Ciência de Dados para Segurança

Trabalho Final

Daniel Osternack Barros Neves - GRR20171603
João Vitor Moreira - GRR20171621

1. Objetivo

Este trabalho tem o objetivo de aplicar técnicas de análise de dados sobre um conjunto de Emails classificados previamente como *benignos* e *fraudulentos*. Foram utilizados dois classificadores e uma rede neural para treinamento e subsequente avaliação dos dados, nos modelos *split percentage* e *K-Fold Cross Validation*.

2. Escolha do dataset

A primeira tarefa para a execução do trabalho foi a escolha do dataset. Procuramos por diversos tópicos relacionados à segurança, mas a maioria dos resultados não nos chamou a atenção, ou possuía algum bloqueio de acesso (paywall, formulário para receber o dataset por email, limitação de uso apenas por acesso remoto à máquina, etc). Decidimos utilizar, por fim, o dataset de emails fraudulentos disponibilizado por Rachael Tatman no Kaggle, disponível no link https://www.kaggle.com/rtatman/fraudulent-email-corpus.

Contudo, ao longo do desenvolvimento do trabalho, percebemos que não seria possível classificar tal *dataset*, pois possui apenas uma classe de emails (fraudulentos, estilo “príncipe nigeriano”). Dessa forma, optamos por juntá-lo a um dataset de emails benignos, e para tal fim optamos pelo dataset da Enron (https://www.cs.cmu.edu/~enron/).

3. Implementação

A implementação foi feita utilizando-se a versão 3 da linguagem Python, com o auxílio das bibliotecas *SciKit Learn* para a manipulação dos dados e *MatplotLib* para exibição de gráficos. Para a classificação, foram escolhidos os algoritmos de *K-Nearest Neighbors*, *Random Forest* e *MultiLayer Perceptron*.

4. Pré-processamento dos datasets

O dataset de emails fraudulentos não precisou de nenhum pré-processamento, pois seus emails já estavam apresentados em uma estrutura contendo um cabeçalho com informações e, logo após, o corpo. A única alteração foi o nome do arquivo, que foi alterado de "**frad**ulent_emails.txt" para "**fraud**ulent_emails.txt".

Já o dataset da Enron não tinha uma estrutura adequada para nosso uso. Este estava dividido em diretórios (um para cada funcionário da Enron), com cada diretório contendo mais diretórios como *"sent"*, *"deleted"* e outras categorias, que então continham arquivos de texto (um para cada email).

De forma a obter apenas emails que poderíamos ter certeza que seriam benignos, extraímos apenas aqueles presentes nos diretórios *"sent"*, presumindo que os funcionários da Enron não teriam utilizado seu email corporativo para enviar emails de *scam*. Além disso, como a ideia inicial era de utilizar o corpo do email, filtramos por aqueles que não continham as palavras "Forwarded", "Original Message" e mais de um "Subject: ", o que indicaria que o email foi encaminhado ou é uma resposta (o que polui o corpo da mensagem com outro email). Os emails extraídos do dataset foram então salvos sequencialmente em outro arquivo, "benign_emails.txt". Todo o pré-processamento dos emails da Enron pode ser encontrado no script "process_enron.py".

5. Extração das características

A extração das características foi feita a partir da leitura dos arquivos pré-processados, verificando pelas informações que possivelmente possam caracterizar um email fraudulento. Inicialmente os campos escolhidos foram o *Assunto*, *Content-Type* e o *corpo* (*conteúdo)* do email.

Posteriormente, foi decidido utilizar apenas o campo de *Assunto,* visto que o *Content-Type* é repetido em grande parte das amostras e o corpo muitas vezes é bastante extenso, causando uma longa demora para o processamento dos dados.

Como os dados extraídos são textuais, utilizamos o módulo de *TfidfVectorizer* da biblioteca *SciKit Learn* para a conversão em dados numéricos, através da função estatística *TF-IDF*.

6. Análise dos dados

Após a conversão das características textuais em dados numéricos, foi possível fazer o treinamento dos classificadores com uma porção dos dados seguido de um teste com o restante das amostras para avaliação dos resultados. Foram feitas duas execuções por classificador, uma utilizando o modelo de *Split Percentage,* nas proporções 80% - 20%, e outra no modelo *K-Fold Cross Validation, com K = 5*.
A seguir detalha-se a execução de cada classificador.

K-Nearest Neighbors: Para este classificador, utilizou-se o valor de K = 2 de forma que a classificação de uma amostra se dá pelos dois vizinhos mais próximos. Este foi o parâmetro com o qual foi possível atingir um maior valor de precisão.

Random Forest: Para o classificador de Random Forest, foram utilizados os parâmetros *max_depth = None* e *random_state = 0*

MultiLayer Perceptron: O MultiLayer Perceptron foi executado utilizando 3 camadas de 20 neurônios cada, um número máximo de 300 iterações e tolerância de 10, padrão do algoritmo na biblioteca *SciKit Learn*.

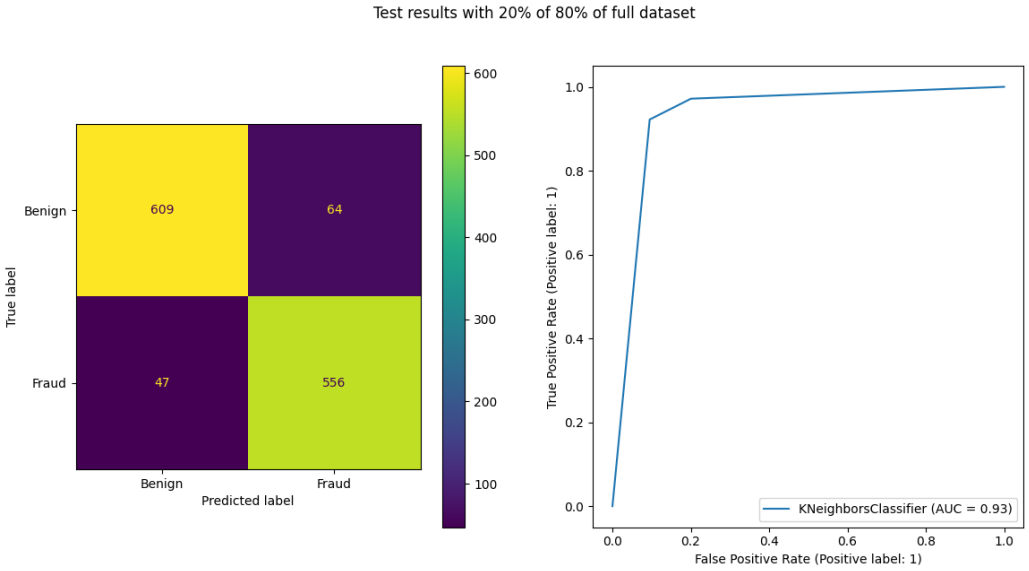
7. Resultados

A seguir, apresenta-se os resultados obtidos para cada classificador em cada modelo.

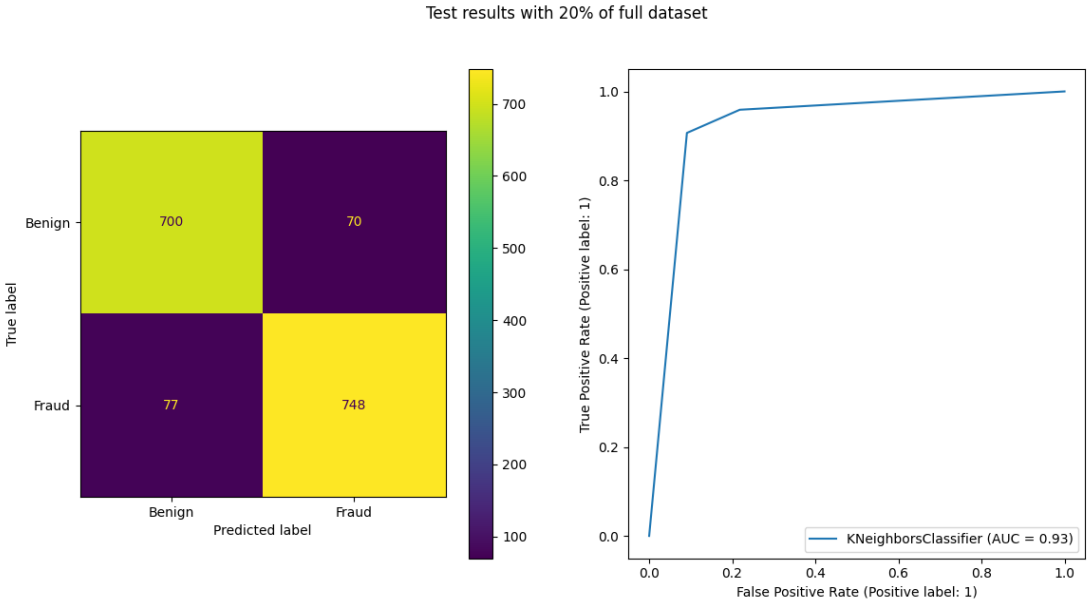
Split Percentage (80% - 20%)

K-Nearest Neighbors treinado com 80% dos 80% do dataset

- Teste com 20% dos 80% do dataset
  - **Precisão:** 0.897
  - **Erro Médio Absoluto:** 0.087

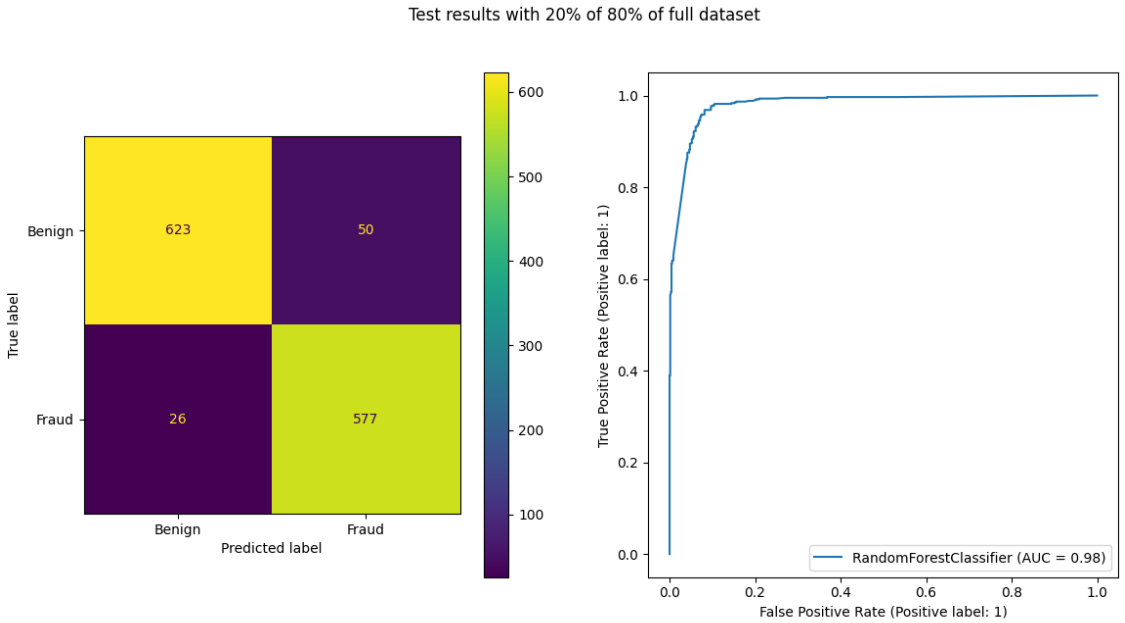


- Teste com os 20% restantes do dataset
  - **Precisão:** 0.914
  - **Erro Médio Absoluto:** 0.092

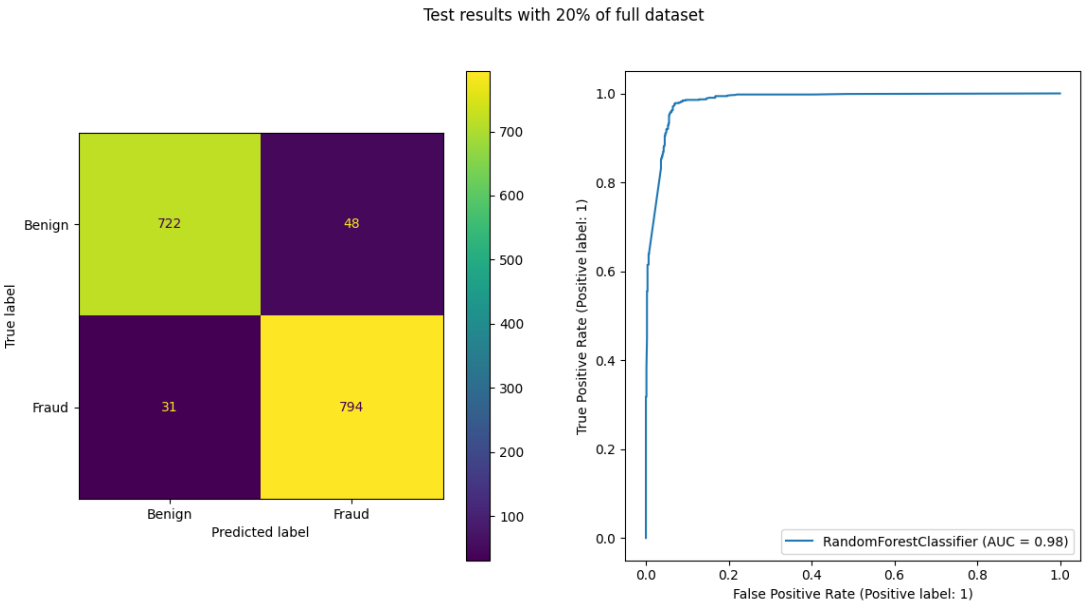


Random Forest treinado com 80% dos 80% do dataset

- Teste com 20% dos 80% do dataset:
  - **Precisão:** 0.920
  - **Erro Médio Absoluto:** 0.060

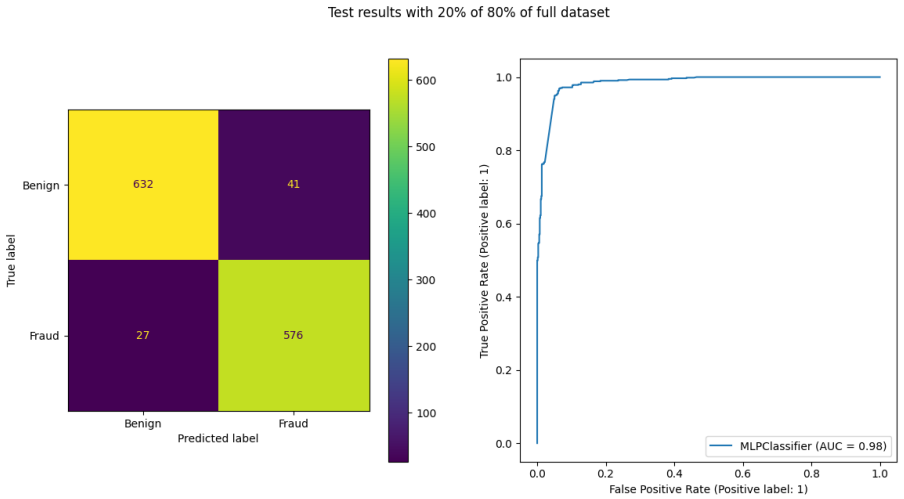


- Teste com os 20% restantes do dataset
  - **Precisão:** 0.943
  - **Erro Médio Absoluto:** 0.050

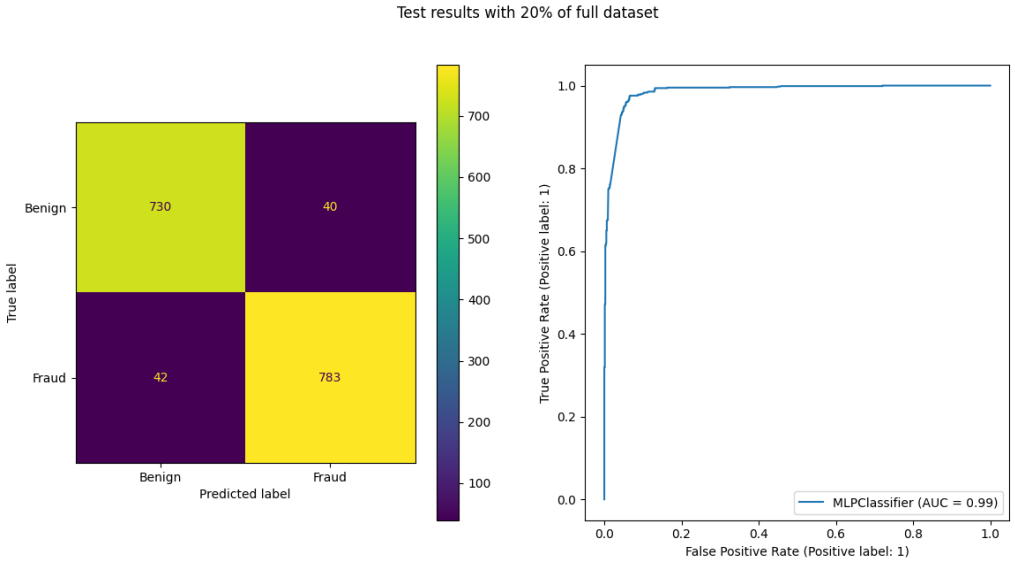


MultiLayer Perceptron treinado com 80% dos 80% do dataset

- Teste com 20% dos 80% do dataset
  - **Precisão:** 0.934
  - **Erro Médio Absoluto:** 0.053



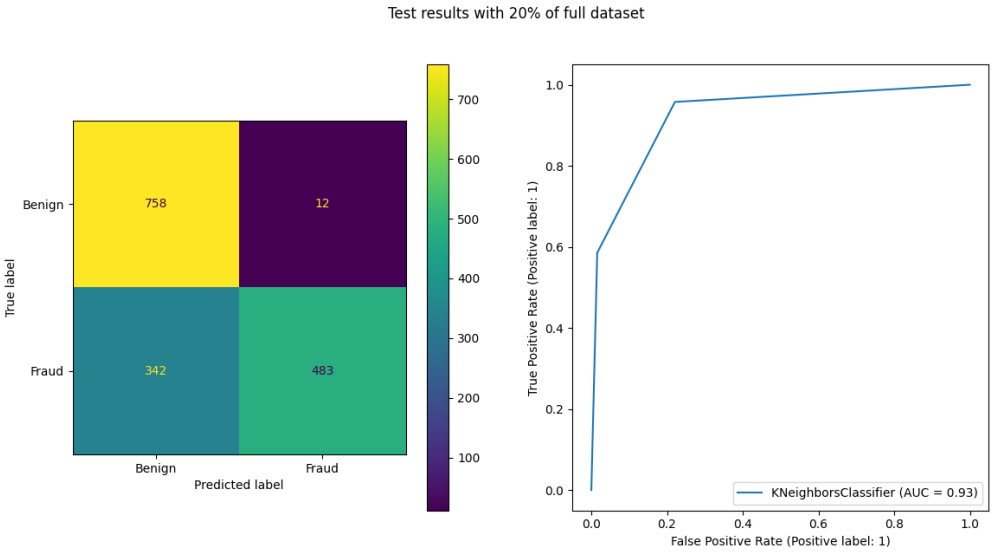
- Teste com os 20% restantes do dataset
  - **Precisão:** 0.951
  - **Erro Médio Absoluto:** 0.051



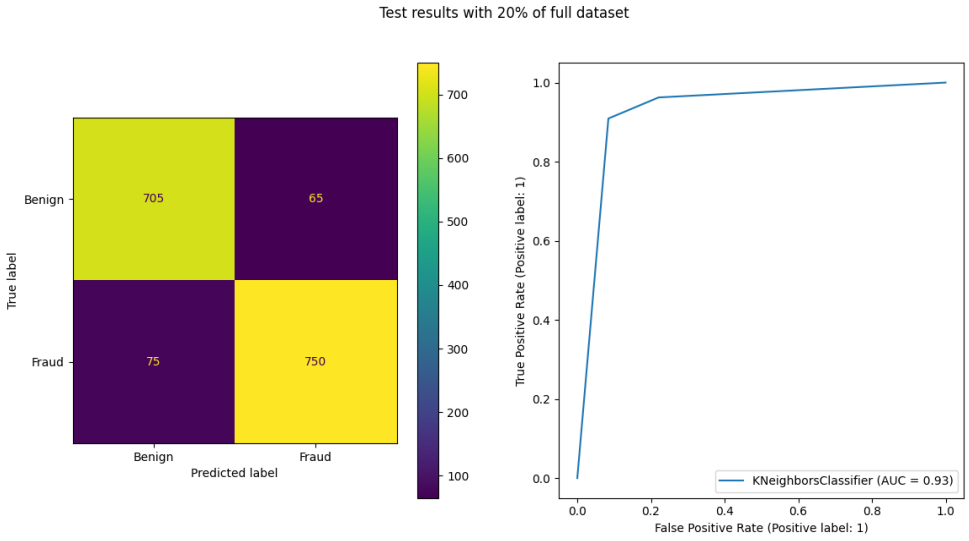
K-Fold Cross Validation (K = 5)

K-Nearest Neighbors treinado com 80% dos 80% do dataset

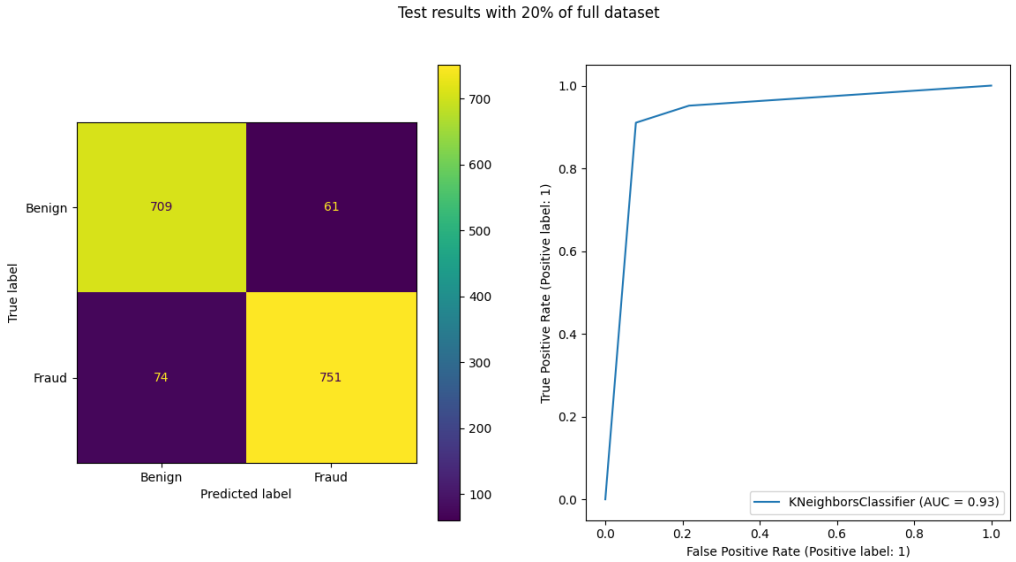
- I) 0 - 20% dos 80% do dataset
  - **Precisão:** 0.969
  - **Erro Médio Absoluto:** 0.203
  - Teste com os 20% restantes do dataset
    - **Precisão:** 0.976
    - **Erro Médio Absoluto:** 0.222



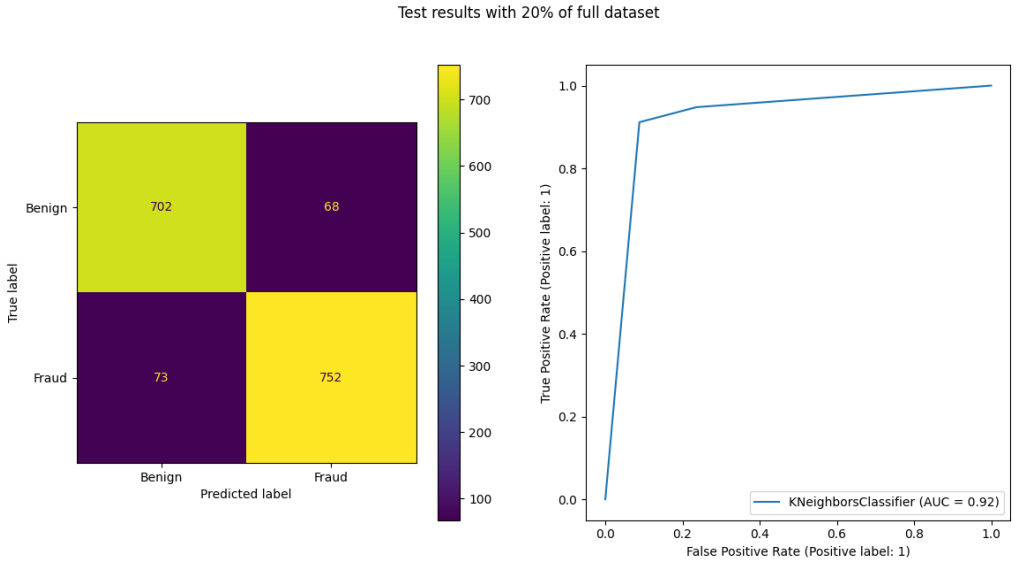
- II) 20% - 40% dos 80% do dataset
  - **Precisão:** 0.936
  - **Erro Médio Absoluto:** 0.079
  - Teste com os 20% restantes do dataset
    - **Precisão:** 0.920
    - **Erro Médio Absoluto:** 0.088



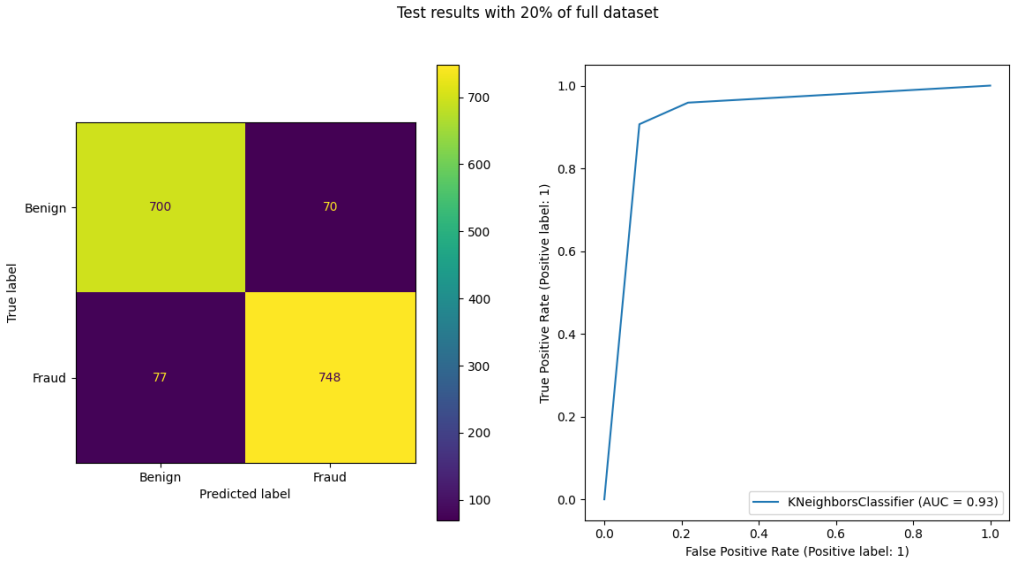
- III) 40% - 60% dos 80% do dataset
    - **Precisão:** 0.899
    - **Erro Médio Absoluto:** 0.087
    - Teste com os 20% restantes do dataset
        - **Precisão:** 0.925
        - **Erro Médio Absoluto:** 0.085



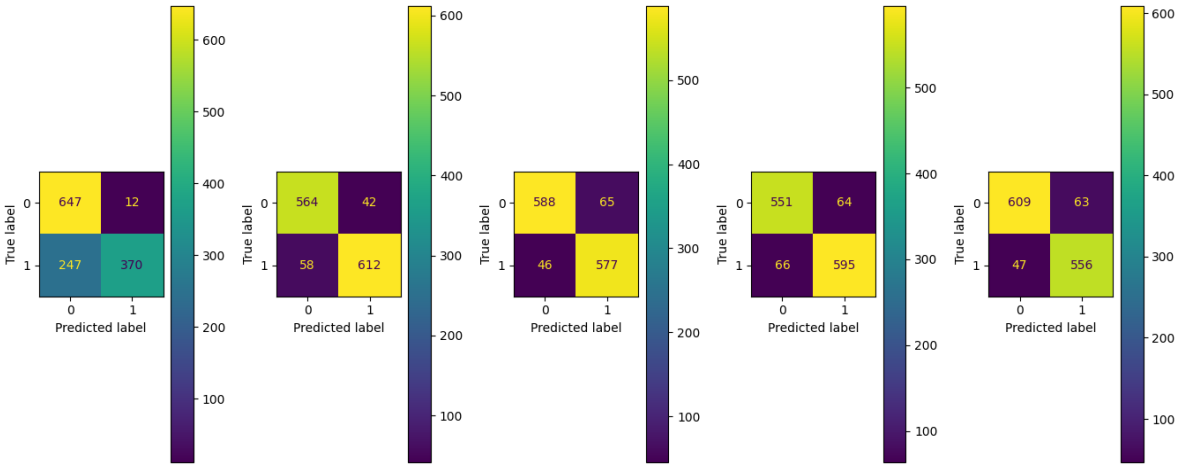
- IV) 60% - 80% dos 80% do dataset
    - **Precisão:** 0.903
    - **Erro Médio Absoluto:** 0.102
    - Teste com os 20% restantes do dataset
        - **Precisão:** 0.917
        - **Erro Médio Absoluto:** 0.088



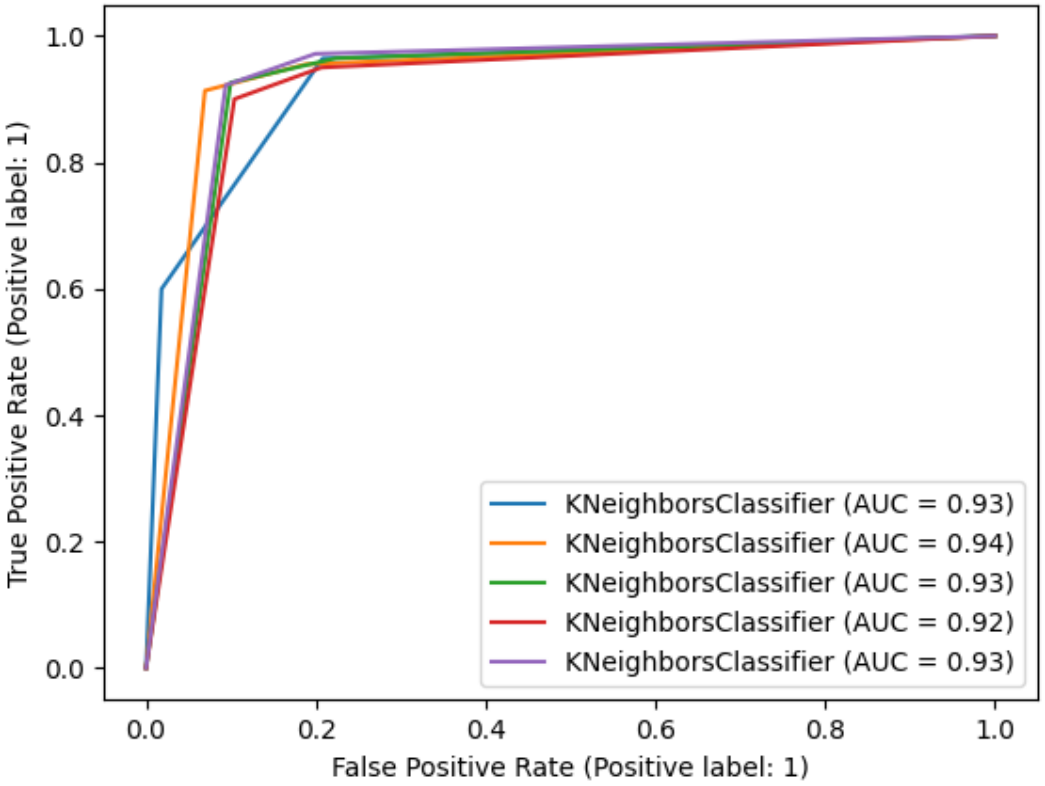
- V) 80% - 100% dos 80% do dataset
    - **Precisão:** 0.898
    - **Erro Médio Absoluto:** 0.086
    - Teste com os 20% restantes do dataset
        - **Precisão:** 0.914
        - **Erro Médio Absoluto:** 0.092



Matrizes de Confusão relativas aos 80% do dataset (da primeira à quinta iteração)

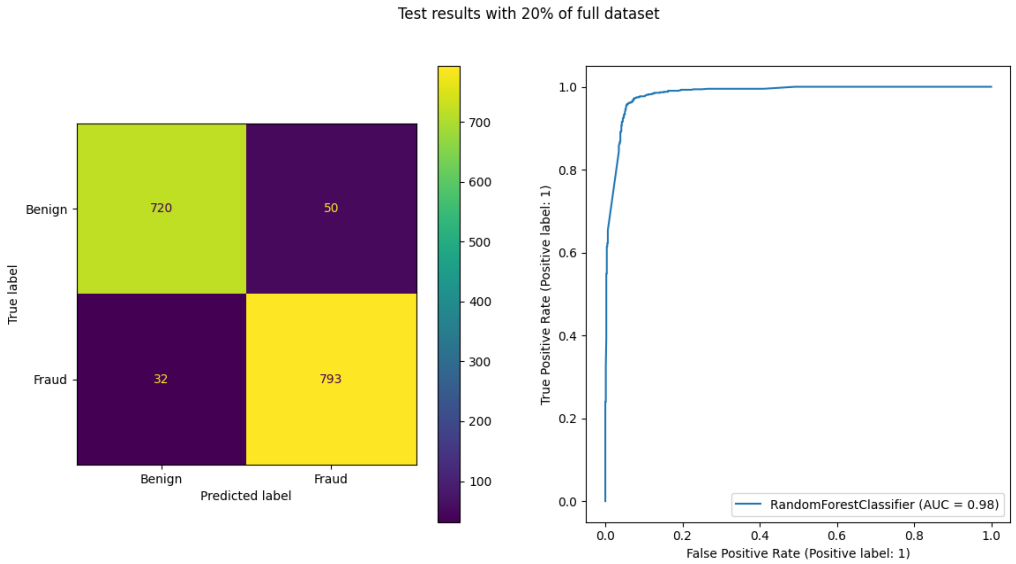


Curvas ROC relativas aos 80% do dataset

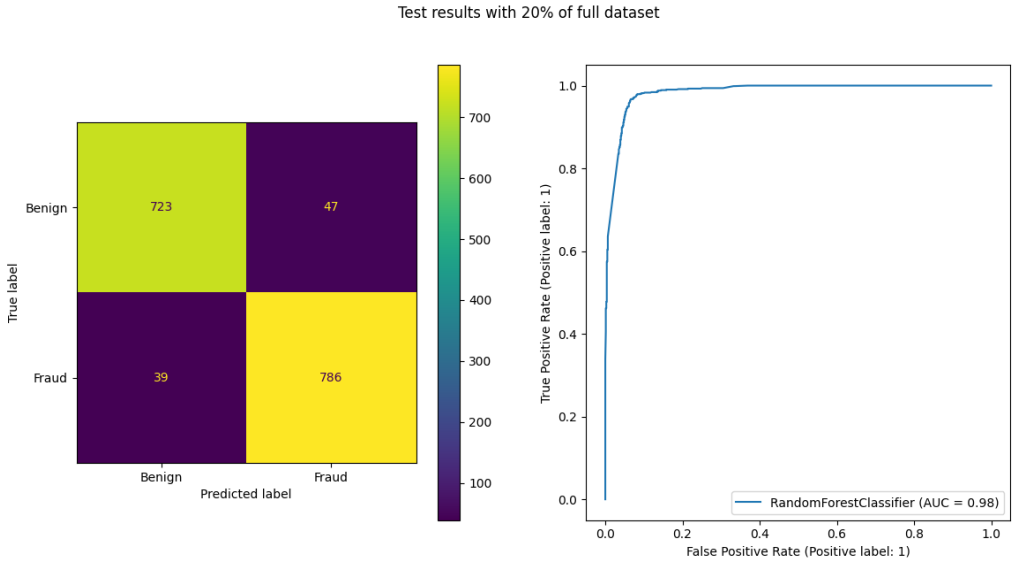


Random Forest treinado com 80% dos 80% do dataset

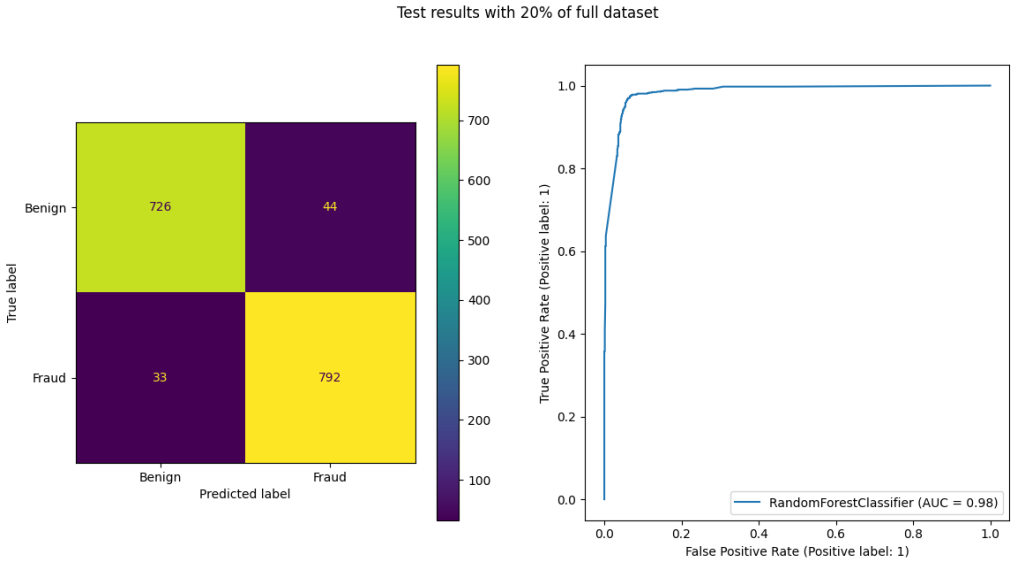
- I) 0 - 20% dos 80% do dataset
  - **Precisão:** 0.922
  - **Erro Médio Absoluto:** 0.062
  - Teste com os 20% restantes do dataset:
    - **Precisão:** 0.941
    - **Erro Médio Absoluto:** 0.051



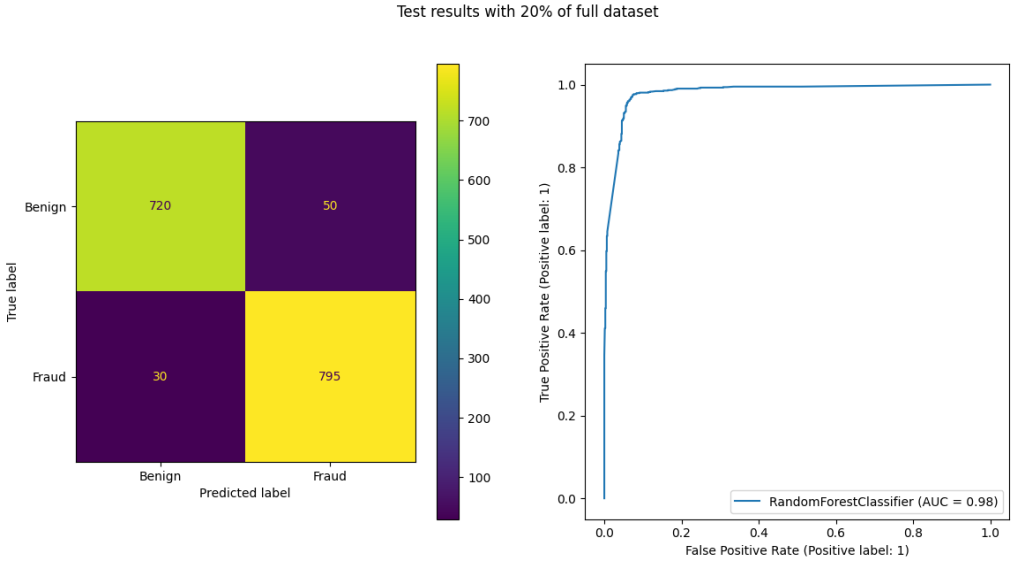
- II) 20% - 40% dos 80% do dataset
  - **Precisão:** 0.959
  - **Erro Médio Absoluto:** 0.049
  - Teste com os 20% restantes do dataset:
    - **Precisão:** 0.944
    - **Erro Médio Absoluto:** 0.054



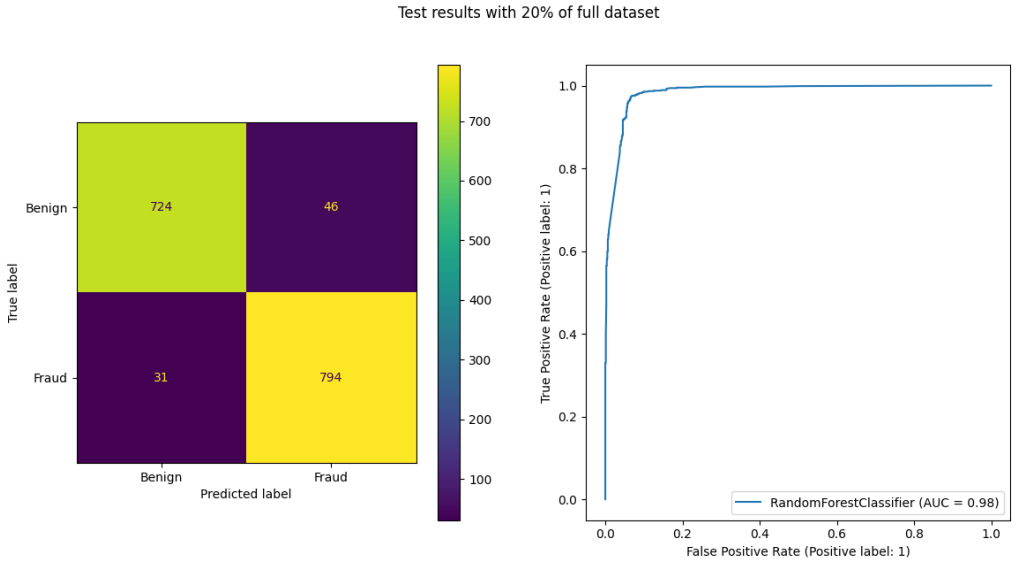
- III) 40% - 60% dos 80% do dataset
  - **Precisão:** 0.924
  - **Erro Médio Absoluto:** 0.063
  - Teste com os 20% restantes do dataset:
    - **Precisão:** 0.947
    - **Erro Médio Absoluto:** 0.048



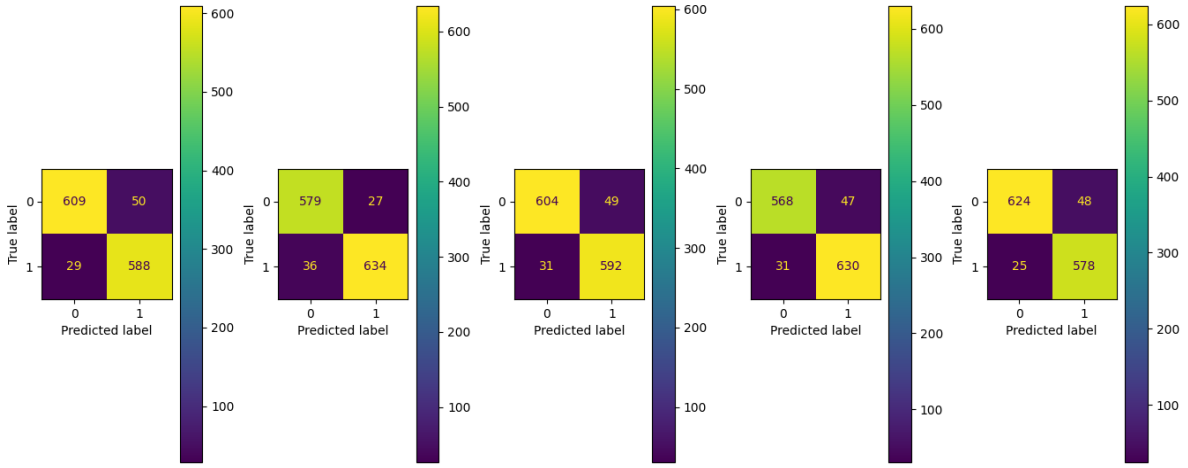
- IV) 60% - 80% dos 80% do dataset
  - **Precisão:** 0.931
  - **Erro Médio Absoluto:** 0.061
  - Teste com os 20% restantes do dataset:
    - **Precisão:** 0.941
    - **Erro Médio Absoluto:** 0.050



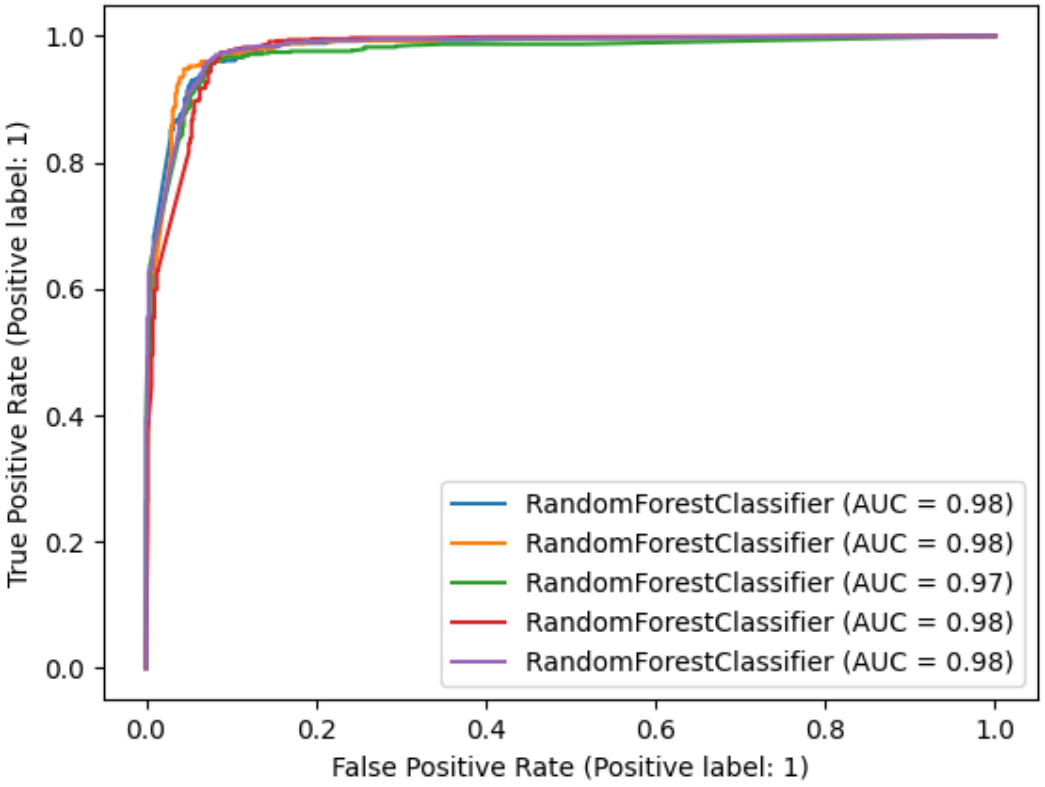
- V) 80% - 100% dos 80% do dataset
  - **Precisão:** 0.923
  - **Erro Médio Absoluto:** 0.057
  - Teste com os 20% restantes do dataset:
    - **Precisão:** 0.945
    - **Erro Médio Absoluto:** 0.048



Matrizes de Confusão relativas aos 80% do dataset (da primeira à quinta iteração)

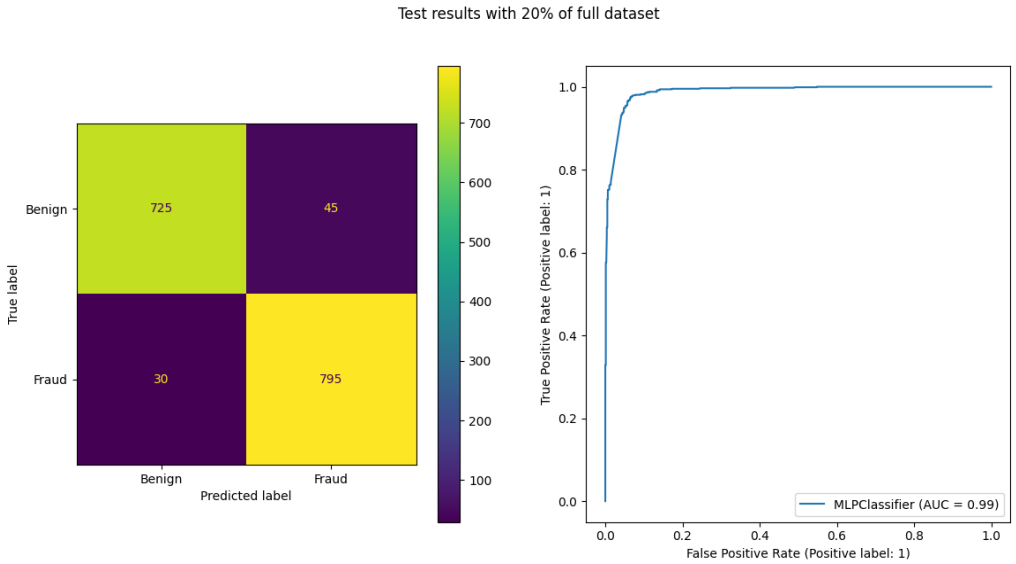


Curvas ROC relativas aos 80% do dataset

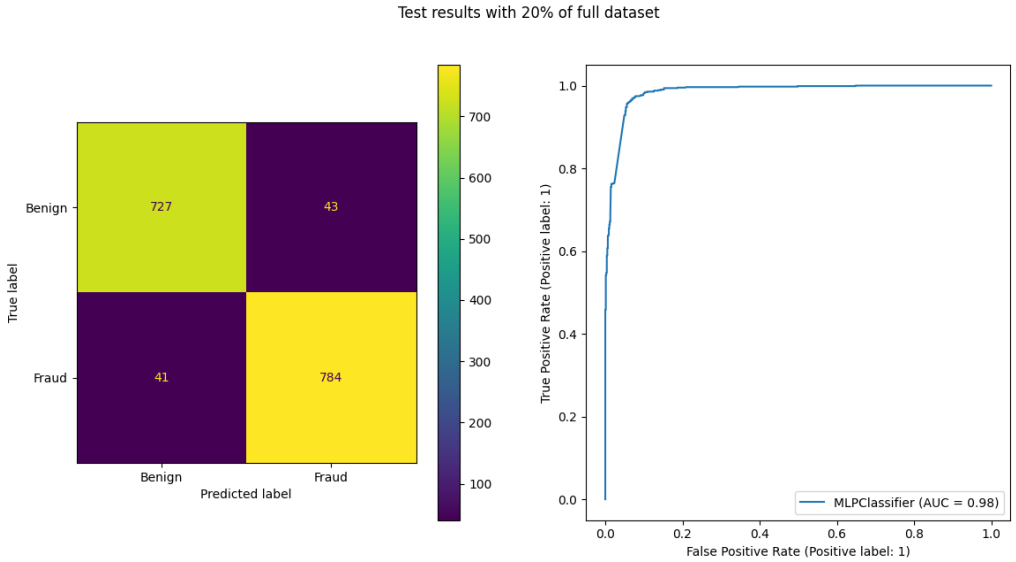


MultiLayer Perceptron treinado com 80% dos 80% do dataset

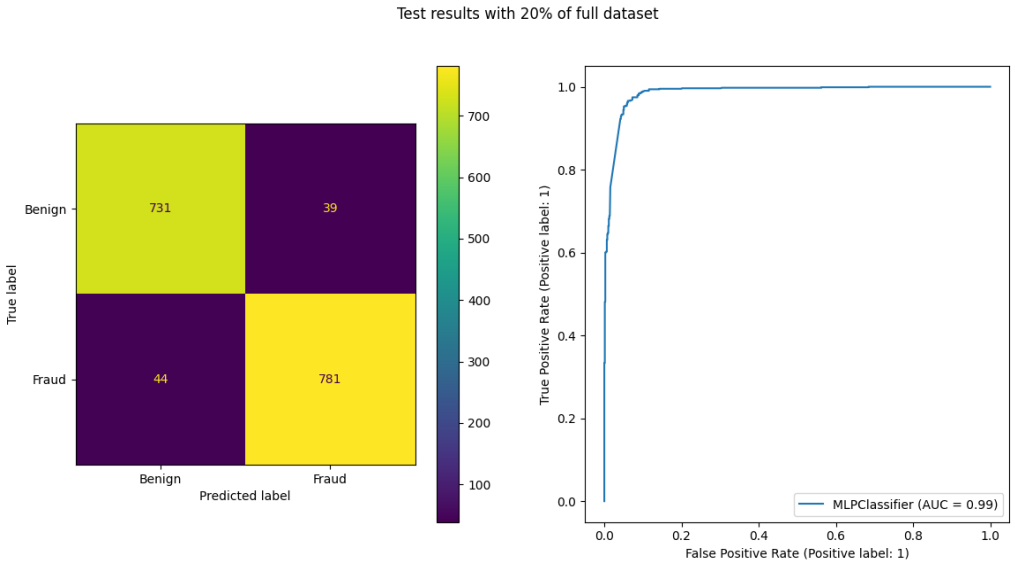
- I) 0 - 20% dos 80% do dataset
  - **Precisão:** 0.935
  - **Erro Médio Absoluto:** 0.056
  - Teste com os 20% restantes do dataset:
    - **Precisão:** 0.946
    - **Erro Médio Absoluto:** 0.047



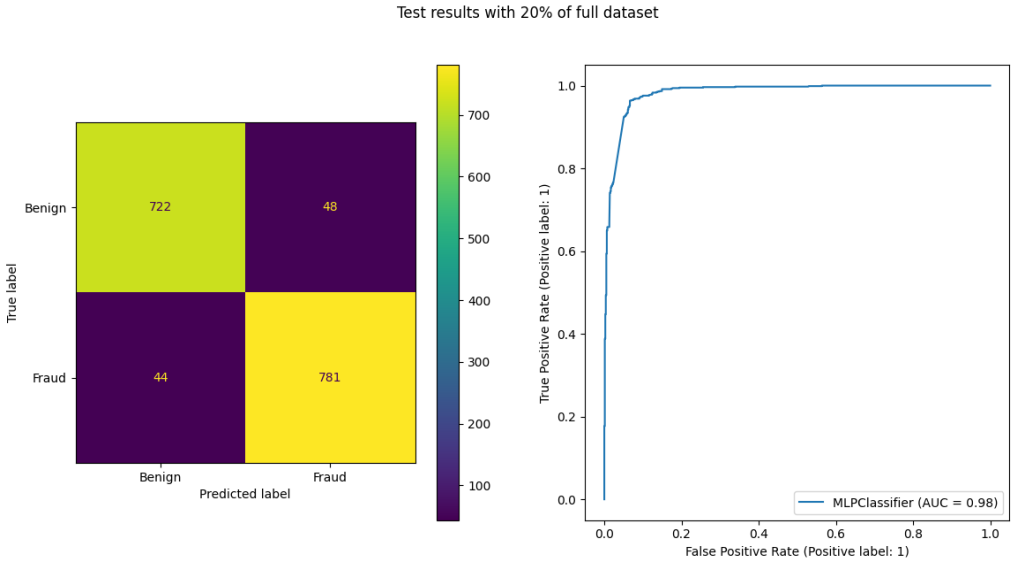
- II) 20% - 40% dos 80% do dataset
  - **Precisão:** 0.959
  - **Erro Médio Absoluto:** 0.051
  - Teste com os 20% restantes do dataset:
    - **Precisão:** 0.948
    - **Erro Médio Absoluto:** 0.053



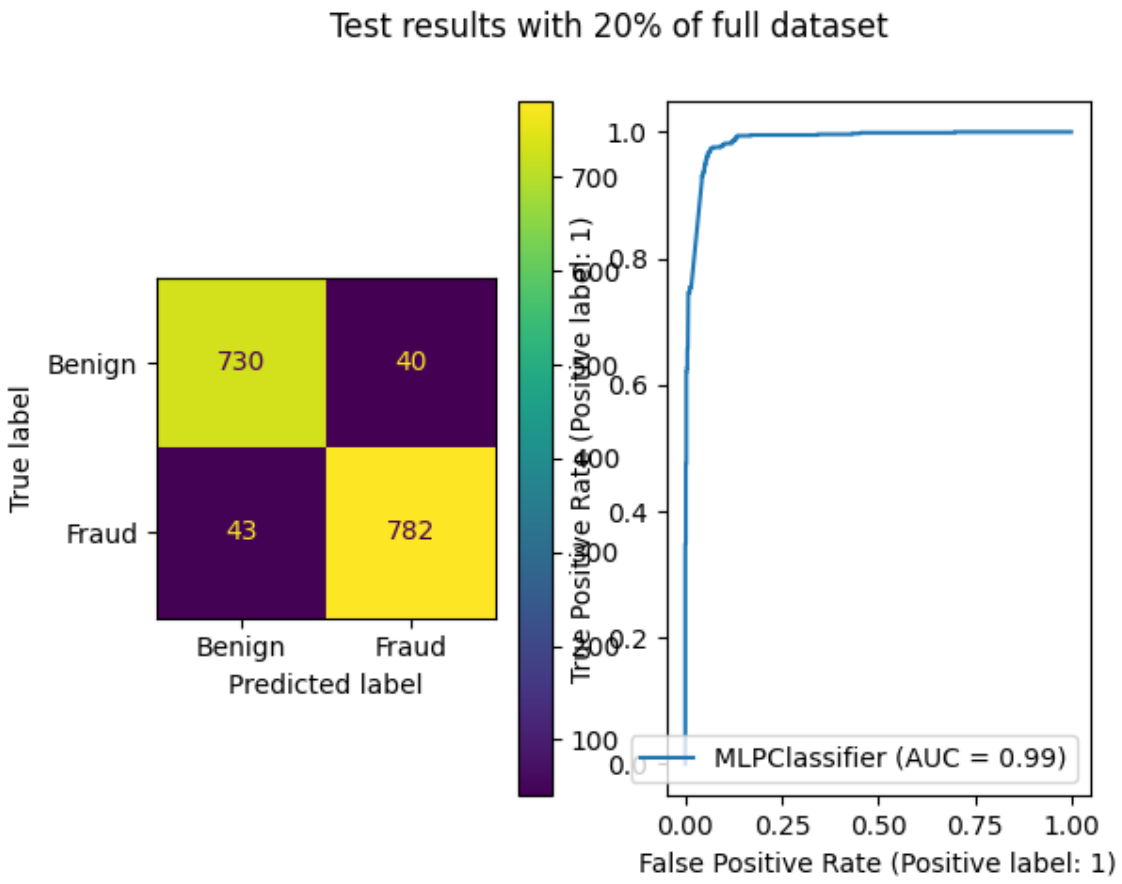
- III) 40% - 60% dos 80% do dataset
  - **Precisão:** 0.946
  - **Erro Médio Absoluto:** 0.051
  - Teste com os 20% restantes do dataset:
    - **Precisão:** 0.952
    - **Erro Médio Absoluto:** 0.052



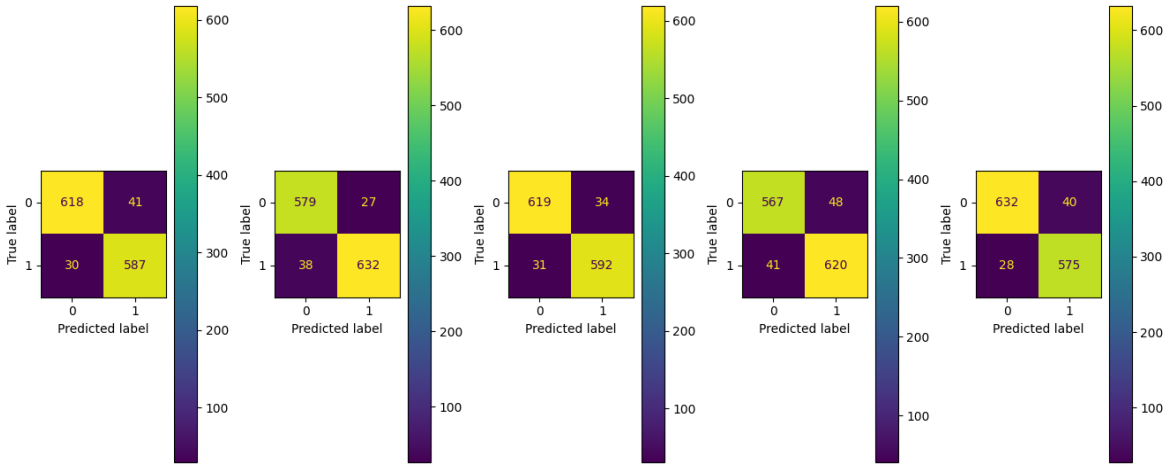
- IV) 60% - 80% dos 80% do dataset
  - **Precisão:** 0.928
  - **Erro Médio Absoluto:** 0.070
  - Teste com os 20% restantes do dataset:
    - **Precisão:** 0.942
    - **Erro Médio Absoluto:** 0.058



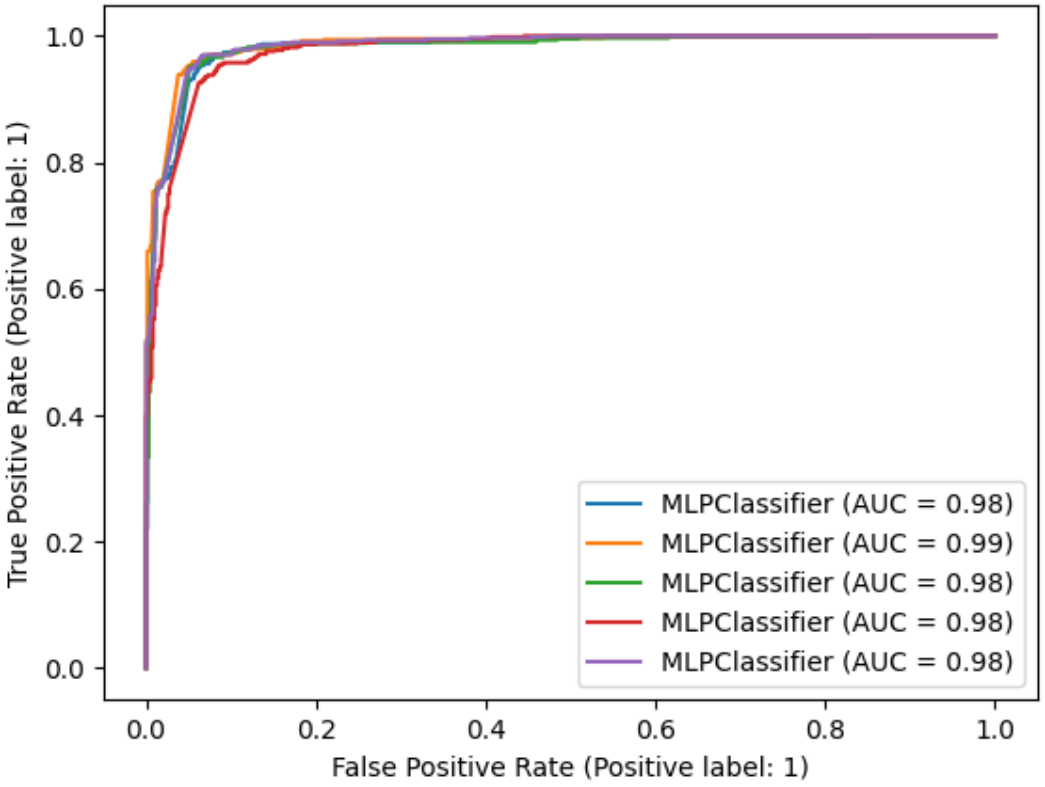
- V) 80% - 100% dos 80% do dataset
  - **Precisão:** 0.935
  - **Erro Médio Absoluto:** 0.053
  - Teste com os 20% restantes do dataset:
    - **Precisão:** 0.951
    - **Erro Médio Absoluto:** 0.052



Matrizes de Confusão relativas aos 80% do dataset (da primeira à quinta iteração)



Curvas ROC relativas aos 80% do dataset



8. Conclusão

Analisando os dados no modelo split percentage, foi possível observar que o MultiLayer Perceptron apresentou os melhores valores de precisão e erro, porém o Random Forest apresentou maior eficiência no treinamento e predição. Já o K-Nearest Neighbors não obteve os menores níveis de precisão.

Em relação ao K-Fold Cross Validation, os algoritmos não apresentaram discrepâncias significativas nos valores, mostrando que é possível obter um bom resultado independentemente de com qual porção do dataset é feito o treinamento.

9. Referências

Radev, D. (2008), CLAIR collection of fraud email, ACL Data and Code Repository, ADCR2008T001, http://aclweb.org/aclwiki

Cohen W. W. (2015), Enron Email Dataset, https://www.cs.cmu.edu/~./enron/