
Esta subseção apresentará as distribuições de dados do Dataset. Como seu tamanho é muito grande (>300.000 linhas), algumas imagens não mostram como todas possíveis features estão distribuídas, retendo-se a apresentar somente aquelas com maiores ocorrências, sabendo que todo resto acontece um número limitado de vezes.

As cores dividem o dataset nas porções de **Treinamento** e **Teste.**

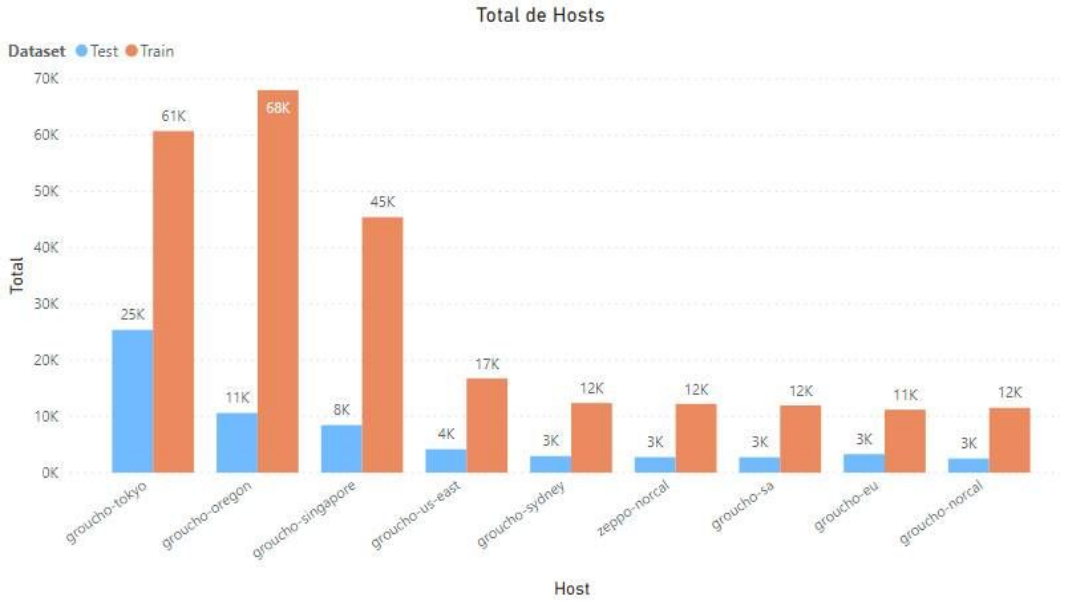
- Países: Existem um total de 140 países diferentes no dataset, a foto abaixo mostra somente aqueles que ocorrem mais de 1000 vezes.



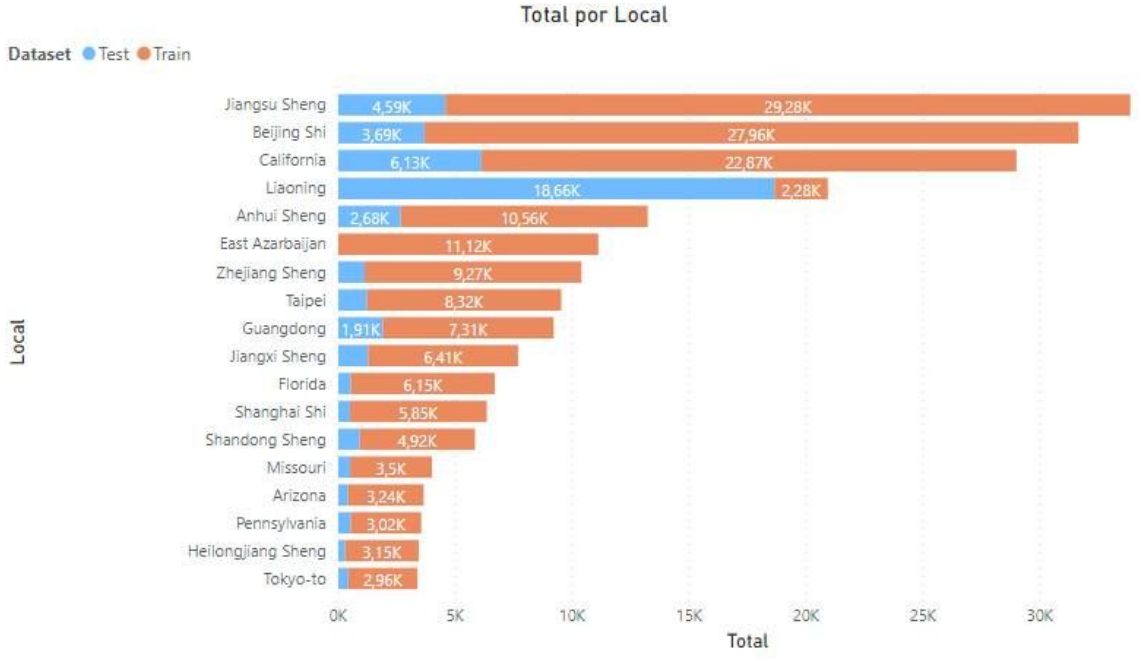
- Data: Visto que todas datas pertencem ao ano de 2013, esta é a distribuição por meses:



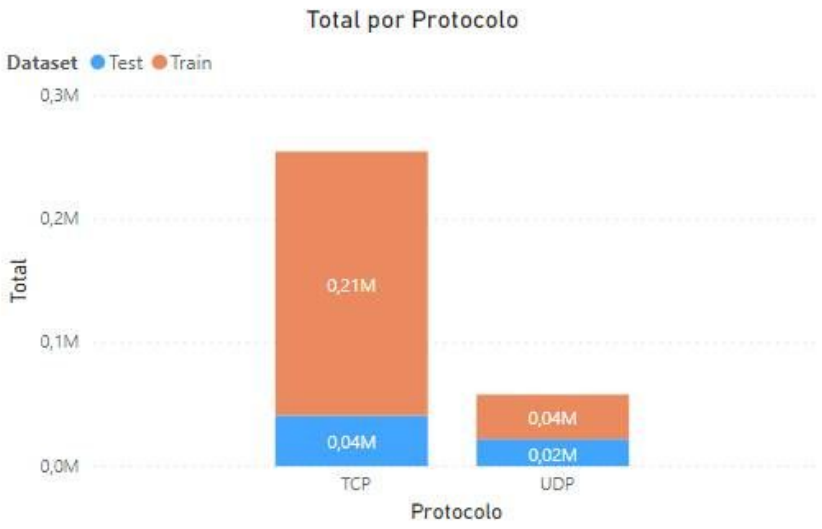
- Hosts: Existem **8 labels de hosts** no dataset, abaixo, suas distribuições:



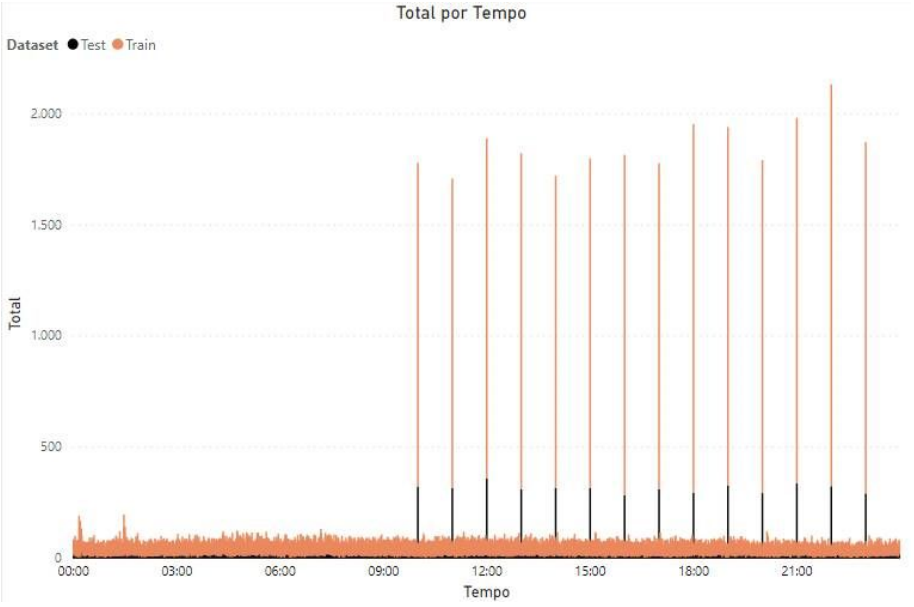
- Local: Algumas vezes a cidade da qual foi realizado o acesso, em outras o estado. Existem mais de 1000 valores diferentes no dataset, abaixo alguns dos principais:



- Protocolos:



- Hora: Desta vez usamos preto para maior discrepância entre as cores, abaixo é possível perceber o *overlapping* do dataset de teste (preto) com o de treino (ainda laranja):



- IP: Transformando os valores em inteiros, temos (todos outros valores ocorrem < 2000 vezes):

srcstr	Test	Train	Total
175146199252	18195		**18195**
2186189218		11116	**11116**
9625417120	339	5074	**5413**
6814516427	1505	1329	**2834**
1231514261	417	2246	**2663**
2202251746		2605	**2605**

- Porta Source: Porta dos protocolos UDP e TCP (todas outras ocorrências são < 2000):

Total de Portas por Valor

Source Port	Test	Train	Total
6000	21336	119532	**140868**
25416	18195		**18195**
10100		11116	**11116**
4445	1627	2811	**4438**
50499	1	2609	**2610**

- Porta Destino: Análoga a Porta Source (demais ocorrências < 10000):

Total de Portas por Valor

Dest Port	Test	Train	Total
1433	14771	83058	**97829**
3389	3549	19188	**22737**
445	2780	19313	**22093**
56338	18195	1	**18196**
80	3448	10508	**13956**
8080	2422	10422	**12844**
3306	1783	11016	**12799**
2193		11116	**11116**
22	1752	9294	**11046**

1.2. Limpeza dos Dados

Para rodarmos os modelos tivemos que primeiramente discretizar os dados neles presentes, visto que não é possível interpretar informações que não estão formatadas em consonância. Abaixo, explicamos o que foi feito para cada uma das colunas alteradas.

1. Data: Como o *dataset* compreendia somente informações entre meses pertencentes aos anos de 2013, retiramos o ano dos campos, assim como as barras "/", para alimentarmos os dados ao modelo como inteiros.
2. Hora: Similar a maneira que a coluna Data foi tratada, retiramos os ":" para tornar o dado um inteiro que pudesse ser recebido pelo modelo. Por fim, retiramos os segundos que foram registrados sempre como "00".
3. Host: Somente classificamos cada host como números inteiros. Como temos 9 hosts diferentes, usamos o intervalo [0,8].
4. Protocolo: Somente 2 protocolos foram usados UDP e TCP, sendo 0 e 1, respectivamente.
5. Portas: Como o dado lido a partir do excel era um float em formato de string, retiramos as partes decimais dos dados.
6. IP: Retiramos os pontos e tratamos os números como inteiros.
7. País e Local: Usamos *Word2Vec,* para discretizar as palavras em vetores de 100 posições que as representavam.

1.3. Objetivo

Com isto, resolvemos realizar a codificação de um modelo preditivo que conseguisse estimar qual ***Host*** (dos *Honeypots)* estava sendo atacado, baseado nas outras 8 colunas do *dataset.* Ou seja, temos um projeto de classificação, onde uma coluna (Host que até então era uma *feature)* tornou-se um *label* do *dataset.*

O racional por trás deste objetivo foi a possibilidade de que determinados Hosts poderiam ser atacados majoritariamente por uma localidade, horário, ou hora específica.

2. Métodos Utilizados

Esta seção descreverá os resultados encontrados por meio dos métodos *K-Nearest-Neighbor, Random-Forest* e um *Multi-Layer Perceptron* todos implementados pela biblioteca *Scikit.*

2.1. K-Nearest-Neighbor

O KNN, é um algoritmo usado tanto para classificação como regressão. Como neste caso queremos classificar a qual *host* uma determinada linha do *dataset* pertence, usamos a versão classificatória da implementação do Scikit. Na fase de classificação do algoritmo, define-se um K inteiro, e um vetor ainda não classificado será assinalado à classe que aparece mais vezes dentre os K pontos mais próximos.

Com isso, obtemos resultados razoavelmente satisfatórios, como mostrado a seguir:

2.1.1. Valor de K

Usando o método *KNeighborsClassifier* da classe sklearn.neighbors, testamos valores de K de 1 até 20. Embora seja aceito que um valor ideal para K seja sqrt(n), onde n é o número de linhas no dataset, como nosso conjunto de dados é demasiado grande, nos limitamos a valores menores por questão de tempo (o teste em si feito demorou diversas horas).

Inicialmente, calculamos as acurácias de predição na porção inteira de testes, sem validação cruzada:

Figura 1. Acurácia na porção de teste de treinamento (20% de 80%).

Percebe-se que a acurácia é maior para valores menores de K, que ultrapassam 60% e decaem até 53%, respectivamente. Portanto, visando melhores resultados, para todos testes e treinamentos adiante, utilizamos K=1.

2.1.2. Treinamento de 80% do Dataset

Aqui treinamos 2 modelos de KNN com K=1, um usando Percentage Split e outro com Validação Cruzada. Depois, usamos 20% da porção 1 para testes. Abaixo, os resultados do modelo treinado com Percentage Split:

[**8971** 325 1234 949 255 220 226 250 250]
[267 **1703** 112 157 205 246 222 262 179]
[1135 162 **5183** 1164 170 122 163 149 183]
[1027 176 1222 **10522** 229 119 153 152 204]
[247 188 151 183 **758** 197 230 183 227]

[197 189 110 114 208 **601** 210 530 199]
[209 192 130 153 216 237 **770** 189 248]
[196 213 106 108 185 455 192 **570** 172]
[232 200 147 164 297 188 226 217 **840**]

Matriz de Confusão no Teste de 20% de 80% para Treino.

Os resultados de Precisão e Erro, foram respectivamente: 0.5978577994484633 e 1.178829782982295.

Para o modelo usando Validação Cruzada como treinamento temos:

[35951 421 1355 936 354 320 316 290 299]
[348 8463 141 110 299 340 270 363 262]
[1379 137 24161 1357 154 137 223 116 145]
[1036 132 1127 41044 155 111 124 116 115]
[362 316 168 157 5060 334 329 321 407]
[344 355 132 99 327 5085 335 696 305]
[344 328 162 139 336 342 5019 333 347]
[367 385 160 127 315 751 288 4527 318]
[294 280 134 108 367 321 323 307 5672]

Matriz de Confusão, Primeira Pasta.

Os resultados de Precisão e Erro foram, respectivamente, 0.8429368087777035 e 0.4712707561839221.

[35867 295 1504 919 315 257 321 293 299]
[356 8572 159 129 301 319 297 308 261]
[1276 144 24179 1167 269 108 192 139 188]
[1033 123 1419 41018 162 51 153 103 141]
[354 291 183 150 5147 321 346 314 375]
[347 337 154 108 367 5021 326 747 322]
[329 319 173 136 361 342 5093 306 338]
[357 363 179 108 327 622 329 4596 315]
[295 250 167 121 379 296 369 273 5543]

Matriz de Confusão, Segunda Pasta.

Os resultados de Precisão e Erro foram, respectivamente, 0.8432740284638395 e 0.4677361942884976 6.

[35995 319 1431 919 314 279 323 328 282]
[357 8456 161 128 284 323 297 337 280]
[1315 127 24276 1112 269 100 204 143 189]
[1019 108 1471 40643 170 75 171 112 141]
[349 273 206 154 5123 347 364 305 419]
[369 342 149 96 344 4986 374 743 293]
[345 313 173 142 351 323 5136 340 337]
[362 365 166 99 293 630 345 4578 344]
[307 257 159 107 402 297 354 269 5645]

Matriz de Confusão, Terceira Pasta.

Os resultados de Precisão e Erro foram, respectivamente, 0.8428306470246607 e 0.4717203824321033.

```
[35861   293  1510   931   323   278   352   317   283]
 [ 363  8549   153   118   306   331   297   342   268]
 [ 1303   135 24146  1149   270   103   203   152   208]
[ 1010   112  1494 40873   157        67   164   109   136]
 [ 374   283   198   156  5146   336   356   305   392]
 [ 338   341   161        97   360  4953   343   728   288]
 [ 353   308   171   125   345   325  5041   303   327]
 [ 367   352   174   116   309   645   335  4604   320]
 [ 308   255   165   119   409   307   352   282  5595]
```

Matriz de Confusão, Quarta Pasta.

Os resultados de Precisão e Erro foram, respectivamente, 0.8416004196511649 e 0.4754235541705957.

```
[35955   311  1452   941   316   266   328   315   271]
 [ 383  8584   147   123   318   329   303   332   269]
 [ 1317   131 24264  1107   265   113   191   149   184]
[ 1024   115  1453 40723   170        75   179   109   129]
 [ 363   278   172   159  5140   318   344   304   405]
 [ 363   317   145        96   351  5014   343   776   295]
 [ 341   323   179   133   347   328  5078   325   337]
 [ 378   372   170   122   300   632   322  4592   302]
 [ 324   237   160   108   383   290   349   262  5615]
```

Matriz de Confusão, Quinta Pasta.

Os resultados de Precisão e Erro foram, respectivamente, 0.8420375562813411 e 0.47296934423260667.

Por fim, as curvas ROC's de ambos algoritmos, para os 20% do dataset reservados na porção 1, com treinamento Percentage Splitting e Cross-Validation:

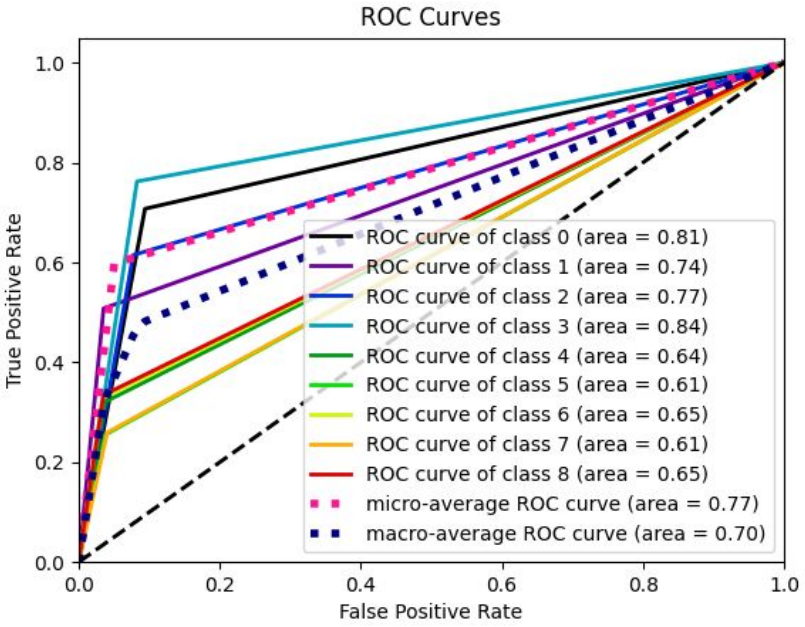


Figura 2. Curva Roc, Modelo Percentage Splitting.

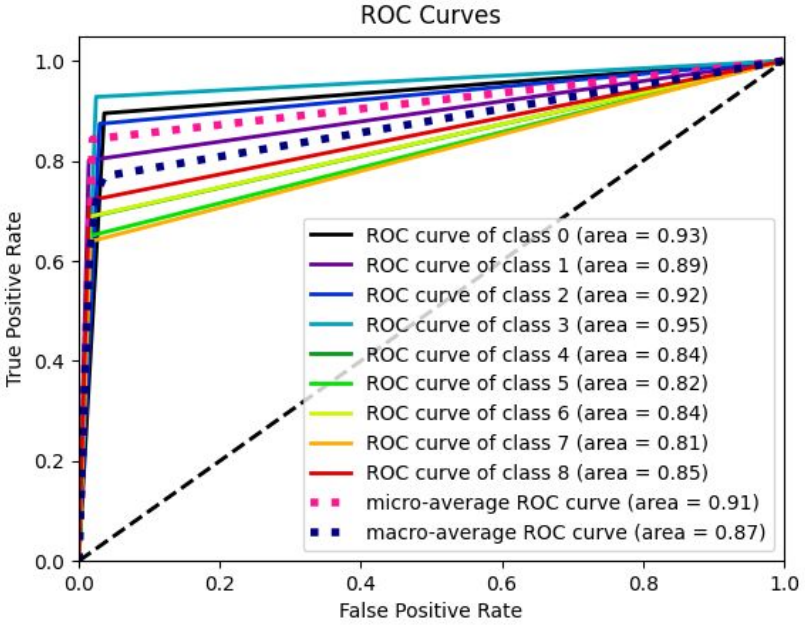


Figura 3. Curva Roc, Pasta 1.

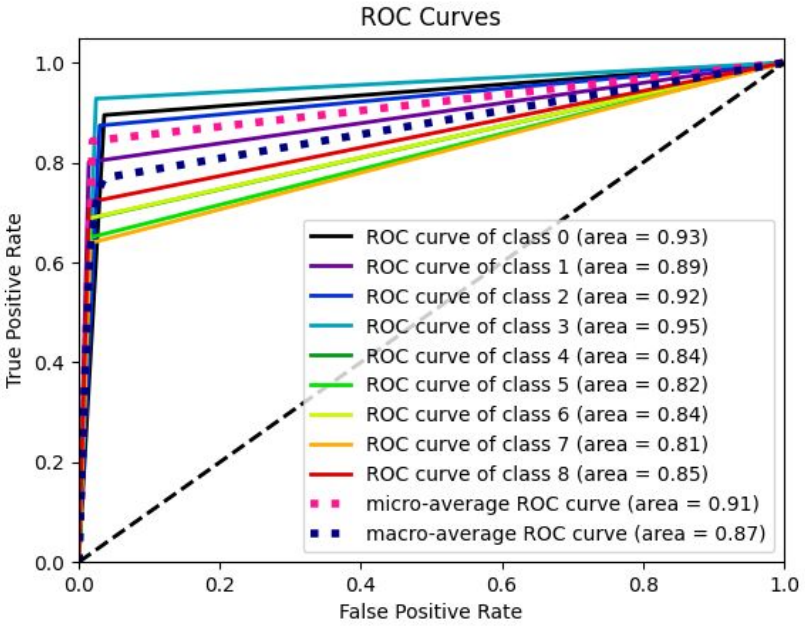


Figura 4. Curva Roc, Pasta 2.

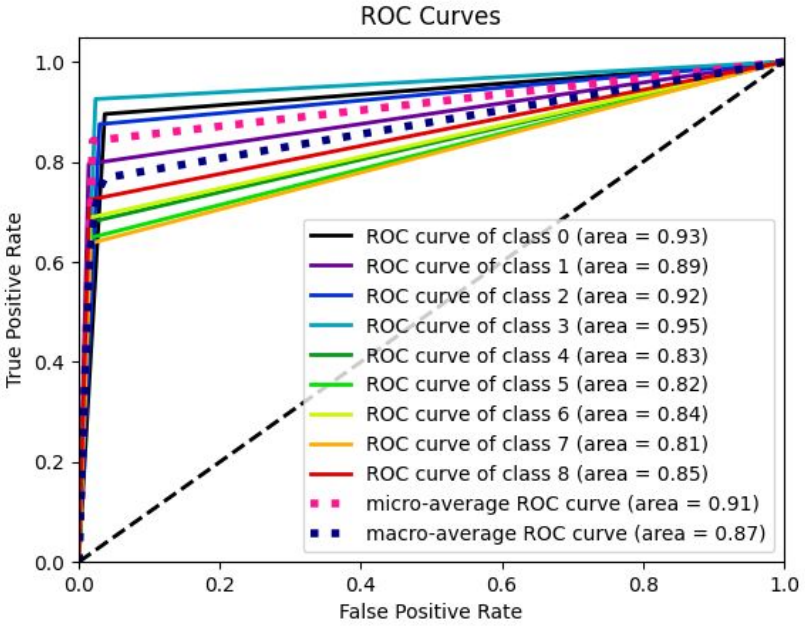


Figura 5. Curva Roc, Pasta 3.

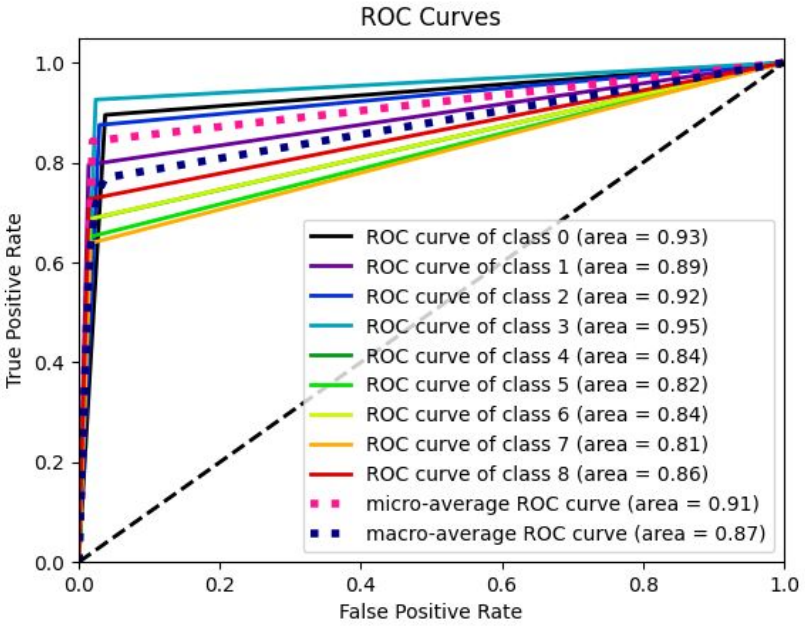


Figura 6. Curva Roc, Pasta 4.

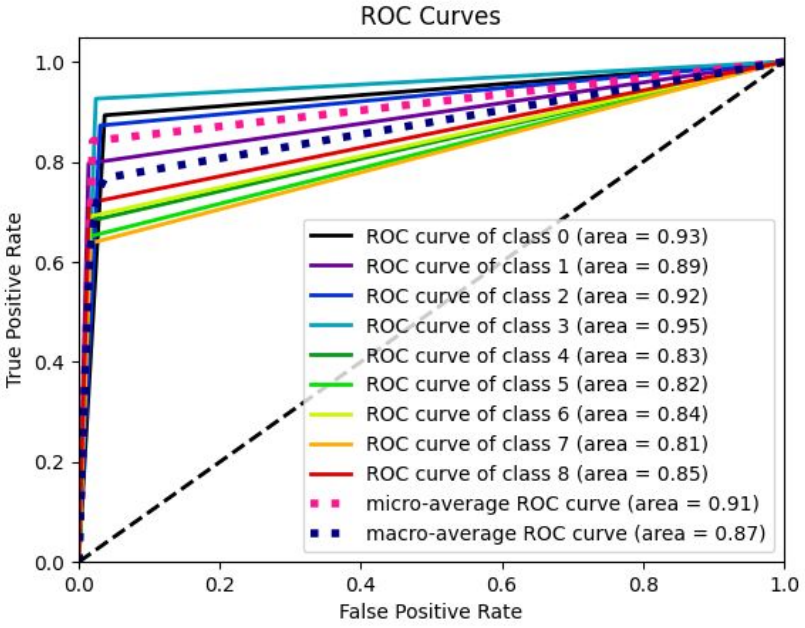


Figura 7. Curva Roc, Pasta 5.

2.1.3. Teste de 20% do Dataset

Esta seção busca enunciar os resultados de testes feitos com os 20% restantes do *dataset.* Primeiro, o modelo com Percentage Split:

[11021 383 1529 1220 300 289 323 311 344]
[341 2104 180 182 264 281 255 318 253]
[1484 183 6509 1556 260 170 177 170 261]
[1276 242 1641 13101 196 150 204 159 243]

[282 244 183 186 941 261 242 227 369]
[277 246 142 144 230 762 254 671 265]
[243 290 170 208 251 273 942 236 279]
[253 279 145 117 237 619 245 690 216]
[294 215 197 202 372 231 314 212 1017]

Matriz de Confusão, Percentage Split.

Os resultados de Precisão e Erro foram, respectivamente, 0.5928892299330168 e 1.199910475916423.

Já para o modelo treinado com Cross-Validation, temos:

[10802 407 1534 1311 296 313 351 345 361]
[336 2056 206 199 254 272 283 320 252]
[1516 210 6366 1567 291 180 189 193 258]
[1322 265 1627 12911 227 183 226 189 262]
[294 243 212 181 892 261 246 246 360]
[275 258 162 159 237 723 275 645 257]
[252 283 185 223 265 266 911 229 278]
[253 290 154 134 254 583 251 653 229]
[305 227 220 216 340 219 314 224 989]

Matriz de Confusão, Modelo Cross-Validation.

Os resultados de Precisão e Erro foram, respectivamente, 0.580355858232219 e 1.2434255751123047.

Sendo as curvas ROC:

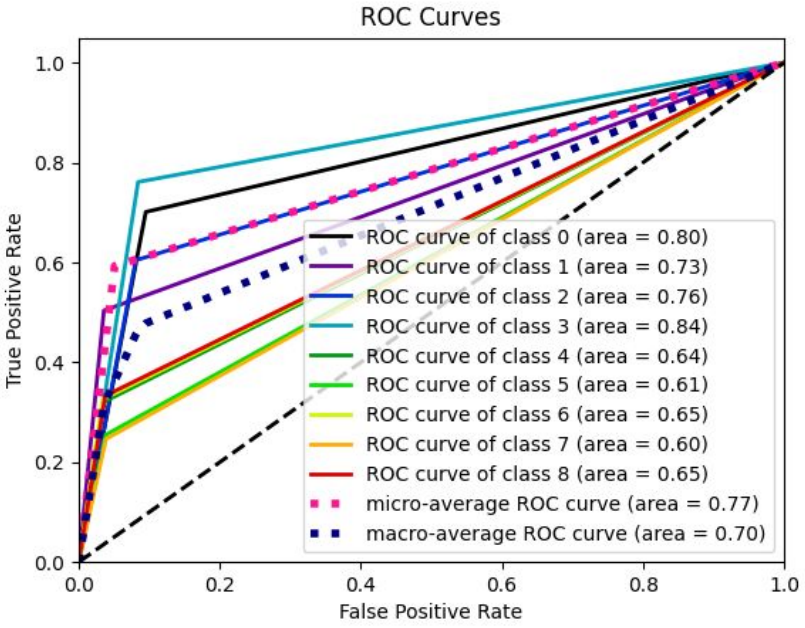


Figura 8. Curva Roc, Modelo Percentage Splitting.

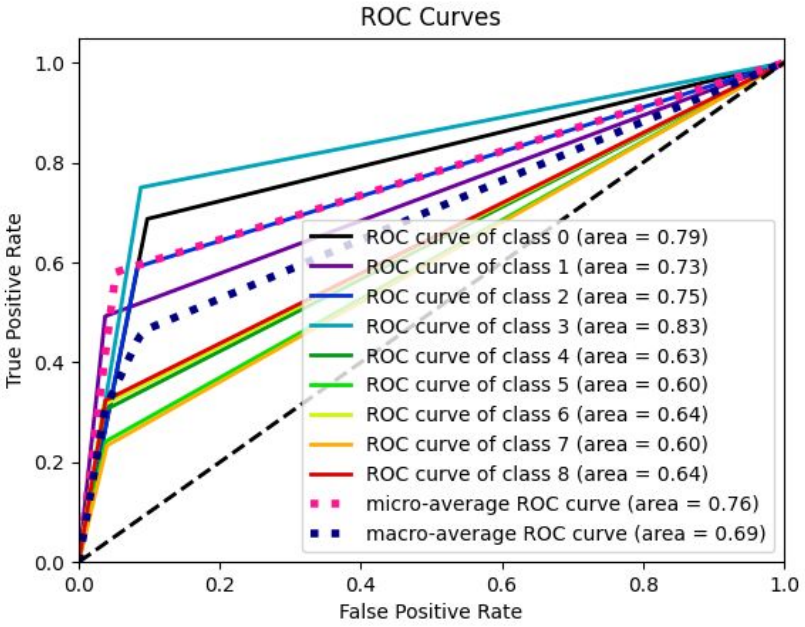


Figura 9. Curva Roc, Modelo com Cross-Validation.

2.2. Random-Forest

Assim como o KNN, o Random Forest também é um algoritmo usado tanto para classificação como regressão (*RandomForestClassifier* e *RandomForestRegressor*). Como o objetivo de classificar a qual *host* uma determinada linha do *dataset* pertence utilizamos o algoritmo como classificador. Nesse caso, o algoritmo funciona construindo várias árvores de decisão e as mesclando-as para obter uma previsão mais precisa e estável.

Usando o método *RandomForestClassifier* da classe sklearn.ensemble e setando o valor do parâmetro *n_estimators* (quantidade de árvores de decisão que são geradas) para 100 (*default*) testamos utilizando a máquina Orval do Departamento de Informática (DINF) da UFPR.

Conforme solicitado na especificação do projeto esses 80% foram divididos na proporção 80/20 e as seções abaixo apresentam os resultados obtidos.

2.2.1 Treinamento 80% do *Dataset*

Nesta seção são apresentados os resultados obtidos durante o treinamento de 80% do *dataset*. Conforme especificado, esses 80% foram novamente divididos na proporção 80/20 sendo 80% para treinamento e 20% dos 80% para teste.

[[**10323** 276 706 594 150 143 164 196 128]
[252 **1674** 117 141 118 278 172 491 110]
[1282 87 **5411** 1197 115 74 75 74 116]
[1139 114 1319 **10779** 143 73 67 54 116]
[216 107 112 140 **783** 182 313 140 371]
[142 127 96 63 149 **677** 218 808 78]
[217 142 117 124 266 437 **725** 119 197]
[141 245 66 79 101 714 174 **616** 61]
[164 110 153 143 503 134 366 95 **843**]]

Matriz de confusão gerada para os 80% do *dataset*.

A precisão (a capacidade do classificador de não rotular como positiva uma amostra negativa) para o treinamento desses 80% do *dataset* foi de 0.6360856880220614, ou seja, aproximadamente 63%. Esse resultado foi obtido através da função `precision_score`.

O erro obtido através da função `mean_absolute_error` que representa a soma dos erros absolutos ao longo da duração das observações/previsões foi de 1.0239998401342871.

As curvas ROC referentes a esse treinamento estão apresentadas abaixo. Enxergamos na figura a seguir que todas as curvas se mantiveram acima da linha pontilhada que divide o gráfico no meio, implicando que é melhor que um decididor aleatório. O pior resultado ainda ficou um pouco acima de um decididor aleatório, com 0.67.

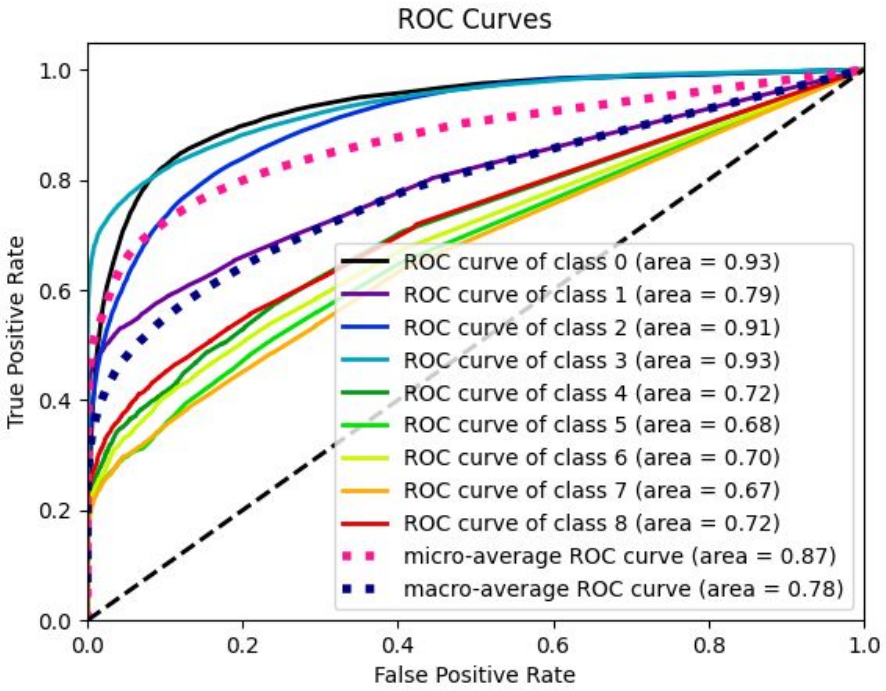


Figura 10. Curvas ROC geradas para o modelo treinado com 80% de 80% do *dataset*.

2.2.2 Treinamento feito usando validação cruzada com 5 pastas

2.2.2.1 Pasta 1

A precisão para esta pasta foi de 0.9976519371518497 e o erro calculado foi de 0.008942622336572328. A matriz de confusão está apresentada abaixo.

```
[ 9955    0    0    0    0    1    2    1    0]
[    1 2752    0    0    0    3    2    4    1]
[    0    0 6840    0    0    0    0    0    0]
[    1    0    1 11080    1    0    0    0    0]
[    0    0    0    0 1902    4    3    1   12]
[    0    2    0    0    2 1920    0    0    1]
[    3    2    1    0    7    0 1861    0    0]
[    1    6    0    1    1    5    1 1754    0]
[    0    0    0    2   21    0    0    0 1875]
```

Matriz de confusão da pasta 1 com Random Forest

2.2.2.2 Pasta 2

A precisão para esta pasta foi de 0.9962530911997602 e o erro calculado foi de 0.01403841830489846. A matriz de confusão está apresentada abaixo.

[**30017** 1 3 3 0 5 12 8 0]
[4 **8133** 0 0 4 18 7 16 1]
[1 0 **20661** 0 2 0 2 1 0]
[0 0 0 **32998** 3 0 0 2 3]
[1 3 1 2 **5619** 14 22 6 71]
[2 10 3 0 11 **5682** 1 15 0]
[7 7 0 0 28 2 **5529** 2 0]
[10 20 3 0 5 13 0 **5298** 0]
[0 3 1 3 83 2 2 1 **5712**]

Matriz de confusão da pasta 2 com Random Forest

2.2.2.3 Pasta 3

A precisão para esta pasta foi de 0.9945794719356531 e erro calculado foi de 0.019953538330877027. A matriz de confusão está apresentada abaixo.

[**50125** 5 5 4 3 12 14 22 1]
[10 **13450** 0 1 21 37 9 41 7]
[5 0 **34545** 0 3 4 3 7 1]
[4 0 0 **54967** 4 0 0 0 4]
[0 11 5 9 **9143** 29 60 14 199]
[14 27 5 0 30 **9399** 2 26 2]
[25 16 1 0 63 8 **9047** 0 6]
[19 33 5 3 19 33 0 **8873** 0]
[0 1 4 7 174 0 6 2 **9531**]

Matriz de confusão da pasta 3 com Random Forest

2.2.2.4 Pasta 4

A precisão para esta pasta foi de 0.9927131545046765 e o erro calculado foi de 0.026681559142279047. A matriz de confusão está apresentada abaixo.

[**70114** 15 3 5 5 31 30 42 3]
[18 **18693** 0 1 24 59 22 62 4]
[13 0 **48361** 0 6 8 0 13 9]
[9 3 1 **77144** 10 0 0 3 8]
[3 42 8 15 **12609** 47 98 22 355]
[19 64 7 0 65 **13094** 12 54 0]
[41 20 6 0 123 14 **12542** 0 17]
[25 92 10 2 29 56 2 **12411** 0]
[0 8 3 8 349 2 15 2 **13221**]

Matriz de confusão da pasta 4 com Random Forest

2.2.2.5 Pasta 5

A precisão para esta pasta foi de 0.9908270121594129, e o erro calculado foi de 0.03375826054616053. A matriz de confusão está apresentada abaixo.

[**90108** 29 14 7 10 48 68 61 2]
[20 **23734** 0 5 43 109 32 130 13]
[13 0 **62252** 0 12 12 5 14 10]
[9 6 2 **99113** 19 0 0 2 14]
[4 39 10 23 **15926** 110 213 43 554]
[46 88 16 0 60 **16911** 21 69 1]
[62 31 5 0 164 29 **16206** 1 24]
[36 135 21 5 42 89 2 **15899** 0]
[4 9 8 10 582 0 38 2 **16843**]

Matriz de confusão da pasta 5 com Random Forest

Para todas as pastas, as curvas ROC geradas apresentam exatamente o mesmo gráfico que está apresentado abaixo.

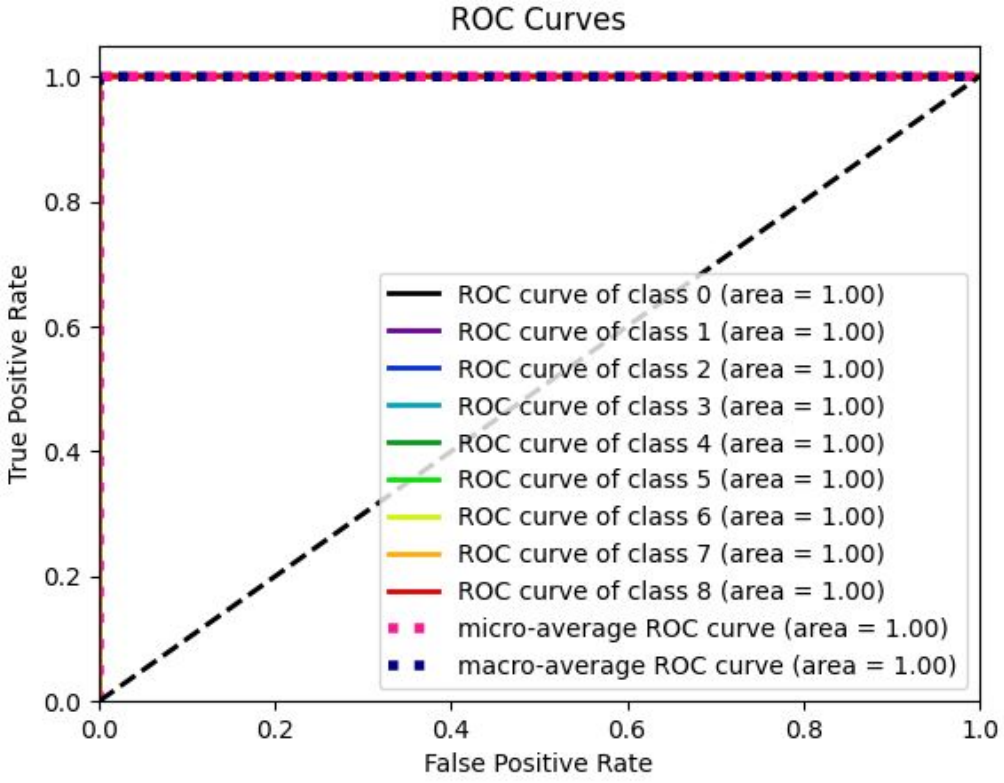


Figura 11. Curvas ROC geradas para a quinta pasta da validação cruzada.

2.2.3 Teste com 20% do *dataset* completo

Com o objetivo de validar os modelos treinados utilizando o classificador Random Forest utilizamos os 20% restantes do *dataset* como entrada. Os resultados obtidos são apresentados nas seções abaixo.

2.2.3.1 Teste com o primeiro modelo

A precisão desse teste foi de 0.6328393522293095, ou seja, aproximadamente 63%, e o erro calculado foi de 1.0388310712515787. A matriz de confusão e as curvas ROC são apresentados abaixo.

[**12699**	350	921	753	191	206	216	232	152]
[295	**2063**	163	182	158	319	215	678	105]
[1723	105	**6854**	1514	147	91	95	95	146]
[1374	133	1733	**13471**	127	57	93	74	150]
[232	142	143	167	**973**	222	363	179	514]
[213	175	118	91	167	**854**	272	1007	94]
[261	173	142	155	351	508	**926**	151	225]
[195	301	115	82	138	958	209	**732**	71]
[238	136	187	164	634	160	423	98	**1014**]

Matriz de confusão dos 20% testados no primeiro modelo treinado

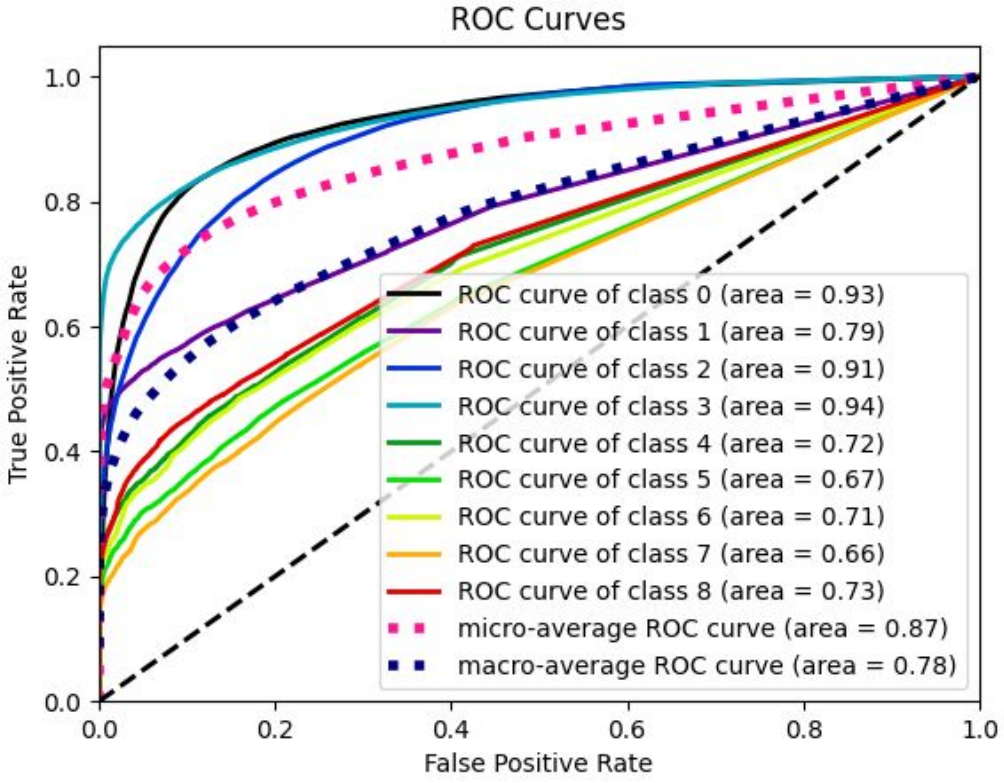


Figura 12. Curvas ROC geradas para o teste do primeiro modelo.

2.2.3.2 Teste com o melhor modelo entre aqueles treinados com validação cruzada

O modelo utilizado para esse teste foi o modelo treinado pela primeira pasta da validação cruzada que obteve uma taxa de precisão de 0.9976519371518497, acima das demais pastas.

A precisão desse teste foi de 0.5734017553114958, ou seja, aproximadamente 57%, e o erro calculado foi de 1.2924559973142775. A matriz de confusão e as curvas ROC são apresentados abaixo.

[**11931**	285	1274	1024	266	256	232	240	212]
[532	**1826**	284	372	228	243	228	265	200]
[2234	163	**6060**	1448	185	173	165	177	165]
[1779	193	1520	**12951**	181	136	151	107	194]
[462	243	282	321	**694**	228	231	220	254]
[476	241	267	322	240	**590**	238	390	227]
[471	276	268	312	211	260	**685**	186	223]
[490	291	249	276	194	415	226	**451**	209]
[501	237	356	345	273	231	234	197	**680**]

Matriz de confusão dos 20% testados no melhor modelo da validação cruzada

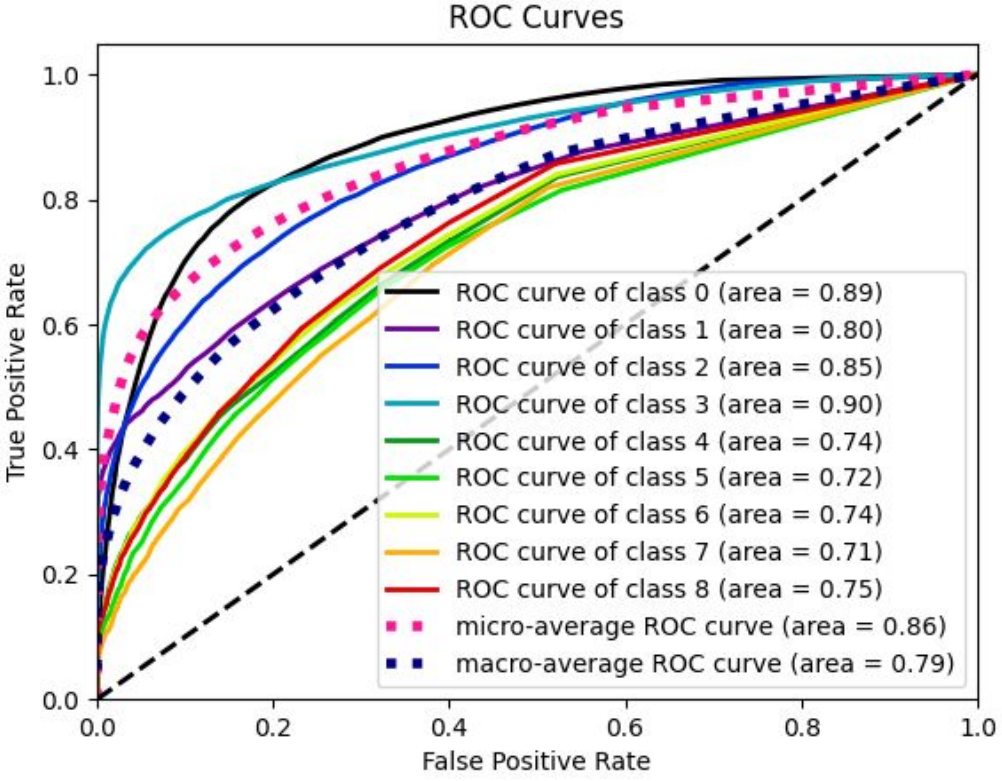


Figura 13. Curvas ROC geradas para o teste do melhor modelo entre aqueles treinados com validação cruzada.

Avaliando esta seção é possível observar que, apesar dos dois testes terem apresentado resultados bastante próximos, ou melhor, com valores nem tão distintos, o primeiro modelo treinado com 80% do *dataset* obteve melhores resultados.

2.3. MLP

O Perceptron multicamadas diferentes das técnicas de Aprendizado de máquinas acima usa a técnica de rede neural para tentar classificar as informações de entrada seguindo um padrão de treino.

Infelizmente os resultados mostram que a rede neural não é capaz de ter uma boa precisão na hora de inferir qual seria o Host. Mesmo usando um sistema de retropropagação.

Mesmo após tentativas de variar os parâmetros da RN observamos que nenhum seria satisfatoriamente comparável aos modelos de ML, então decidimos treinar e colocar no relatório o modelo default com o Adam de otimização.

Conforme solicitado na especificação do projeto esses 80% foram divididos na proporção 80/20 e as seções abaixo apresentam os resultados obtidos.

2.3.1 Treinamento 80% do *Dataset*

Nesta seção são apresentados os resultados obtidos durante o treinamento de 80% do *dataset*. Conforme especificado, esses 80% foram novamente divididos na proporção 80/20 sendo 80% para treinamento e 20% dos 80% para teste.

```
[[  51  632    2 11979    5    9    0    0    2]
 [   0  230    0  3122    0    1    0    0    0]
 [  17  274    0  8137    2    1    0    0    0]
 [  35  434    0 13326    3    2    0    2    2]
 [   0  144    0  2220    0    0    0    0    0]
 [   0  159    0  2198    0    1    0    0    0]
 [   2  142    0  2200    0    0    0    0    0]
 [   0  132    0  2063    0    0    0    1    1]
 [   1  184    0  2325    0    1    0    0    0]]
```

Matriz de confusão gerada para os 80% do *dataset*.

A precisão (a capacidade do classificador de não rotular como positiva uma amostra negativa) para o treinamento desses 80% do *dataset* foi de 0.2719515606890212, ou seja, aproximadamente 27%. Esse resultado foi obtido através da função `precision_score`.

O erro obtido através da função `mean_absolute_error` que representa a soma dos erros absolutos ao longo da duração das observações/previsões foi de 1.7850405659246233.

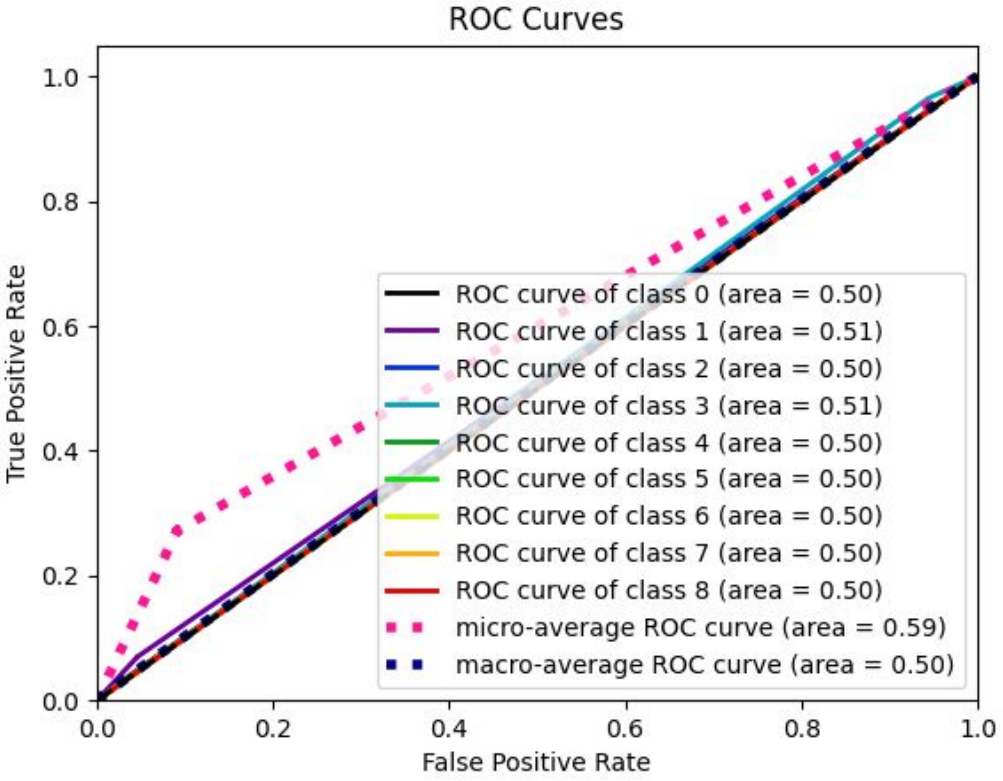


Figura 14. Curvas ROC geradas para o modelo treinado com 80% de 80% do *dataset*.

2.3.2 Treinamento feito usando validação cruzada com 5 pastas

2.3.2.1 Pasta 1

A precisão para esta pasta foi de 0.04601175273054273 e o erro calculado foi de 4.398156532382457. A matriz de confusão está apresentada abaixo.

```
[[  12  727    0  468    0  245    0 38596  194]
 [  13  352    0  172    0   13    0 10015   31]
 [  32  233    0  422    0   51    0 27002   69]
 [  68  396    0  250    0   65    0 43079  102]
 [   6  203    0  203    0   11    0  6965   66]
 [   6  220    0  231    0   14    0  7152   55]
 [   4  199    0  100    0   12    0  6988   47]
 [   6  213    0  249    0   14    0  6703   53]
 [   6  228    0   96    0   13    0  7426   37]]
```

Matriz de confusão da pasta 1 com Random Forest

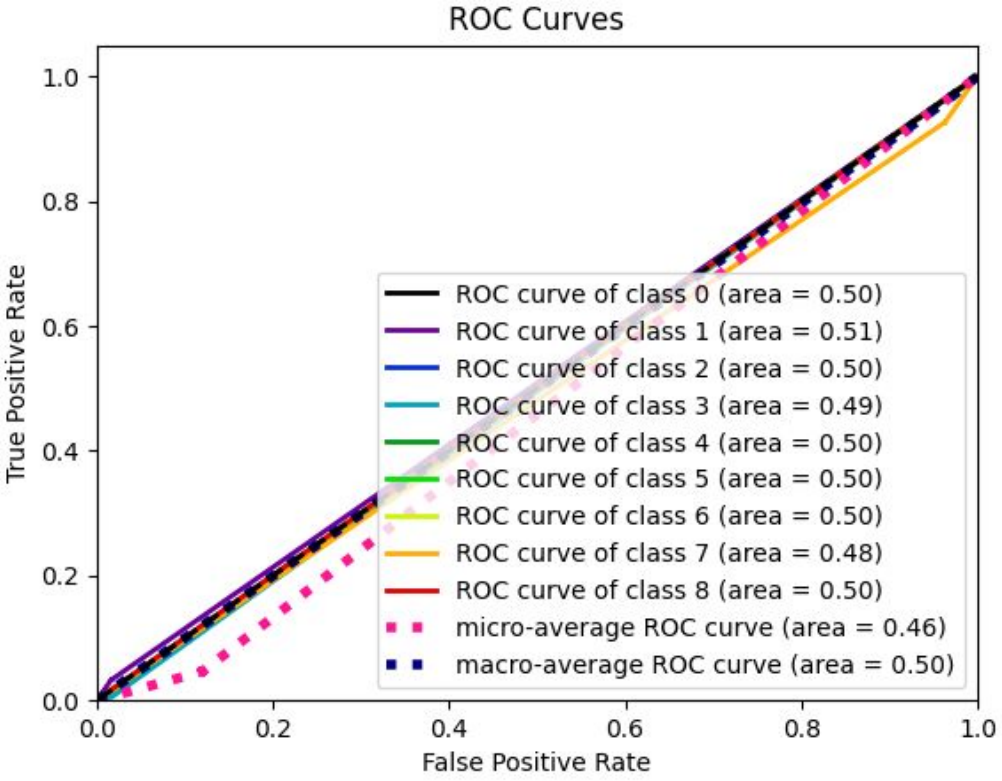


Figura 15. Curvas ROC geradas para a primeira pasta da validação cruzada.

2.3.2.2 Pasta 2

A precisão para esta pasta foi de 0.17146372078209987 e o erro calculado foi de 1.8522603086184608. A matriz de confusão está apresentada abaixo.

```
[[   87     5 38060    75    49    44   477  1273     0]
 [    2     0 10059    51     6    12   125   447     0]
 [   31     8 26889    79    15    43    99   498     0]
 [  138    37 42957    88    17    34   164   766     2]
 [    2     1  6994    83     4     7    50   340     0]
 [    1     4  7190   106     8    13    57   350     0]
 [    1     7  6989    17     1    15    47   320     0]
 [    1     2  6704    98     1     9    52   329     0]
 [    1     8  7223    28     4     9    72   348     0]]
```

Matriz de confusão da pasta 2 com Random Forest

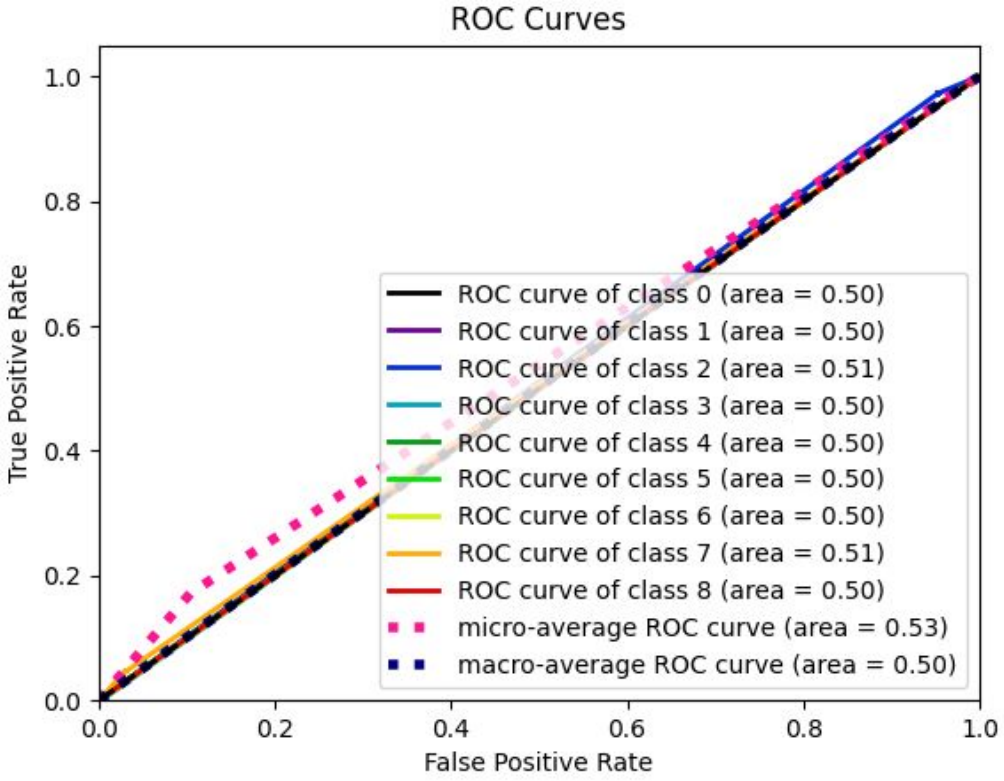


Figura 16. Curvas ROC geradas para a segunda pasta da validação cruzada.

2.3.2.3 Pasta 3

A precisão para esta pasta foi de 0.2633186164001174 e erro calculado foi de 1.873511393654025. A matriz de confusão está apresentada abaixo.

```
[[ 165    9    0 36613  482    0  2921    0    0]
 [   4    5    0  9220   75    0  1319    0    0]
 [  43   14    0 25885  120    0  1673    0    0]
 [ 131    9    0 41200  187    0  2379    4    0]
 [   7    9    0  6688   40    0   796    0    0]
 [   3    2    0  6815   68    0   808    0    0]
 [   2    4    0  6648   50    0   756    0    0]
 [   2    8    0  6359   53    0   760    0    0]
 [   2    1    0  6566   56    0  1171    1    0]]
```

Matriz de confusão da pasta 3 com Random Forest

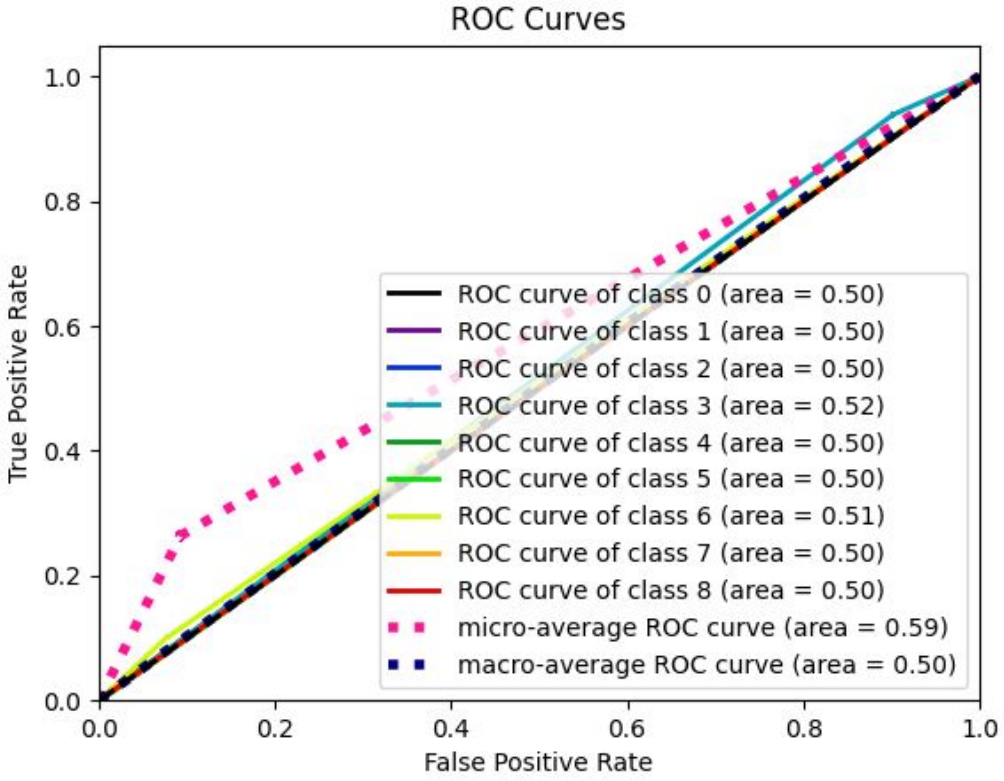


Figura 17. Curvas ROC geradas para a terceira pasta da validação cruzada.

2.3.2.4 Pasta 4

A precisão para esta pasta foi de 0.24319159698500623 e o erro calculado foi de 2.6651908101390718. A matriz de confusão está apresentada abaixo.

```
[[38302    33     0   620     0    12     0   646   542]
 [10079     7     0   280     0     2     0   282   138]
 [26704     3     0   611     0    37     0   207   159]
 [42959     3     0   330     0   122     0   325   238]
 [ 6933     3     0   260     0     1     0   216    70]
 [ 7068     6     0   326     0     0     0   217    83]
 [ 6956     1     0   151     0     0     0   206    77]
 [ 6598     0     0   303     0     1     0   213    75]
 [ 7303     3     0   123     0     2     0   206    91]]
```

Matriz de confusão da pasta 4 com Random Forest

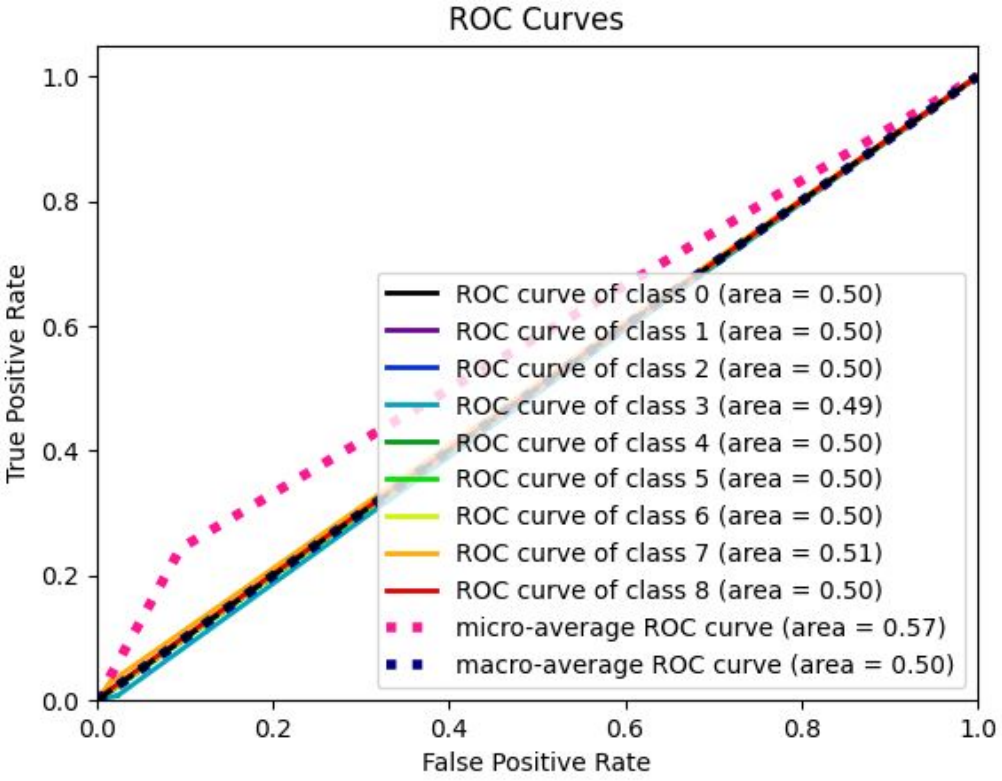


Figura 18. Curvas ROC geradas para a quarta pasta da validação cruzada.

2.3.2.5 Pasta 5

A precisão para esta pasta foi de 0.24619535011521673, e o erro calculado foi de 2.05523533562726. A matriz de confusão está apresentada abaixo.

```
[[ 185    0    1 33487   57  3953   17    0  2448]
 [   5    0    1  8259    4  1413    5    0  1040]
 [  42    0    0 23317    0  3158    5    0  1147]
 [ 155    0    2 37503    7  4665   10    1  1779]
 [   2    0    0  6139    0   862    1    0   542]
 [   1    0    0  6125    2   848    2    0   631]
 [   3    0    0  5930    2   790    2    0   571]
 [   3    0    1  5809    1   838    4    0   566]
 [   4    0    2  5817    2  1081    0    0   886]]
```

Matriz de confusão da pasta 5 com Random Forest

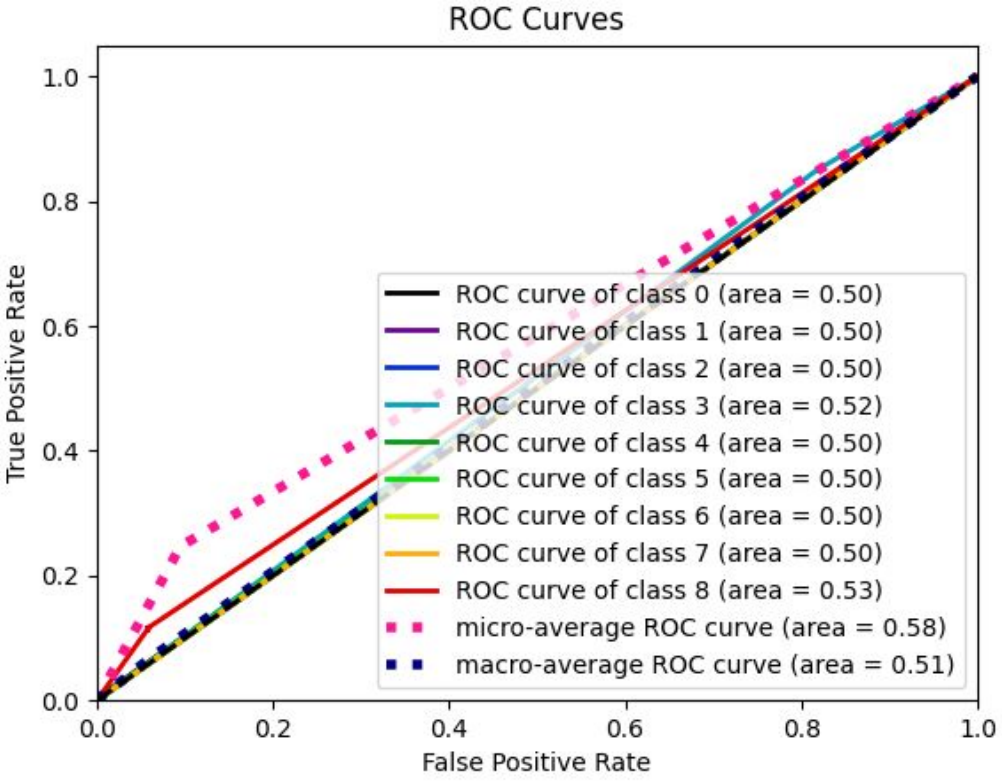


Figura 19. Curvas ROC geradas para a quinta pasta da validação cruzada.

2.3.3 Teste com 20% do *dataset* completo

Com o objetivo de validar os modelos treinados utilizando o classificador MLP Classifier utilizamos os 20% restantes do *dataset* como entrada. Os resultados obtidos são apresentados nas seções abaixo.

2.3.3.1 Teste com o primeiro modelo

A precisão desse teste foi de 0.2697232746630857, ou seja, aproximadamente 27%, e o erro calculado foi de 1.7813374258628683. A matriz de confusão e as curvas ROC são apresentados abaixo.

```
[[  58  808    0 14835    4   11    0    3    1]
 [   0  278    0  3899    0    0    0    0    1]
 [  14  367    0 10382    5    1    0    1    0]
 [  50  620    0 16534    1    2    0    5    0]
 [   0  171    0  2764    0    0    0    0    0]
 [   2  197    0  2789    0    2    0    1    0]
 [   5  195    0  2690    0    1    0    1    0]
 [   1  172    0  2628    0    0    0    0    0]
 [   1  226    0  2825    0    0    0    2    0]]
```

Matriz de confusão dos 20% testados no primeiro modelo treinado

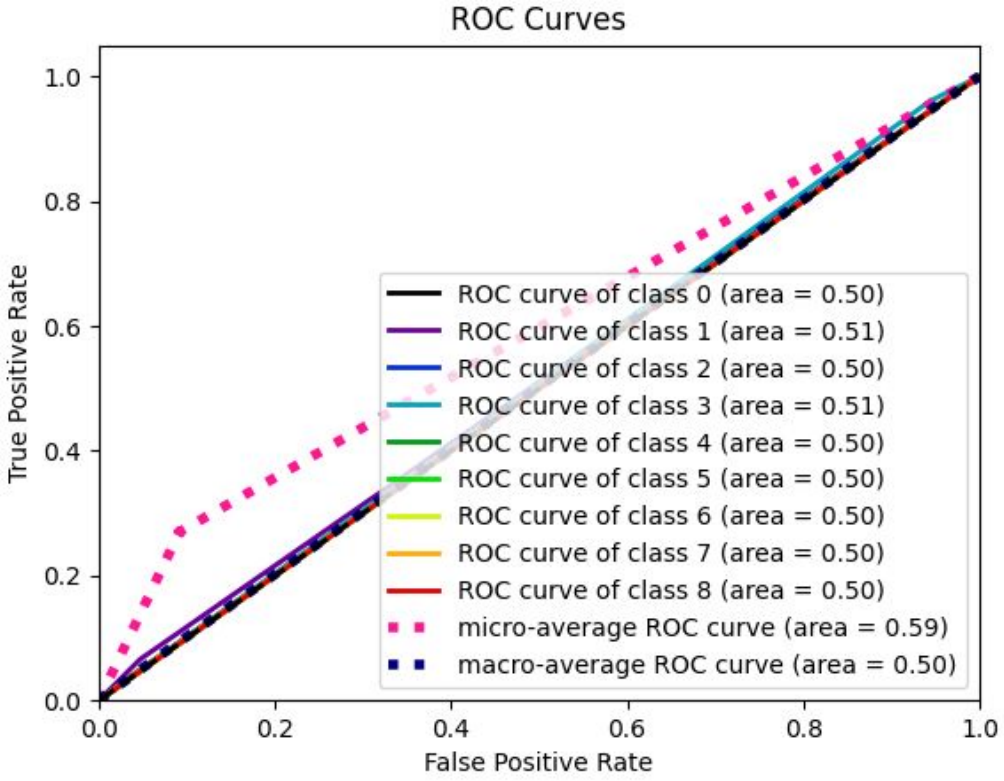


Figura 20. Curvas ROC geradas para o teste do primeiro modelo.

2.3.3.2 Teste com o melhor modelo entre aqueles treinados com validação cruzada

O modelo utilizado para esse teste foi o modelo treinado pela terceira pasta da validação cruzada que obteve uma taxa de precisão de 0.2633186164001174, acima das demais pastas.

A precisão desse teste foi de 0.26348856169967866, ou seja, aproximadamente 26%, e o erro calculado foi de 1.8773679919428325. A matriz de confusão e as curvas ROC são apresentados abaixo.

```
[[  52    5    0 14331  185    0 1147    0    0]
 [   0    0    0  3649   27    0  502    0    0]
 [  18    6    0 10014   52    0  680    0    0]
 [  48    4    0 16100   79    0  978    3    0]
 [   0    3    0  2628   15    0  289    0    0]
 [   3    2    0  2647   25    0  314    0    0]
 [   3    1    0  2557   15    0  315    1    0]
 [   1    2    0  2499   24    0  275    0    0]
 [   3    0    0  2579   24    0  448    0    0]]
```

Matriz de confusão dos 20% testados no melhor modelo da validação cruzada

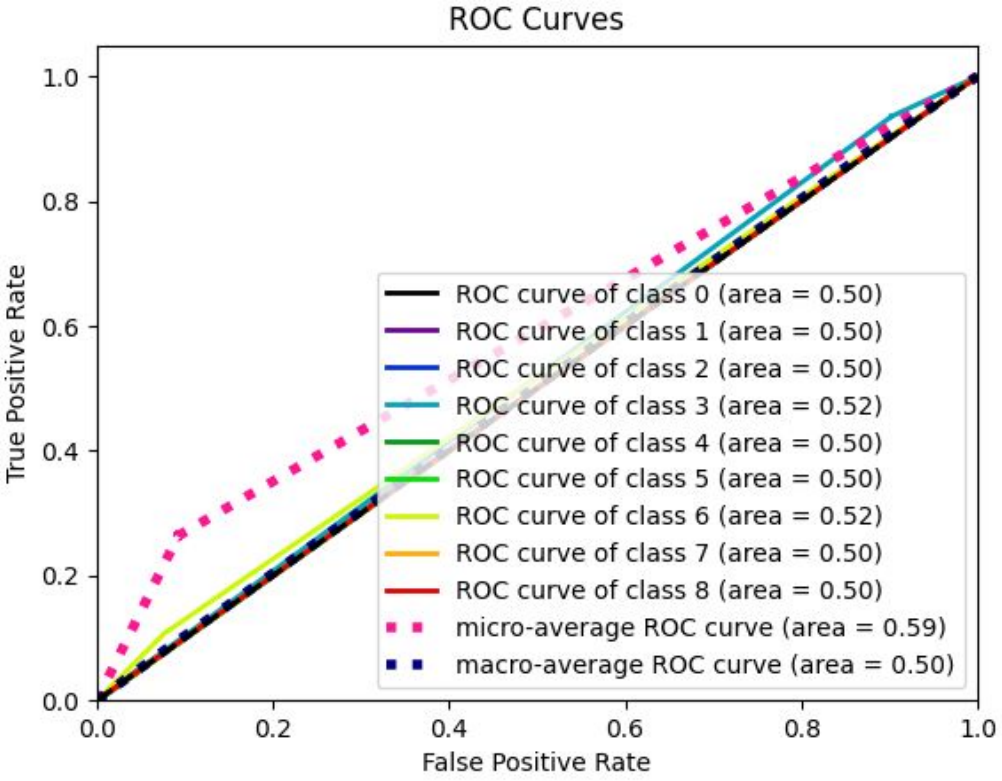


Figura 21. Curvas ROC geradas para o teste do melhor modelo entre aqueles treinados com validação cruzada.

Avaliando esta seção é possível observar que, apesar dos dois testes terem apresentado resultados bastante próximos, ou melhor, com valores nem tão distintos, o primeiro modelo treinado com 20% do *dataset* obteve melhores resultados. Porém nenhum satisfatório.

References

COLE, Eric; NORTHCUTT, Stephen. Honeypots: A Security manager's guide to Honeypots.**https://www.sans.edu/cyber-research/security-laboratory/article/honeypots-guide. Last accessed: August**, v. 2, 2018.