Exploração e classificação de um dataset de URLs maliciosas

CI1030 - Ciência de Dados para Segurança
Aluno: Christian Debovi Paim Oliveira (GRR20186713)
Professor: André Gregio

Dataset - Pesquisa

- **Paper:** Detecting Malicious URLs Using Lexical Analysis.
- **Autores:**
  - Mohammad Saiful Islam Mamun
  - Mohammad Ahmad Rathore
  - Arash Habibi Lashkari
  - Natalia Stakhanova
  - Ali A. Ghorbani
- **Universidade:**
  - University of New Brunswick , Fredericton, NB, Canada
- **Objetivo:**
  - Avaliar o uso de características léxicas para a classificação de URLs
  - Avaliar uso de técnicas de ofuscação nas urls coletadas

Datasets - URLs

Arquivos texto com urls:

- Total de 165.366 URLs

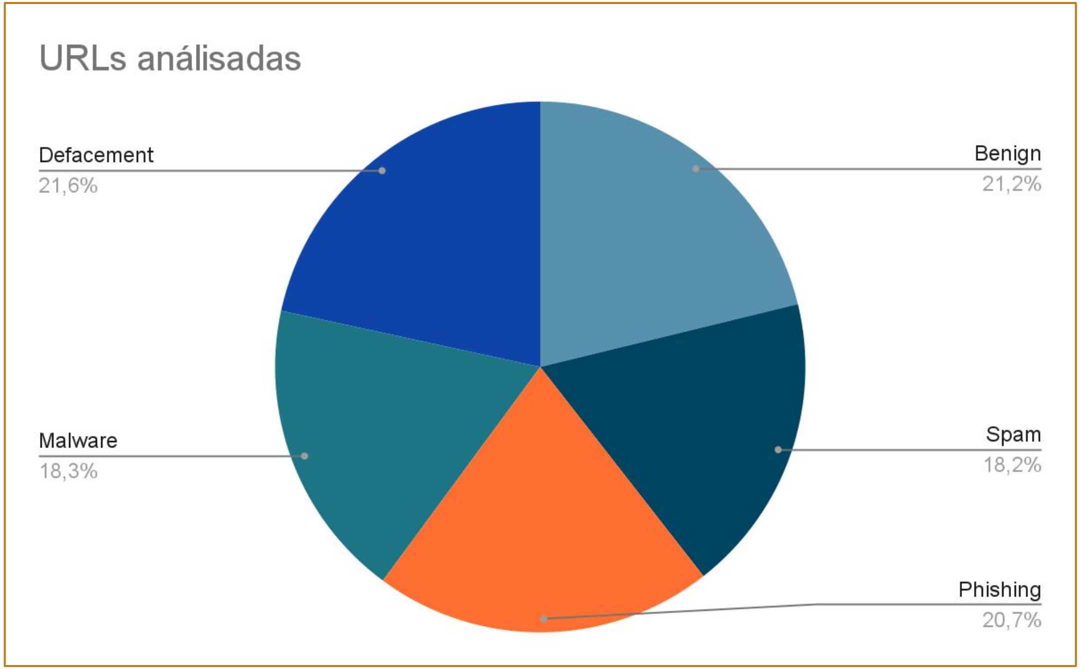
Labels/Classes:

- **Benign:** Alexa top sites + crawler + VirusTotal (35.378 URLs).
- **Spam:** dataset WEBSPAM-UK2007 (12.000 URLs).
- **Phishing:** repositório OpenPhish (9965 URLs).
- **Malware:** lista DNS-BH (11.566 URLs).
- **Defacement:** em Alexa top sites (96.457 URLs).

Dataset - Informações de URLs

CSVs com informações léxicas:

- Extraídas de 36.707 URLs do total
- 79 atributos léxicos + label



Dataset - Atributos

Tipos:

- **Entropy:** variação nos tokens em certas partes da url.
    - Entropy_Domain, Entropy_Extension

- **CharacterContinuityRate:** soma da continuidade dos tokens divididos pelo tamanho da URL.
    - **Ex:** abc567ti = (3 + 3 + 1)/9 = 0.77

- **Ratios:** número de tokens de uma parte da URL divididos pelo número de tokens da outra.
    - argPathRatio, argUrlRatio, argDomainRatio, domainUrlRatio, pathUrlRatio, PathDomainRatio.

- **NumberRate:** proporção de dígitos nas partes da URL.
    - NumberRate_Domain, NumberRate_DirectoryName, NumberRate_FileName, NumberRate_URL, NumberRate_AfterPath.

- Outros atributos relacionados ao tamanho e contagem de tokens, dígitos e símbolos de diferentes partes da URL.

Datasets - Seleção de Atributos (WEKA)

Infogain:

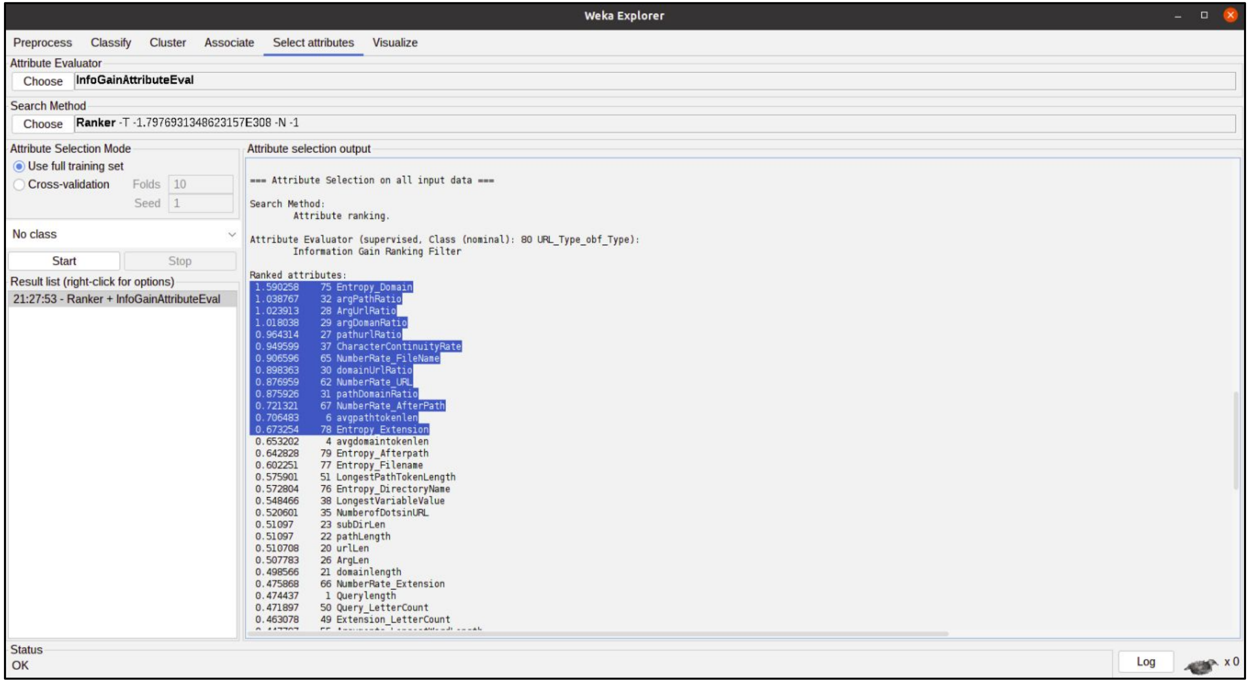
Determina o “peso” de um atributo através da medição do ganho de informação em respeito à classe

InfoGain(Class, Attribute) =

H(Class) - H(Class | Attribute).

Ranker

Ordena atributos de acordo com sua avaliação.



Dataset - Atributos escolhidos

Escolhidos 12 de 79 atributos

- Entropy_Domain
- argPathRatio
- ArgUrlRatio
- argDomanRatio
- pathurlRatio
- CharacterContinuityRate
- NumberRate_FileName
- domainUrlRatio
- NumberRate_URL
- pathDomainRatio
- NumberRate_AfterPath
- avgpathtokenlen

Dataset - Reavaliando características

Removidos 3 características (restam 9)

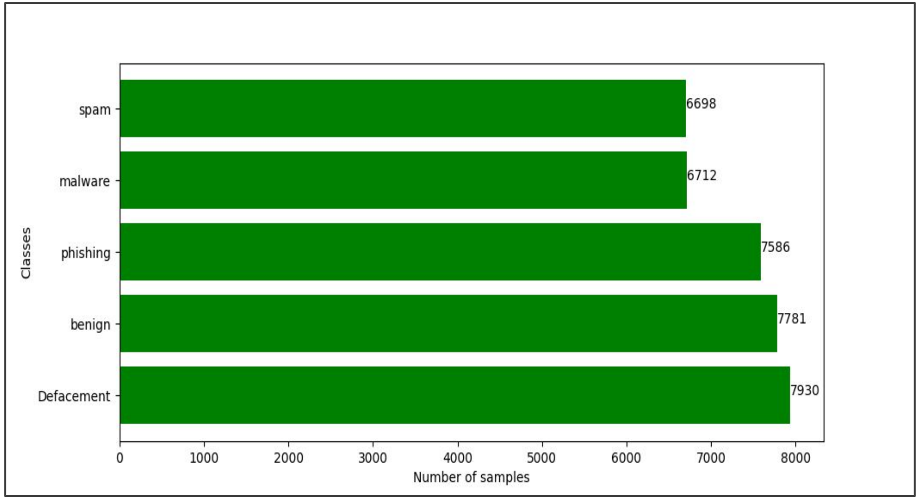
- Entropy_Domain (Fórmula não especificada)
- CharacterContinuityRate (Maneira de calcular ambígua)
- avgpathtokenlen(Não especifica relação da média)

Dataset - Atributos escolhidos

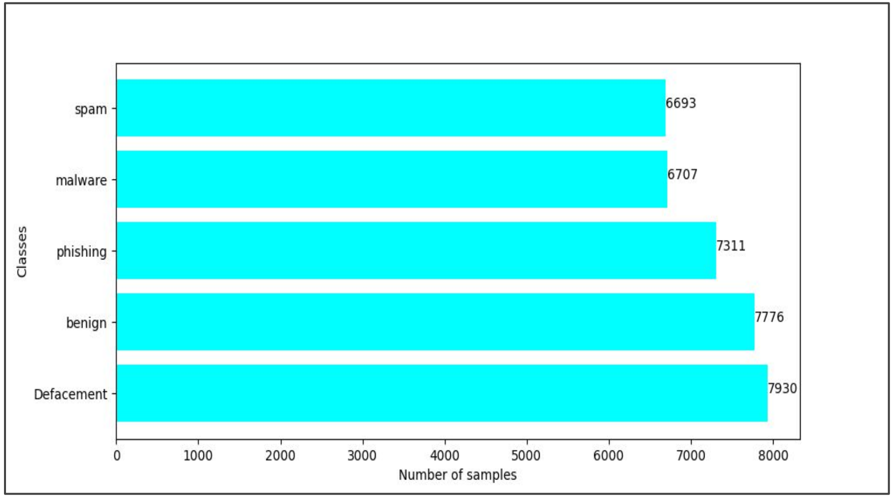
	pathurlRatio	ArgUrlRatio	argDomanRatio	domainUrlRatio	pathDomainRatio	argPathRatio	NumberRate_URL	NumberRate_FileName	NumberRate_AfterPath
count	36417.000000	36417.000000	36417.000000	36417.000000	36417.000000	36417.000000	36417.000000	36417.000000	36417.000000
mean	0.674727	0.237958	2.176043	0.227918	4.705858	0.326423	0.095932	0.111093	-0.471727
std	0.150243	0.272202	6.445867	0.120900	6.493607	0.338607	0.095880	0.321863	0.600680
min	0.041000	0.000000	0.000000	0.011000	0.044000	0.000000	0.000000	-1.000000	-1.000000
25%	0.604000	0.029000	0.125000	0.147000	2.095000	0.044000	0.023000	0.000000	-1.000000
50%	0.705000	0.050000	0.200000	0.197000	3.571000	0.111000	0.065000	0.070000	-1.000000
75%	0.778000	0.518000	2.667000	0.288000	5.286000	0.703000	0.149000	0.214000	0.104000
max	0.985000	0.975000	92.533000	0.930000	93.467000	0.990000	0.762000	1.000000	1.000000

Dataset - Distribuição

Todos os dados



Infogain + Dropnan



Experimento - Máquina

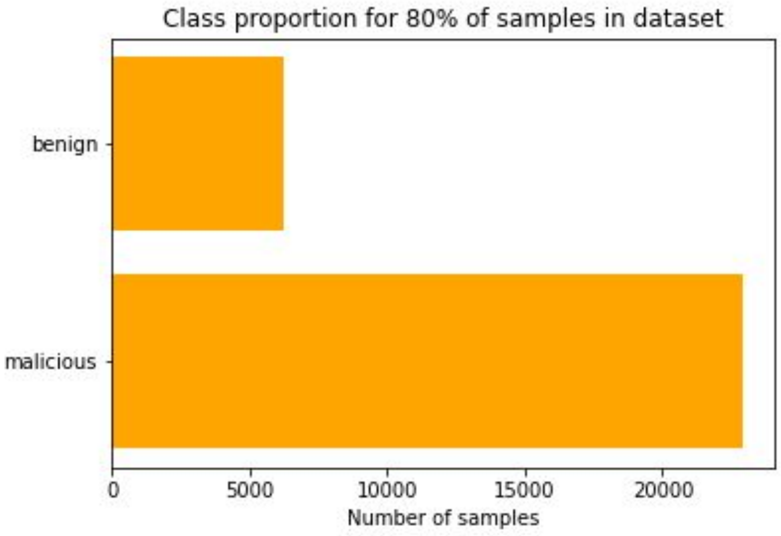
- **Google Colab**
  - **Processador:** Intel(R) Xeon(R) CPU @ 2.20GHz
  - **RAM:** 12 GB
  - **OS:** Linux Ubuntu 18.04.5 LTS (Bionic Beaver)

Experimento - Classes

- Transformar em problema de classificação binário:
  - **Classe positiva:** phishing, malware, Defacement, spam -> 'malicious'
  - **Classe negativa:** 'benign'
- Separação do dataset em 80% e 20%:
  - Treino e validação com 80% das amostras
  - Teste final com 20% das amostras
- Nos 80% das amostras, temos:

```
malicious    22899

benign        6234
```



Experimento - GridSearch

- **Classificadores:** Knn, RandomForest, MLPClassifier
- **Hiperparâmetros:**
    - k = [1, 3 ,5]
    - n_estimators (n_trees) = [50, 100]
    - epocs = [1000, 5000]
    - random_state fixo
- **Métrica:**
    - Otimizar Recall
- **Cross-validation:**
    - Default = K-fold (k=5)

Experimento - Resultados do GridSearch

```
KNeighborsClassifier(n_neighbors=1)
Score: 0.8967506835891944
Best K: 1
```

```
RandomForestClassifier(n_estimators=50, random_state=5)
Score: 0.9328934707644783
Best n_trees: 50
```

```
MLPClassifier(max_iter=1000, random_state=5)
Score: 0.8513734548086107
Best epocs: 1000
```

```python
# Exemplo de execucao do GridSearch
gs = GridSearchCV(
   estimator=KNeighborsClassifier(),
   param_grid={'n_neighbors':[1, 3, 5]},
   scoring='recall_macro',
   refit='recall_macro'
)
gs.fit(X, Y)
k = gs.best_params_['n_neighbors']
print(gs.best_estimator_)
print(gs.best_score_)
print("Best K:", k)
```

Resultados e Validação

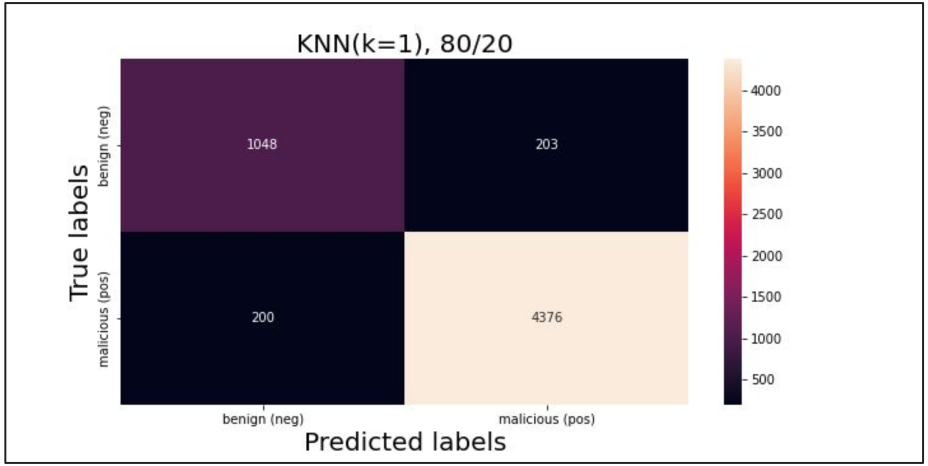
Resultado Experimento X Resultado Paper

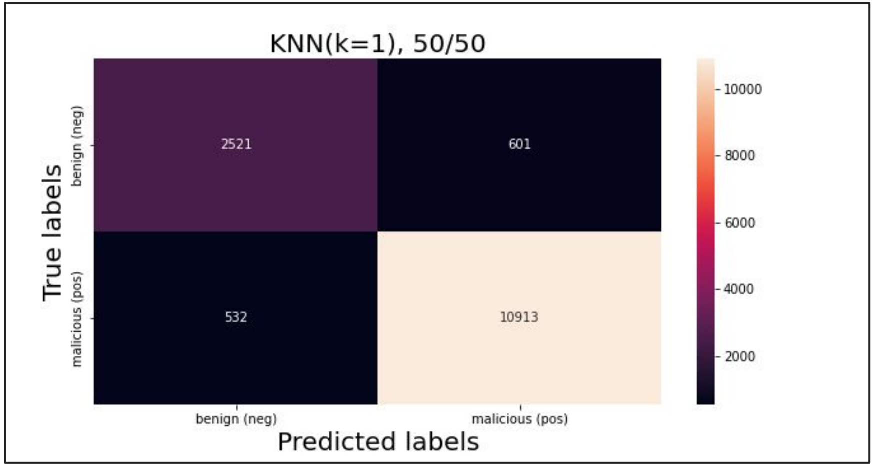
	classifier	accuracy	precision	recall	f1-score	fit_time
0	KNN(k=1), 80/20	0.930839	0.897705	0.897012	0.897358	0.043961
1	KNN(k=1), 50/50	0.922221	0.886774	0.880506	0.883585	0.026628
2	KNN(k=1), Fold 1	0.931354	0.894435	0.898257	0.896327	0.049317
3	KNN(k=1), Fold 2	0.935130	0.906815	0.903144	0.904961	0.055274
4	KNN(k=1), Fold 3	0.926720	0.894427	0.887804	0.891055	0.048658
5	KNN(k=1), Fold 4	0.927738	0.890813	0.894255	0.892518	0.050014
6	KNN(k=1), Fold 5	0.932029	0.897723	0.900511	0.899107	0.050552
7	RandomForest(n_trees=50), 80/20	0.953321	0.933607	0.927008	0.930255	1.355263
8	RandomForest(n_trees=50), 50/50	0.951054	0.929669	0.924128	0.926861	0.842649
9	RandomForest(n_trees=50), Fold 1	0.956581	0.935589	0.932131	0.933846	1.334119
10	RandomForest(n_trees=50), Fold 2	0.958469	0.941387	0.936903	0.939120	1.386449
11	RandomForest(n_trees=50), Fold 3	0.955723	0.936456	0.932428	0.934422	1.337093
12	RandomForest(n_trees=50), Fold 4	0.950566	0.926357	0.925854	0.926105	1.344452
13	RandomForest(n_trees=50), Fold 5	0.955544	0.933033	0.934806	0.933915	1.346822
14	MLP(max_iter=1000), 80/20	0.889995	0.842254	0.823669	0.832393	22.908726
15	MLP(max_iter=1000), 50/50	0.891536	0.843824	0.827086	0.834997	17.382036
16	MLP(max_iter=1000), Fold 1	0.900463	0.858799	0.828867	0.842497	31.123947
17	MLP(max_iter=1000), Fold 2	0.903381	0.854630	0.870732	0.862253	37.121134
18	MLP(max_iter=1000), Fold 3	0.889480	0.828460	0.883558	0.850301	32.187398
19	MLP(max_iter=1000), Fold 4	0.889633	0.856566	0.795453	0.820288	24.470428
20	MLP(max_iter=1000), Fold 5	0.902163	0.858598	0.844884	0.851451	34.114614

Dataset	Algorithm	Result	
		Pr	Re
Spam	C4.5	0.98	0.98
	KNN	0.98	0.98
	RF	**0.99**	**0.99**
Phishing	C4.5	0.97	0.97
	KNN	0.97	0.97
	RF	**0.99**	**0.99**
Malware	C4.5	0.98	0.98
	KNN	0.98	0.98
	RF	**0.99**	**0.99**
Defacement	C4.5	0.99	0.99
	KNN	0.99	0.99
	RF	**0.99**	**0.99**

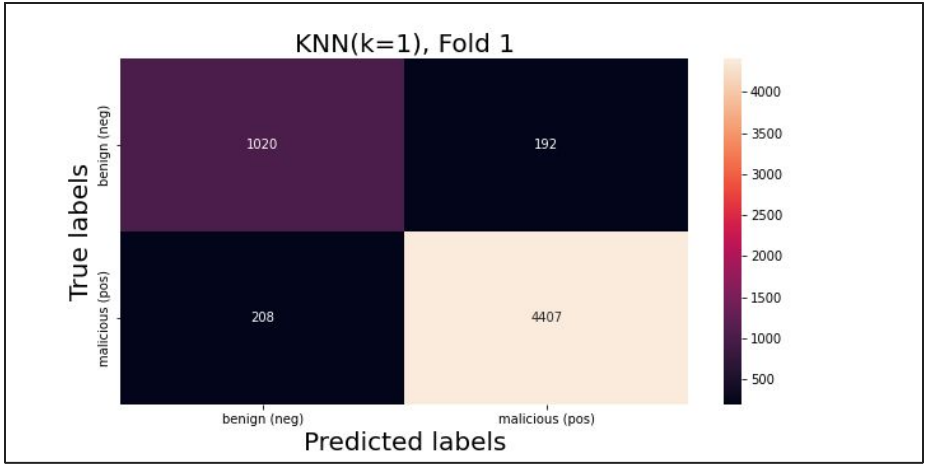
(a) Lexical Features

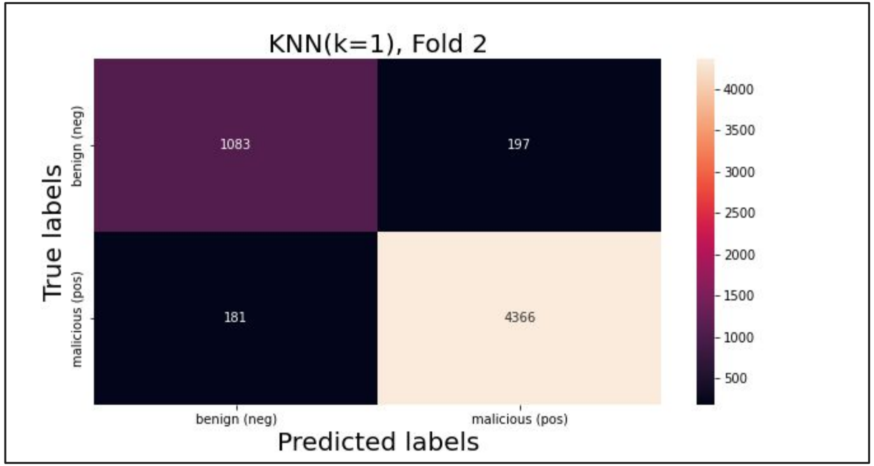
Experimento - Matrizes de Confusão (KNN)



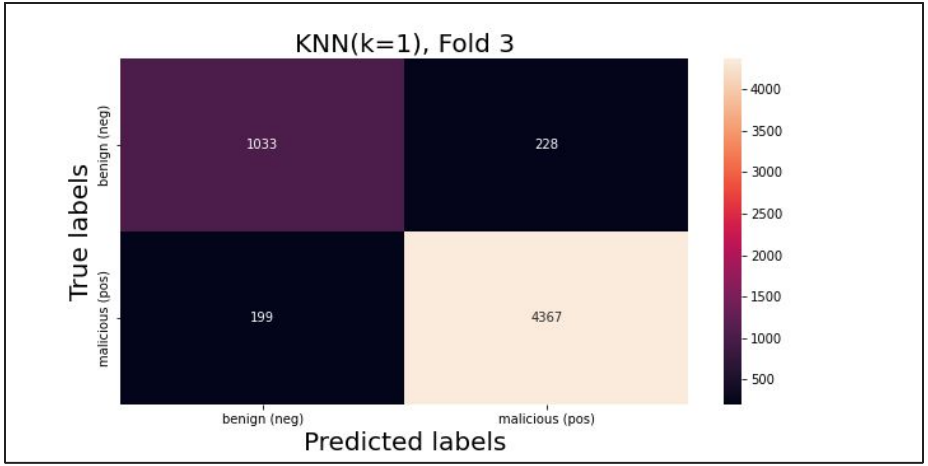


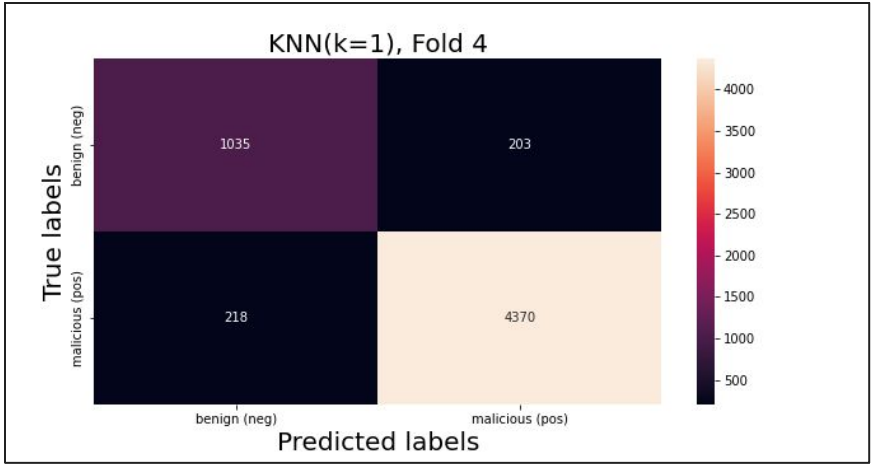
Experimento - Matrizes de Confusão (KNN)



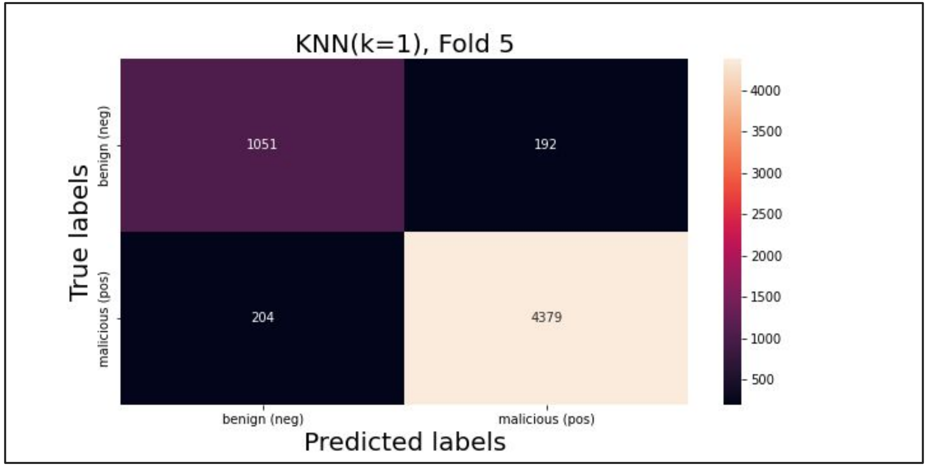


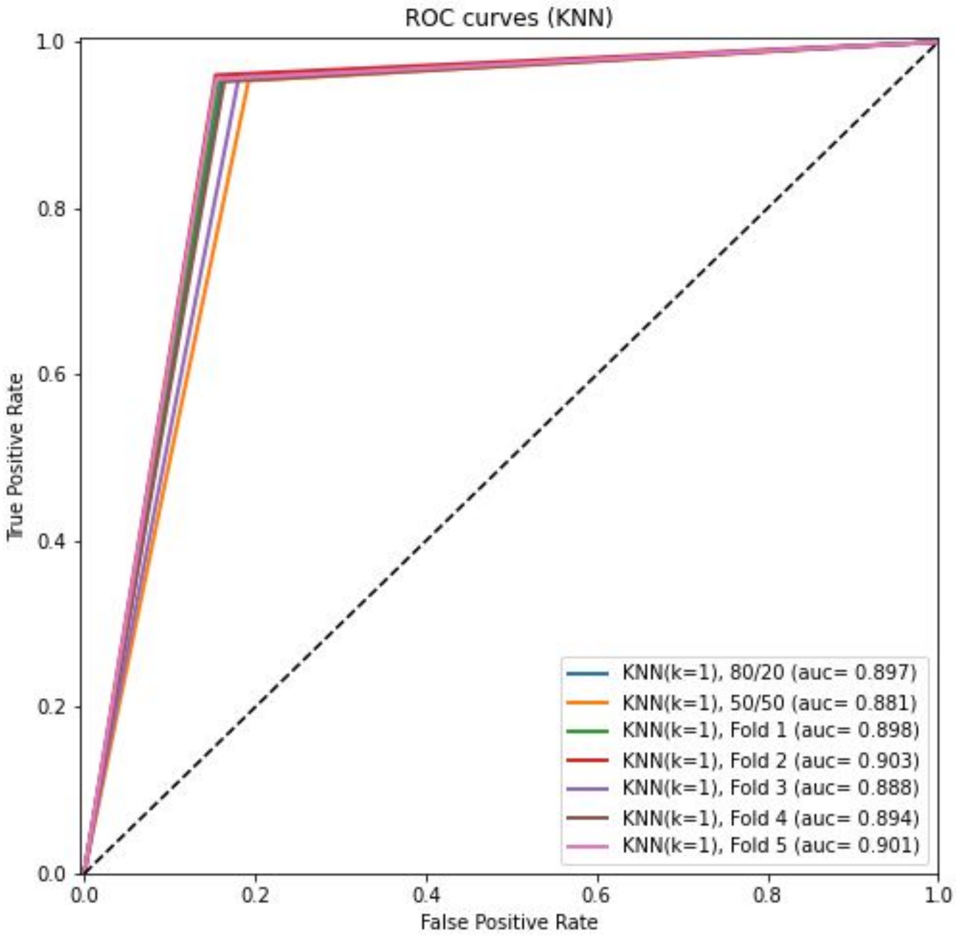
Experimento - Matrizes de Confusão (KNN)



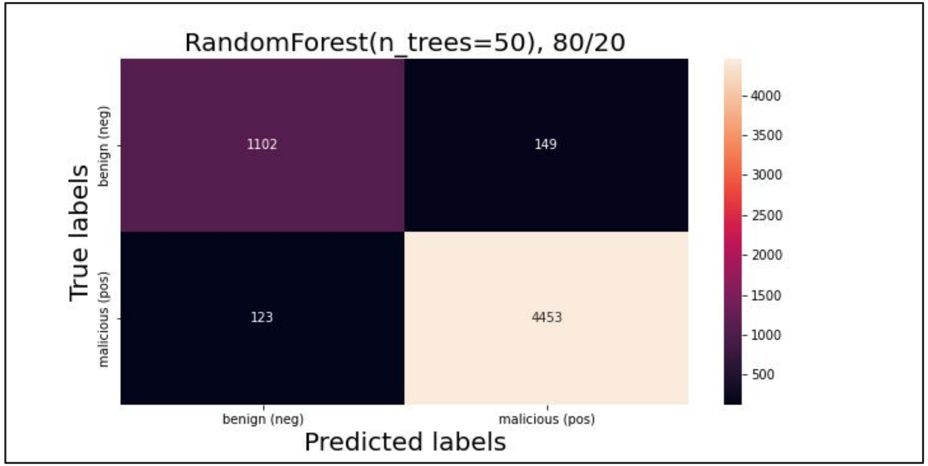


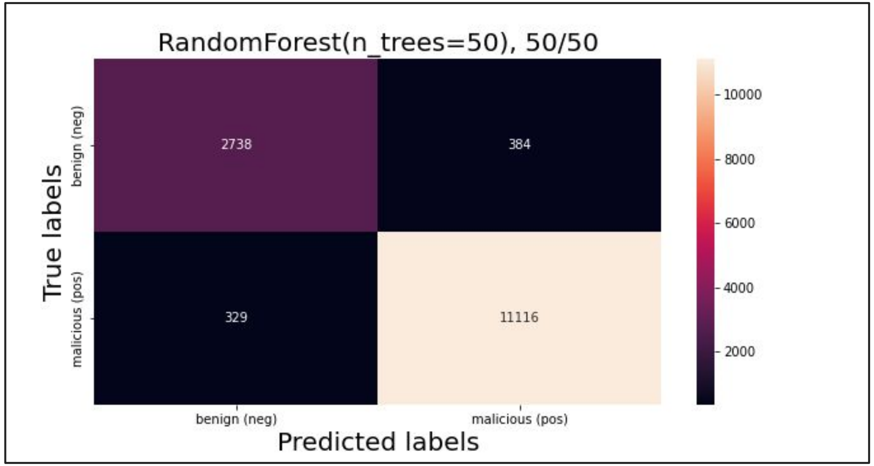
Experimento - Matrizes de Confusão (KNN)



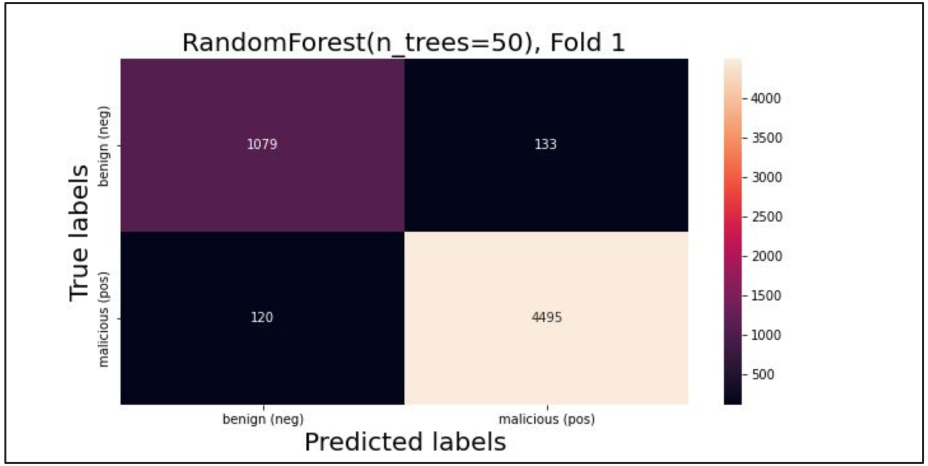

ROC curves (KNN)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
KNN(k=1), 80/20 (auc= 0.897)
KNN(k=1), 50/50 (auc= 0.881)
KNN(k=1), Fold 1 (auc= 0.898)
KNN(k=1), Fold 2 (auc= 0.903)
KNN(k=1), Fold 3 (auc= 0.888)
KNN(k=1), Fold 4 (auc= 0.894)
KNN(k=1), Fold 5 (auc= 0.901)

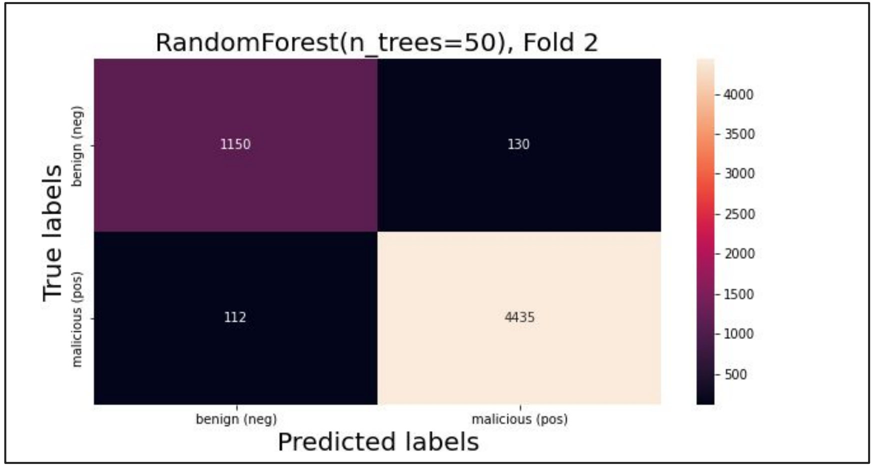
Experimento - Matrizes de Confusão (RF)



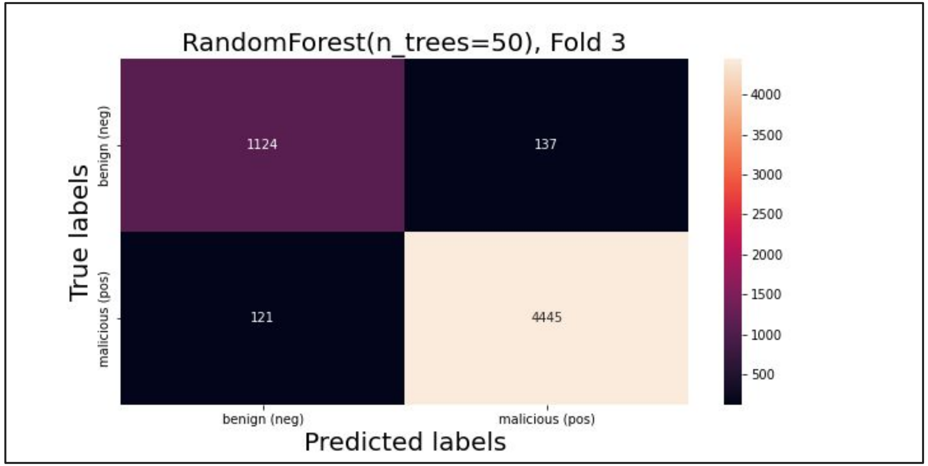


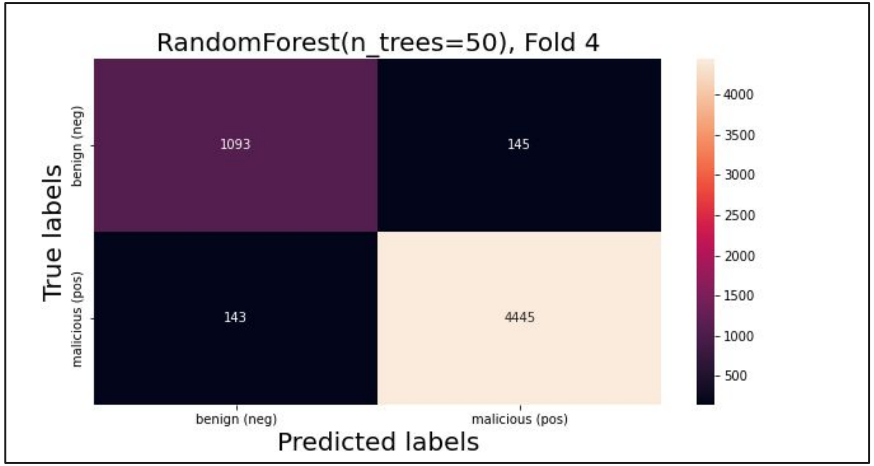
Experimento - Matrizes de Confusão (RF)



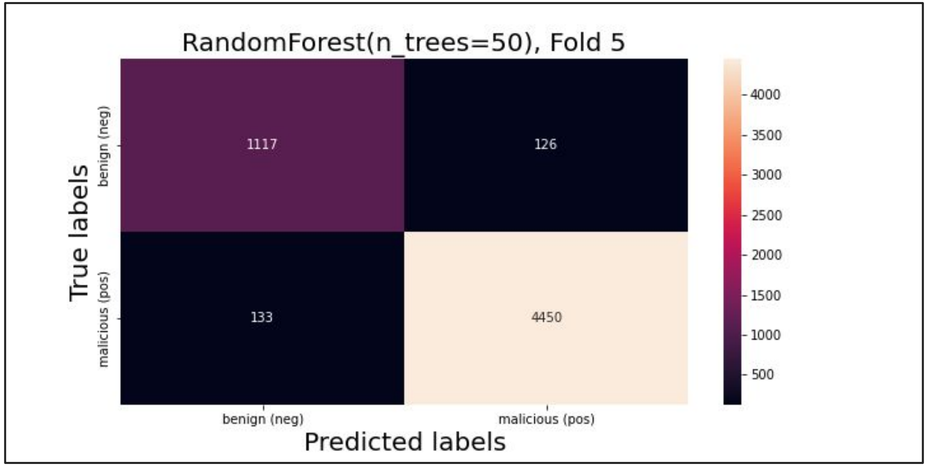


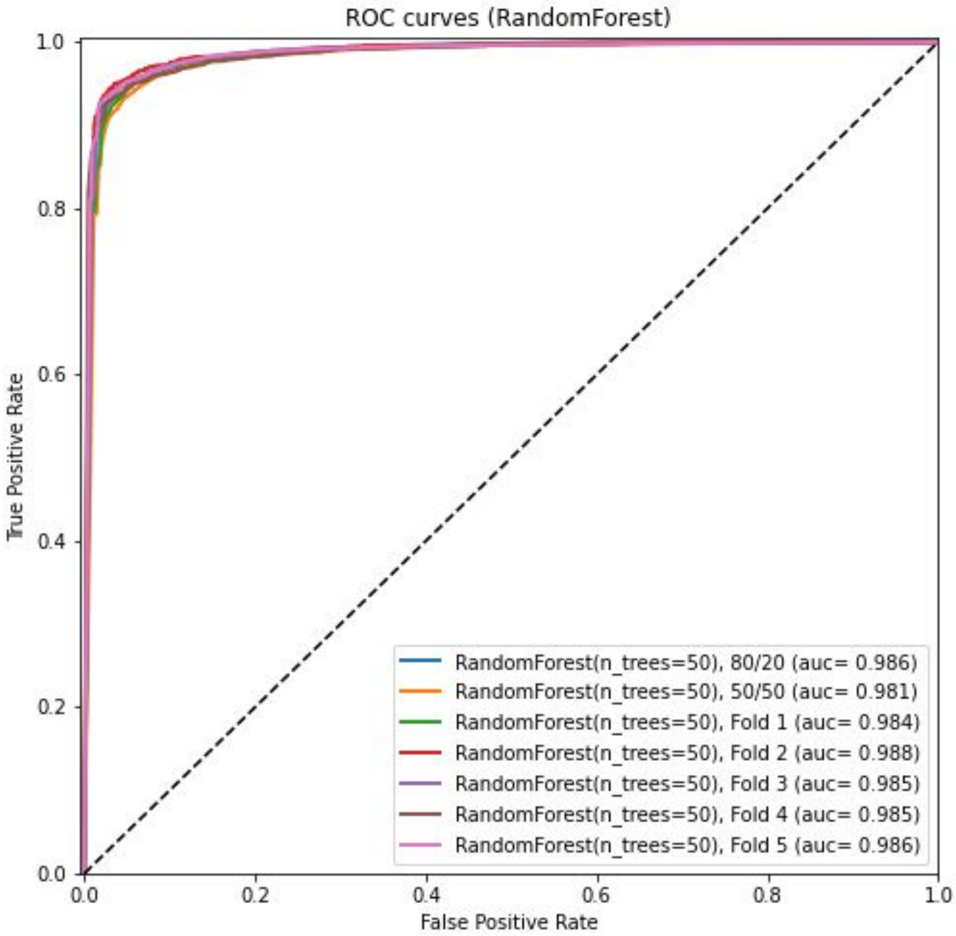
Experimento - Matrizes de Confusão (RF)



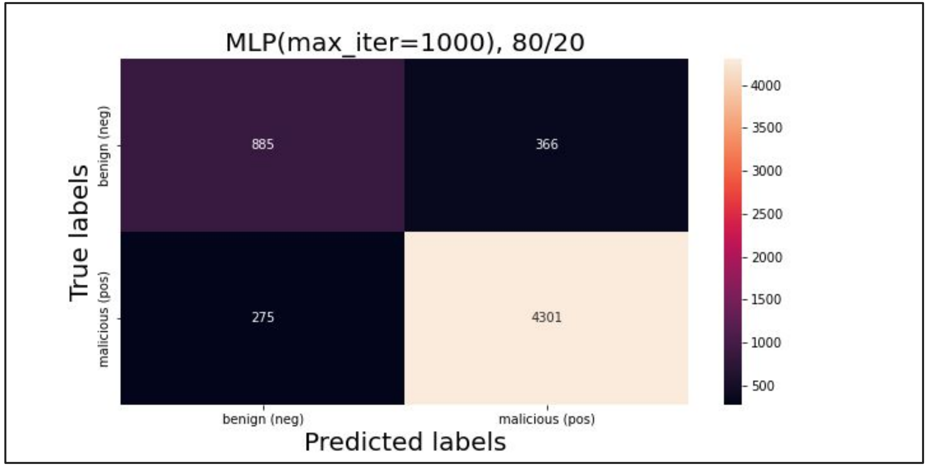


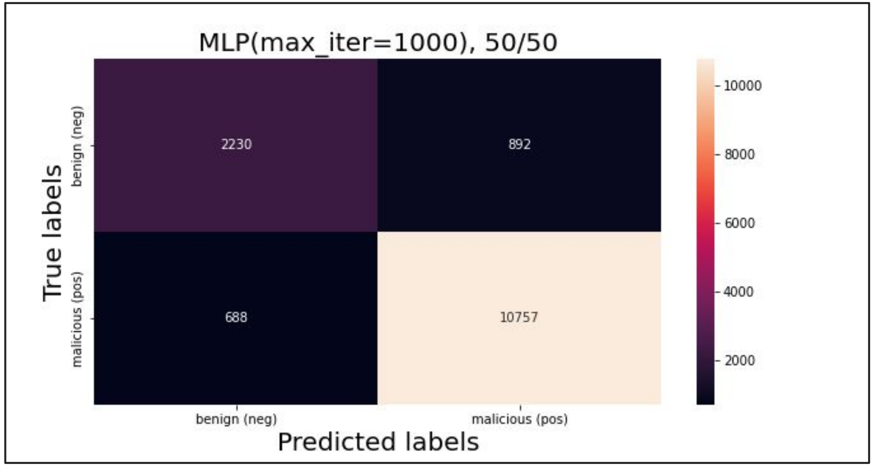
Experimento - Matrizes de Confusão (RF)



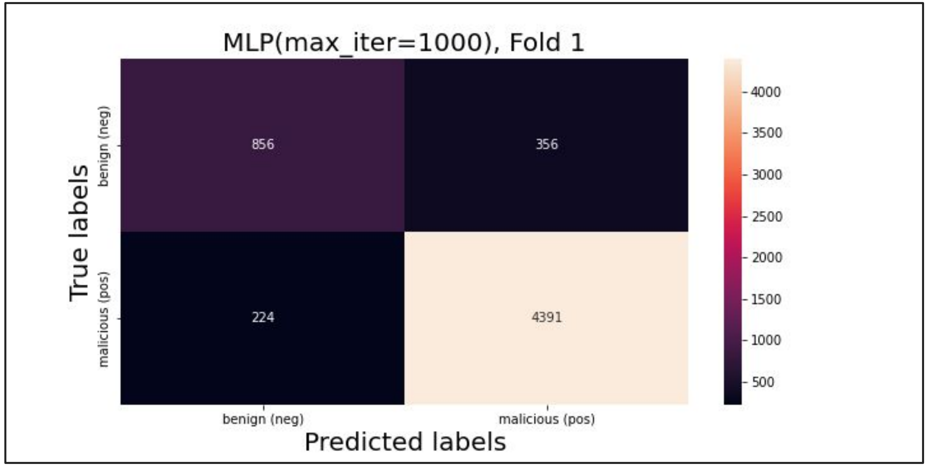

ROC curves (RandomForest)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
RandomForest(n_trees=50), 80/20 (auc= 0.986)
RandomForest(n_trees=50), 50/50 (auc= 0.981)
RandomForest(n_trees=50), Fold 1 (auc= 0.984)
RandomForest(n_trees=50), Fold 2 (auc= 0.988)
RandomForest(n_trees=50), Fold 3 (auc= 0.985)
RandomForest(n_trees=50), Fold 4 (auc= 0.985)
RandomForest(n_trees=50), Fold 5 (auc= 0.986)

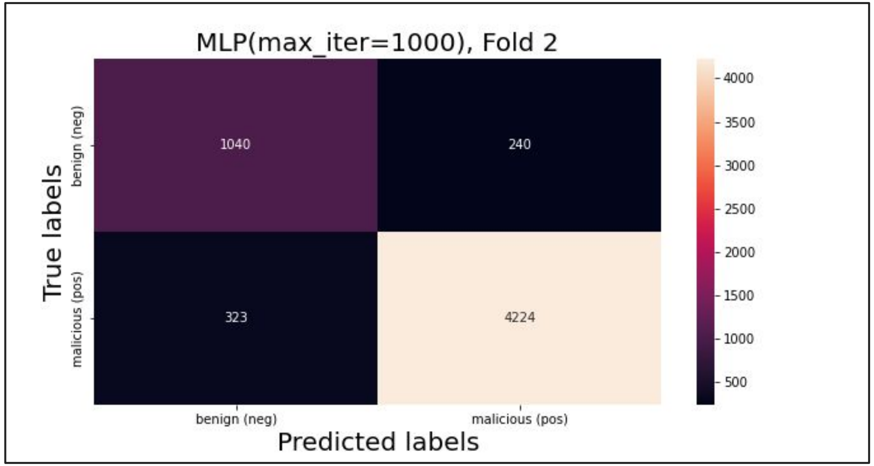
Experimento - Matrizes de Confusão (MLP)



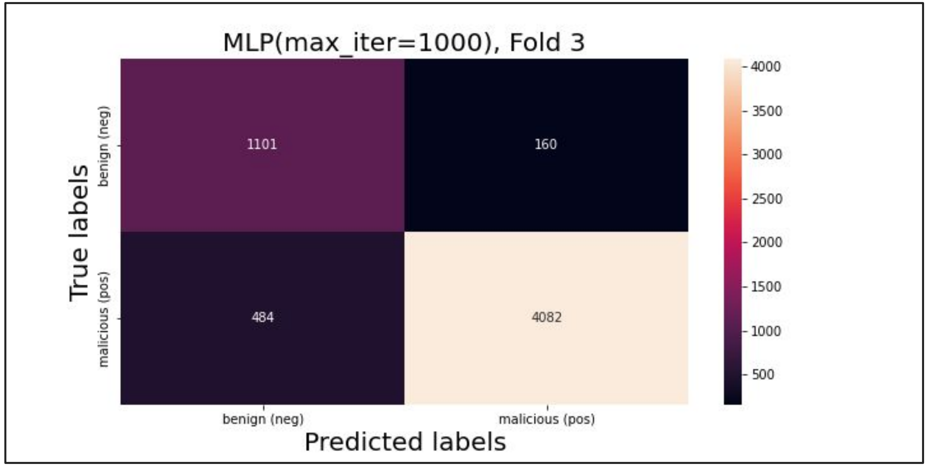


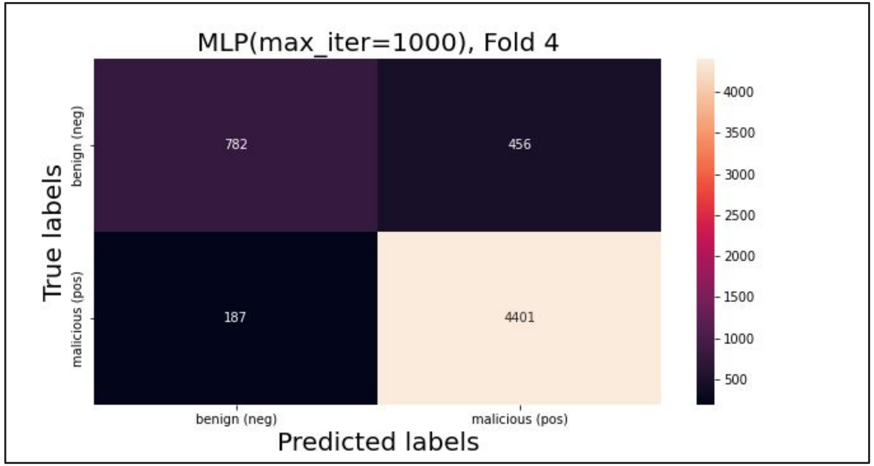
Experimento - Matrizes de Confusão (MLP)



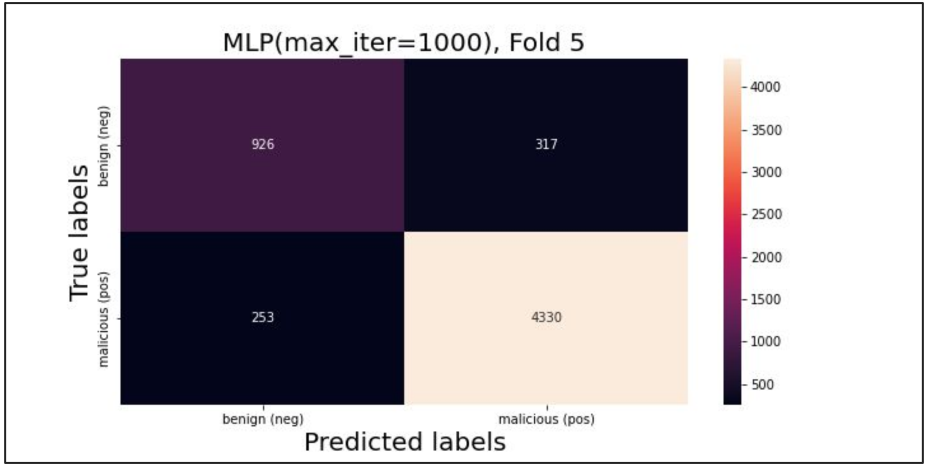


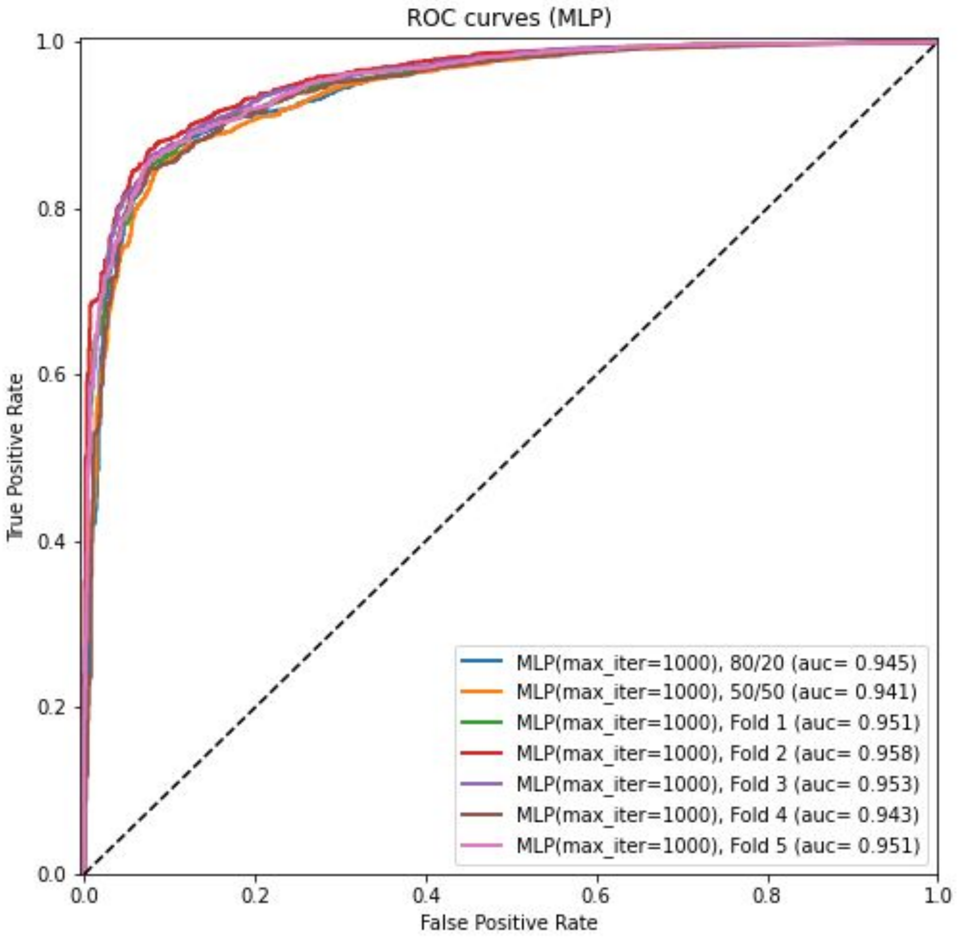
Experimento - Matrizes de Confusão (MLP)





Experimento - Matrizes de Confusão (MLP)





ROC curves (MLP)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
MLP(max_iter=1000), 80/20 (auc= 0.945)
MLP(max_iter=1000), 50/50 (auc= 0.941)
MLP(max_iter=1000), Fold 1 (auc= 0.951)
MLP(max_iter=1000), Fold 2 (auc= 0.958)
MLP(max_iter=1000), Fold 3 (auc= 0.953)
MLP(max_iter=1000), Fold 4 (auc= 0.943)
MLP(max_iter=1000), Fold 5 (auc= 0.951)

Fontes

- Dataset utilizado:
  - https://www.unb.ca/cic/datasets/url-2016.html
- Paper “Detecting Malicious URLs Using Lexical Analysis”:
  - https://www.researchgate.net/publication/308365207_Detecting_Malicious_URLs_Using_Lexical_Analysis

Obrigado pela atenção!