

Ambiente para benchmark de técnicas de Machine Learning aplicado à Cibersegurança

Licenciatura em Engenharia Informática

Rafael José Varanda Esteves Espadinha

Ricardo Lopes Sousa

Leiria, setembro de 2021

Net Traffic Classifier

Licenciatura em Engenharia Informática

Rafael José Varanda Esteves Espadinha

Ricardo Lopes Sousa

Trabalho de Projeto da unidade curricular de Projeto Informático realizado sob a orientação do Professor Doutor Carlos Manuel da Silva Rabadão, do Professor Doutor Leonel Filipe Simões Santos e do Doutor Rogério Luís de Carvalho Costa

Leiria, setembro de 2021

Resumo

Este projeto consiste num *benchmark* de algoritmos de *machine learning* aplicados à cibersegurança. Tratando-se de um *benchmark* o objetivo é obter métricas e relatórios comparativos dos diversos algoritmos implementados, por forma a termos uma visão de qual o algoritmo que proporciona melhores resultados em determinado conjunto de dados.

A aplicação permite a criação de projetos. Sumariamente, as etapas inerentes a cada projeto consistem em escolher um *dataset*, tratar/limpar o *dataset*, dividir o *dataset* em conjunto de treino e conjunto de teste, aplicar os algoritmos de *machine learning* ao *dataset* e obter métricas e comparações dos resultados de cada algoritmo. A aplicação permite ainda guardar os modelos de treino dos algoritmos, para que eventualmente possam ser usados mais tarde na classificação de outros *datasets*.

Palavras-chave: *benchmark, machine learning, cibersegurança, dataset*

Abstract

This project consists of a benchmark of machine learning algorithms applied to cybersecurity. Being a benchmark, the main goal is to obtain metrics and comparative reports of the several implemented algorithms, in order have an idea of which algorithm provides the best results in a given dataset.

The application allows the creation of projects. Briefly, the steps inherent to each project consist of choosing a dataset, cleaning de dataset, dividing de dataset, splitting the dataset into training and testing sets, applying the machine learning algorithms to the dataset and obtaining the metrics and comparative reports of the results of each algorithm. The application also allows user to save the models of the trained algorithms, so they can eventually be used later to classify another dataset.

Keywords: benchmark, machine learning, cybersecurity, dataset

Lista de Figuras

Figura 1 - Função de ativação da função sigmoide	4
Figura 2 - Exemplo visual do funcionamento do algoritmo KNN	5
Figura 3 - Support Vector Machine.....	6
Figura 4 - Tipos de aprendizagem.....	7
Figura 5 - Matriz de confusão com várias classes	10
Figura 6 - Gráfico da curva ROC	12
Figura 7 - AUC para um modelo de classificação excelente.....	13
Figura 8 - AUC = 0.7	13
Figura 9 - AUC = 0.5	13
Figura 10 - AUC = 0	14
Figura 11 - Valores do Agile Manifesto.....	19
Figura 12 - Metodologias ágeis mais usadas	21
Figura 13 - Práticas gerais do Kanban.....	22
Figura 14 - Exemplo de quadro de kanban.....	23
Figura 15 - Políticas do kanban	24
Figura 16 - Arquitetura geral da aplicação	30
Figura 17 - Estrutura do ficheiro log.txt.....	31
Figura 18 - Estrutura do ficheiro syslog.txt.....	32
Figura 19 - Estrutura de ficheiros e diretorias	33
Figura 20 - Página inicial da aplicação	34
Figura 21 - Interface da preparação dos dados	35
Figura 22 - Interface da classificação	36
Figura 23 - Variáveis do construtor no ficheiro File.py	38
Figura 24 - Construtor da classe <i>Table</i>	41
Figura 25 - Construtor da classe <i>LogFile.py</i>	43
Figura 26 - Exemplo de uma chamada a função <code>logActions()</code>	44
Figura 27 - Exemplo do ficheiro <code>syslog.txt</code>	44
Figura 28 - Construtor da classe <i>ProjectLogFile</i>	45

Figura 29 - Escolher nome do projeto	46
Figura 30 - Seleção do ficheiro para carregamento de projeto	47
Figura 31 - Visualizar dados numa tabela	48
Figura 32 - Exemplo de tabela com dados.....	49
Figura 33 - Visualizar Dados e Estatísticas	49
Figura 34 - Zona reservada ao tratamento de colunas	51
Figura 35 - Exemplo do antes e depois da remoção das colunas selecionadas	52
Figura 36 - Exemplo de renomeação da coluna "std_iat".....	52
Figura 37 - Exemplo do antes e depois da renomeação da coluna "std_iat"	53
Figura 38 - Exemplo de gravação da coluna objetivo.....	53
Figura 39 - Exemplo de keyword gerada após a gravação da coluna objetivo	54
Figura 40 - Escolher algoritmo a utilizar	54
Figura 41 - Lista de algoritmos da aplicação	55
Figura 42 - Exemplo de algoritmo que necessita de parâmetros especiais	55
Figura 43 - Exemplo de dois algoritmos selecionados	56
Figura 44 - Exemplo de um algoritmo presente na aplicação	57
Figura 45 - Diferentes métodos de divisão de dados	58
Figura 46 - Zona para iniciar/parar a classificação	59
Figura 47 - Área reservada aos resultados	59
Figura 48 - Exemplo de curva de Precision-Recall	61
Figura 49 - Exemplo de curva de ROC.....	61
Figura 50 - Exemplo de uma matriz de confusão	61
Figura 51 - Exemplo da mostra de resultados após classificação dos dados	62
Figura 52 - Mensagem apresentada aquando do término da classificação	63
Figura 53 - <i>Frame</i> para classificação de dados com base na escolha de um modelo	63
Figura 54 - Exemplo de um modelo selecionado.....	64
Figura 55 - Exemplo de um ficheiro com os dados classificados com base no modelo selecionado	64
Figura 56 - Carregamento e remoção de métricas	65
Figura 57 - Exemplo de uma matriz de confusão aberta numa aplicação externa	65
Figura 58 - Comparar resultados obtidos.....	66

Figura 59 - Exemplo de comparação entre duas classificações.....	66
Figura 60 - Ecrã inicial da aplicação.....	74
Figura 61 - Inserção do nome do projeto.....	75
Figura 62 - Escolha do ficheiro de dados	75
Figura 63 - Escolha do ficheiro para carregar o projeto	76
Figura 64 - Interface da preparação dos dados	76
Figura 65 - Dados da coluna selecionada	77
Figura 66 - Estatísticas da coluna selecionada	77
Figura 67 - Secção de tratamento dos dados	78
Figura 68 - Ecrã da classificação dos <i>datasets</i>	79
Figura 69 - Escolha dos algoritmos a serem usados.....	79
Figura 70 - Escolha do tipo de divisão de dados	79
Figura 71 - Secção de treino e teste dos algoritmos	80
Figura 72 - Métricas dos algoritmos aplicados.....	81
Figura 73 - Classificação de um <i>dataset</i> com um modelo já treinado.....	81
Figura 74 - Comparação de resultados e visualização de gráficos	81

Lista de tabelas

Tabela 1 - Exemplo de uma matriz de confusão.....	9
---	---

Lista de siglas e acrónimos

AUC	Area Under the Curve
CSV	Coma Separated Values
ESTG	Escola Superior de Tecnologias e Gestão
IA	Inteligência Artificial
IP	Internet Protocol
IPLeiria	Instituto Politécnico de Leiria
ROC	Receiver Operating Characteristics
XP	Extreme Programming

Índice

Resumo.....	iv
Abstract.....	v
Lista de Figuras.....	vi
Lista de tabelas	ix
Lista de siglas e acrónimos	x
1. Introdução	1
2. Conceitos e Tecnologias Utilizadas	2
2.1. Inteligência Artificial.....	2
2.2. Machine Learning	3
2.2.1. Aprendizagem Supervisionada	3
2.2.2. Aprendizagem Não Supervisionada	6
2.3. Preparação dos dados.....	7
2.3.1. Ordinal Encoding.....	7
2.3.2. Z-Score	8
2.4. Métricas	8
2.4.1. Accuracy Score.....	8
2.4.2. Confusion Matrix.....	8
2.4.3. Precision	10
2.4.4. Recall.....	10
2.4.5. F1 Score.....	11
2.4.6. Logarithmic Loss.....	11
2.4.7. ROC AUC	11
2.5. Tecnologias.....	14
2.5.1. Linguagem Python.....	14
2.5.2. IDE PyCharm	15
3. Metodologias	17

3.1.	Agile Manifesto	18
3.1.1.	Valores	18
3.1.2.	Princípios.....	19
3.2.	Scrum	20
3.3.	Kanban	21
3.3.1.	Práticas Gerais.....	22
3.4.	Extreme Programming (XP)	25
3.5.	Metodologia Utilizada	25
4.	Arquitetura e Requisitos Funcionais	27
4.1.	Requisitos Funcionais	27
4.1.1.	Projetos.....	27
4.1.2.	Dados de Entrada	28
4.1.3.	Tarefas de Pré-Processamento	28
4.1.4.	Tarefas de Treino, Teste e Classificação.....	29
4.1.5.	Métricas e Geração de Relatórios.....	30
4.2.	Arquitetura	30
4.2.1.	Estrutura de ficheiros	31
4.2.2.	Backend.....	33
4.2.3.	User Interface	34
4.3.	Tecnologias e bibliotecas.....	36
4.3.1.	Tkinter	36
4.3.2.	Scikit-learn	36
4.3.3.	Pandas.....	37
4.3.4.	Statistics	37
5.	Implementação	38
5.1.	Sistema de Carregamento e Edição dos Ficheiros de Dados.....	38
5.2.	Sistema de Cálculo de Estatísticas, Normalizações e <i>Encoding</i>	40

5.3.	Sistema de Logs.....	43
5.4.	Sistema de Gestão dos Dados e Projetos da Aplicação.....	44
5.5.	Sistema de Criação ou Carregamento de Projetos	45
5.6.	Sistema de Visualização de Dados e Estatísticas	48
5.7.	Sistema de Tratamento de Colunas	50
5.8.	Sistema de Seleção e Adição de Algoritmos	54
5.8.1.	Importação e Seleção de Algoritmos.....	54
5.8.2.	Adição de Algoritmos.....	56
5.9.	Sistema de Divisão de Dados	57
5.10.	Sistema de Classificação dos Dados	58
5.10.1.	Iniciar Classificação	59
5.10.2.	Parar Classificação	63
5.11.	Sistema de Carregamento de Modelos.....	63
5.12.	Sistema de Carregamento de Resultados	64
5.13.	Sistema de Visualização de Gráficos.....	65
5.14.	Sistema de Comparação de Resultados	66
6.	Testes	67
6.1.	Testes Unitários	67
6.2.	Testes de Integração	68
6.3.	Testes de Sistema	69
7.	Conclusão	70
8.	Bibliografia.....	71
9.	Anexos.....	74
	Anexo A - Manual de Utilização Da Aplicação	74

1. Introdução

O presente relatório foi elaborado no âmbito da Unidade Curricular de Projeto Informático, pertencente ao ciclo de estudos da Licenciatura em Engenharia Informática, do Instituto Politécnico de Leiria, do ano letivo 2020/21, e tem como tema o desenvolvimento de um *benchmark* de algoritmos de *machine learning* aplicados à cibersegurança.

Sendo um *benchmark*, o objetivo da aplicação é obter resultados comparativos da aplicação de diversos algoritmos de *machine learning* aplicados em *datasets* contendo informações de fluxo de dados na rede. Como tal, esta comparação é feita num contexto de cibersegurança.

A crescente importância da segurança na internet e os diversos contextos e possíveis utilizações de técnicas de *machine learning* levaram-nos ao desenvolvimento desta aplicação para que possamos saber se um determinado pacote enviado de um IP para outro IP é ou não malicioso.

No decorrer do relatório serão abordados os temas que nos ajudaram na realização deste projeto. Primeiramente, serão abordados os conceitos e tecnologias utilizadas no desenvolvimento do projeto. No terceiro capítulo serão introduzidas as metodologias ágeis de desenvolvimento de *software* e também a metodologia usada. Seguidamente serão apresentados os requisitos funcionais e a arquitetura de aplicação bem como as tecnologias e bibliotecas aplicadas. No quinto capítulo, iremos explicar aprofundadamente todos os componentes e funcionalidades desenvolvidas de modo a garantir que a aplicação vai de encontro com os objetivos propostos. Por fim, no último capítulo denominado Testes serão apresentados alguns testes realizados para garantir o correto funcionamento da aplicação.

2. Conceitos e Tecnologias Utilizadas

Na primeira reunião sobre o Net Traffic Classifier, foram-nos propostas várias linguagens e plataformas que poderíamos utilizar para o desenvolvimento do projeto. Visto tratar-se de um *benchmark* relacionado com inteligência artificial e *machine learning*, optámos por utilizar uma linguagem mais direcionada para esse tema com inúmeras bibliotecas existentes que nos facilitaram o desenvolvimento do projeto. Neste capítulo iremos abordar primeiro os conceitos base inerentes ao projeto e posteriormente a linguagem e plataforma utilizadas.

2.1. Inteligência Artificial

O termo “Inteligência Artificial” (IA) existe desde o ano de 1956 quando John McCarthy, um cientista da computação estadunidense, se referiu ao tema numa conferência de especialistas definindo-a como “a ciência e engenharia de produzir sistemas inteligentes”. É uma ciência que não tem apenas uma definição, podendo referirmo-nos a ela como sendo “a inteligência similar à humana exibida por sistemas de software” ou como alguns investigadores se referem a ela “o estudo e projeto de agentes inteligentes”, onde o agente inteligente é um sistema que percebe o seu ambiente e toma atitudes maximizando as chances de sucesso. Para além destas definições, também Andreas Klapan e Michael Haenlein, professores de Marketing na *ESCP Business School* em França, se referem à inteligência artificial como “uma capacidade do sistema para interpretar corretamente dados externos, aprender a partir desses dados e utilizar essas aprendizagens para atingir objetivos e tarefas específicas através de adaptação flexível” [1].

Apesar de existirem inúmeras definições para este termo, o principal objetivo dos sistemas de IA é igual para qualquer que seja a definição apresentada. Executar funções consideradas inteligentes caso executadas por um ser humano, facilitando assim a realização de algumas tarefas do dia a dia, poupando tempo e trabalho. Tem sido também uma área de estudo muito utilizada para obter avanços em pesquisas científicas, como por exemplo no tratamento de doenças [2].

2.2. Machine Learning

Considerado um subcampo da inteligência artificial, um cientista da computação estadunidense de seu nome Arthur Samuel, em 1959, definiu “*Machine Learning*” como o “campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados”, ou seja, é um ramo da IA em que os sistemas conseguem aprender com dados, identificar padrões e tomar de decisões sem ser preciso quase nenhuma intervenção por parte dos humanos. Para além disso, cria automaticamente modelos de representação de conhecimento com base num conjunto de dados aquando da execução de algoritmos sobre estes [3].

Esta área tem como objetivo criar os melhores modelos para que estes consigam efetuar análises e previsões de qualidade em situações futuras e com dados diferentes. Cada vez mais vemos este tipo de análise de dados presente no nosso dia a dia, como por exemplo, nas recomendações da Amazon ou nas pesquisas que efetuamos no Google.

No nosso projeto vamos usar métodos de aprendizagem supervisionada, contudo, de seguida, vamos explicar os principais conceitos tanto da aprendizagem supervisionada como da aprendizagem não supervisionada.

2.2.1. Aprendizagem Supervisionada

Este tipo de aprendizagem requer um conhecimento prévio dos *outputs* dos exemplos usados nas etapas de treino e teste dos algoritmos de aprendizagem. Assim sendo, o principal objetivo deste tipo de aprendizagem é gerar um modelo/função que relacione os dados de *input* com os *outputs* desejados da melhor forma, permitindo assim classificar mais tarde um determinado conjunto de dados de *input*, tentando obter o *output* correto [4].

Dentro desta aprendizagem supervisionada há 2 tipos de problemas: problemas de classificação e problemas de regressão. Os problemas de classificação focam-se em classificar um conjunto de dados de *input* num determinado valor de *output* discreto ou nominal. Dando o exemplo do nosso projeto, pretendemos classificar um conjunto de dados de *input*, num *output* que será “ataque” ou “não ataque”. Nos problemas de regressão os algoritmos tentam mapear uma função a partir dos valores de *input* para obter um valor numérico contínuo [5]. Por exemplo estimar a altura de um indivíduo a partir de informações sobre o indivíduo como *input*.

Neste projeto, visto que estamos na presença de problemas de classificação vamos usar algoritmos de *machine learning* dessa natureza tais como *Logistic Regression*, *k-Nearest Neighbors*, *Gaussian Naive Bayes* ou *Support Vector Machine*, sendo possível adicionar outros.

De seguida vamos abordar sucintamente os algoritmos de aprendizagem supervisionada utilizados no projeto e acima mencionados.

Logistic Regression

O algoritmo *Logistic Regression* é utilizado em problemas de classificação e baseia-se na função sigmoide. Supondo que temos a reta $Z = mx + b$, e que aplicamos a função sigmoide sobre o *output* da função da reta $h\Theta(x) = \text{sigmoid}(Z)$ [6].

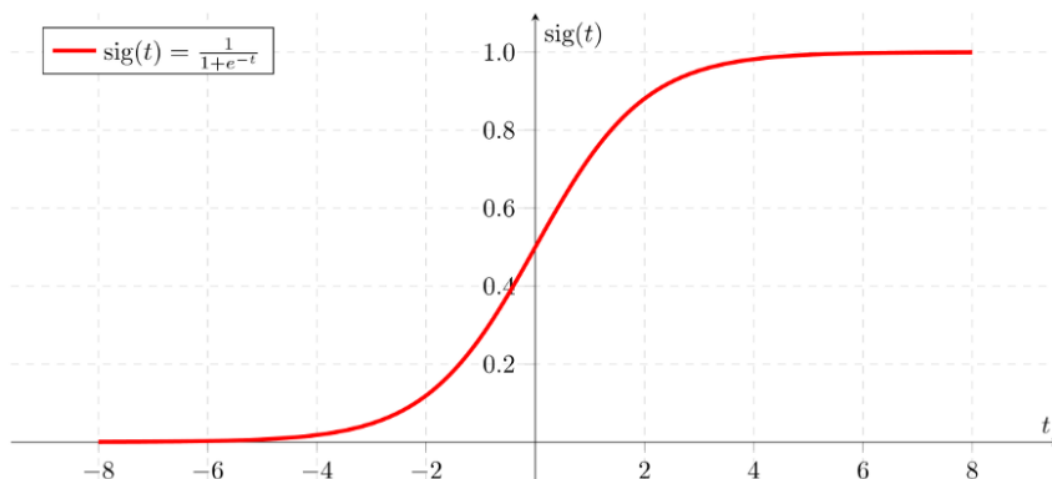


Figura 1 - Função de ativação da função sigmoide¹

Se o Z tender para infinito a função sigmoide irá retornar 1, caso o Z tenda para infinito negativo, a função sigmoide retornará 0.

K-Nearest Neighbors

Este algoritmo pode ser usado tanto em problemas de classificação (como é o nosso caso), como em problemas de regressão. A sua maior desvantagem, é ficar mais lento quanto maior for o tamanho do ficheiro de dados usado.

¹ <https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc>

O princípio deste algoritmo passa por assumir que coisas similares estão perto umas das outras. Ou seja, o algoritmo supõe que se os valores de uma amostra são similares entre si, então o *output* será também similar. Na imagem abaixo podemos ver que o triângulo preto está mais perto dos triângulos verdes, ou seja, o algoritmo *KNN* iria classificar aquela amostra como “*Legitimate*” [7].

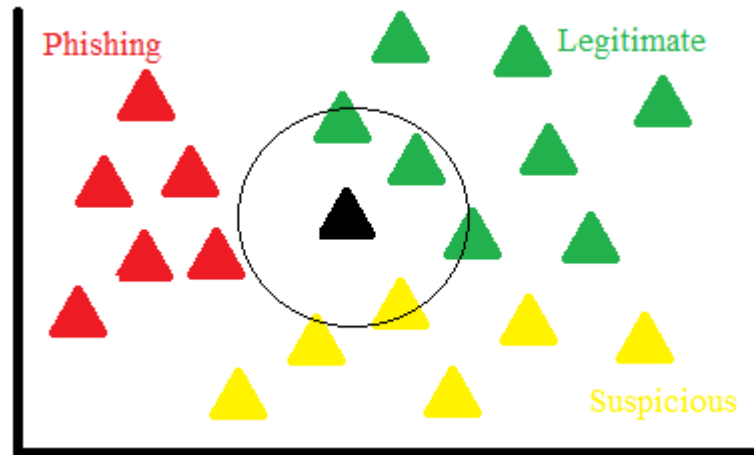


Figura 2 - Exemplo visual do funcionamento do algoritmo KNN²

Gaussian Naive Bayes

Este algoritmo inicia-se com o cálculo das probabilidades condicionais referentes a cada classe do atributo objetivo. No caso de termos 3 classes x_1 , x_2 e x_3 , a probabilidade condicional da classe x_1 seria:

$$P(c|x_1) = \frac{n_{x1}}{n_{x1} + n_{x2} + n_{x3}}$$

Em que n_{x1} corresponde ao número de amostras da classe x_1 , n_{x2} , corresponde ao número de amostras da classe x_2 e n_{x3} corresponde ao número de amostras da classe x_3 .

Após isto é calculado, para cada classe, a média e o desvio padrão dos atributos, que permitem contruir uma curva normal. O objetivo é calcular uma curva normal de cada classe e para cada atributo que permita prever novas amostras da melhor forma possível [8].

² https://www.researchgate.net/figure/A-k-nearest-neighbor-KNN-classifier-KNN-is-explained-as-follows_fig1_318096864

Support Vector Machine

O objetivo deste algoritmo é encontrar uma linha ou plano num espaço com N dimensões, sendo que N é o número de atributos de cada amostra, que consegue distinguir as várias classes em causa. A imagem abaixo ilustra o algoritmo [9].

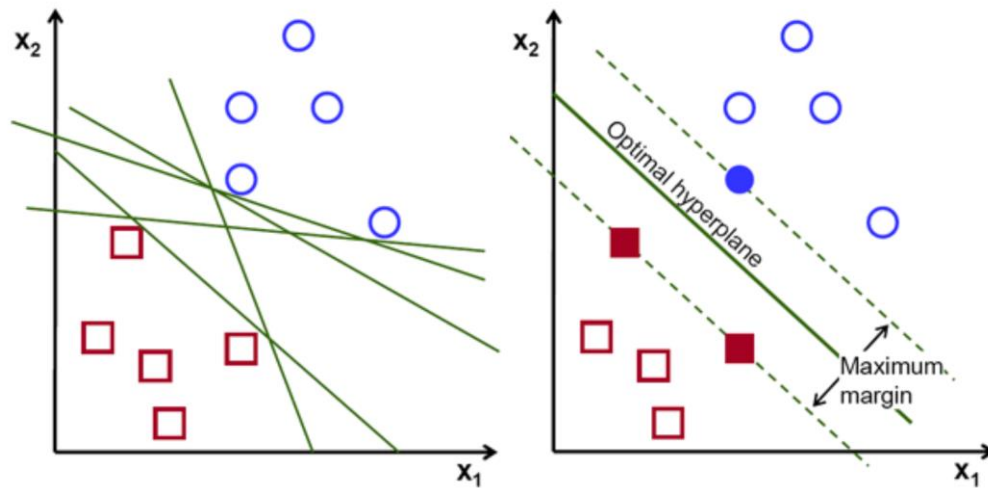


Figura 3 - Support Vector Machine³

Quanto maior for a distância entre a reta/plano, e as classes melhor será o resultado do algoritmo.

2.2.2. Aprendizagem Não Supervisionada

Uma vez que não vamos pôr em prática este tipo de aprendizagem no projeto, não lhe vamos dar tanto ênfase como à aprendizagem supervisionada [10].

Ao contrário da aprendizagem supervisionada, a aprendizagem não supervisionada não necessita de um conjunto de dados de output já classificados.

Para perceber melhor este conceito, imaginemos um exemplo de um bebé que vê um cão pela primeira vez, passado umas semanas vê outro cão diferente. Sem ter conhecimento prévio, o bebé vai reconhecer o novo animal como um cão analisando as semelhanças que tem com o animal que conhecia anteriormente.

Podemos ainda dividir este método de aprendizagem em 2 tipos principais de problemas: problemas de *clustering*, e problemas de *dimensionality reduction*.

³ <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>

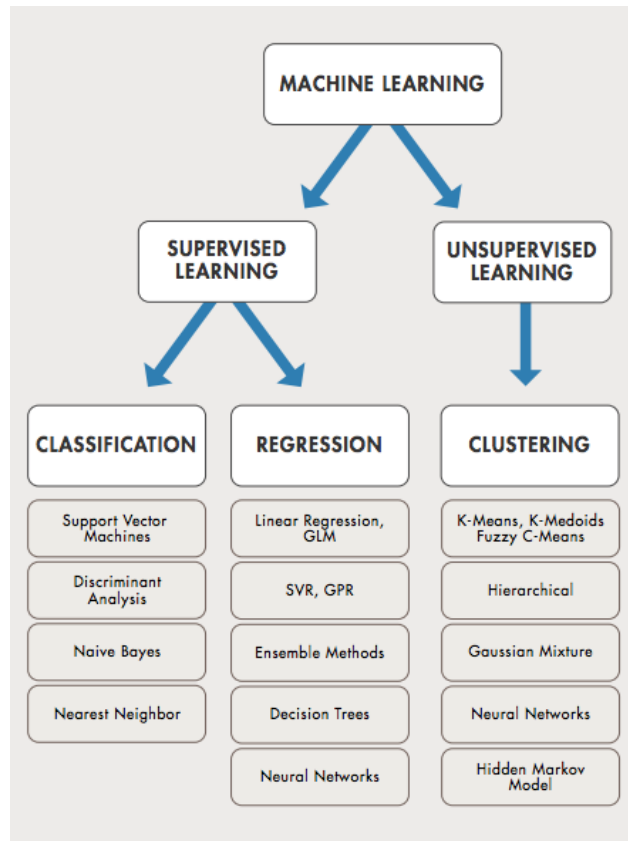


Figura 4 - Tipos de aprendizagem⁴

2.3.Preparação dos dados

Antes de aplicar quaisquer algoritmos de *machine learning* sobre o *dataset*, é importante que este esteja “limpo” e com valores aceitáveis para o algoritmo. Assim sendo, e para além dos tipos de limpeza de dados mais básicos, tais como a eliminação de colunas ou substituição de valores em falta, é relevante preparar os dados com técnicas próprias, como técnicas de *encoding* ou normalização. De seguida, vamos abordar as técnicas usadas neste projeto.

2.3.1. Ordinal Encoding

Esta técnica de *encoding* transforma valores nominais em valores numéricos. Quando usada sem parâmetros, atribui um número a uma palavra por ordem alfabética [11]. Exemplificando, no caso de termos uma coluna com os valores *baixo*, *médio*, *alto*, após a aplicação do *ordinal encoding* nesta coluna ficaríamos com a seguinte correspondência:

alto – 0, baixo – 1, médio – 2.

⁴ <https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94>

2.3.2. Z-Score

Esta técnica de normalização aproxima os valores numéricos de uma coluna do 0, sendo que todos os valores dessa coluna iguais à média serão substituídos por 0 [12]. Esta técnica de normalização é calculada da seguinte forma:

$$Z - Score = \frac{x - \mu}{\sigma}$$

Sendo,

x – valor da célula

μ - média dos valores da coluna

σ – desvio padrão dos valores da coluna

2.4.Métricas

Para entender quais os algoritmos que apresentam melhores resultados, é essencial compreendermos as métricas por estes geradas. Neste projeto usamos 7 métricas distintas para comparar os resultados dos algoritmos usados: *Accuracy Score*, *Confusion Matrix*, *Precision*, *Recall*, *F1 Score*, *Logarithmic Loss (Log Loss)* e *Roc AUC* [13].

2.4.1. Accuracy Score

Esta métrica é muitas vezes vista como a métrica mais importante, contudo, o facto de obtermos bons resultados com o *Accuracy Score*, não significa que as outras métricas retornem igualmente bons resultados. Esta métrica é bastante simples de entender, sendo obtida pela divisão entre o número de predições corretas e o número total de predições feitas pelo algoritmo [14].

$$Accuracy\ Score = \frac{\text{número de predições corretas}}{\text{número total de predições}}$$

2.4.2. Confusion Matrix

A matriz de confusão (*confusion matrix* em inglês), dá-nos uma matriz como output que descreve sucintamente o desempenho do algoritmo. No caso do nosso projeto, queremos classificar um determinado conjunto de dados como sendo “ataque” (1) ou “não ataque” (0).

Imaginemos que tínhamos um conjunto de dados com 100 amostras, e que um determinado algoritmo aplicado sobre estas 100 amostras, devolve a seguinte matriz de confusão [15]:

Tabela 1 - Exemplo de uma matriz de confusão

Nº de amostras: 100	Predição 0	Predição 1
Realidade: 0	35	10
Realidade: 1	15	40

Analisando esta matriz de confusão há 4 conceitos importantes que temos de retirar:

- **Verdadeiros Positivos:** número de casos em que o algoritmo previu 1(ataque) e o output real é 1 (célula verde)
- **Verdadeiros Negativos:** número de casos em que o algoritmo previu 0(não ataque) e o output real é 0 (célula vermelha)
- **Falsos Positivos:** número de casos em que o algoritmo previu 1(ataque) e o output real é 0 (célula azul)
- **Falsos Negativos:** número de em que o algoritmo previu 0(não ataque) e o output real é 1 (célula amarela)

No caso de o atributo que queremos prever ter mais de 2 classes, estes 4 conceitos acima descritos não são aplicados de forma tão linear. A seguinte imagem ilustra esta situação.

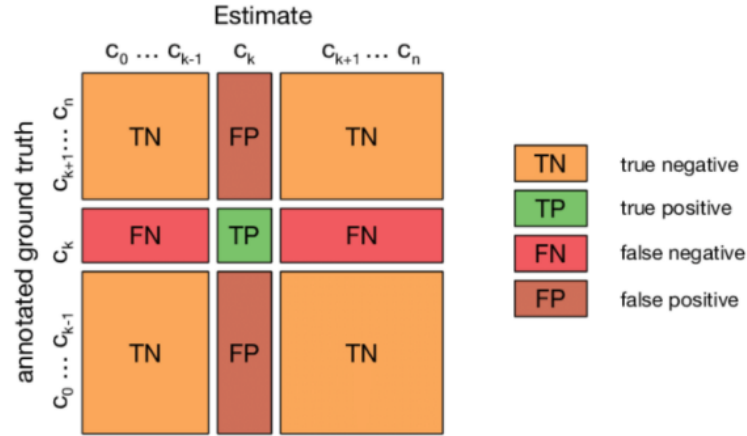


Figura 5 - Matriz de confusão com várias classes⁵

Neste caso:

- **Previsões corretas:** todos os valores que se encontram na diagonal principal da matriz
- **Previsões incorretas:** restantes valores da matriz

A divisão entre a soma dos valores da diagonal principal da matriz e o número total de amostras dá-nos a *accuracy* da mesma.

$$Accuracy = \frac{\text{Soma valores diagonal principal}}{\text{total de amostras}}$$

2.4.3. Precision

A precisão é a divisão entre os verdadeiros positivos e a soma dos verdadeiros positivos com os falsos positivos [14].

$$Precision = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Positivos}}$$

2.4.4. Recall

O *Recall* é a divisão entre os verdadeiros positivos e o número de todas as amostras relevantes (todas as amostras que deviam ter sido classificadas como positivas) [14].

$$Recall = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Negativos}}$$

⁵ https://www.researchgate.net/figure/Confusion-matrix-for-multi-class-classification-The-confusion-matrix-of-a_fig7_314116591

2.4.5. F1 Score

Uma vez que o valor da *Accuracy* pode ser influenciado no caso de haver um grande número de verdadeiros negativos, o *F1 Score* é uma medida que permite obter valores mais seguros nessas situações [14].

O *F1 Score* é a média harmónica entre a *Precision* e o *Recall*, e diz-nos o quão preciso e robusto é o algoritmo usado. A amplitude de valores desta métrica varia sempre entre 0 e 1, inclusive para ambos. Sendo que quanto maior for o valor melhor é a performance do algoritmo. A fórmula seguinte expressa como é calculado o *F1 Score*:

$$F1\ Score = 2 \times \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

2.4.6. Logarithmic Loss

Esta métrica penaliza as classificações incorretas do algoritmo. Apesar de ser uma métrica mais adequada para classificações onde existem mais de 2 classes como output possível, é também possível usá-la no nosso caso. Supondo que existem n amostras pertencentes a m classes, a *Log Loss* é calculada da seguinte forma:

$$Log\ Loss = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m y_{ij} \times \log(p_{ij})$$

Onde:

- y_{ij} indica se a amostra i pertence à classe j ou não
- p_{ij} indica a probabilidade de a amostra i pertencer à classe j

A amplitude de valores da *Log Loss* não tem limite superior, $[0, \infty[$. Quanto mais perto do 0 for o valor, mais preciso é o modelo gerado pelo algoritmo.

2.4.7. ROC AUC

Esta métrica é das mais importantes para medir a performance de qualquer algoritmo de classificação. *ROC (Receiver Operating Characteristics)* é uma curva de probabilidade e a área *AUC (Area Under The Curve)* o grau de separabilidade. Esta medida diz-nos o quão capaz é o modelo de distinguir as classes existentes no problema de classificação. Quanto

maior for a área AUC, mais capaz será o algoritmo de prever classes “0” como “0” e classes “1” como “1”, por exemplo [16].

A curva *ROC* é descrita pela função que relaciona frequência de falsos positivos (*FPR*) com a frequência de verdadeiros positivos (*TPR*). A *TPR* ou *Recall*, como já foi visto acima é calculada da seguinte forma:

$$Recall = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Negativos}$$

Para calcularmos a *FPR* temos de recorrer a outra métrica, a especificidade (ou *Specificity*, em inglês):

$$Specificity = \frac{Verdadeiros\ Negativos}{Verdadeiros\ Negativos + Falsos\ Positivos}$$

Assim sendo a *FPR* é calculada da seguinte forma:

$$FPR = 1 - Specificity$$

O gráfico abaixo ilustra o que foi dito anteriormente:

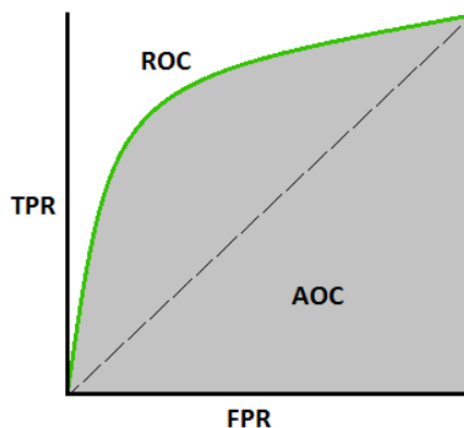


Figura 6 - Gráfico da curva ROC⁶

Com os conceitos assimilados acima, podemos concluir que, numa situação ideal, um modelo de classificação excelente teria um valor da área *AUC* perto de 1, o que significa que esse modelo saberia sempre distinguir as classes entre si, como podemos ver na figura seguinte.

⁶ <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>

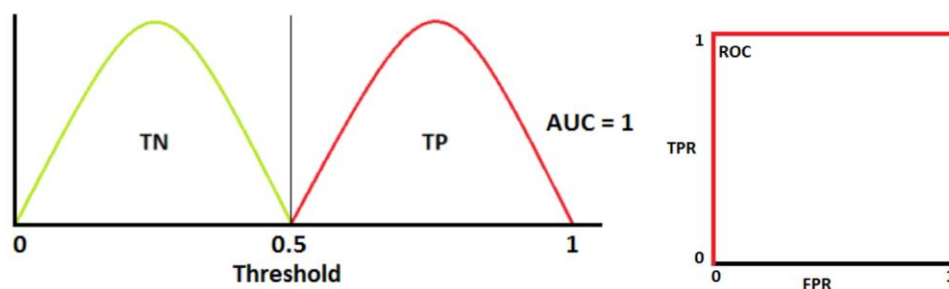


Figura 7 - AUC para um modelo de classificação excelente⁷

A imagem seguinte mostra uma área AUC igual a 0.7, ou seja, há 70% de probabilidade de o modelo distinguir corretamente as classes.

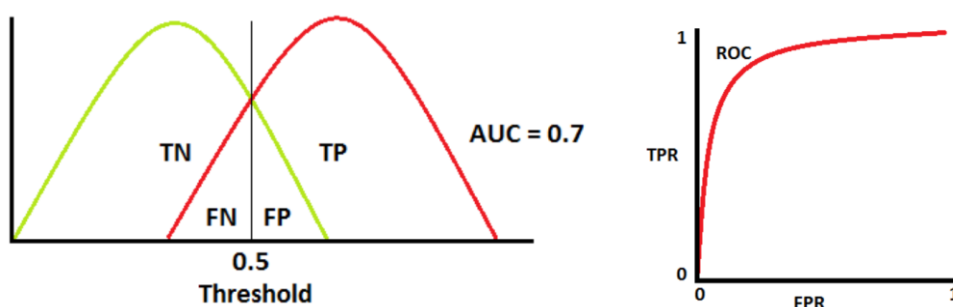


Figura 8 - $AUC = 0.7$ ⁷

A pior situação é quando o valor da área AUC é 0.5, ou seja, o modelo não consegue distinguir as classes.

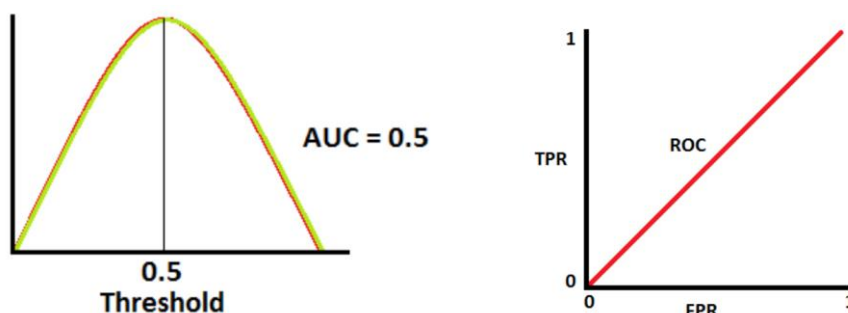
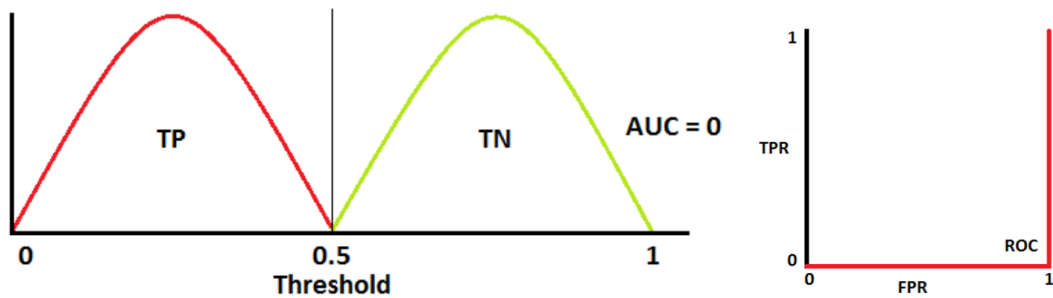


Figura 9 - $AUC = 0.5$ ⁷

No caso de o valor da área AUC ser igual a 0, significa que o modelo está a prever classes “0” como “1” e classes “1” como “0”.

⁷ <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>

Figura 10 - AUC = 0⁸

2.5. Tecnologias

2.5.1. Linguagem Python

Uma das 5 linguagens mais populares do mundo da programação, *Python* é uma linguagem de alto nível, dinâmica, interpretada, modular, multiplataforma e orientada a objetos, ou seja, é focada na criação de objetos que contêm tanto os dados como as funcionalidades. Atualmente, gerido pela organização sem fins lucrativos *Python Software Foundation*, possui um modelo de desenvolvimento comunitário e de código aberto. Para além disso, esta linguagem foi pensada e estruturada para o ensino da programação daí ser de sintaxe relativamente simples e de fácil compreensão, priorizando a legibilidade do código ao invés da velocidade ou expressividade e, por isso, tem ganho popularidade entre profissionais da indústria tecnológica que não são programadores [17].

Uma vez que é uma linguagem que pode receber atualizações por parte de terceiros, esta possui um grande número de bibliotecas, tornando-a difundida e útil numa grande variedade de setores, nomeadamente em áreas com análise de dados, *machine learning* e inteligência artificial.

Como falado anteriormente, as vantagens de aprender *Python* são muitas e vamos passar a explicar algumas delas e o porquê desta linguagem estar a crescer cada vez mais e a ser utilizada pelos programadores e pelas empresas.

1. **Simple e fácil de aprender:** por ser uma linguagem que tem uma sintaxe muito acessível e ter sido criada com o objetivo de aumentar a agilidade e produtividade de quem a utiliza, é absorvida rápida e facilmente, daí ter uma curva de aprendizagem relativamente baixa.

⁸ <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>

2. **Multiplataforma e extensível:** uma vez que é uma linguagem multiplataforma, o *Python* pode ser utilizado em qualquer sistema operacional desde que o seu interpretador esteja instalado. Para além disso, é uma linguagem muito conhecida por ser extensível, disponibilizando aos utilizadores mais de cento e vinte e cinco mil bibliotecas muito versáteis.
3. **Licença de uso público:** visto que é uma linguagem “*open source*”, ou seja, de código aberto, é totalmente gratuita e para a utilizar basta apenas instalar e começar a programar.
4. **Escassez de profissionais:** uma das grandes vantagens de ser um profissional especializado em *Python* deve-se ao facto de a concorrência profissional ser baixíssima, ou seja, ao nos destacarmos como bons profissionais desta linguagem teremos trabalho assegurado devido à falta de programadores na área.

2.5.2. IDE PyCharm

No mundo da programação, existem inúmeros programas que se podem utilizar para criarmos o código relativo ao projeto que temos em mãos. Esses programas são denominados de “*Integrated Development Environment*” (IDE) ou, em português, ambientes de desenvolvimento integrados. PyCharm não é mais do que um IDE usado em programação de computadores, especificamente para a linguagem *Python*. Desenvolvido pela empresa JetBrains, empresa essa que tem vários IDE’s para diversas linguagens, este é multiplataforma, ou seja, funciona tanto em sistemas operativos Linux, Windows e macOS, e possui duas versões, uma delas é grátis apesar de não ter todas as funcionalidades disponíveis, *Community Edition*, e a outra versão é paga, mas fornece recursos e funcionalidades extras, *Professional Edition* [18].

Apesar de se poder criar código em qualquer programa em que seja possível escrever, como por exemplo o Word ou o Bloco de Notas, nenhum destes conseguem fazer o que os IDE’s fazem. Neste caso, o *PyCharm* fornece aos programadores assistência e análise de código, com conclusão de código, destaque de sintaxe e ajudas rápidas; um *debugger*, ou seja, um programa examinador de código que o executa numa máquina virtual com o objetivo de detetar erros que possam existir nesse mesmo código; um testador unitário integrado, com cobertura de código linha a linha; contém integração com sistemas de controlo de versão (VCS), isto é, tem um sistema responsável por gerir configurações ou alterações no software; e também suporta desenvolvimento web bem como ciência dos

dados. Para além destas características, ainda permite navegar pelo projeto e pelo código, seja visualizar a estrutura do projeto ou navegar entre ficheiros, classes, funções e variáveis e também permite de uma maneira rápida extrair e renomear métodos, introduzir variáveis e constantes.

3. Metodologias

Várias foram as metodologias ágeis aprendidas ao longo da licenciatura e que nos ajudaram a desenvolver métodos de trabalho mais organizados e eficazes.

Um dos grandes desafios quando se inicia a realização de um projeto, deve-se ao facto de nós, programadores, conseguirmos aceder aos pedidos dos clientes garantindo que a sua execução é bem feita e que a entrega final está de acordo com o pretendido inicialmente pelo cliente. Para atingirmos esses objetivos da melhor maneira possível, houve a necessidade de criar estratégias e métodos capazes de nos auxiliar durante o desenvolvimento do projeto e daí surgiram as metodologias ágeis.

As metodologias ágeis surgiram com o objetivo de resolver problemas comuns a diversas empresas que precisam de gerir projetos e entregar produtos finais aos clientes. Lançadas na indústria da Tecnologia da Informação, estas vieram resolver problemas como a falta de comunicação e clareza de ideias entre os desenvolvedores e os clientes ou como a falta de orientação no que toca à duração das etapas de produção e à data de entrega do produto final ao cliente, garantindo que todos os projetos que fossem desenvolvidos estivessem bem organizados com princípio, meio e fim [19].

Ao contrário dos métodos mais antigos, as metodologias ágeis propõem ciclos de vida mais curtos, com entregas do produto bem definidas e focam-se na melhoria contínua da gestão da equipa e da comunicação desta com o cliente. Com o aparecimento destes novos métodos, o processo de identificar erros ou falhas durante a execução do projeto foi simplificado e as pessoas envolvidas ganharam mais flexibilidade e facilidade para alterar este último e evitar que certos contratempos afetassem o resultado final esperado.

Existem inúmeras metodologias ágeis e muitas organizações utilizam uma combinação destas, de acordo com as suas necessidades, com o objetivo de conseguirem que o processo de desenvolvimento seja mais rápido e eficiente. De acordo com o “*13th Annual State of Agile Report*”, as metodologias ágeis mais usadas são o *Scrum*, *Kanban* e *Extreme Programming (XP)* [20].

Nos próximos tópicos iremos, um pouco mais em detalhe, abordar os valores e princípios das metodologias ágeis e os três tipos de metodologias faladas no parágrafo anterior que nos ajudaram a realizar o projeto de uma maneira mais fácil e responsável.

3.1. Agile Manifesto

Em 2001, um grupo composto por dezassete pessoas debateu sobre novas abordagens de gestão de projetos. Apesar de algumas metodologias já existirem antes desta data, foi nesta reunião que foi oficializada a existência de metodologias ágeis e foram estabelecidos os princípios/regras que as caracterizam. Nessa reunião foi criado o *Agile Manifesto*.

De acordo com as experiências de desenvolvimento de software, as dezassete pessoas presentes na reunião escreveram num documento, quatro valores e doze princípios acerca do desenvolvimento ágil.

3.1.1. Valores

Após a finalização da reunião, foram definidos quatro valores que passamos a apresentar de seguida [21].

1. **“Indivíduos e interações** mais do que processos e ferramentas.”
2. **“Software funcional** mais do que a documentação abrangente.”
3. **“Colaboração com o cliente** mais do que negociação contratual.”
4. **“Responder à mudança** mais do que seguir um plano.”

Apesar de eles reconhecerem valor nos itens da direita, valorizam mais os itens à esquerda. Scott Ambler, um engenheiro de software canadiano, consultor e autor de vários livros sobre o desenvolvimento de software *Agile* e um dos dezassete elementos presentes na reunião, explicou um pouco mais aprofundadamente os quatro valores acima apresentados.

1. Apesar das ferramentas e os processos serem importantes, é mais importante ter pessoas competentes a trabalhar juntas de forma eficiente.
2. O ponto principal do desenvolvimento é criar o software e não a documentação, ainda que, uma boa documentação é benéfica para ajudar as pessoas a perceberem como o software foi criado e para as elucidar de como este deve ser utilizado.
3. Embora um contrato seja importante, um trabalho próximo aos clientes para perceber as suas necessidades não pode ser substituído.
4. Ter um plano pré-estabelecido é importante, mas não deve ser muito rígido para que seja possível fazer mudanças na tecnologia, aceder às prioridades e

preocupações dos clientes e para que estes entendam os problemas e as soluções encontradas.

Estes valores implementados veem demonstrar que a flexibilidade e a colaboração entre as partes são mais relevantes do que a rigidez de processos e do que o planeamento usado antigamente.

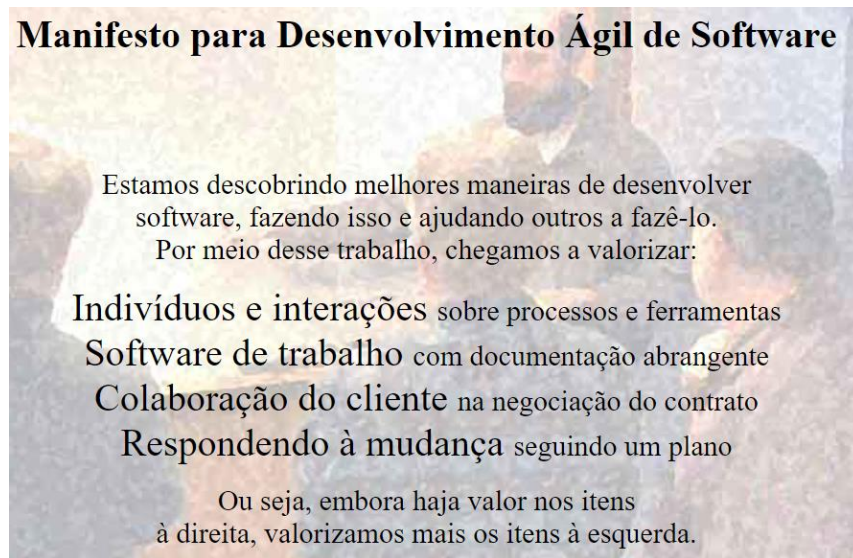


Figura 11 - Valores do Agile Manifesto⁹

3.1.2. Princípios

Da mesma maneira que os valores, também os princípios acerca das metodologias ágeis foram tornados públicos após a tal reunião que envolveu dezassete engenheiros. Foram doze esses princípios e iremos enunciá-los a seguir [22].

1. Garantir a satisfação do cliente, procurando entregar ao cliente software funcional de um modo rápido e contínuo.
2. Quaisquer mudanças ao projeto, sejam elas no início, meio ou fim, são bem-vindas.
3. Software funcional é entregue frequentemente. Deve ser entregue semanalmente ou mensalmente, dando preferência a períodos mais curtos.
4. O cliente e os desenvolvedores devem estar em cooperação constante, durante todo o decorrer do projeto.

⁹ <https://agilemanifesto.org/iso/ptpt/manifesto.html>

5. Usar indivíduos motivados para desenvolver software, oferecendo-lhes todo o apoio possível, confiando que conseguem cumprir todos os objetivos com distinção.
6. Passar informação entre a equipa de desenvolvimento e o cliente através de conversas pessoais e diretas é o melhor método e o mais eficaz.
7. A principal medida de progresso do projeto é software funcional.
8. O desenvolvimento deve ser sustentável, ou seja, todos os intervenientes no projeto devem manter o mesmo ritmo do início ao fim.
9. Deve-se estar constantemente atento à excelência técnica e ao design, garantindo assim o aumento da agilidade.
10. Simplicidade é essencial.
11. Equipas que se organizam autonomamente garantem melhores arquiteturas, requisitos e designs.
12. Em intervalos regulares, a equipa conversa sobre o que devem fazer para se tornar mais eficaz e posteriormente, ajusta o seu comportamento.

3.2. Scrum

Utilizado em mais de 60% dos projetos ágeis em todo o mundo, o *Scrum* é uma metodologia ágil para gestão e planeamento de projetos de software. É uma *framework* na qual as pessoas planeiam e desenvolvem produtos complexos e adaptativos com o mais alto valor possível. Não é uma técnica ou um processo para criar produtos, mas sim uma estrutura que pode empregar vários processos e técnicas. É uma metodologia que segue uma abordagem iterativa e incremental e um dos seus princípios é ter requisitos variáveis e de fácil alteração, garantindo assim que as empresas desenvolvedoras estão sempre preparadas para receber pedidos de alteração dos requisitos por parte dos clientes que mudam de ideias ao longo de todo o desenrolar do processo. Tem uma estrutura bem definida pois cada *role* tem tarefas bem estabelecidas e delineadas. O *Scrum* também segue uma abordagem empírica, isto é, no momento de início do desenvolvimento, o processo não é definido como compreendido a cem por cento, focando-se o *Scrum* em maximizar o número de entregas por parte da equipa, ajustar-se ao mercado atual e aceitar e realizar trocas de requisitos feitos pelos clientes [23].

AGILE METHODS AND PRACTICES

Agile Methodologies Used

Scrum and Scrum/XP Hybrid (64%) continue to be the most common agile methodologies used by respondents' organizations.

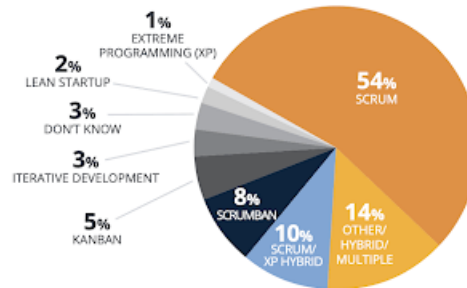


Figura 12 - Metodologias ágeis mais usadas¹⁰

3.3. Kanban

Fundador da *Kanban University*, *David J. Anderson* adaptou o *Kanban* como uma técnica para ser utilizada no desenvolvimento de software. Neste contexto, foi definido como “Um método para definir, gerar e melhorar serviços que entregam **trabalho de conhecimento**, tais como serviços profissionais, atividades criativas e o design de produtos físicos e de software”. Pode ser caracterizado como um método de “**começar a partir do que se conhece**”, reduzindo a resistência a mudanças benéficas de acordo com os objetivos da organização. Utiliza um sistema visual, de forma a representar o fluxo de trabalho.

A maior característica desta metodologia é o quadro de *Kanban*. Todos os envolvidos no processo devem visualizar o quadro, uma que vez, é nele que se acompanha o progresso do desenvolvimento desde o início até ao fim. Como em todas as metodologias, o *Kanban* não tem só uma característica que a define, mas sim várias, como é o caso do limite da quantidade de trabalho em andamento (*WIP – Work In Progress*). A existência destes limites conjugada com a representação do processo de desenvolvimento, cria um *pull system*. Este *pull system* consiste em puxar as tarefas de acordo com a disponibilidade da equipa de desenvolvimento em vez destas serem empurradas para o local pretendido quando terminadas. Estes limites vieram melhorar o fluxo de trabalho [24].

¹⁰ <https://stateofagile.com/#ufh-i-613553418-13th-annual-state-of-agile-report/7027494>

3.3.1. Práticas Gerais

O *Kanban* é uma metodologia que se rege por sete práticas gerais presentes na Figura 10. As principais práticas são: *Visualize*, *Limit Wip*, *Manage Flow*, *Make Policies Explicit*, *Feedback Loops* e por fim, *Improve & Evolve*.

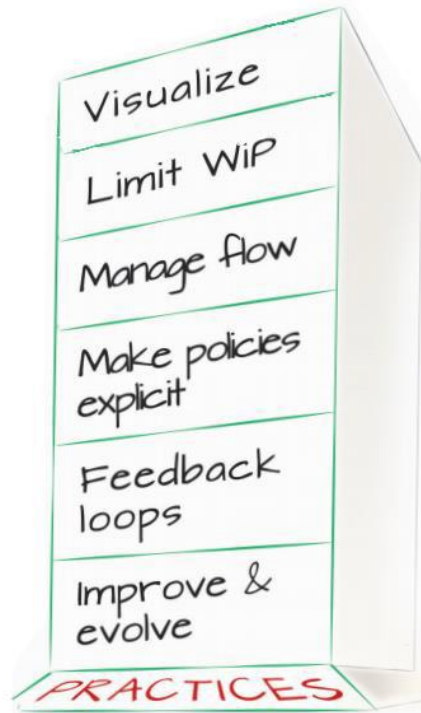


Figura 13 - Práticas gerais do Kanban¹¹

Quadro de *Kanban* (Visualize)

O quadro de *Kanban* (Figura 14) é uma ferramenta para visualização e um dos componentes chave da metodologia associada. Dividido por colunas, que representam uma etapa do fluxo de trabalho e por raia, que representam diferentes tipos de atividades. É importante que esteja bem definido para que seja possível fazer uma avaliação acerca do que precisa de ser feito antes de mover um item de uma coluna para a outra. A estrutura das colunas pode variar consoante o contexto em que estejamos inseridos, uma vez que esta metodologia não se restringe a apenas uma única estrutura. Para não afetar o fluxo de trabalho, é fundamental salientar os itens bloqueados.

¹¹https://ead.ipleiria.pt/2020-21/pluginfile.php/396871/mod_folder/content/0/kanban/7_ATSE_Kanban_white.pdf?forcedownload=1

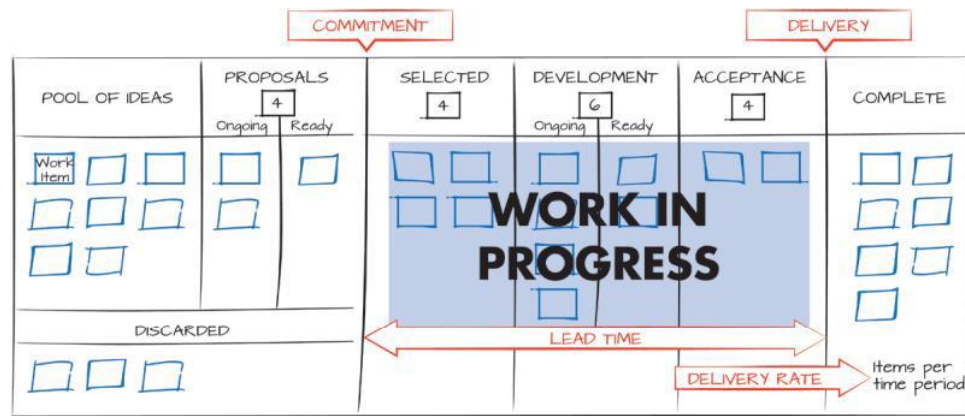


Figura 14 - Exemplo de quadro de kanban¹²

Limitar trabalho em andamento (*Limit WIP*)

Esta prática transforma um *push system* num *pull system*, ou seja, o trabalho passa a ser puxado de acordo com a quantidade de trabalho em vez de ser empurrado para o devido processo quando pedido. A existência de muito trabalho parcialmente completo acaba por ser muito dispendiosa a nível monetário, prolongado assim os prazos de entrega e impede que a organização seja responsiva, ou seja, não consegue ser tão competitiva como os seus competidores diretos. É fundamental observar, limitar e otimizar, para que o limite dos prazos de entrega e a qualidade melhorem, garantindo assim o aumento das taxas de entregas e a satisfação dos clientes.

Gerir fluxo de trabalho (*Manage Flow*)

O principal objetivo desta prática consiste em maximizar a entrega de itens de valor e minimizar o tempo de entrega. Assim irá ser possível entregar software com a melhor qualidade possível no menor curto espaço de tempo possível. Para atingir tal objetivo, é necessário existir boa comunicação, cooperação e organização e um bom fluxo de trabalho por parte de todos os membros da equipa de desenvolvimento.

Criar políticas específicas (*Make Policies Explicit*)

As políticas criadas devem ser bem definidas, visíveis e sempre aplicadas de forma que sejam facilmente modificáveis. Estas políticas devem ser explícitas e alteradas quando alguma delas for contraproduktiva ou não aplicada.

¹²https://ead.ipleiria.pt/2020-21/pluginfile.php/396871/mod_folder/content/0/kanban/7_ATSE_Kanban_white.pdf?forcedownload=1

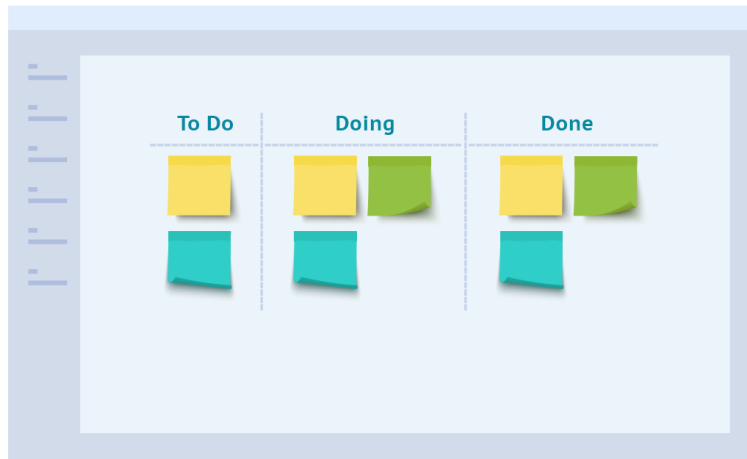


Figura 15 - Políticas do kanban¹³

Ciclos de feedback (*Feedback Loops*)

Os ciclos de feedback são essenciais para garantir um desenvolvimento controlado e fundamental para mudanças evolutivas. É com esta prática que se analisa como está a correr o processo de desenvolvimento e se tentar otimizar e melhorar este.

São várias as áreas de melhoria em que se incidem estes ciclos:

- Estratégia
- Coordenação Operacional
- Gestão de riscos
- Melhoramento do serviço
- Reposição
- Fluxo
- Entregas ao cliente

Evoluir e melhorar (*Improve and Evolve*)

A metodologia *Kanban* está diretamente relacionada com melhoria. Sabendo que os processos estão em constante mudança, esta prática assenta sobre a procura contínua e incremental de melhorias, com o objetivo de evoluir a sua forma de desenvolvimento para melhor.

¹³ <https://www.digite.com/pt-br/kanban/o-que-e-kanban/>

A evolução envolve diferenciação e constante manutenção e amplificação de alterações úteis entre ciclos, garantindo deste modo um bom fluxo de desenvolvimento.

3.4. Extreme Programming (XP)

O *Extreme Programming* (XP) é uma metodologia ágil que tem como objetivo o excelente desenvolvimento de software. Esta metodologia garante que o software pode ser desenvolvido a um custo menor, com menos defeitos e com maior produtividade permitindo assim um grande retorno sobre o investimento feito.

É uma metodologia leve, baseada na abordagem de restrições no desenvolvimento de software e aborda riscos de todos os níveis de desenvolvimento. Consegue trabalhar com equipas de qualquer tamanho e adapta-se a requisitos vagos ou que mudam rapidamente. Valoriza totalmente o esforço feito, avalia a contribuição para os objetivos partilhados da equipa e pede para que algumas das necessidades humanas sejam atendidas através do desenvolvimento de software [25].

Para além disso, o XP assenta numa filosofia de desenvolvimento de software baseada em valores; tem um conjunto de práticas úteis cujo objetivo é melhorar o desenvolvimento; contém um conjunto de princípios complementares que traduzem valores em práticas; conta com uma comunidade que partilha valores e práticas. Defende lançamentos frequentes e ciclos de desenvolvimento curtos, tendo como objetivo aumentar e melhorar a produtividade e responder rapidamente a alterações impostas pelo cliente. Recomenda a programação em pares, pois duas ideias diferentes podem valorizar o software de várias maneiras e enquanto um elemento está a programar, o outro tem mais facilidade em encontrar erros, tornando o processo de desenvolvimento mais eficiente. Para além disso, esta metodologia permite a revisão contínua do código, evita a programação de funcionalidades que não são necessárias e faz uso de código claro e simples.

3.5. Metodologia Utilizada

De acordo com todas as metodologias mencionadas nos tópicos acima, a que se adequa mais ao sistema que utilizámos durante o processo de desenvolvimento deste projeto, foi a *Extreme Programming* (XP). A maior parte do desenvolvimento do software pedido foi feito em *pair programming* (programação em pares), ou seja, enquanto um elemento do grupo estava a escrever o código da aplicação, o outro estava a ver e a tentar descobrir possíveis

problemas ou *bugs* que iam aparecendo. Passado algum tempo, as posições eram trocadas e quem estava a programar, ia ficar a ver e a descobrir problemas. Esta técnica garantia que todo o código era conhecido por ambos os membros dos grupos e tornou o seu desenvolvimento mais rápido. Para além disso, tentámos ao máximo desenvolver apenas funcionalidades pedidas e absolutamente necessárias para o correto funcionamento do software pretendido.

No início do desenvolvimento foi-nos proposto um conjunto de funcionalidades que o projeto deveria ter e o que elas deveriam fazer, no entanto, com o avanço do desenvolvimento e das reuniões semanais que iam acontecendo, alguns ajustes e alterações foram feitas de acordo com o pretendido por parte dos orientadores e também com algumas sugestões provenientes da nossa parte, de forma que o software fosse mais funcional, rápido e sugestivo.

4. Arquitetura e Requisitos Funcionais

4.1.Requisitos Funcionais

Após várias reuniões onde foram abordados vários tópicos acerca do tema inteligência artificial e *machine learning*, foi feito pela nossa parte, uma pesquisa aprofundada na tentativa de encontrar requisitos e funcionalidades a implementar na aplicação. Posteriormente, foi-nos fornecido um conjunto de pontos chave, que em conjunto com os requisitos por nós obtidos, viriam a ser implementados para que nos fosse possível desenvolver um software funcional, sugestivo e que fosse ao encontro daquilo que era pretendido pelos orientadores. Como já referido anteriormente, o objetivo da aplicação é conseguir comparar métodos de *machine learning* (de classificação de dados de tráfego de rede) executados em condições idênticas. Para tal, o desenvolvimento do software foi dividido em cinco partes, das quais surgiram os “Projetos”, “Dados de Entrada”, “Tarefas de Pré-Processamento”, “Tarefas de Treino, Teste e Classificação”, e por fim, “Métricas e Geração de Relatórios”. Seguidamente, estes cinco tópicos irão ser explicados de uma forma sucinta, uma vez que no próximo capítulo iremos abordar a maneira como os implementámos.

4.1.1. Projetos

Qualquer aplicação, relacionada com inteligência artificial ou não, que permita gravar dados, dá a possibilidade ao utilizador de escolher o local onde pretende gravar esses dados. Neste caso, a aplicação foi desenvolvida de maneira que o utilizador pudesse criar um projeto, ou seja, uma pasta que contém todas as configurações feitas pelo utilizador, desde o ficheiro de dados utilizado, as transformações aplicadas a esse ficheiro e os parâmetros e métodos de classificação que foram utilizados. Por outras palavras, cada projeto representa a avaliação de um ou mais algoritmos em determinado ficheiro de dados.

Para além disso, também deveria ser possível interromper e reiniciar execuções longas de projetos, como por exemplo, parar a meio o treino de um modelo.

Uma vez que o utilizador irá ter a possibilidade de fazer várias ações ao longo de todo o processo de classificação dos dados, um ficheiro será criado aquando da criação do projeto e irá contemplar todas as ações feitas pelo utilizador, desde as informações sobre as

configurações utilizadas, os dados de entradas e operações realizadas, registos com a data e hora da realização de uma determinada tarefa, entre outras.

Para que o utilizador não precise de estar constantemente a criar um novo projeto sempre que precise de realizar alguma ação a um ficheiro de dados anteriormente utilizado e gravado num projeto antigo, será disponibilizada uma opção para carregar esse projeto. Quando carregado, o ficheiro de dados, os modelos e os relatórios sobre todas as ações realizadas anteriormente estarão disponíveis para consulta por parte do utilizador e este, poderá fazer mais alterações que irão ser guardadas no mesmo projeto.

4.1.2. Dados de Entrada

Para que seja possível avaliar, através de um ou mais algoritmos, um determinado conjunto de dados, necessitamos de especificar esses mesmos dados. Como tal, o utilizador irá ter de selecionar o ficheiro de dados que pretende classificar. Estes serão guardados no sistema e apresentados em formato de tabela. Ao analisarmos o ficheiro selecionado, veremos um conjunto de linhas e colunas, em que, o sistema irá identificar a primeira linha como a linha que contém informação sobre o nome de cada coluna. Estes nomes poderão ser alterados pelo utilizador.

De modo a ser possível aos algoritmos classificar os dados fornecidos, o utilizador terá, obrigatoriamente, de selecionar a coluna “*target*”, ou seja, a coluna que irá servir de comparação aquando da classificação com os diversos algoritmos.

4.1.3. Tarefas de Pré-Processamento

As tarefas de pré-processamento dão origem a um conjunto de dados modificados, que servirão de input para tarefas de treino e teste. Para facilitar a análise dos dados, a aplicação irá permitir a exploração destes através de funções de sumarização e exibição de estatísticas sobre cada uma das colunas. Entende-se por estatísticas o valor médio, a mediana, o valor máximo e mínimo, quantidade de linhas e a quantidade de linhas com valor vazio.

Estas tarefas permitem também fazer a limpeza/tratamento dos dados, como por exemplo, remover um conjunto de linhas do conjunto dos dados. As linhas podem ser eliminadas com base nos valores de uma coluna, ou seja, o utilizador especifica a coluna, o valor e se pretende eliminar as linhas cujo valor seja superior, inferior ou igual ao valor por ele inserido. O utilizador poderá também remover as linhas da coluna escolhida onde o valor

está vazio. Caso este queira manter as células que têm os valores vazios, terá a possibilidade de preencher esses campos com o valor da média ou da mediana da coluna selecionada.

A aplicação permitirá aplicar técnicas de *encoding*, isto é, converter valores nominais para valores numéricos. Escolhida a coluna, será possível utilizar estas e outras técnicas, como as técnicas de normalização.

Para que seja possível classificar os dados de entrada, com base nos algoritmos escolhidos, o sistema possibilitará a divisão destes em conjuntos de treino e de teste. Esta divisão poderá ser feita usando diferentes critérios, tais como baseado em percentual de linhas (*stratified split*, *percentual split*) e variações de *cross validation* (*stratified k-fold*, *k-fold cross validation*). Estes critérios irão ser aprofundados no capítulo seguinte. Todas estas instruções irão ser gravadas num ficheiro para que seja possível ao utilizador saber todas as ações que realizou.

4.1.4. Tarefas de Treino, Teste e Classificação

Após a realização das tarefas de tratamento dos dados, a próxima etapa será aplicar as tarefas de treino, teste e classificação. Para a primeira tarefa, o sistema irá permitir ao utilizador a escolha de um que será aplicado aos dados resultantes das tarefas anteriores ou aos dados originais, caso estes não tenham sofrido nenhuma alteração até este momento. No entanto, como o objetivo da aplicação é comparar resultados obtidos através da utilização de diferentes algoritmos em condições semelhantes, esta permitirá ao utilizador selecionar mais do que um algoritmo. Como todos os algoritmos são diferentes, o sistema permitirá a especificação dos parâmetros necessários em função do algoritmo escolhido.

No final do treino, ou seja, após a especificação dos algoritmos e dos seus parâmetros, o modelo treinado será guardado num ficheiro na pasta do projeto. A aplicação permitirá realizar tarefas de teste e/ou de classificação com ou sem a realização do treino de modelos, isto é, o utilizador poderá realizar estas tarefas usando o modelo treinado anteriormente ou escolher um modelo guardado de outro projeto. Neste último caso, o utilizador terá a possibilidade de escolher o modelo que quer utilizar.

A aplicação terá um conjunto de algoritmos por nós fornecidos, no entanto, o utilizador poderá inserir ficheiros com algoritmos desenvolvidos por ele. Estes algoritmos têm de seguir um conjunto de regras também por nós definidas pois, caso contrário, o algoritmo

inserido não irá funcionar. Após a inserção do novo ficheiro com o novo algoritmo, o utilizador poderá logo utilizá-lo e ficará guardado para sempre na aplicação.

4.1.5. Métricas e Geração de Relatórios

Terminada a realização das tarefas de treino, teste e classificação e a construção de modelos treinados, é necessário apresentar os resultados e comparar os valores obtidos. Para tal, o sistema irá gerar valores de métricas de interesse sobre os resultados das operações de teste e classificação. As métricas a ser geradas são: *precision*, *recall*, *f1-score*, *log loss* e *ROC-AUC*. Para além disso, também será criada a matriz de confusão (*Confusion Matrix*). Todas estas métricas e a matriz de confusão estão explicados no capítulo 2 “Conceitos e Tecnologias Utilizadas”, no tópico “Métricas”.

Como falado no tópico anterior, o projeto poderá contemplar mais do que um classificador. Neste caso, serão gerados relatórios comparativos entre os métodos utilizados.

Para que seja possível consultar todas as operações realizadas, todos os parâmetros de configuração do projeto e parâmetros utilizados para treinar os modelos, um ficheiro será criado contemplando toda esta informação e também informação sobre a data e hora de execução de cada tarefa.

4.2.Arquitetura

Atendendo ao facto de não termos usado componentes externos no nosso projeto, a arquitetura pode-se dizer relativamente simples. O projeto é constituído por um *backend*, onde foi desenvolvido todo o código para lidar com os dados e parâmetros relativos à aplicação, e um *frontend*, que permite ao utilizador interagir com a aplicação. Os dados da aplicação e dos projetos criados são guardados numa estrutura de ficheiros, estrutura essa que vamos detalhar de seguida.

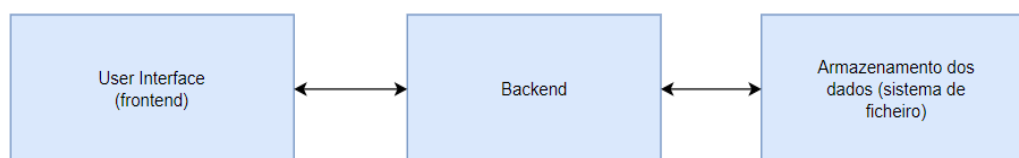


Figura 16 - Arquitetura geral da aplicação

4.2.1. Estrutura de ficheiros

Uma vez que tivemos de lidar com ficheiros, nomeadamente ficheiros CSV, para carregar os dados necessários para o correto funcionamento da aplicação, decidimos também escolher a utilização de ficheiros para gravar os dados relativos à aplicação.

Todos os dados resultantes da escolha do dataset, do tratamento dos dados, da aplicação dos algoritmos e da utilização da aplicação são guardados em ficheiros.

Aquando da criação de um novo projeto na aplicação, é criada uma pasta relativa a esse projeto. Esta pasta contém as seguintes ficheiros e diretorias:

Na diretoria raiz do projeto:

log.txt – ficheiro onde são guardadas as principais ações realizadas na aplicação, bem como a data e hora em que foram realizadas.

```

102 2021-06-21 16:28:51,098 - Stratified K-Fold applied to split data with 7 splits
103 2021-06-21 16:28:51,270 - Stratified K-Fold applied to split data with 7 splits
104 2021-06-21 16:28:51,435 - Stratified K-Fold applied to split data with 7 splits
105 2021-06-21 16:28:51,581 - Stratified K-Fold applied to split data with 7 splits
106 2021-06-21 16:29:36,318 - Column names of datafile shown
107 2021-06-21 16:29:36,319 - Column names printed in listbox
108 2021-06-21 16:29:45,903 - Percentage split applied to split data with test_size equals to 0.26
109 2021-06-21 16:34:33,331 - Column names of datafile shown
110 2021-06-21 16:34:33,331 - Column names printed in listbox
111 2021-06-21 16:34:40,716 - Stratified K-Fold applied to split data with 8 splits
112 2021-06-21 16:42:59,080 - Column names of datafile shown
113 2021-06-21 16:42:59,080 - Column names printed in listbox
114 2021-06-21 16:43:08,314 - Percentage split applied to split data with test_size equals to 0.22
115 2021-06-21 16:43:08,330 - Logistic Regression used
116 2021-06-22 15:16:13,195 - Column names of datafile shown
117 2021-06-22 15:16:13,195 - Column names printed in listbox
118 2021-06-22 15:16:27,025 - Percentage split applied to split data with test_size equals to 0.3
119 2021-06-22 15:16:27,043 - Logistic Regression used
120 2021-06-22 15:16:54,440 - Column names of datafile shown
121 2021-06-22 15:16:54,440 - Column names printed in listbox
122 2021-06-22 15:34:03,894 - Column names of datafile shown
123 2021-06-22 15:34:03,895 - Column names printed in listbox
124 2021-06-22 15:34:35,720 - Stratified K-Fold applied to split data with 3 splits

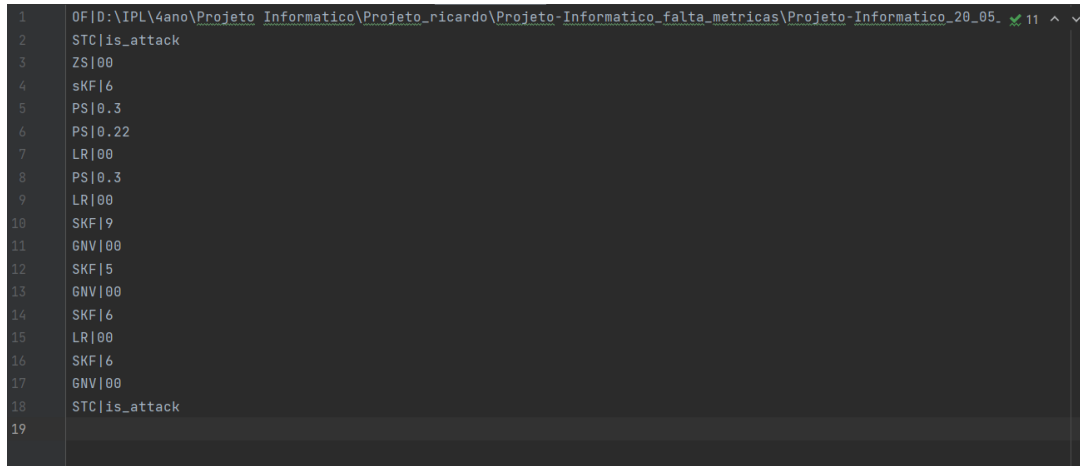
```

Figura 17 - Estrutura do ficheiro log.txt

DataFile{Nome_do_projeto}.csv – ficheiro resultante da limpeza e tratamentos dos dados do ficheiro escolhido. É o ficheiro que vai ser utilizado nos seguintes passos da aplicação.

{Nome_do_projeto}syslog.txt – ficheiro destinado a guardar os passos realizados durante a criação do projeto, para ser possível mais tarde carregar e correr esse projeto. Este ficheiro está estruturado da seguinte forma: cada linha guarda um keyword relativa ao passo que a aplicação deve executar, seguida dos parâmetros (caso existam) utilizados

na execução desse passo. Cada um destes parâmetros é separado pelo caracter ‘|’. Caso não existam parâmetros, existirá apenas uma separação pelo | e o segundo parâmetro será “00”. A imagem seguinte ilustra a estrutura acima descrita.



```

1 0FID:\IPL\4ano\Projeto_Informatico\Projeto_ricardo\Projeto-Informatico_falta_metricas\Projeto-Informatico_20_05. 11 ^ v
2 STC|is_attack
3 ZS|00
4 SKF|6
5 PS|0.3
6 PS|0.22
7 LR|00
8 PS|0.3
9 LR|00
10 SKF|9
11 GNV|00
12 SKF|5
13 GNV|00
14 SKF|6
15 LR|00
16 SKF|6
17 GNV|00
18 STC|is_attack
19

```

Figura 18 - Estrutura do ficheiro syslog.txt

Dentro da diretoria “Models”:

Contem todos os ficheiros, com extensão “.pkl”, que guardam o modelo de cada algoritmo treinado no projeto. Os nomes destes ficheiros têm o seguinte formato: **{Sigla_do_algoritmo}model_{data_hora_do_treino}.pkl**

Dentro da diretoria “Plots”:

Contem os gráficos resultantes da aplicação dos algoritmos. Os nomes destes ficheiros têm o seguinte formato: **{Sigla_do_algoritmo}model_{data_hora_do_treino}_{nome_do_grafico}.png**

Dentro da diretoria “Results”:

Contem os ficheiros com as métricas resultantes do treino e teste de cada algoritmo sobre o dataset. Os nomes destes ficheiros têm o seguinte formato: **{Sigla_do_algoritmo}metrics_{data_hora_do_treino}.txt**

A estrutura de ficheiro explicada acima pode ser visualizada na seguinte imagem:

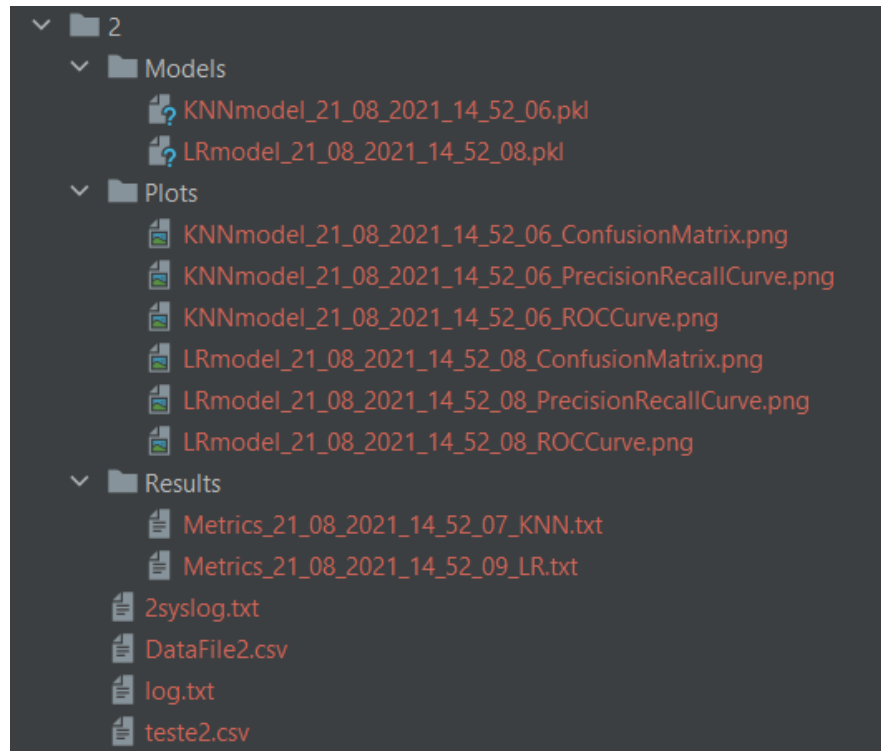


Figura 19 - Estrutura de ficheiros e diretorias

4.2.2. Backend

Relativamente à componente do nosso projeto que trata os dados fornecidos pelo utilizador, aplica os algoritmos e gera as métricas resultantes da aplicação de cada algoritmo no *dataset* escolhido, está estruturada da seguinte forma:

File.py – este ficheiro contém todas as funções necessárias para o carregamento de um determinado *dataset* (ficheiro .csv), criação da cópia desse ficheiro, que será usada no resto da aplicação e alterações feitas no ficheiro cópia, tais como alterações de valores ou eliminação de linhas ou colunas.

Table.py – contem todas as funções relativas à alteração e limpeza de dados realizadas na fase do tratamento do *dataset* escolhido pelo utilizador, tais como calculo de médias, medianas, contagem de ‘nulls’, aplicação de técnicas de normalização e *encoding*.

LogFile.py – contem as funções necessárias para registar as operações feitas na aplicação, estes ficheiros de log estão segmentados pelos vários projetos criados.

ProjectLogFile.py – contem as funções necessárias para guardar os passos realizados pelo utilizador no seu projeto, permitindo que este seja retomado mais tarde em

caso de necessidade, ou que eventualmente seja corrido de novo com os parâmetros e especificações guardadas.

Classify.py – contem as funções que permitem aplicar os algoritmos no *dataset* previamente preparado.

GUI.py – contem o código responsável pela interface apresentada ao utilizador. Trata também da receção dos dados introduzidos na aplicação e da apresentação de resultados.

Na diretoria **Algoritmos**, estão os ficheiros *.py* que contem os algoritmos existentes no projeto. O formato do nome destes ficheiros é: {Nome_do_algoritmo}.py.

4.2.3. User Interface

Tal como foi mencionado acima, a implementação referente à interface do utilizador encontra-se no ficheiro GUI.py. Esta interface é composta por uma página inicial com o nome da aplicação e onde podemos escolher se pretendemos criar um projeto novo, ou carregar um projeto já existente.

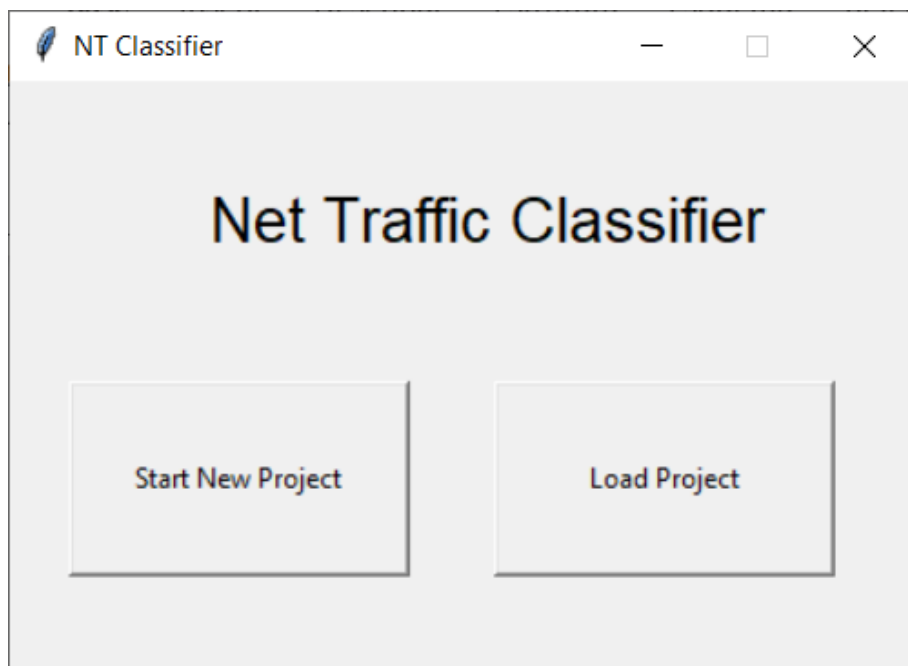


Figura 20 - Página inicial da aplicação

Criando um projeto novo, a aplicação pede ao utilizador que escolha um *dataset*, que poderá depois modificar conforme necessitar. Após esta etapa somos levados para o ecrã

seguinte, onde o utilizador pode editar o *dataset* e deixá-lo pronto a ser usado pelos algoritmos de *machine learning*.

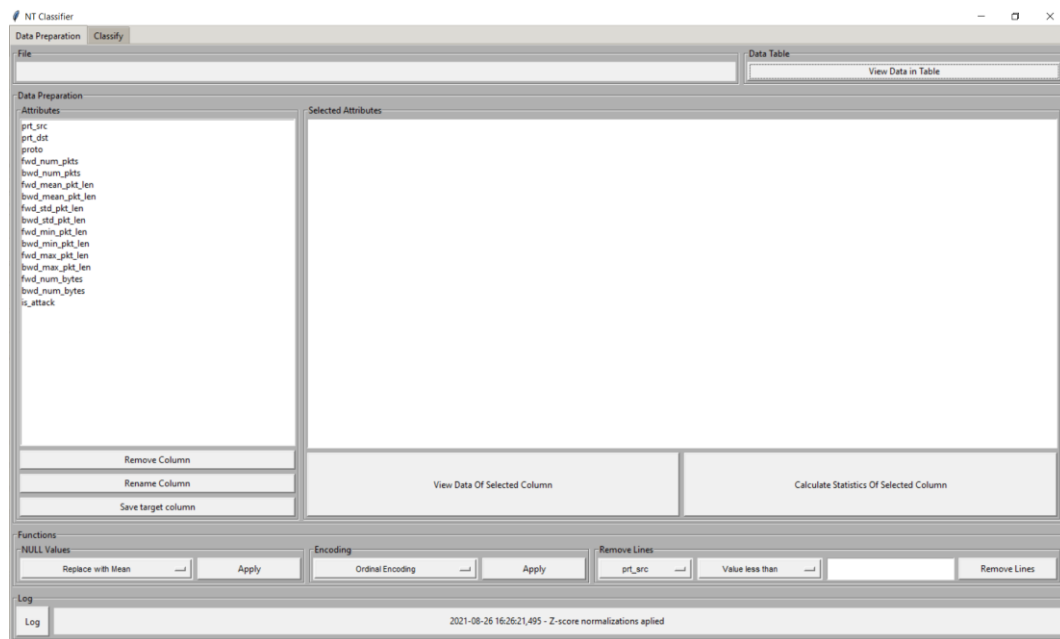


Figura 21 - Interface da preparação dos dados

Uma vez que a interface do utilizador está organizada por separadores, como se pode ver na imagem, após o *dataset* estar pronto a ser usado, seguimos para a interface da classificação do *dataset*, onde são aplicados os algoritmos. Aqui é possível escolher o tipo de divisão de treino e teste do *dataset* que queremos. De seguida aplicamos 1 ou mais algoritmos sobre o *dataset*. Após este passo podemos comparar as métricas geradas por cada algoritmo aplicado sobre o *dataset*.

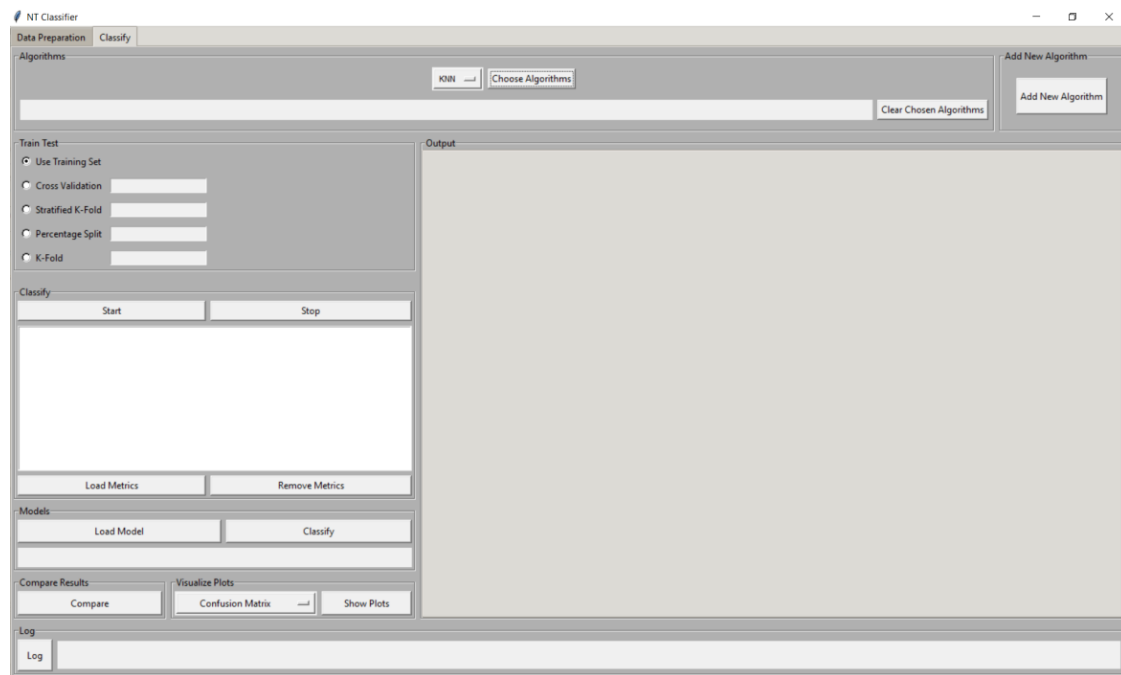


Figura 22 - Interface da classificação

4.3. Tecnologias e bibliotecas

4.3.1. Tkinter

No desenvolvimento desta aplicação utilizámos bibliotecas que nos ajudaram no nosso trabalho. Relativamente ao desenvolvimento da interface gráfica, recorremos à biblioteca do Python **Tkinter**. Com várias funções para desenho de botões, caixas de textos e outros elementos de visualização gráfica, esta biblioteca tem a vantagem de ser multiplataforma, ou seja, o mesmo código funciona tanto em Windows, macOS ou Linux. Os elementos visuais são renderizados usando elementos visuais nativos dos sistemas operativos, de forma que as aplicações aparentam pertencer à plataforma onde estão a correr [26].

4.3.2. Scikit-learn

A biblioteca *Scikit-learn*, ou *Sklearn* na sua forma abreviada, é uma biblioteca de *machine learning open-source*, para a linguagem de programação *python*. Esta biblioteca contém várias funções, que utilizamos no nosso projeto, para nos auxiliar tanto no treino e teste dos algoritmos de *machine learning* como na divisão dos *datasets* em conjuntos de treino e de teste, ou mesmo na geração das métricas resultantes da aplicação dos algoritmos [27].

4.3.3. Pandas

Para nos ajudar na manipulação dos dados dos *datasets*, utilizamos a biblioteca Pandas. Apesar das várias funções presentes nesta biblioteca, as funções para lidar com ficheiros *.csv* e com *arrays* de dados, foram as principais funções utilizadas [28].

4.3.4. Statistics

Esta biblioteca contém várias funções de calculo matemático. No nosso projeto usamo-la no cálculo das estatísticas relativas ao *dataset* escolhido, nomeadamente a média, mediana, desvio padrão, máximo e mínimo.

Utilizamos ainda outras bibliotecas para nos auxiliar em várias funções do projeto, particularmente funções para lidar com ficheiros no Windows, como a biblioteca **Path** e **os**, e bibliotecas para manipular os modelos gerados pelo treino dos algoritmos, como a biblioteca **jobLib**.

5. Implementação

A aplicação final resultou de um conjunto de pequenas implementações desenvolvidas consoante as necessidades desta. Nos próximos subcapítulos iremos abordar os diversos sistemas e métodos desenvolvidos e de que forma interagem com a aplicação.

5.1. Sistema de Carregamento e Edição dos Ficheiros de Dados

Para que a aplicação funcione de forma correta o primeiro passo é escolher um ficheiro de dados, ou *dataset*, para ser carregado e modificado pelo utilizador para ficar pronto a ser utilizado pelos algoritmos de *machine learning*. Para tal criamos uma sequência de funções que lidam com o ficheiro de dados escolhido. Todas estas funções encontram-se no ficheiro **File.py**.

No construtor encontramos várias variáveis que são usadas nas restantes funções do ficheiro.

```
class File:

    def __init__(self):
        self.filePath = ""
        # reference to the open file
        self.fileReference = None
        # number of lines and columns of the file
        self.fileNumLines = 0
        self.fileNumColumns = 0
        # array that stores the lines in the file
        self.lines = []
```

Figura 23 - Variáveis do construtor no ficheiro File.py

Este processo inicia-se na função *initDataFile()*. Esta função recebe como parâmetros a diretoria do ficheiro origem (ficheiro para ser carregado para a aplicação) e a diretoria onde vai ser criada a cópia do ficheiro, usada nos restantes passos da aplicação, respetivamente. Esta função copia linha a linha do ficheiro origem para o ficheiro cópia.

De seguida, a função *openDataFile()*, recebendo a diretoria do ficheiro cópia criado anteriormente como parâmetro, trata de abri-lo para poder ser lido e editado conforme os dados são modificados na aplicação.

Sempre que é necessário obter um determinado valor do ficheiro cópia para ser usado na aplicação, recorremos á função *getValue()*. Esta função recebe por parâmetros o índice da linha e o índice da coluna correspondentes ao valor que se pretende obter.

Para a guardar as alterações feitas no ficheiro cópia temos a função *setValue()*, com o índice da linha, o índice da coluna e o novo valor para ser substituído como argumentos de entrada. Esta função é também utilizada para definir o valor da célula especificada como sem valor. Neste caso, o parâmetro **newValue**, deve conter o valor “None”.

A função *getLine()* é responsável pelo retorno de uma determinada linha, especificada no parâmetro de entrada.

Para melhorar o desempenho da aplicação, no caso de querermos alterar todos os valores de uma linha, criámos a função *setLine()*, que altera toda a linha especificada no parâmetro **lineIndex** com os valores recebidos no parâmetro **newLine**, evitando assim demasiadas chamadas à função *setValue()* e acelerando o processo. Tal como a função **setValue**, esta função também é usada para eliminar a linha contida no parâmetro de entrada **lineIndex**. Para tal o valor do parâmetro **newLine** deve ser “None”.

À semelhança do que acontece com a duas funções descritas imediatamente acima que, no caso, são funções que incidem na manipulação das linhas do ficheiro, as próximas duas incidem na manipulação das colunas do ficheiro. A função *getColumn()* recebe, como argumento de entrada, o índice da coluna pretendido e retorna a coluna correspondente a esse índice.

A função *setColumn()* recebe como parâmetros o índice da coluna e um *array* com a coluna nova a ser escrita no ficheiro. Assim, como na função *setLine()*, no caso do parâmetro “*newColumn*” ter o valor “None”, a coluna é eliminada. Caso contrário, a coluna é reescrita com o valor contido nessa variável.

Uma vez que o ficheiro de dados carregado pode conter células sem valor, neste caso, em termos visuais corresponde a duas virgulas seguidas (Ex: 2,4,5) uma vez que se trata de um ficheiro .csv, tivemos a necessidade de criar uma função para lidar com estes casos, para

que nas restantes etapas do projeto estas células sem valor possam ser apagadas ou substituídas mais facilmente. Para tal criámos a função *fixEmptyCommas()*, que recebe como argumento de entrada a linha especificada na chamada à função e acrescenta o carácter ‘*’ sempre que houver uma célula sem valor. No caso do exemplo dado acima ficaria 2,*,4,5.

Uma vez que a leitura do ficheiro por parte da aplicação é feita linha a linha no caso das linhas e coluna a coluna no caso das colunas de forma a otimizar o processo, precisamos de uma função que divida essa linha ou coluna em componentes, onde cada componente é um valor, de forma a ficarmos com um *array* em que cada posição é um valor da linha/coluna. O método *stringToComponents()*, faz precisamente essa função, recebendo como parâmetro de entrada a linha especificada.

No caso contrário, quando queremos escrever no ficheiro necessitamos de uma função que junte todos os valores de uma linha ou coluna numa *string*, que será a linha ou coluna escrita no ficheiro. A função responsável por esta tarefa é a função *componentsToString()*, que recebe como argumentos de entrada um *array* com os valores a serem escritos no ficheiro.

A função *updateFile()*, é responsável por atualizar todas as alterações feitas na aplicação no ficheiro cópia.

5.2.Sistema de Cálculo de Estatísticas, Normalizações e *Encoding*

A aplicação permite ao utilizador ver várias estatísticas acerca do *dataset*, nomeadamente a média, mediana, máximo, mínimo, número de amostras e número de linhas sem valor. Para além disso é possível também aplicar normalizações nos dados nomeadamente o *Z-score*, aplicado em colunas numéricas, e *encoding*, aplicado nas colunas nominais, para passar esses valores para valores numéricos, de forma a poderem ser usados pelos algoritmos de *machine learning*.

Todos estes cálculos são feitos no ficheiro *Table.py*. O código presente neste ficheiro faz a ligação entre a interface do utilizador e o ficheiro *File.py* que lida diretamente com o ficheiro de dados. Em termos de argumentos, esta classe apenas precisa de receber uma referência para o ficheiro dos dados.

```
class Table:

    def __init__(self, dataFile):
        self.dataFile = dataFile
        # self.statFile = statFile
```

Figura 24 - Construtor da classe *Table*

Para o cálculo das estatísticas acima mencionadas e aplicação das normalizações e *encoding*, para além das funções que realizam estas operações são precisas funções auxiliares, que criamos e passamos a explicar.

A função *setValue()* recebe como parâmetros o índice da linha e da coluna e o valor a ser escrito no ficheiro. Esta função chama a função *setValue* presente no ficheiro *File.py*.

A função *getValue()*, tal como a função acima descrita, chama a função *getValue()* do ficheiro *File.py* explicada anteriormente. Esta função recebe como argumentos de entrada o índice da linha e da coluna das quais se pretende obter o valor.

Para termos acesso ao nome das colunas do ficheiro criamos a função *getColumnNames()*. Esta função acede ao ficheiro para devolver apenas a primeira linha, ou seja, a linha que contem o nome das colunas.

As funções *getLine()* e *getColumn()* devolvem a linha e a coluna especificada nos parâmetros de entrada respetivamente. Tando uma como outra chamam as funções com o mesmo nome presente no ficheiro *File.py*, previamente explicadas.

Para apagar uma linha ou uma coluna recorreremos à função *deleteLine()* e *deleteColumn()* respetivamente. Estas recebem como parâmetro o índice da linha e o índice da coluna respetivamente. Para apagar uma linha recorreremos à função *setLine()* do ficheiro *File.py* passando como argumento o valor “None” para que a linha seja apagada, tal como foi explicado no tópico anterior. O mesmo se verifica com a função *deleteColumn()* no caso de querermos apagar uma coluna. Todas estas funções explicadas acima serão usadas como funções auxiliares para o resto dos métodos presentes neste ficheiro, que englobam o cálculo de estatísticas, normalização e *encoding*.

Para o cálculo da média e da mediana de cada coluna do ficheiro de dados, criámos respetivamente as funções *calculateMean()* e *calculateMedian()*. Ambas as funções recebem como argumento de entrada o índice da coluna, e percorrem todos os elementos dessa coluna guardando-os num *array*. Tanto para o cálculo da média, como para o cálculo da mediana recorremos a duas funções presentes na biblioteca *statistics* (função *mean* para a média e função *median* para a mediana), passando o *array* dos elementos da coluna como argumento de entrada.

As funções *calculateMax()* e *calculateMin()*, calculam o máximo e o mínimo da coluna especificada, e utilizam a função *max(array)* para o calculo do valor máximo e *min(array)* para o calculo do valor mínimo. Ambas estas funções recebem um *array* contendo os valores da coluna.

A função *quantityOfValues()* é uma função muito simples, que apenas retorna o número de amostras do ficheiro ou seja, o número de linhas. Para tal esta função recorre ao atributo *fileNumLines* presente na instância do objeto *File* criada. A função *numOfLinesWithNullValues()* é uma função similar, mas que percorre as linhas do ficheiro procurando por células com o caracter ‘*’, que tal como foi explicado acima são células sem valor. Esta função retorna o número de linhas que não contêm valores.

Para evitar que o *dataset* seja classificado com algumas células sem valor, tivemos a necessidade de criar uma função que os substitua. O método *replaceNullValues()* é responsável por essa tarefa. Recebendo da interface do utilizador o índice da coluna e o valor que irá substituir as células sem valor dessa coluna, a função percorre os valores da coluna e sempre que encontrar o caracter ‘*’, altera-o pelo valor especificado.

Pode também ser necessário substituir um determinado valor numa coluna, por ser por exemplo, um *outlier*. Assim sendo, criamos a função *replaceCertain()*. Esta função recebe como parâmetros o índice da coluna, o novo valor a ser escrito e o valor que será substituído, percorre todos os valores da coluna e sempre que encontrar o *targetValue* substitui pelo novo valor.

A função *checkIfNotNumeric()* é uma função auxiliar e recebe o índice de uma coluna e verifica se essa coluna é ou não numérica, retornando *False* se for numérica e *True* se for nominal.

Passando para as funções de *encoding*, temos a função *ordinalEncoding()*, que recebe como parâmetro a diretoria do ficheiro de dados e aplica a função *OrdinalEncoder()*, presente na biblioteca *sklearn.preprocessing*. Esta função aplicada num *dataset* da forma como o fizemos, sem parâmetros, converte valores nominais em valores numéricos por ordem alfabética, ou seja, no caso de termos uma coluna com os valores B, A, C, A, após aplicar o ordinal *encoding* a coluna passaria a ter os valores 1, 0, 2, 0.

Relativamente à normalização dos valores, utilizamos um tipo de normalização chamado *Z-Score*, explicado no tópico 2.3.2. Esta função recebe como parâmetros a diretoria do ficheiro de dados e a coluna target, ou seja, a coluna objetivo da predição do algoritmo, para evitar que a normalização seja aplicada nessa coluna. As normalizações são aplicadas em todos os valores exceto nos valores da coluna target e nos valores nominais. O método *z_score()*, é o responsável por este papel.

5.3.Sistema de Logs

Tal como todas as aplicações, o NT Classifier tem também um sistema para registar tudo o que foi feito na aplicação. Este registo faz-se usando ficheiros de *logs* segmentados, ou seja, cada projeto criado tem o seu próprio ficheiro de *logs*, o que significa que tudo o que esteja presente naquele ficheiro foi feito no projeto ao qual pertence. Esta implementação encontra-se no ficheiro *LogFile.py*.

O código presente neste ficheiro é breve e bastante simples. Aquando da criação do projeto é criada uma instância do objeto, contendo como argumento a diretoria do projeto. O ficheiro de *logs* é criado na raiz da diretoria do projeto.

```
class LogFile:

    def __init__(self, projectPath):
        # path to save the file with the project configurations
        self.filePath = f"{projectPath}\\log.txt"
        # reference to the opened file
        self.fileReference = None
        # number of lines in the file
        self.numOfLines = 0
```

Figura 25 - Construtor da classe LogFile.py

A única função presente nesta classe, *logActions()*, trata de escrever os registos das atividades no ficheiro de *logs* que tem o nome *log.txt*. Esta função recebe como parâmetro de entrada a mensagem a ser escrita, que deve conter a ação realizada. A esta mensagem é adicionado o momento em que a ação foi realizada no seguinte formato 2021-08-21 16:56:16,742. Para nos auxiliar no desenvolvimento desta função recorremos à biblioteca *logging*.

```
# write in the log file
self.log.logActions(f"File: {self.filePath} opened")
self.log.logActions(f"Project data file {self.copiedFilePath} created and opened")
```

Figura 26 - Exemplo de uma chamada a função *logActions()*

5.4. Sistema de Gestão dos Dados e Projetos da Aplicação

Para gerir os dados da aplicação como já foi referido neste relatório, decidimos utilizar ficheiros, mais precisamente ficheiros no formato *.txt*. O ficheiro de dados é guardado conforme vai sendo atualizado, contudo, precisamos de saber o que foi feito na aplicação para que o projeto, quando for carregado, seja reaberto tal como foi fechado. Esses passos que permitem guardar o que foi feito na aplicação ficam guardados em forma de “*keywords*” num ficheiro com o nome no formato *{Nome_do_projeto}syslog.txt*. A forma como esta implementação foi pensada foi a seguinte: cada linha do ficheiro corresponde a um passo na aplicação. Essa linha tem 2 ou mais componentes, separados pelo carater ‘|’. O primeiro componente da linha corresponde à *keyword* que está associada a uma determinada função, que será a função a executar quando o projeto for carregado. O segundo e restantes componentes serão os parâmetros de entrada da função correspondente à *keyword* do primeiro componente da linha.

```
OF|D:\IPL\4ano\Projeto Informatico\NTClassifier\weka_informatico\2\DataFile2.csv
STC|is_attack|9
```

Figura 27 - Exemplo do ficheiro *syslog.txt*

Na figura acima podemos ver um exemplo do que acabou de ser descrito. Tomando como exemplo a primeira linha, o primeiro componente da linha, *OF*, é a *keyword* referente à função *openFile()*, que irá abrir o ficheiro contido no segundo componente da linha, o parâmetro de entrada.

Passando ao código presente no ficheiro *ProjectLogFile.py*, ficheiro responsável por esta funcionalidade, é relativamente similar ao código explicado no tópico anterior. Quando o projeto é criado, é criada uma instância da classe *ProjectLogFile* que recebe como argumento a diretoria do projeto.

```
class ProjectLogFile:
    def __init__(self, filePath):
        # path to save the file with the project configurations
        self.filePath = filePath
        # reference to the opened file
        self.fileReference = open(self.filePath, 'a+')
```

Figura 28 - Construtor da classe ProjectLogFile

Esta classe contém depois 2 funções, uma responsável pela escrita no ficheiro *syslog.txt* à medida que vão sendo realizadas ações no projeto, e outra responsável pela leitura desse ficheiro quando se carrega um projeto anteriormente criado, para que este seja reaberto como foi deixado. Estas funções são *writeLine()* e *readLine()* respetivamente.

O ficheiro contém ainda a função *closeFile()*, para fechar o ficheiro *syslog.txt*, guardando as alterações efetuadas.

5.5. Sistema de Criação ou Carregamento de Projetos

A página inicial da aplicação, como demonstra a Figura 17, corresponde à página que o utilizador vê primeiro assim que inicia a aplicação. Nesta, o utilizador tem duas possibilidades de entre as quais, a criação de um novo projeto (botão do lado esquerdo) ou o carregamento de um projeto criado anteriormente (botão do lado direito).

A classe *Main* presente no ficheiro *main.py*, contém todos os atributos e métodos/funções responsáveis pela criação e apresentação da página inicial. Dentre esses, encontramos o método *createProject()*, responsável pela criação de um novo projeto e o método *loadProject()*, responsável pelo carregamento de um projeto. Ambos os métodos criam uma nova janela composta por dois separadores diferentes, em que o primeiro, *Data Preparation*, conta com todas as funcionalidades relativas à limpeza e tratamento de dados e o segundo, *Classify*, composto por todas as funcionalidades relacionadas com a divisão de dados, escolha de algoritmos, classificação de dados e geração e visualização de resultados.

Para tal, é necessário instanciar duas classes. Primeiro a classe *Classify* que receberá como parâmetro o *frame* correspondente ao separador *Classify* e depois a classe GUI, que receberá como parâmetros o *frame* correspondente ao separador *Data Preparation* e a instância anterior, *Classify*. Estas duas classes irão ser abordadas mais afincadamente no próximo tópico.

A grande diferença entre os dois principais métodos da janela inicial deve-se ao facto de, no caso da criação de um novo projeto, ser invocada a função *chooseNameOfProject()* presente na classe GUI, que recebe como parâmetro a classe GUI instanciada anteriormente. Esta função irá apresentar uma caixa de texto onde o utilizador terá de escolher o nome que pretende dar ao projeto.

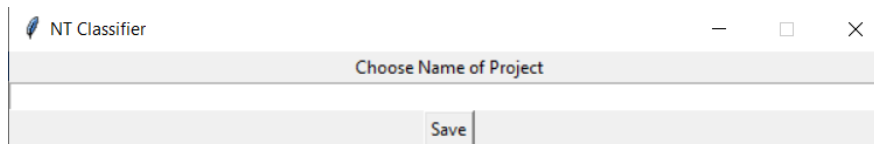


Figura 29 - Escolher nome do projeto

Após a escolha do nome do projeto, o utilizador irá clicar no botão *Save*, presente na figura acima, e a função *openFile()* será iniciada. Este terá de escolher o ficheiro de dados (.csv) que pretende classificar e posteriormente, o sistema de ficheiros será criado com uma pasta *Results*, onde ficarão todos os ficheiros (.txt) relativos aos resultados obtidos da classificação dos dados com os diferentes algoritmos; uma pasta *Models*, com os ficheiros (.pkl) relativos aos modelos obtidos da classificação dos dados; uma pasta *Plots*, com os ficheiros (.png) relativos aos gráficos obtidos também da classificação dos dados; um ficheiro *{NomeDoProjeto}syslog.txt*, composto pelas *keywords* respetivas a algumas ações feitas pelo utilizador, como por exemplo, a escolha do ficheiro de dados ou da coluna objetivo, *keywords* essas que serão necessárias para carregar o projeto; um ficheiro *Datafile{NomeDoProjeto}.csv*, que terá uma cópia do ficheiro de dados selecionado anteriormente e que irá ser tratado/limpo pelo utilizador para que estes possam depois ser classificados; e um ficheiro *log.txt*, composto pelas ações que o utilizador irá realizar ao longo do projeto, em formato textual e de fácil leitura e compreensão, utilizado apenas para consultar as mesmas.

Após isso, os atributos da classe *GUI* relativos ao caminho do projeto, ao ficheiro de dados e aos ficheiros de *logs*, que tinham sido criados a “None”, serão atualizados. Também o atributo *table*, criado a “None”, passará a ser uma instância da classe *Table* que recebe como parâmetro o ficheiro de dados e também será apresentado o nome das colunas dos dados na mesma lista. Os restantes botões presentes na aplicação serão ativados e o caminho completo do ficheiro de dados selecionado será apresentado numa caixa de texto não editável.

Será escrito no ficheiro que contempla as *keywords*, *{NomeDoProjeto}syslog.txt*, uma linha correspondente ao ficheiro de dados selecionado e, no ficheiro *log.txt*, a informação acerca da criação de um novo projeto e o ficheiro escolhido.

Para finalizar, vai ser feita uma chamada à função *addFilePathProjectFolderPath()* da classe *Classify*, que foi recebida como parâmetro quando a classe *GUI* foi instanciada, e vai ser passado como parâmetro o caminho completo do ficheiro de dados (.csv) e o caminho completo da pasta do projeto criado, de modo a que seja possível aceder aos ficheiros necessários do projeto no separador *Classify*.

Já no caso do carregamento de um projeto, será invocada a função *loadProject()* presente na classe *GUI* e que recebe também como parâmetro a classe *GUI* instanciada anteriormente. Esta função irá abrir um explorador de ficheiros e o utilizador terá de selecionar o ficheiro *{NomeDoProjeto}syslog.txt* do projeto que pretende carregar.

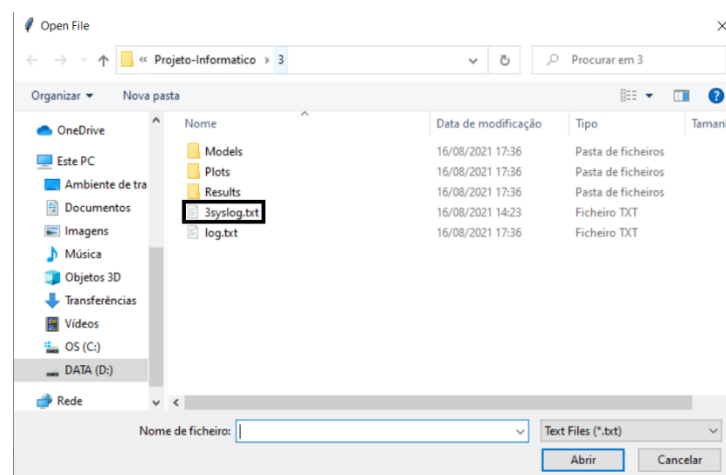


Figura 30 - Seleção do ficheiro para carregamento de projeto

A função irá ler todas as *keywords* presentes nesse ficheiro e executar o código correspondente a cada uma. Para conseguir carregar um projeto, a primeira *keyword* presente

no ficheiro terá de conter o caminho completo para o ficheiro de dados (.csv) relativo ao projeto a carregar, caso contrário não será possível carregar o mesmo. Após isso, os atributos da classe *GUI* relativos ao caminho do projeto, ao ficheiro de dados e aos ficheiros de *logs* serão atualizados.

Tal como na criação de um novo projeto, o atributo *table* que tinha sido criado a *None*, passará a ser uma instância da classe *Table* que recebe como parâmetro o ficheiro de dados e também será apresentado o nome das colunas dos dados na mesma lista. Também os restantes botões presentes na aplicação serão ativados, tal como o caminho completo do ficheiro de dados será apresentado na caixa de texto não editável.

Por fim, também nesta função será feita uma chamada à função *addFilePathProjectFolderPath()* da classe *Classify*, que foi recebida como parâmetro quando a classe *GUI* foi instanciada, e vai ser passado como parâmetro o caminho completo do ficheiro de dados (.csv) e o caminho completo da pasta do projeto criado, de modo a que seja possível aceder aos ficheiros necessários do projeto no separador *Classify* e, para além disso, será feita uma chamada à função *setTargetColumn()*, também esta da classe *Classify*, e vai ser passado como parâmetro, caso tenha sido lido do ficheiro de *logs* que foi carregado, a coluna objetivo e o índice desta, para que seja possível posteriormente utilizar essa coluna na classificação dos dados.

5.6. Sistema de Visualização de Dados e Estatísticas

Para que os utilizadores possam ver em tempo real os dados que estão a limpar/tratar, foi criada a função *showDataInTable()* que é despoletada quando estes pressionam o botão “View Data In Table”, do *frame* “Data Table”, presente na figura abaixo.

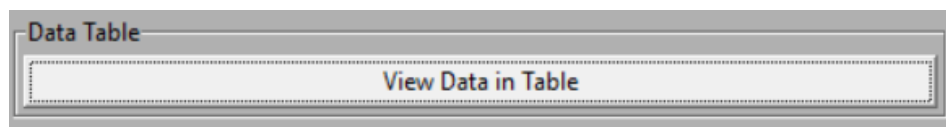


Figura 31 - Visualizar dados numa tabela

A função acima cria uma nova janela com uma tabela composta por todos os dados presentes no ficheiro de dados selecionado pelo utilizador e associado ao projeto atual. Nessa mesma tabela é possível visualizar que as colunas apresentam, como fundo, uma cor cinzenta mais escura e as linhas associadas apresentam cor branca, de forma que seja fácil distinguir ambas. A figura 29 mostra parte de um ficheiro de dados presente na tabela.

num_pkts	std_jit	mean_pkt_len	num_bytes	min_pkt_len	max_pkt_len	is_attack
-1.307750331001981	0.0	83.0	-0.06892758003934337	1.823701833120097	0.5559780477650073	1.0
-1.307750331001981	0.0	83.0	-0.06892758003934337	1.823701833120097	0.5559780477650073	1.0
103.4005935059755	0.06085463465339165	1344.0	128.51877236111343	90.05657083345204	76.05716204903268	0.0
103.4005935059755	0.06085463465339165	1344.0	128.51877236111343	90.05657083345204	76.05716204903268	0.0
-1.3121370478684486	0.0	87.0	-0.0692436140216933	87.0	0.795461814733901	0.0
-1.307750331001981	0.0	60.0	-0.06911160316830663	0.21437808370880843	-0.8210522770581938	0.0
-1.3121370478684486	0.0	87.0	-0.0692436140216933	2.10335840504090166	0.795461814733901	0.0
2.0	0.0	67.0	-0.06905559656383954	0.704172963964418	-0.40195608706852387	0.0
1.0	0.0	71.0	-0.06930761928394139	0.9840551812533377	-0.1624725336014108	0.0
-1.307750331001981	0.0	87.0	-0.06928761763948886	67.0	-0.40195608706852387	0.0
-1.3121370478684486	0.0	67.0	-0.06928761763948886	2.10335840504090166	0.795461814733901	0.0
-1.3121370478684486	0.0	76.0	-0.06928761763948886	1.3339079528644875	0.1368818637753374	0.0
1.0	0.0	75.0	-0.06929161796837936	1.2639373985422575	0.0770108003482417	0.0
-1.307750331001981	0.0	67.0	-0.06905559656383954	0.704172963964418	-0.40195608706852387	0.0
-1.307750331001981	0.0	67.0	-0.06905559656383954	0.704172963964418	-0.40195608706852387	0.0
-1.307750331001981	0.0	67.0	-0.06905559656383954	0.704172963964418	-0.40195608706852387	0.0
-1.3121370478684486	0.0	71.0	-0.06930761928394139	0.9840551812533377	-0.1624725336014108	0.0
316.0	4.83804132579402	52.22784810126582	-0.003357021462636107	-0.34538535086903105	-1.1802775716207679	1.0
-1.2989768972690452	1.3389961161180084	60.0	-0.06863156370144598	0.21437808370880843	-0.8210522770581938	0.0
-1.2989768972690452	1.2545942843404897	60.0	-0.06863156370144598	0.21437808370880843	-0.8210522770581938	0.0
-1.3121370478684486	0.0	86.0	-0.0692436140216933	2.0336134960867867	0.7355960804629844	0.0
-1.307750331001981	0.0	102.0	-0.06877357554150417	3.153142365242466	1.6935348328798256	0.0
-1.3121370478684486	0.0	75.0	-0.06929161796837936	1.2639373985422575	0.0770108003482417	0.0
0.0696787650689098	4.83804132579402	52.22784810126582	-0.003357021462636107	-0.34538535086903105	-1.1802775716207679	1.0
-1.2989768972690452	2.3595178097814886	83.0	332.0	1.823701833120097	0.5559780477650073	0.0
-1.2989768972690452	1.2547675058319836	60.0	-0.06863156370144598	0.21437808370880843	-0.8210522770581938	0.0
-0.0854304212391182	4.432575140003395	89.75357142857143	0.030940622712124	-0.2054442422343712	4.806810771088082	1.0
-1.2989768972690452	2.359507975314286	83.0	-0.068633344351947	1.823701833120097	0.5559780477650073	0.0
-1.2989768972690452	2.359205529729083	67.0	-0.06851955480251182	0.704172963964418	-0.40195608706852387	1.0
-1.2989768972690452	2.3591620448388415	67.0	-0.06851955480251182	0.704172963964418	-0.40195608706852387	1.0
-1.3121370478684486	0.0	76.0	-0.06851955480251182	1.3339079528644875	0.1368818637753374	0.0
-1.2989768972690452	2.3595446713700503	83.0	-0.0682633344351947	1.823701833120097	0.5559780477650073	0.0
0.1530268553179732	11.388480769597046	54.4	18224.0	52.0	-0.8210522770581938	1.0
-1.2989768972690452	2.359507975314286	83.0	-0.0682633344351947	1.823701833120097	0.5559780477650073	0.0
0.174599988413634	9.47895260687254	54.4	0.00439644052362055	-0.34538535086903105	-0.8210522770581938	1.0
0.3005614736581704	9.530481086391052	22.088	22.088	-0.34538535086903105	0.855324646904058	1.0
-1.2989768972690452	1.2546361516589498	60.0	-0.06863156370144598	0.21437808370880843	-0.8210522770581938	0.0

Figura 32 - Exemplo de tabela com dados

Ainda no que toca à visualização de dados, foi criado um *frame*, “*Selected Attributes*”, composto por uma caixa de texto e dois botões. Na figura 30, o botão do lado esquerdo, “*View Data Of Selected Column*”, despoleta a função *printColumnData()*, em que esta apresenta na caixa de texto acima do botão, o tipo de dados da coluna selecionada da lista de colunas (lista que será abordada num subtópico seguinte) e informação sobre a presença de linhas sem valores.

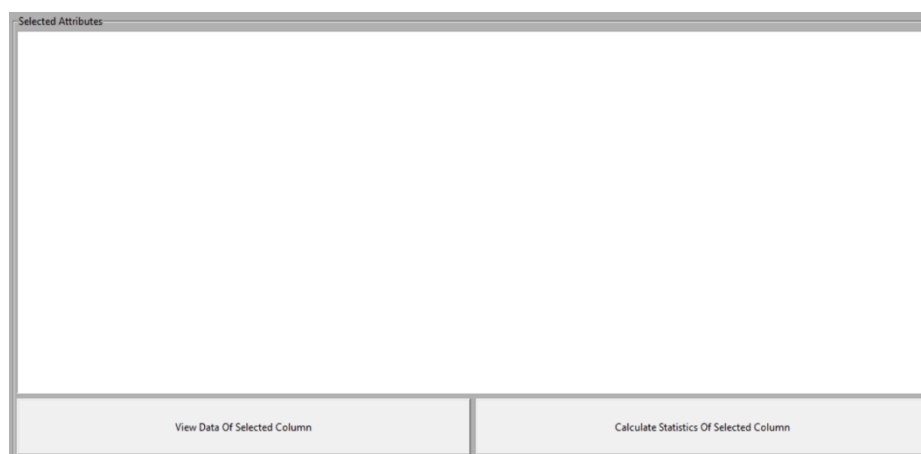


Figura 33 - Visualizar Dados e Estatísticas

No que diz respeito à visualização de estatísticas, o botão do lado direito da figura acima, “*Calculate Statistics Of Selected Column*”, despoleta a função *calculateStatistics()*. Esta função calcula e apresenta na caixa de texto informações como a média e a mediana dos dados da coluna selecionada, o valor máximo e mínimo da mesma e também a

quantidade de linhas e a quantidade de linhas sem valores. Para tal, foi necessário criar uma função para cada informação.

No caso do cálculo da média e da mediana, é feita a chamada a duas funções presentes na classe *Table*, *calculateMean()* e *calculateMedian()*, e ambas recebem como parâmetro o índice da coluna selecionada. A partir do índice, vamos buscar os valores todos da coluna e, caso estes sejam todos numéricos, é calculada a média e a mediana. Caso contrário, é apresentada uma mensagem de erro.

Já no caso do valor máximo e mínimo, também é feita a chamada a duas funções da classe *Table*, *calculateMax()* e *calculateMin()*, recebendo como parâmetro o índice da coluna selecionada mas não será apresentada uma mensagem de erro caso todos os valores não sejam numéricos. Essa mensagem apenas será mostrada caso todos os valores forem não numéricos, uma vez que, só assim não existirá um valor máximo e mínimo.

As últimas estatísticas apresentadas referem-se à quantidade de linhas e à quantidade de linhas sem valor. Para tal foram criados dois métodos na classe *Table*. Para a primeira estatística foi criado o método *quantityOfValues()* que lê o numero de linhas presentes no ficheiro e para a segunda, o método *numOfLinesWithNullValues()*, que lê todas as linhas do ficheiro, analisa-as verificando se têm algum valor, seja este numérico ou não, e mostra a quantidade das que não apresentem valores. Ao contrário das estatísticas explicadas anteriormente, estas últimas são sempre apresentadas ao utilizador.

Para finalizar, as estatísticas calculadas são guardadas no ficheiro *log.txt*, em formato textual, para que o utilizador as possa consultar sempre que pretender sem ter que abrir a aplicação.

5.7. Sistema de Tratamento de Colunas

Após a escolha do ficheiro de dados, este poderá não ter os dados todos corretos e, para tal, será necessário fazer o seu tratamento/limpeza. Como consequência, as ações de tratamento/limpeza foram divididas em duas partes distintas, em que a primeira parte refere-se ao tratamento das colunas e a segunda parte refere-se ao tratamento das linhas associadas. Neste tópico iremos abordar apenas a primeira parte e abordaremos a segunda parte no tópico seguinte.

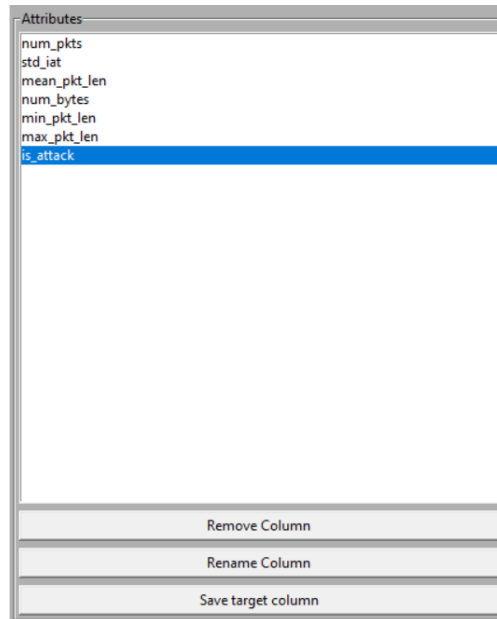


Figura 34 - Zona reservada ao tratamento de colunas

Como podemos ver na figura acima, foi criado um *frame* composto por uma lista e 3 botões. A lista será preenchida com o nome das colunas do ficheiro seleccionado assim que este for seleccionado, através da função *printColumns()* da classe *GUI*, e os botões realizarão ações consoante a(s) coluna(s) escolhidas da lista.

Seleccionadas uma ou mais colunas, será possível, através do botão “*Remove Column*”, remover estas do ficheiro de dados e todas as linhas associadas. Para que isso aconteça, foi necessário criar a função *removeMultipleColumns()*, que guarda numa lista os índices das colunas seleccionadas e posteriormente, percorre a lista, chamando em cada iteração o método *deleteColumn()* da classe *Table* e passando como parâmetro o índice da coluna atual. Será nesta última função que se procederá à eliminação da coluna e das respetivas linhas do ficheiro. Sempre que uma coluna for eliminada, a lista de colunas será atualizada através da função acima descrita, *printColumns()*. No final, será apresentada uma mensagem com informação sobre as colunas eliminadas e será escrito a mesma no ficheiro de informações que o utilizador pode consultar, *log.txt*.

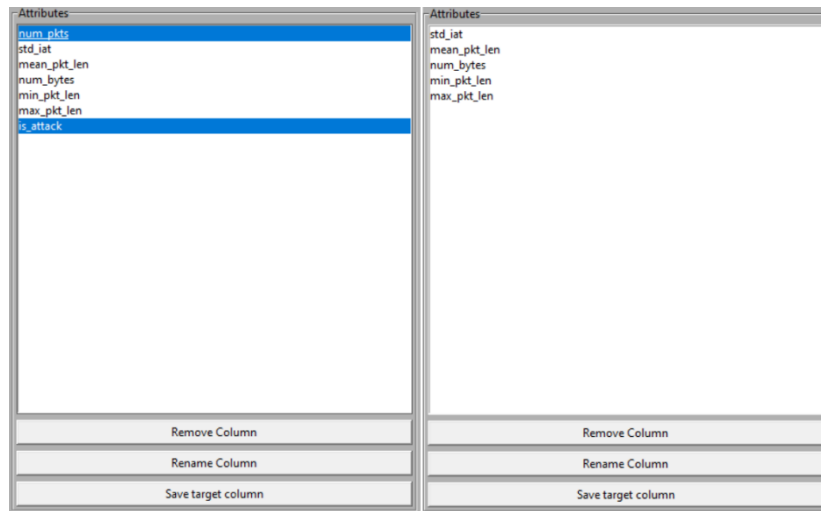


Figura 35 - Exemplo do antes e depois da remoção das colunas selecionadas

As outras duas possibilidades de tratamento de colunas são renomear e gravar coluna objetivo, através dos botões “*Rename Column*” e “*Save Target Column*”, respetivamente. Ao contrário do que acontece com a eliminação de colunas, nestes casos apenas é possível tratar uma coluna de cada vez.

No que toca a renomear colunas, foram desenvolvidas duas funções, *renameColumns()* e *applyRenaming()*, em que a primeira cria uma janela com uma lista com todas as colunas do ficheiro e uma caixa de texto no qual o utilizador terá de escrever o nome que pretende dar à coluna escolhida da lista. Apresenta também um botão “*Rename Column*”, que irá despoletar a função *applyRenaming()*. Esta receberá como parâmetro a janela anterior, com a coluna selecionada e o novo nome. Caso o utilizador tenha escrito um novo nome, será aplicada uma outra função, *setValue()* da classe *Table*, enviando o índice da linha (será 0 pois trata-se da linha onde estão presentes os nomes das colunas todas), o índice da coluna anterior e o novo nome, e será então realizada a alteração no ficheiro de dados. Posteriormente será mostrada uma mensagem de sucesso, a ação será escrita no ficheiro *log.txt* e a função *printColumns()* será chamada para atualizar a lista de colunas. Caso contrário, será apresentada uma mensagem de erro.

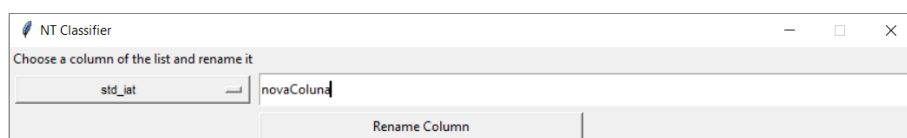


Figura 36 - Exemplo de renomeação da coluna “std_iat”

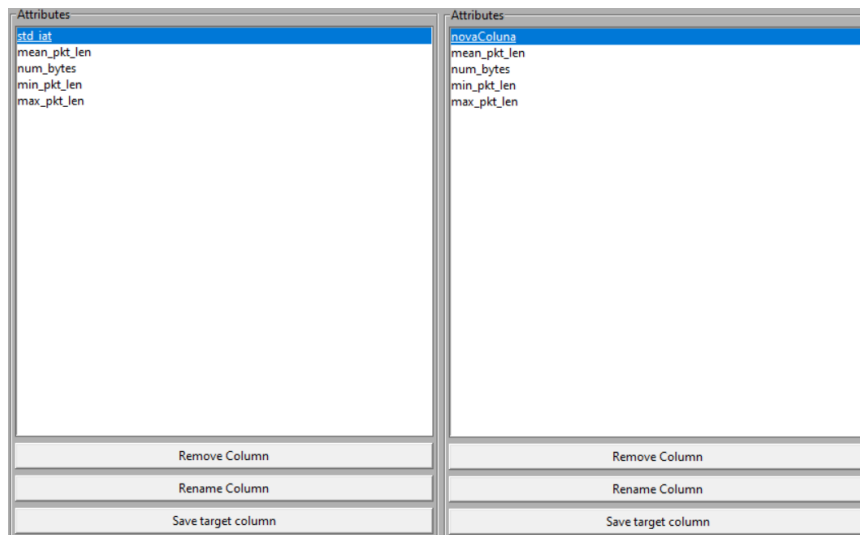


Figura 37 - Exemplo do antes e depois da renomeação da coluna "std_iat"

A última ação relativa a colunas está relacionada com a gravação da coluna objetivo através do último botão presente na figura 20. Selecionada a coluna pretendida e pressionado o botão “*Save Target Column*”, será chamada a função *saveTargetColumn()* que guarda a coluna e o índice desta. Estes dados serão passados como parâmetros através de uma outra função, *setTargetColumn()* presente na classe *Classify*, para que se possa depois realizar a classificação dos dados com base nos dados da coluna selecionada. Tal como nas outras funções anteriores, também nesta, *saveTargetColumn()*, é apresentada uma mensagem de sucesso que depois será escrita no ficheiro *log.txt* e, posteriormente será escrito no ficheiro *{NomeDoProjeto}syslog.txt*, uma linha composta pela *keyword* “STC|{NomeDaColuna}|{IndiceDaColuna}” que será utilizada para carregar a coluna objetivo quando o utilizador carregar o projeto.

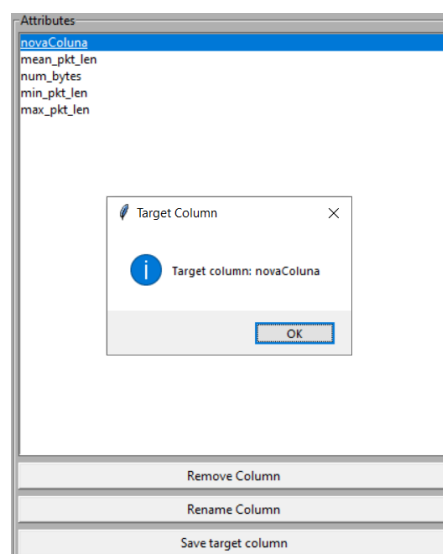


Figura 38 - Exemplo de gravação da coluna objetivo

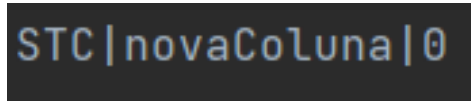


Figura 39 - Exemplo de keyword gerada após a gravação da coluna objetivo

5.8. Sistema de Seleção e Adição de Algoritmos

Como referido anteriormente, a aplicação “Net Traffic Classifier” foi desenhada e desenvolvida para classificar conjuntos de dados obtidos através do tráfego de rede. Para conseguirmos classificar esses dados, necessitamos de recorrer a algoritmos de *machine learning*. Tivemos então de recorrer a uma biblioteca fornecida pela linguagem *Python*, *sklearn*, para escolhermos os algoritmos que iam ser adicionados à aplicação. Apesar de alguns algoritmos já terem sido adicionados, o utilizador poderá também adicionar outros, desde que sigam algumas regras que serão abordadas de seguida.

5.8.1. Importação e Seleção de Algoritmos

Após os dados estarem devidamente tratados, é necessário trocar de separador para o “*Classify*” para podermos então decidir como queremos classificar os dados. Como consequência, foi criada uma lista composta por todos os algoritmos presentes na pasta “*Algorithms*”, da qual vamos escolher os algoritmos a usar e um botão “*Choose Algorithms*” para concluir a escolha.

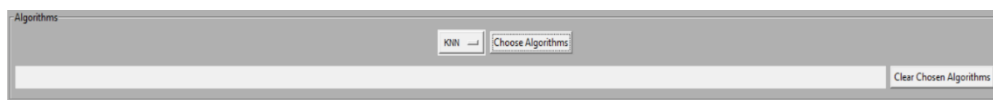


Figura 40 - Escolher algoritmo a utilizar

Uma vez que os utilizadores poderão adicionar os seus algoritmos à aplicação, ficando também estes alocados na pasta “*Algorithms*”, foi necessário criar uma função que percorresse essa pasta e importa-se o ficheiro, a classe e a função de cada algoritmo. Essa função, *importAlgorithmsInstanceDynamically()* da classe *Classify*, verifica para cada ficheiro presente na pasta acima descrita, se tem a extensão *.py*. Caso isso se verifique, guarda o nome do ficheiro sem a extensão e acede à função *import_module* da biblioteca *importlib* (biblioteca não desenvolvida por nós) passando como parâmetros a nome da pasta e o nome do ficheiro atual sem extensão, garantindo assim que o ficheiro será importado. Posteriormente serão adicionados a dois dicionários de dados criados, assim que a aplicação se inicia, uma chave, que corresponde ao nome do algoritmo atual e um valor, que

corresponde à classe presente no ficheiro desse algoritmo, no caso do primeiro dicionário de dados e, uma chave, que corresponde ao nome do algoritmo atual e um valor, que corresponde à função que será usada para classificar os dados presente na classe do ficheiro, no caso do segundo dicionário de dados. Esta função é chamada assim que a aplicação inicia, garantindo que o utilizador consiga logo ver os algoritmos presentes nesta.

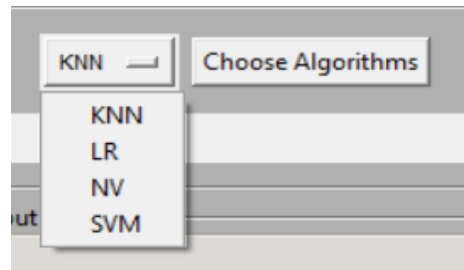


Figura 41 - Lista de algoritmos da aplicação

Para que o utilizador complete a seleção dos algoritmos, terá de pressionar o botão “Choose Algorithm”. Este irá chamar o método *chosenAlgorithms()*, que verificará se o algoritmo existe, e caso exista, adiciona o seu nome a uma lista de algoritmos selecionados, procurando-o posteriormente no primeiro dicionário de dados abordado anteriormente para verificar se este necessita de parâmetros especiais. Se necessitar, será criada uma nova janela com os vários parâmetros e com uma caixa de texto para cada um. Após a especificação dos valores para os parâmetros especiais, será feita a chamada à função *instantiateAlgorithm()* que receberá o valor correspondente à chave do algoritmo selecionado e instancia assim a classe do algoritmo. Um novo dicionário de dados, correspondente aos algoritmos instanciados, será criado, tendo como chave, o nome do algoritmo e como valor, a instância deste.

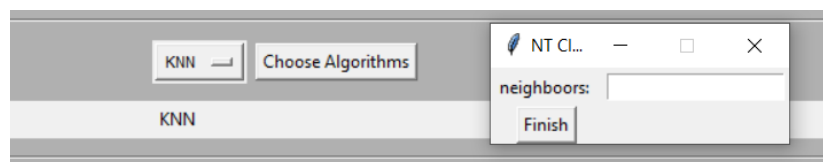


Figura 42 - Exemplo de algoritmo que necessita de parâmetros especiais

Caso se verifique que o algoritmo selecionado não necessita de outros parâmetros, será logo criada uma instância deste, adicionando mais uma entrada no dicionário de dados dos algoritmos instanciados.

Após a escolha e instanciação de cada algoritmo, o nome deste será adicionado à caixa de texto presente na figura seguinte.

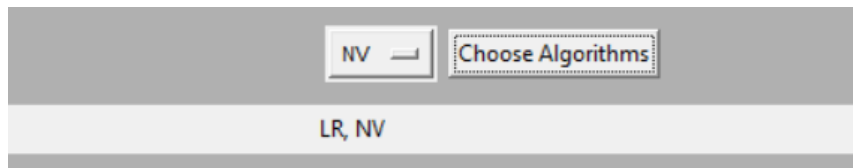


Figura 43 - Exemplo de dois algoritmos selecionados

O utilizador poderá limpar a lista de algoritmos selecionados, utilizando o botão “*Clear Chosen Algorithms*”. Este invocará a função *cleanChosenAlgorithms()*, que irá também limpar a caixa de texto acima.

5.8.2. Adição de Algoritmos

Como dito anteriormente, a aplicação fornece aos utilizadores a possibilidade de eles puderem adicionar e utilizar o algoritmo que eles pretendem. Para isso é necessário que o ficheiro com o algoritmo que se pretende adicionar, siga umas certas instruções:

1. O ficheiro a adicionar terá de ter a extensão .py.
2. O nome do ficheiro, o nome da classe e o nome da função do algoritmo têm de ser iguais.
3. O código da classe tem de ser semelhante ao modelo apresentado.

Estas instruções são apresentadas assim que o utilizador clicar no botão “*Add New Algorithm*”. Aquando do clique, é iniciada a função *addAlgorithms()* da classe *Classify*, em que, num primeiro instante apresenta as instruções e, posteriormente apresenta uma janela composta por uma caixa de texto e um botão, “*Add Algorithm*”. Esta, apresenta o código presente num dos ficheiros dos algoritmos da aplicação e o botão será utilizado para o utilizador escolher o ficheiro, que contém o algoritmo, que pretende adicionar. A função *add()* irá verificar se já existe algum ficheiro com o mesmo nome que o ficheiro escolhido. Caso não exista, criará uma cópia desse ficheiro na pasta *Algorithms* e irá realizar os mesmos passos da função *importAlgorithmsInstanceDynamically()* para importar o ficheiro e a classe, apesar de não chamar a função.

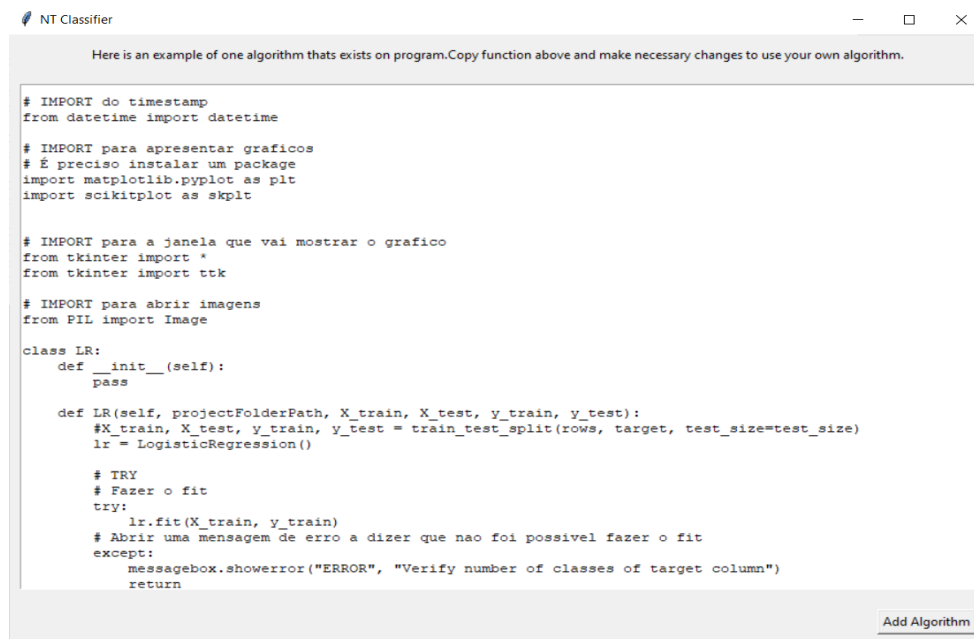


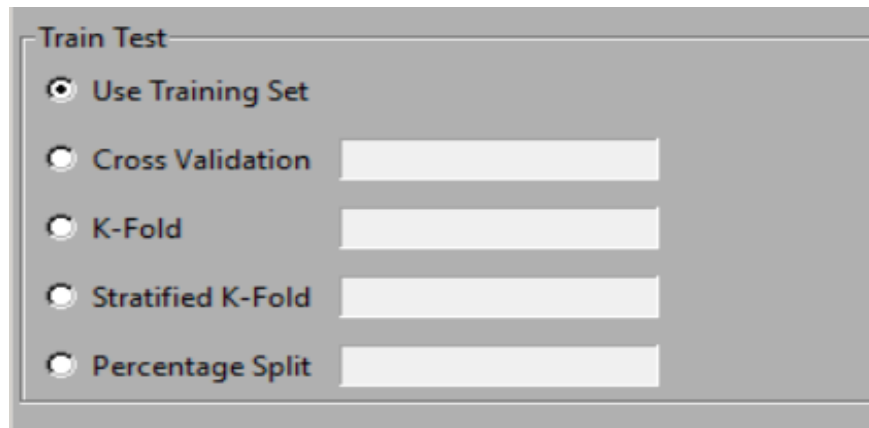
Figura 44 - Exemplo de um algoritmo presente na aplicação

Após a importação ter sido concluída, será chamado o método `rewriteAlgorithmsListBox()`, que percorrerá a pasta `Algorithms` e mostrará na lista da figura 27, todos os algoritmos existentes na aplicação. Para além disso, irá escrever no ficheiro `log.txt` (ficheiro para consulta), informação sobre a adição de uma nova coluna.

5.9. Sistema de Divisão de Dados

Para que seja possível classificar o conjunto de dados selecionado, será necessário dividi-lo em duas partes: dados de treino e dados de teste. Os dados de treino são dados que vão ser apresentados aos algoritmos de *machine learning*, para que estes construam um modelo, e os dados de teste são dados que irão testar o modelo, simulando previsões reais que este realizará. Normalmente, os dados de treino costumam representar uma maior percentagem do conjunto de dados.

De modo que o utilizador possa escolher qual o melhor método de divisão de dados, foram adicionados à aplicação, cinco métodos diferentes.



The image shows a 'Train Test' dialog box with a title bar. Inside, there are five radio buttons for selecting a data splitting method. The first radio button, 'Use Training Set', is selected. To the right of the other four radio buttons are empty text input fields. The methods are: 'Use Training Set', 'Cross Validation', 'K-Fold', 'Stratified K-Fold', and 'Percentage Split'.

Figura 45 - Diferentes métodos de divisão de dados

Como podemos observar na figura acima, o primeiro método, “*Use Training Set*”, utiliza todos os dados para treino e todos os dados para teste. Os restantes métodos apresentam uma caixa de texto, que será ativada conforme o método escolhido, utilizando a função *activateEntry()*, em que o utilizador poderá decidir como pretende dividir os dados.

No caso dos métodos “*Cross Validation*” e “*Percentage Split*”, os valores que o utilizador pode inserir nas respetivas caixas de texto, terão de ser valores percentuais, ou seja, compreendidos entre 0.0 e 1.0, e serão valores relativos à percentagem de dados que irão ser usados como dados de teste.

No caso dos métodos “*K-Fold*” e “*Stratified K-Fold*”, os valores inseridos terão de ser valores inteiros.

Os vários métodos de divisão de dados serão sempre os mesmos e o utilizador não poderá adicionar outros. Validações acerca dos valores inseridos só serão efetuadas quando o utilizar iniciar a classificação.

5.10. Sistema de Classificação dos Dados

Concluído o tratamento de dados, a escolha do algoritmo e feita a divisão dos dados em treino e teste, o utilizador poderá iniciar a classificação.

Foram então adicionados à aplicação dois botões, uma lista e uma área para os resultados.

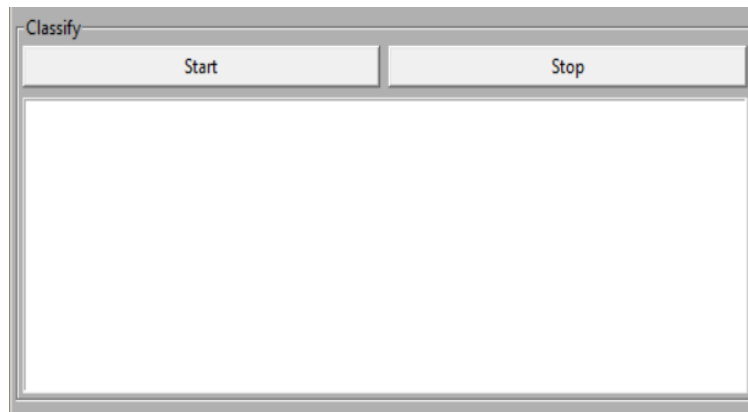


Figura 46 - Zona para iniciar/parar a classificação

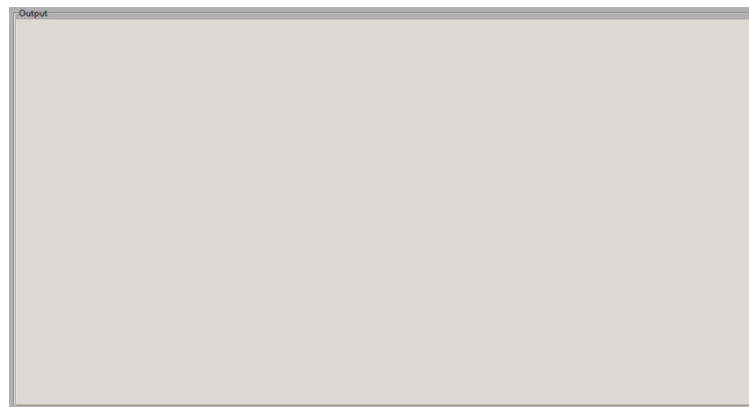


Figura 47 - Área reservada aos resultados

Na Figura 46, o botão “*Start*” iniciará a classificação, enquanto o botão “*Stop*” parará essa mesma classificação. A área a branco irá conter a lista de todas as classificações realizadas no formato “*{dia-mês-ano} {hora:minuto:segundo} – {NomeDoAlgoritmoSelecionado}*”.

As métricas e resultados obtidos da classificação irão ser escritos em separadores diferentes consoante os vários algoritmos escolhidos para classificar os dados e estes serão apresentados na área presente na Figura 47.

5.10.1. Iniciar Classificação

Assim que o utilizador pressionar o botão “*Start*”, irá ser despoletada a função *startWithThread()*, que irá criar uma nova atividade da aplicação para classificar os dados, garantindo que o programa principal possa continuar a ser utilizado sem ser necessário esperar que a classificação termine. Irá também criar uma variável de controlo para saber se a atividade está ativa ou não.

A nova atividade irá correr a função *start()*, que verificará se o utilizador selecionou algum algoritmo. Se não tiver sido especificado nenhum algoritmo, é mostrada uma mensagem de erro e a atividade é terminada. Caso contrário, irá abrir o ficheiro de dados para leitura (caso este não esteja aberto) e irá percorrer a lista de algoritmos selecionados.

Para cada algoritmo da lista, uma chamada à função *readTrainTestSplit()* será efetuada. Esta função recebe como parâmetro o algoritmo atual e verificará o método de divisão de dados selecionado pelo utilizador. Consoante o método escolhido, uma função diferente será despoletada, no entanto todas elas recebem como parâmetros, o valor especificado para dividir os dados em treino e teste e o algoritmo atual.

Nesta última função, sabendo que a coluna objetivo já foi escolhida, cada método dividirá os dados em treino e teste consoante as suas diretrizes e o valor especificado, tendo por base esta. Posteriormente será criado um dicionário de dados de informações com os seguintes dados:

1. Nome do algoritmo utilizado.
2. Método de divisão de dados utilizado.
3. Valor para dividir os dados em treino e teste.
4. Nome da coluna objetivo.

Este dicionário de dados, juntamente com o nome do algoritmo e com a função correspondente ao algoritmo selecionado, serão passados como parâmetros para a função *addTabs()*. A função correspondente ao algoritmo selecionado, terá de receber como parâmetros a instância do algoritmo (criada quando este foi selecionado), o caminho completo da pasta do projeto e todos os dados de treino e teste.

Esta última, vai criar o modelo e treiná-lo com base nos dados de treino. Após o treino, irá fazer a previsão dos dados com base nos dados de teste. O modelo será guardado na pasta *Models* presente na pasta do projeto com o formato “*{NomeDoAlgoritmo}model_{dia_mes_ano_hora_minuto_segundo}.pkl*”. Posteriormente serão calculadas as métricas consoante os dados reais da coluna objetivo e os dados previstos pelo modelo, utilizando as funções específicas de cada métrica presentes na biblioteca “*sklearn*”, no módulo “*metrics*”. Será criado então um dicionário de dados com todas as métricas:

1. Accuracy Score
2. F1-Score
3. Precision
4. Recall
5. Log Loss
6. ROC AUC
7. Confusion Matrix

Posteriormente, serão gerados três gráficos que irão gravados em três ficheiros .png. O primeiro ficheiro irá conter a curva de ROC, o segundo irá conter a matriz de confusão e o terceiro, a curva de *Precision-Recall*. Os três ficheiros serão guardados na pasta *Plots* presente na pasta do projeto no formato “{NomeDoAlgoritmo}model_{dia_mes_ano_hora_minuto_segundo}_{NomeDoGráfico}.png”. No final da função, será devolvido o dicionário de dados composto por todas as métricas.

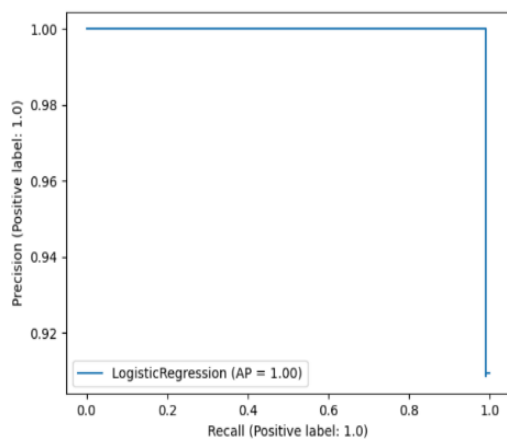


Figura 48 - Exemplo de curva de Precision-Recall

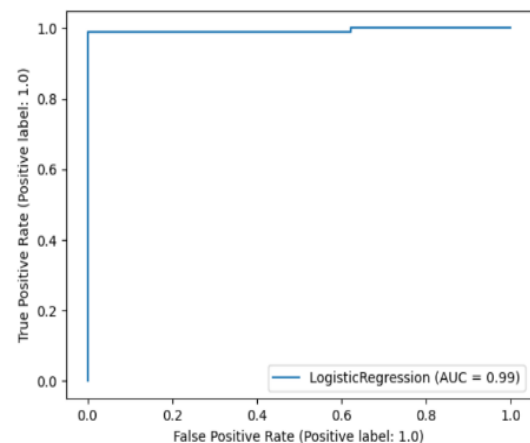


Figura 49 - Exemplo de curva de ROC

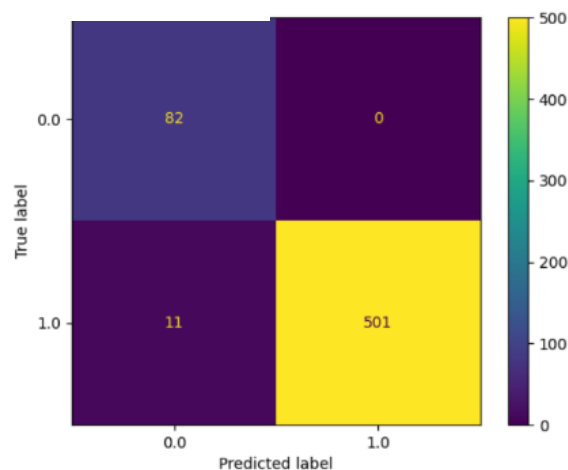
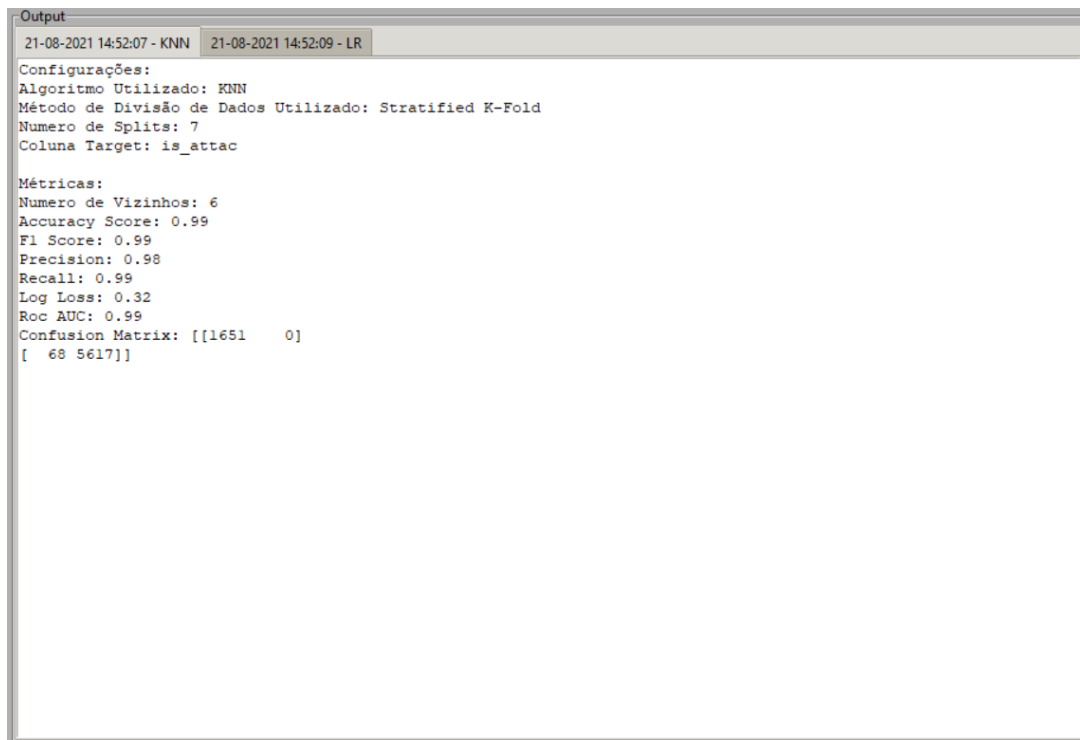


Figura 50 - Exemplo de uma matriz de confusão

Após se obter o dicionário de dados composto pelas métricas, é necessário apresentar os resultados na aplicação. A função `addTabs()` irá criar para cada algoritmo diferente, um separador com o nome do algoritmo e a data da realização da classificação, e irá escrever na zona representada na Figura 51 e num ficheiro que será criado na pasta *Results* com o formato “*Metrics_{dia_mes_ano_hora_minuto_segundo}_{NomeDoAlgoritmo}.txt*”, os dados presentes no dicionário de dados das informações e no dicionário de dados das métricas.



```

Output
21-08-2021 14:52:07 - KNN  21-08-2021 14:52:09 - LR
Configurações:
Algoritmo Utilizado: KNN
Método de Divisão de Dados Utilizado: Stratified K-Fold
Numero de Splits: 7
Coluna Target: is_attac

Métricas:
Numero de Vizinhos: 6
Accuracy Score: 0.99
F1 Score: 0.99
Precision: 0.98
Recall: 0.99
Log Loss: 0.32
Roc AUC: 0.99
Confusion Matrix: [[1651    0]
 [ 68 5617]]
  
```

Figura 51 - Exemplo da mostra de resultados após classificação dos dados

Por fim, irá escrever no ficheiro de consulta, *log.txt*, os dados presentes no dicionário de dados das informações.

A função `start()` só terminará assim que todos os algoritmos seleccionados sejam utilizados ou caso o utilizador pare a classificação. Se o utilizador parar a classificação, a função saberá devido à variável de controlo falada anteriormente.

Após realizar todas as classificações, o ficheiro de dados será fechado recorrendo à função `close()` da classe *File*.

5.10.2. Parar Classificação

A aplicação permitirá que o utilizador possa parar a classificação, no entanto, a classificação não será mesmo parada/cancelada. A função associada ao botão “*Stop*”, *stop()*, apenas alterará a variável de controlo abordada no tópico anterior. Assim que o utilizar mandar parar a classificação, este terá de esperar que a classificação atual termine. Após o término desta, as restantes classificações que o utilizador tinha pedido para serem realizadas, não acontecerão.

Quando o utilizador pedir para terminar, uma mensagem será mostrada com a informação de que a classificação foi parada, apesar de este ter de esperar que a atual termine.

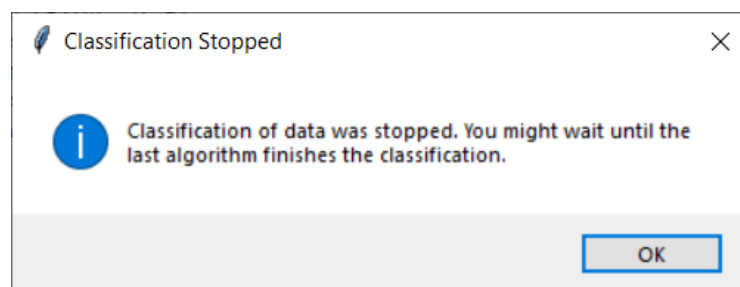


Figura 52 - Mensagem apresentada aquando do término da classificação

5.11. Sistema de Carregamento de Modelos

Para que o utilizador possa classificar o conjunto de dados sem que necessite de estar sempre a treinar um novo modelo, foi desenhado na aplicação, um *frame* dedicado à seleção de um modelo anteriormente treinado. Neste *frame* (ver figura seguinte), foram criados dois botões e uma caixa de texto de leitura. O botão do lado esquerdo, “*Load Model*”, permite que o utilizador selecione um modelo que tenha sido anteriormente treinado e o botão do lado direito, “*Classify*”, classifica o conjunto de dados, utilizando o modelo selecionado. A caixa de texto irá receber o nome do algoritmo selecionado e o caminho completo da pasta onde este se encontra.

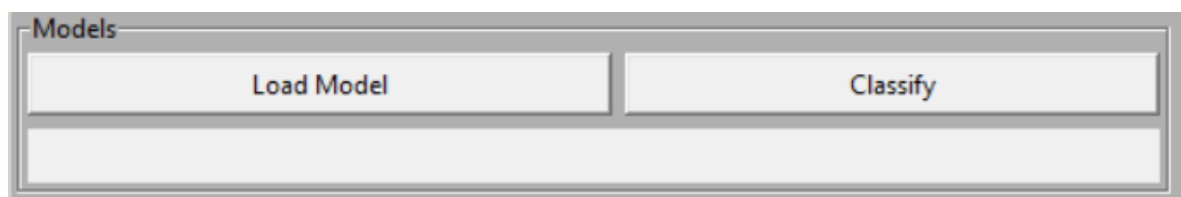


Figura 53 - *Frame* para classificação de dados com base na escolha de um modelo

A função *loadModel()* será desencadeada assim que o utilizador pressionar o botão “*Load Model*”. Esta abrirá um explorador de ficheiros, no qual o utilizador terá de seleccionar um ficheiro .pkl (extensão dos ficheiros que guardam os modelos) do modelo pretendido. O caminho completo do ficheiro será guardado numa variável que será acedida posteriormente. Após a escolha do modelo, a caixa de texto da figura acima, será atualizada e será adicionada mais uma linha ao ficheiro *log.txt*, correspondente ao carregamento do modelo selecionado.

Após a escolha do modelo, o utilizador poderá classificar os dados com base neste. Para isso o utilizador terá de seleccionar o botão “*Classify*” e a função *classifyWithExistentModel()* será iniciada. Esta irá carregar o modelo e irá prever os valores da coluna objetivo, especificada durante o tratamento dos dados. Os valores resultantes da previsão irão ser escritos num ficheiro novo na pasta *Results*, presente na pasta do projeto, com o formato “*Classified_{NomeDoModelo}_{dia_mes_ano_hora_minuto_segundo}.txt*”.

Classificados os dados com base no modelo selecionado, uma mensagem de sucesso será mostrada e a mesma será escrita no ficheiro de consulta, *log.txt*. Caso a classificação não ocorra com sucesso, uma mensagem de erro será apresentada e, também esta, será escrita no ficheiro de consulta *log.txt*.

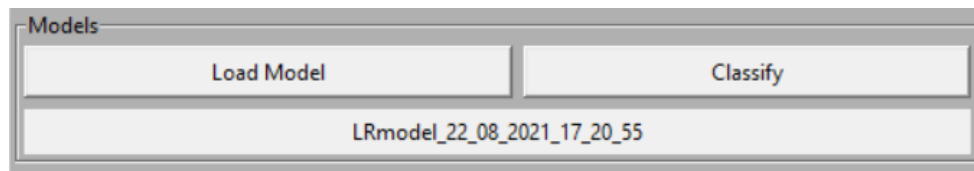


Figura 54 - Exemplo de um modelo selecionado

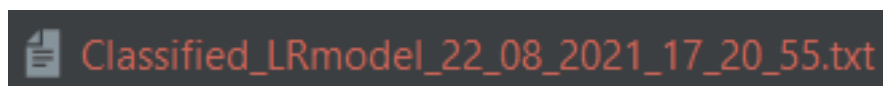


Figura 55 - Exemplo de um ficheiro com os dados classificados com base no modelo selecionado

5.12. Sistema de Carregamento de Resultados

De modo que o utilizador consiga visualizar resultados obtidos a partir de classificações feitas anteriormente, ou seja, classificações realizadas antes de carregar o projeto, a aplicação fornece uma funcionalidade que permite que este escolha as classificações feitas no passado e que pretende ver ou comparar com as realizadas no momento. Para tal, dois botões foram adicionados ao programa, “*Load Metrics*” e “*Remove Metrics*”, em que o primeiro despoleta a função *loadMetrics()*, no qual o utilizador escolherá o ficheiro de métricas que pretende

visualizar e a informação deste será apresentada num separador parecido ao da Figura 51. Também será adicionado à lista de classificações (ver Figura 46), uma linha correspondente ao ficheiro selecionado.

Por outro lado, o segundo botão, “Remove Metrics”, chama a função `deleteMetrics()` em que esta verifica as linhas selecionadas da lista de classificações realizadas, remove as da lista e posteriormente remove os separadores associados.



Figura 56 - Carregamento e remoção de métricas

5.13. Sistema de Visualização de Gráficos

Como falado anteriormente, dois gráficos e a matriz de confusão irão ser criados e guardados na pasta “Plots” presente na pasta do projeto. Para que o utilizador possa aceder a estes, foi criado uma lista que irá conter os possíveis gráficos a visualizar (*ROC Curve*, *Confusion Matrix* e *Precision Recall Curve*) e um botão “Visualize Plots”. Este irá chamar a função `showPlots()`, que irá ler a(s) classificação(ões) realizada(s), presente na lista da figura 46, e o gráfico selecionado. Posteriormente irá abrir a imagem resultante da escolha feita pelo utilizador numa aplicação externa.

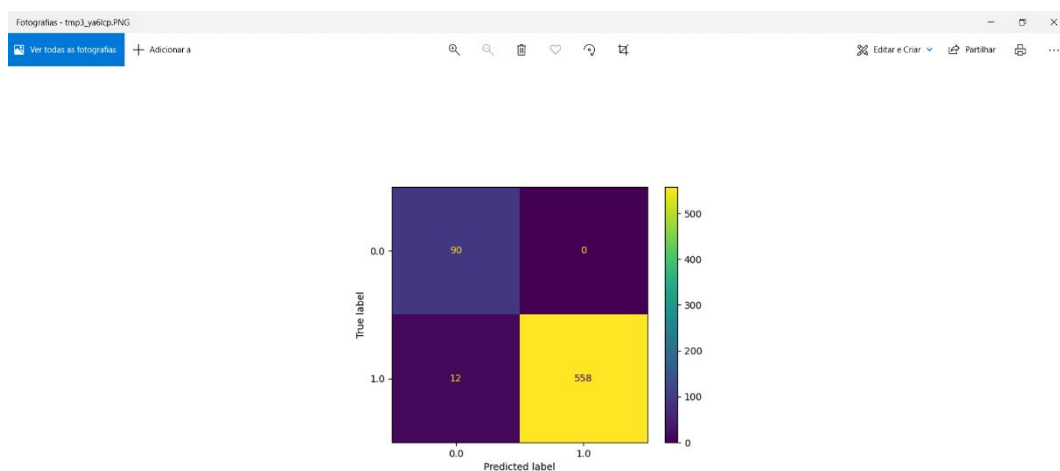


Figura 57 - Exemplo de uma matriz de confusão aberta numa aplicação externa

5.14. Sistema de Comparação de Resultados

A aplicação permitirá que o utilizador possa comparar os resultados obtidos a partir das classificações que efetuou. Terá de selecionar as classificações que pretende comparar e pressionar o botão “Compare”.

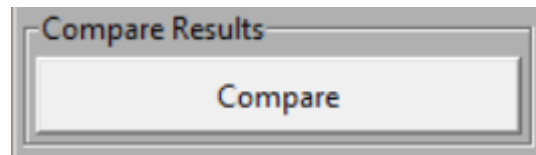


Figura 58 - Comparar resultados obtidos

A função *compareResults()* irá verificar se o utilizador selecionou alguma classificação e caso não tenha selecionado, uma mensagem de erro será apresentada. Caso contrário, uma nova janela será aberta com uma tabela. Esta tabela terá como colunas, os nomes das classificações efetuadas e, como linhas, as métricas e o valor correspondente a cada métrica de cada classificação. Para que tal seja possível, será necessário, para cada classificação, abrir o ficheiro de resultados associado presente na pasta “Results” e retirar a informação relevante, guardando-a numa lista. Esta lista é que será responsável por enviar os dados para a tabela que será mostrada ao utilizador.

 A screenshot of the NT Classifier application window. The window has a title bar with the text "NT Classifier" and standard window controls. Below the title bar, there is a table titled "Table". The table has three columns: "Métricas/Algoritmos", "23-08-2021 15:15:07 - LR", and "23-08-2021 15:15:07 - NV". The table contains six rows of data representing different metrics for two different models.

Métricas/Algoritmos	23-08-2021 15:15:07 - LR	23-08-2021 15:15:07 - NV
Accuracy-Score	0.98	0.97
F1-Score	0.96	0.93
Precision	0.93	0.90
Recall	0.99	0.98
Log-Loss	0.81	1.16
ROC-AUC	0.99	0.98

Figura 59 - Exemplo de comparação entre duas classificações

No capítulo 9, Anexos, encontra-se um manual de utilização para que o utilizador possa consultar e descobrir como realmente funciona a aplicação desenvolvida, contendo os passos desde a iniciação de um novo projeto até à visualização dos resultados das classificações.

6. Testes

Os testes fazem parte do desenvolvimento de qualquer software. Caso uma falha da aplicação ocorra nas mãos de um utilizador, o desenvolvedor não terá oportunidade de explicar por que razão aquele problema aconteceu. Então, a melhor maneira de garantir a qualidade do software desenvolvido é através de testes.

Os testes de software são divididos em três categorias: testes unitários, testes de integração e testes de sistema. Testes unitários são feitos em partes isoladas do código, ou seja, permitem testar uma função ou classe. Testes de integração são testes unitários feitos em mais do que uma parte do código, ou seja, juntam múltiplos componentes e verificam a comunicação e integração entre os mesmos. E por fim, testes de sistema que validam todo o tipo de comportamento possível dentro da aplicação, simulando a atividade do utilizador final [29].

Para este projeto não foram realizados testes com base em código, não houve automação dos mesmos, ou seja, todos os testes foram efetuados pelos desenvolvedores após a finalização de qualquer funcionalidade, verificando qual seria o comportamento da funcionalidade e alterando-a caso não estivesse de acordo com o pretendido.

6.1. Testes Unitários

Os testes unitários têm como objetivo validar pequenas partes do software desenvolvido tendo em as entradas possíveis e as saídas desejadas. Estes testes são usados para testar uma única função ou classe que recebe argumentos e devolve um determinado valor ou que efetua alguma ação cujo resultado possa ser visualizado [29].

Como dito anteriormente, nenhum dos testes foi executado com base em código, mas sim através da utilização dos desenvolvedores, ou seja, neste projeto um dos testes realizados que pode ser contabilizado como teste unitário será a alteração das linhas sem valor para valores como a média ou a mediana, uma vez que, conhecemos as entradas e sabemos quais seriam as saídas desejadas. Se após a verificação do ficheiro de dados, as linhas em questão continuassem sem valor ou tivessem valores diferentes da média ou da mediana dos valores daquela coluna, significava que a funcionalidade tinha sido mal desenvolvida e necessitava de ser modificada.

Outro exemplo de teste unitário realizado foi para testar a remoção de colunas. Neste teste, quando o utilizador tenta-se remover as colunas seleccionadas, estas teriam de ser removidas do ficheiro de dados e da lista de colunas. Caso contrário, a funcionalidade não teria sido bem desenvolvida.

No caso da visualização dos gráficos criados aquando da classificação dos dados, a funcionalidade foi concebida para que estes fossem apresentados numa janela nova, criada pela própria aplicação. No entanto, após vários testes e tentativas, não conseguimos que os gráficos fossem apresentados no local pretendido. Para tal, tivemos de optar por outra solução, que passa por apresentar os gráficos através de uma aplicação externa à aplicação desenvolvida.

6.2. Testes de Integração

Testes de integração testam um conjunto de funcionalidades presentes na aplicação. Estes, unem vários componentes e verificam a comunicação e integração entre os mesmos, garantindo que funcionam em conjunto e apresentam os resultados esperados. Estes testes são realizados após ter sido garantido o bom funcionamento individual dos componentes a unir [30].

Como teste de integração podemos considerar, por exemplo, a classificação de um conjunto de dados com base num algoritmo e num método de divisão de dados selecionado pelo utilizador. Após a classificação terminar, será esperado que os resultados sejam apresentados ao utilizador. Este teste juntará três componentes diferentes, que serão a escolha do algoritmo, a escolha do método de divisão de dados e a classificação desses dados e testará o funcionamento destes três módulos. Uma vez que se trata algoritmos de *machine learning* e de conjuntos de dados muito grandes, não temos como garantir que os resultados apresentados estão corretos, mas caso os resultados sejam apresentados ao utilizador com algum critério, assumimos que as funcionalidades foram bem desenvolvidas e, portanto, passam no teste.

Se por algum motivo, os resultados da classificação dos dados não forem apresentados ao utilizador, assumimos que os testes falharam e, como tal, as funcionalidades foram mal desenvolvidas e terão de ser modificadas.

Foi realizado outro teste de integração para testar a classificação dos dados com base num algoritmo adicionado pelo utilizador. Como tal, tivemos de realizar o teste mencionado anteriormente e um teste para verificar se o ficheiro a adicionar, é adicionado à aplicação. Posteriormente, testámos ambas as situações em conjunto. Se a classificação dos dados com utilização do novo algoritmo ocorre-se sem falhas, significava que ambas as funcionalidades estavam operacionais e que o algoritmo adicionado tinha sido bem desenvolvido pelo utilizador. Caso contrário, significaria que o algoritmo adicionado não seguia as regras especificadas por nós e, como tal, o teste falhava.

6.3. Testes de Sistema

Este último tipo de testes validam todo o tipo de comportamento possível dentro da aplicação. O objetivo destes testes é executar a aplicação sob o ponto de vista do utilizador final, testando todas as funcionalidades em busca de falhas que possam ser encontradas em relação aos objetivos originais do projeto. Estes testes são executados em condições similares àquelas que o utilizador final encontrará na manipulação da aplicação no seu dia a dia [31].

De recordar que, apesar de não terem sido executados testes baseados em código, a aplicação foi testada como um todo, desde a criação de um novo projeto até à visualização de resultados após a classificação, passando também pelo tratamento de dados.

7. Conclusão

O desenvolvimento deste projeto possibilitou aos elementos do grupo aprender novos conhecimentos em áreas pouco abordadas durante a licenciatura, como é o caso de inteligência artificial e *machine learning*. Este projeto permitiu-nos investigar e aprofundar temas que têm sido cada vez mais utilizados em áreas como a saúde, segurança ou educação, trazendo benefícios e avanços tecnológicos para a nossa sociedade. Também desenvolvemos novas habilidades no que toca à aprendizagem de uma nova linguagem, *Python*, e de algumas das suas inúmeras bibliotecas relacionadas com o tema do projeto.

Durante o desenvolvimento, fomos encontrando algumas dificuldades, nomeadamente relacionadas com os algoritmos de *machine learning* e com a necessidade de aprendizagem de uma nova linguagem, mas com alguma pesquisa e trabalho de equipa, essas dificuldades foram superadas com sucesso.

Para concluir, os objetivos do projeto foram atingidos com sucesso, obtendo uma aplicação totalmente funcional dedicada a utilizadores que necessitem de fazer previsões sobre os seus dados. No entanto, existem alguns aspetos que podem ser melhorados como é o caso da interface visual. Esta apresenta alguns problemas quando se tenta expandir e quando se fazem inúmeras classificações. A aplicação apresenta também algumas limitações ao nível da rapidez da leitura do ficheiro de dados. Estas melhorias trariam uma maior estabilidade e rapidez à aplicação, o que poderia representar um aumento do número de utilizadores e, consequentemente, a possibilidade de esta se tornar rentável.

8. Bibliografia

- 1] [Wikipédia, “Artificial Intelligence” [Online]. Available: https://en.wikipedia.org/wiki/Artificial_intelligence. [Acedido em 07 06 2021].
- 2] [Stoodi, “Inteligência artificial: o que é, como funciona e aplicações!” [Online]. Available: <https://www.stoodi.com.br/blog/atualidades/inteligencia-artificial/>. [Acedido em 07 06 2021].
- 3] [Wikipédia, “Machine Learning” [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning. [Acedido em 08 06 2021].
- 4] [Opensanca, "Aprendizagem de Máquina: Supervisionada ou Não Supervisionada?" [Online]. Available: <https://medium.com/opensanca/aprendizagem-de-maquina-supervisionada-ou-n%C3%A3o-supervisionada-7d01f78cd80a>. [Acedido em 10 06 2021]
- 5] [Tatiana Escovedo, "Machine Learning: Conceitos e Modelos - Parte I: Aprendizado Supervisionado*" [Online]. Available: <https://tatianaesc.medium.com/machine-learning-conceitos-e-modelos-f0373bf4f445>. [Acedido em 10 06 2021]
- 6] [Saishruthi Swaminathan, “Logistic Regression - Detailed Overview” [Online]. Available: <https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc>. [Acedido em 10 06 2021]
- 7] [Onel Harrison, “Machine Learning Basics with the K-Nearest Neighbors Algorithm” [Online]. Available: <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>. [Acedido em 10 06 2021]
- 8] [Stefan Hrouda-Rasmussen, “(Gaussian) Naive Bayes” [Online]. Available: <https://towardsdatascience.com/gaussian-naive-bayes-4d2895d139a>. [Acedido em 10 06 2021]
- 9] [Rohith Gandhi, “Support Vector Machine - Introduction to Machine Learning Algorithms” [Online]. Available: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>. [Acedido em 10 06 2021]

- 10] [Aliger, "Entenda o aprendizado não supervisionado no Machine Learning" [Online]. Available: <https://www.aliger.com.br/blog/machine-learning-entenda-o-que-e-aprendizado-nao-supervisionado/>. [Acedido em 11 06 2021]
- 11] [Baijayanta Roy, "All about Categorical Variable Encoding" [Online]. Available: <https://towardsdatascience.com/all-about-categorical-variable-encoding-305f3361fd02> [Acedido em 11 04 2021]
- 12] [Dr. Saul McLeod, "Z-Score: Definition, Calculation and Interpretation" [Online]. Available: <https://www.simplypsychology.org/z-score.html> [Acedido em 13 04 2021]
- 13] [Aditya Mishra, "Metrics to Evaluate your Machine Learning Algorithm" [Online]. Available: <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234> [Acedido em 10 04 2021]
- 14] [Koo Ping Shung, "Accuracy, Precision, Recall or F1?" [Online]. Available: <https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9>. [Acedido em 10 04 2021]
- 15] [Sarang Narkhede, "Understanding Confusion Matrix" [Online]. Available: <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>. [Acedido em 10 04 2021]
- 16] [Sarang Narkhede, "Understanding AUC - ROC Curve" [Online]. Available: <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>. [Acedido em 11 04 2021]
- 17] [Wikipedia, "Python (programming language)" [Online]. Available: [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language)) [Acedido em 23 04 2021]
- 18] [JetBrains, "PyCharm" [Online]. Available: <https://www.jetbrains.com/pycharm/> [Acedido em 27 04 2021]
- 19] [T. Huff, "What is Agile Software Development?" [Online]. Available: <https://www.outsystems.com/blog/posts/agile-software-development/>. [Acedido em 15 06 2021]
- 20] [CollabNet, "13th Annual State of Agile Development" [Online]. Available: <https://stateofagile.com/#ufh-i-613553418-13th-annual-state-of-agile-report/7027494>. [Acedido em 14 06 2021]
- 21] [Agile Manifesto, "Manifesto for Agile Software Development" [Online]. Available: <https://agilemanifesto.org/>. [Acedido em 15 06 2021]

- 22] [Agile Manifesto, "Principles behind the Agile Manifesto" [Online]. Available: <https://agilemanifesto.org/principles.html>. [Acedido em 16 06 2021]
- 23] [C. Reis, "Scrum" [Online]. Available: https://ead.ipleiria.pt/2020-21/pluginfile.php/389148/mod_folder/content/0/scrum/5_ATSE_Scrum_white.pdf?forcedownload=1 [Acedido em 17 06 2021]
- 24] [C. Reis, "Kanban" [Online]. Available: https://ead.ipleiria.pt/2020-21/pluginfile.php/396871/mod_folder/content/0/kanban/7_ATSE_Kanban_white.pdf?forcedownload=1 [Acedido em 18 06 2021]
- 25] [C. Reis, "Extreme Programming" [Online]. Available: https://ead.ipleiria.pt/2020-21/pluginfile.php/392330/mod_folder/content/0/xp/6_ATSE_XP_white.pdf?forcedownload=1 [Acedido em 21 06 2021]
- 26] [Scikit-Learn, "Scikit-Learn: Machine Learning in Python" [Online]. Available: <https://scikit-learn.org/stable/> [Acedido em 25 04 2021]
- 27] [Tkinter, "Tkinter – Python Interface" [Online]. Available: <https://docs.python.org/3/library/tkinter.html> [Acedido em 25 04 2021]
- 28] [Pandas, "Pandas" [Online]. Available: <https://pandas.pydata.org/> [Acedido em 26 04 2021]
- 29] [DevMedia, "Guia completo sobre teste de software" [Online]. Available: <https://www.devmedia.com.br/guia/guia-de-testes-de-software/34403>. [Acedido em 25 08 2021]
- 30] [Wikipedia, "Testes de Software" [Online]. Available: https://pt.wikipedia.org/wiki/Teste_de_software#Teste_de_integra%C3%A7%C3%A3o. [Acedido em 26 08 2021]
- 31] [Wikipédia, "Teste de Sistema" [Online]. Available: https://pt.wikipedia.org/wiki/Teste_de_sistema. [Acedido em 26 08 2021]

9. Anexos

Anexo A - Manual de Utilização Da Aplicação

Uma vez que a interface da aplicação é relativamente complexa, decidimos fazer um manual de utilização para que seja mais fácil ao utilizador perceber como usar o NTClassifier.

Antes do utilizador poder iniciar a aplicação, terá de verificar se tem o *Python3* instalado no seu computador e algumas bibliotecas necessárias para o correto funcionamento da aplicação. As bibliotecas a instalar são: *pandas*, *scikit-learn*, *Pillow*, *tk*, *scikit-plot*. Estas poderão ser instaladas através da utilização do comando “*pip install <nome_da_biblioteca>*” na linha de comandos.

Posteriormente, o utilizador iniciará a aplicação a partir da linha de comandos, posicionada na pasta “NEW” presente na pasta do projeto, usando o comando “*python3 main.py*”.

Após iniciar a aplicação, pode escolher criar um projeto novo ou carregar um já existente, como mostra a figura abaixo.

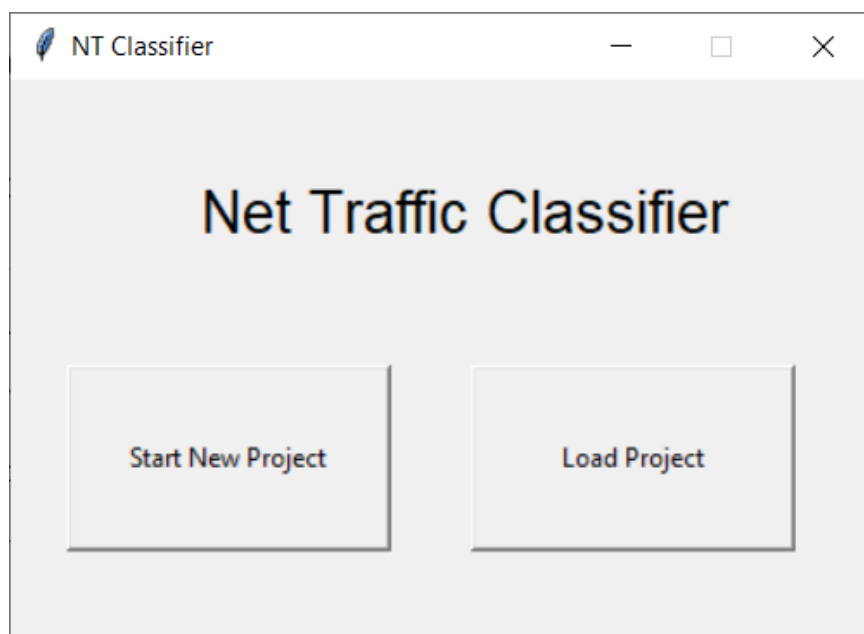


Figura 60 - Ecrã inicial da aplicação

Ao escolher a criação de um novo projeto, o utilizador tem de escolher um nome para o projeto. Depois disso tem de escolher um ficheiro de dados em formato *csv* para ser usado na aplicação.

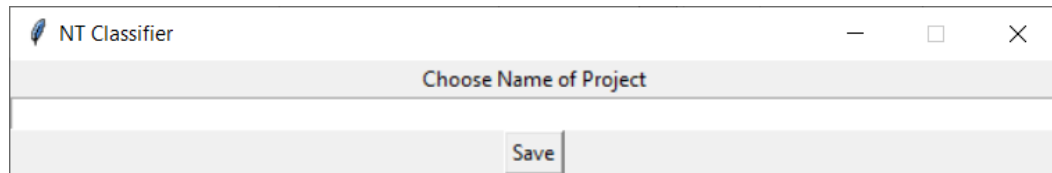


Figura 61 - Inserção do nome do projeto

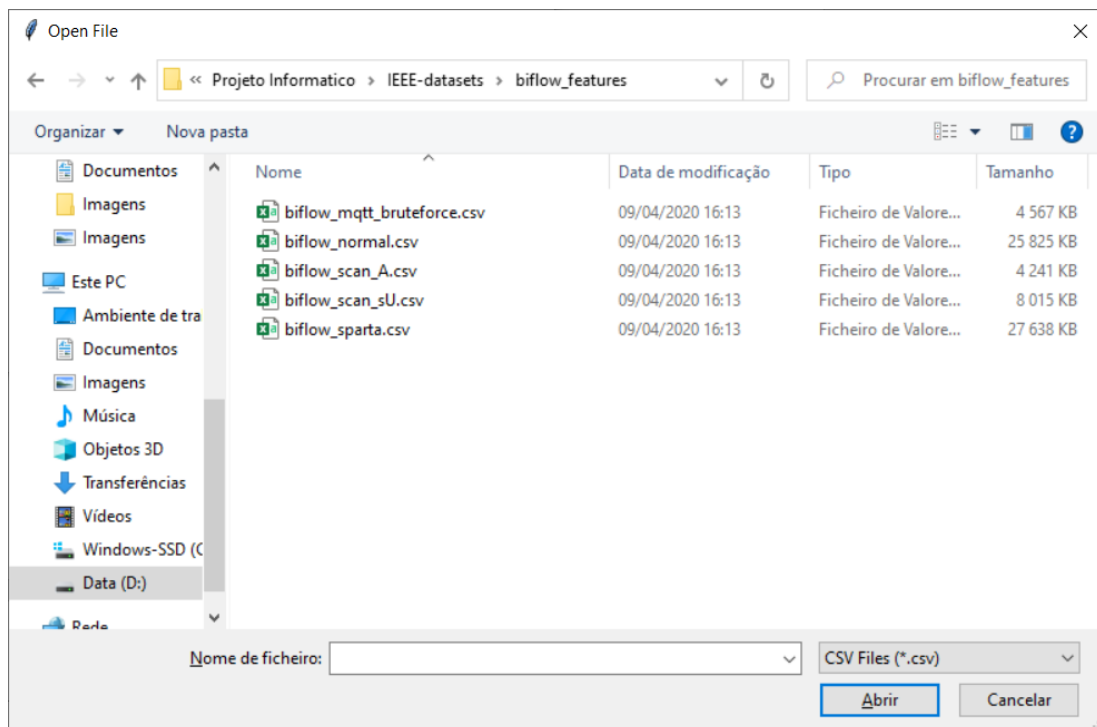


Figura 62 - Escolha do ficheiro de dados

Caso o utilizador escolha a opção de carregar um projeto já existente, terá de ir até à diretoria do projeto e escolher o ficheiro *{Nome_do_prjeto}syslog.txt* para abrir o projeto, como se vê na imagem seguinte.

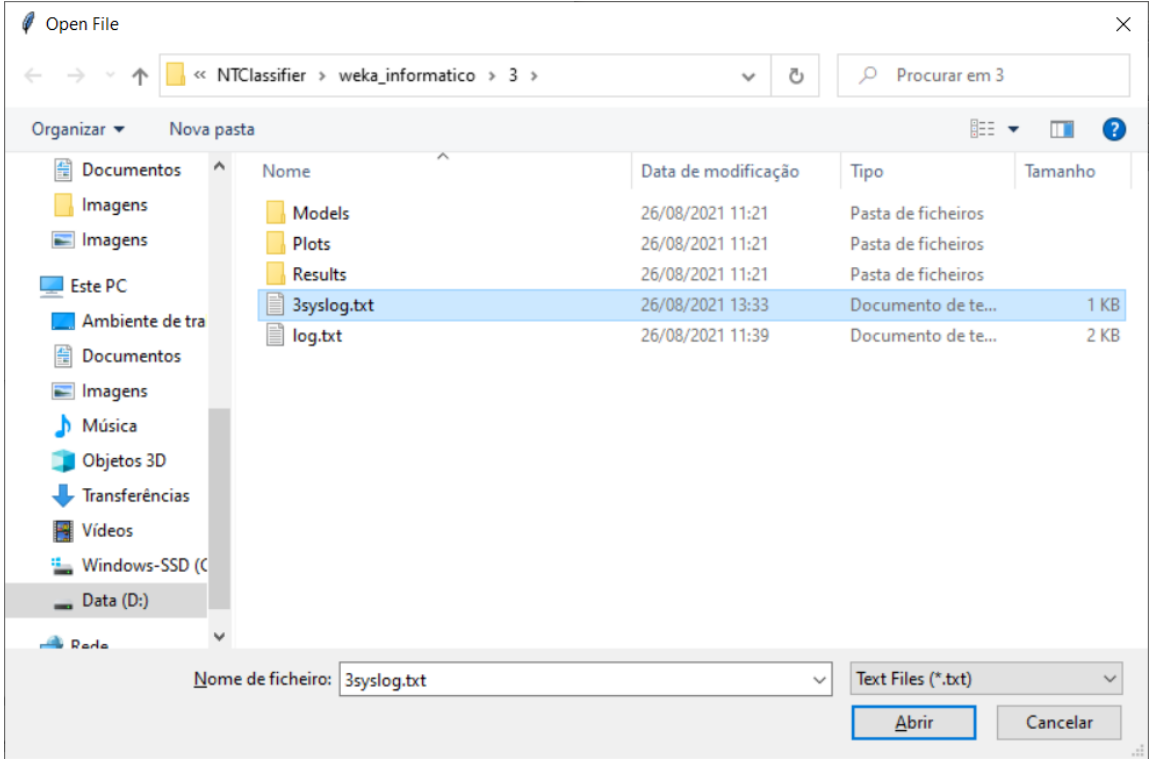


Figura 63 - Escolha do ficheiro para carregar o projeto

De seguida o *dataset* é carregado e o utilizador é direcionado para a página da preparação dos dados.

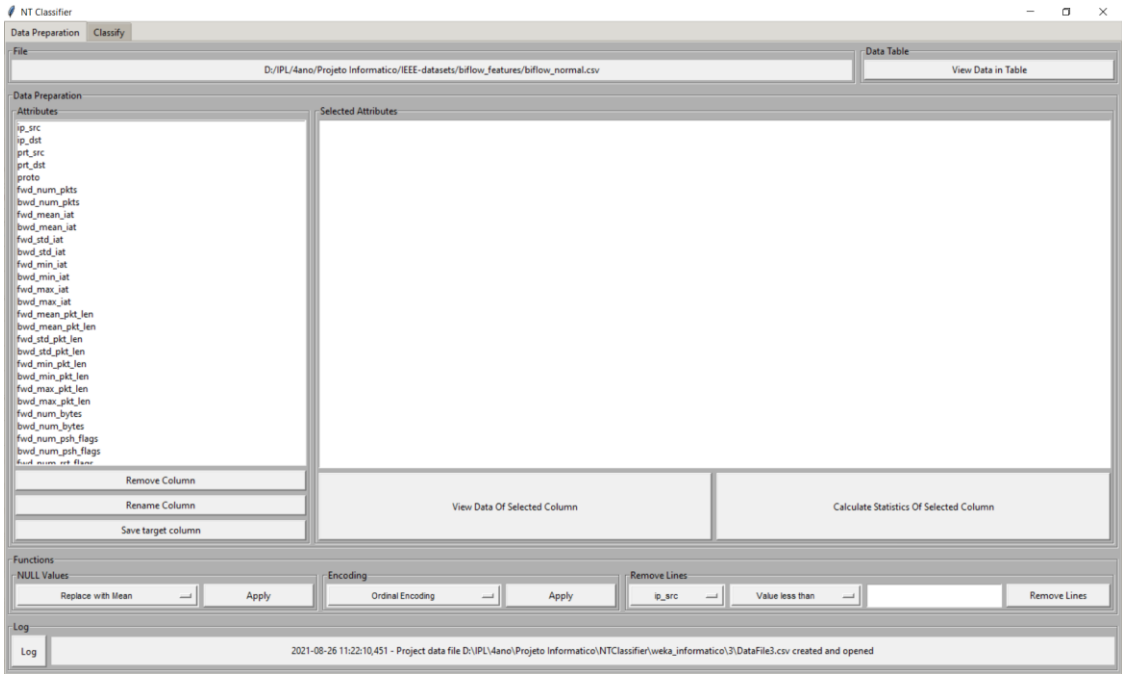


Figura 64 - Interface da preparação dos dados

Na imagem acima podemos ver em cima a diretoria do ficheiro escolhido seguido de um botão onde é possível visualizar os dados em formato de tabela. Do lado esquerdo do ecrã temos uma *textbox* com todas as colunas do ficheiro de dados, com 3 botões em baixo para remover as colunas seleccionadas, alterar o nome de uma coluna e definir qual a coluna *target*, ou seja a coluna que é o objetivo da predição do algoritmo. Do lado direito desta *textbox* temos outra, com 2 botões. O primeiro botão permite saber alguns dados acerca da coluna seleccionada e o segundo botão permite obter várias estatísticas acerca da coluna.

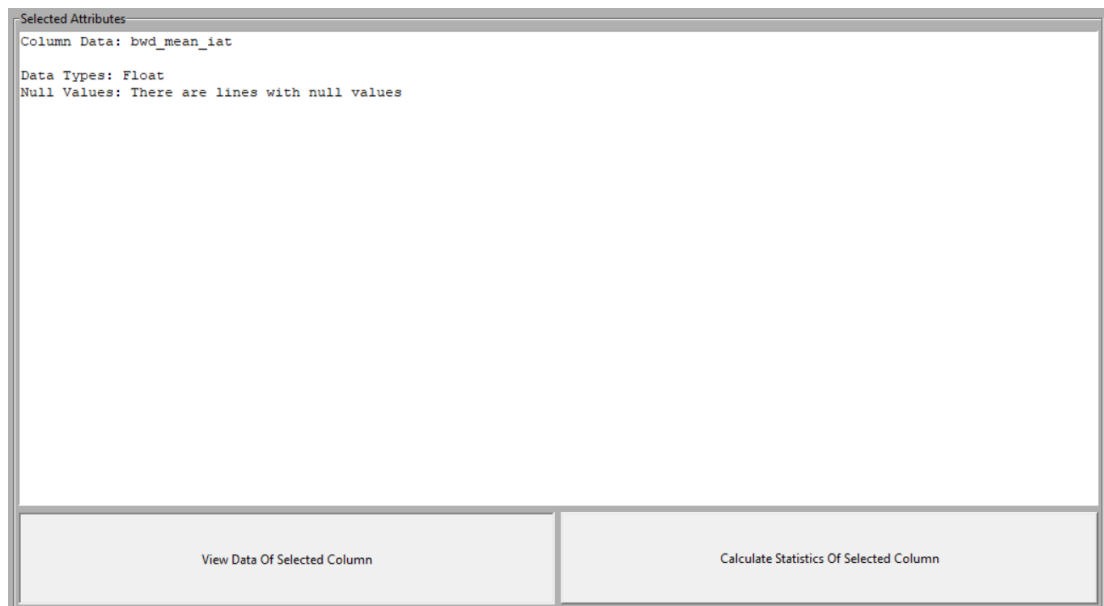


Figura 65 - Dados da coluna seleccionada

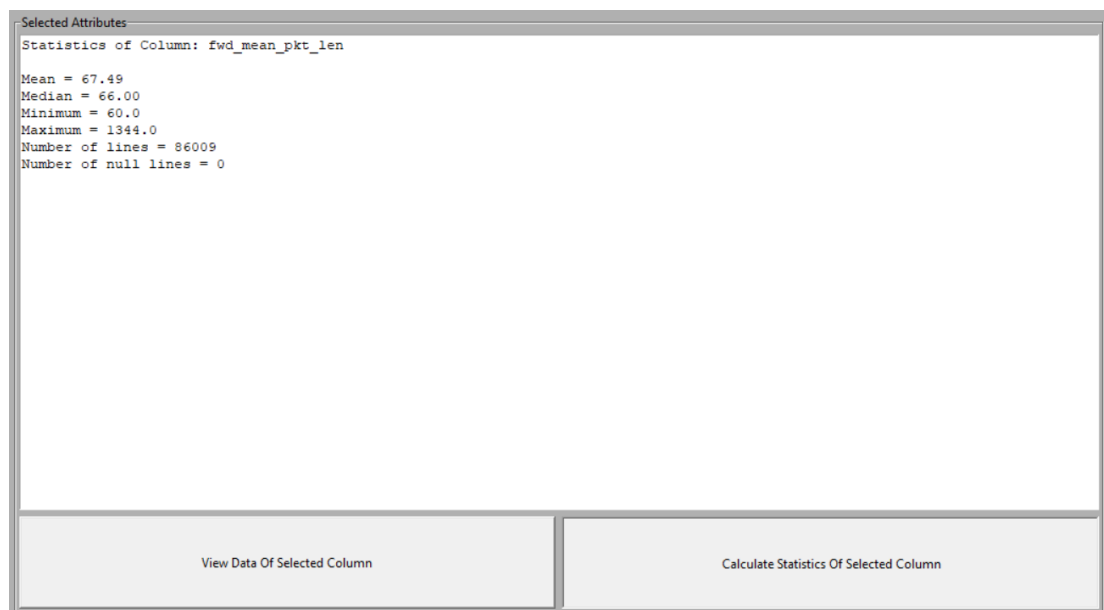


Figura 66 - Estatísticas da coluna seleccionada

Abaixo destas duas caixas de texto temos uma secção que permite modificar os dados.

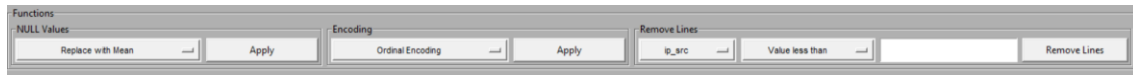


Figura 67 - Secção de tratamento dos dados

Da esquerda para a direita temos:

- Um botão que permite substituir as células sem valor pelo valor escolhido na *Dropbox*
- Um botão que permite aplicar técnicas de *encoding* nas colunas nominais e técnicas de normalização nas colunas numéricas, conforme o que for escolhido na *dropdown*. As normalizações devem ser aplicadas antes do *encoding* para que a aplicação funcione bem.
- Por último temos uma secção que permite remover linhas com um valor numa determinada coluna maiores, menores ou iguais ao especificado na caixa de texto. É também possível remover linha com células sem valor.

Por último, nesta aba da preparação dos dados temos uma secção que mostra a última linha do ficheiro de *logs*, ou seja, a última alteração feita no projeto.

Passando para o separador do trieno, teste e classificação, a aparência do ecrã é a que se vê na imagem abaixo.

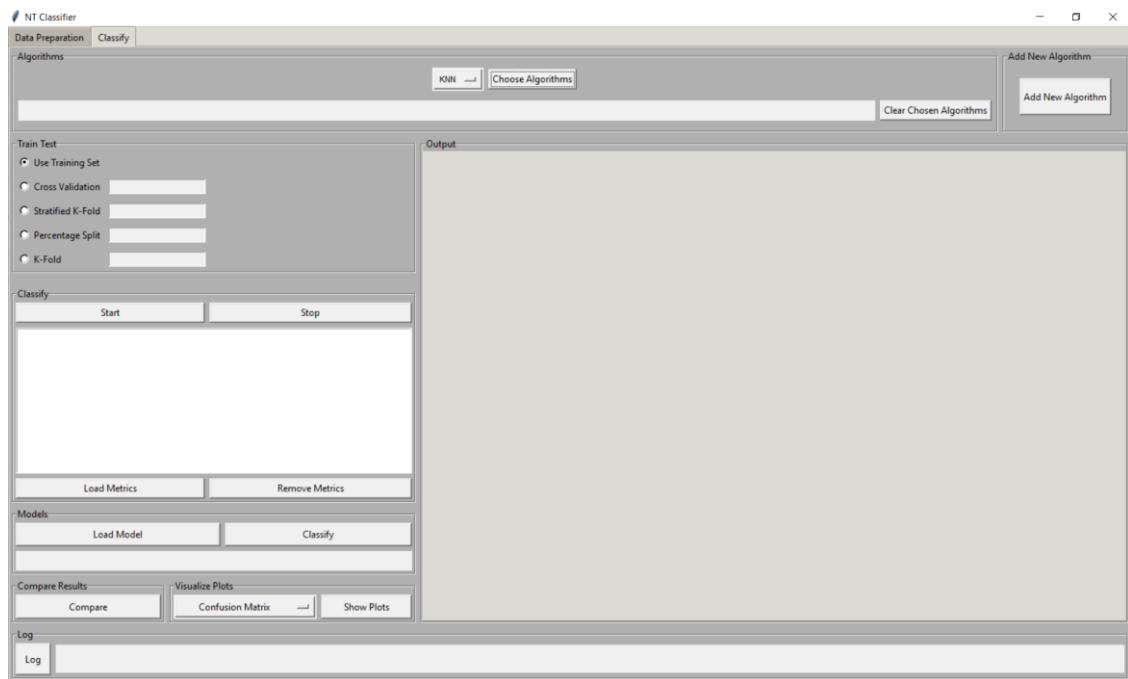


Figura 68 - Ecrã da classificação dos *datasets*

A primeira secção permite escolher um ou mais algoritmos para serem usados naas tarefas de treino e teste, bem como apagar essa lista de algoritmos a serem usados. Ao lado temos um botão que permite adicionar um novo algoritmo.

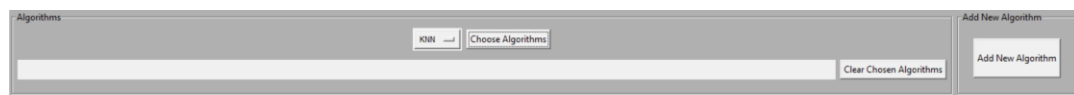


Figura 69 - Escolha dos algoritmos a serem usados

A secção seguinte permite escolher o tipo de divisão de dados que o utilizador pretende utilizar para as tarefas de treino e de teste.

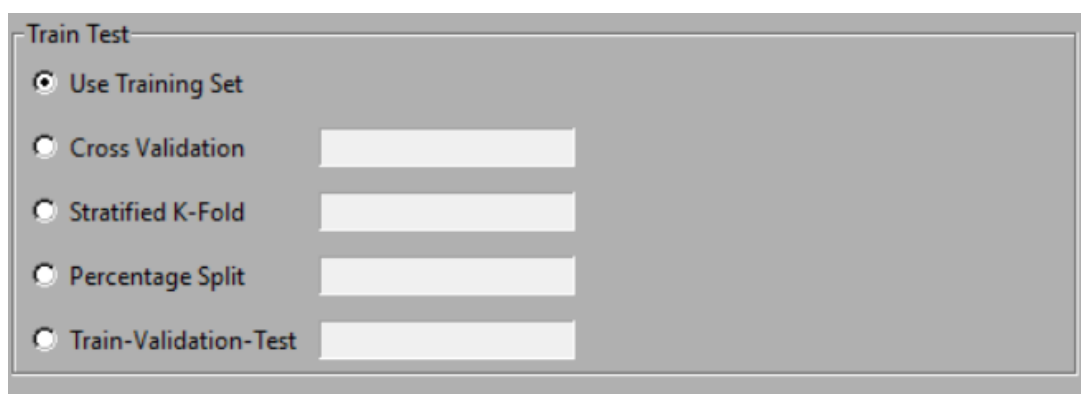


Figura 70 - Escolha do tipo de divisão de dados

Após a divisão dos dados, o botão *Start* presente na figura abaixo, permite iniciar o treino e teste dos algoritmos selecionados. O botão *Stop* permite interromper o processo apenas entre algoritmos, ou seja, neste caso, após o algoritmo LR começar a correr se o utilizador clicar no botão *Stop* o processo só para quando esse algoritmo terminar. Quando os algoritmos selecionados terminarem é possível ver na caixa de texto abaixo a data em que os algoritmos foram corridos. Com o botão *Load Metrics* podemos carregar as métricas de um algoritmo já treinado, e com o botão *Remove Metrics* podemos apagar os algoritmos e respectivas métricas selecionados na lista.

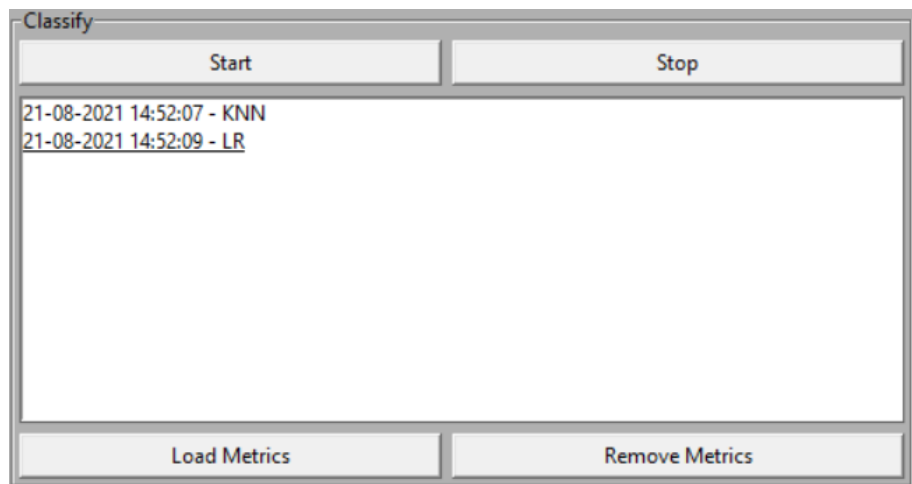


Figura 71 - Secção de treino e teste dos algoritmos

Na secção ao lado, é possível ver as métricas resultantes da aplicação dos algoritmos.

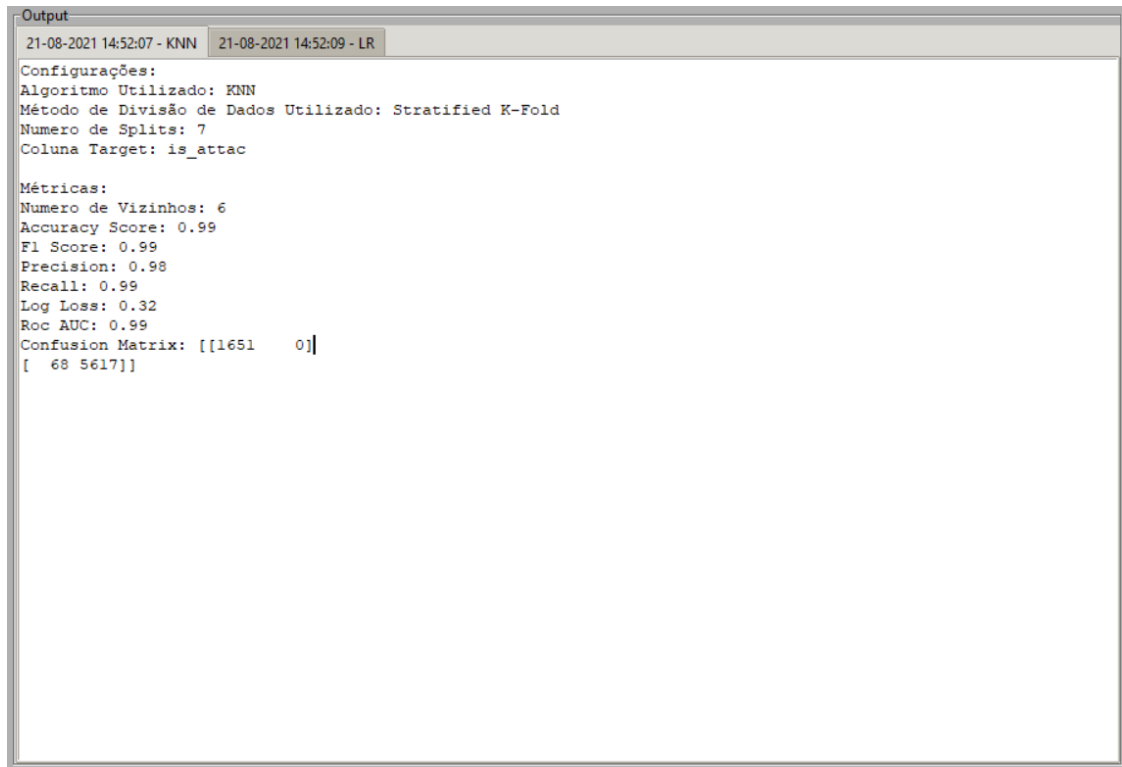


Figura 72 - Métricas dos algoritmos aplicados

É possível ainda escolher um *dataset* para ser classificado com um modelo já treinado.

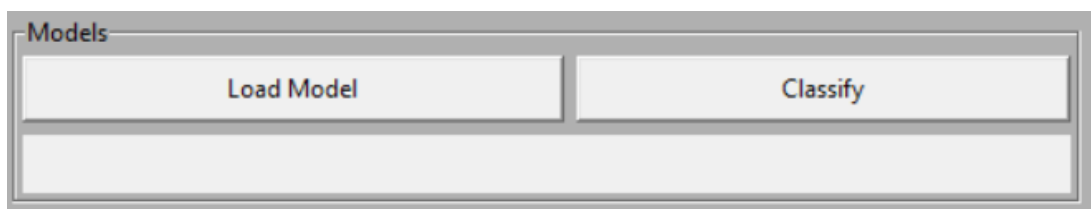


Figura 73 - Classificação de um *dataset* com um modelo já treinado

Após o treino e teste dos algoritmos é possível visualizar alguns gráficos e comparar os resultados de 2 ou mais algoritmos caso tenham sido treinados mais do que 1 algoritmo.

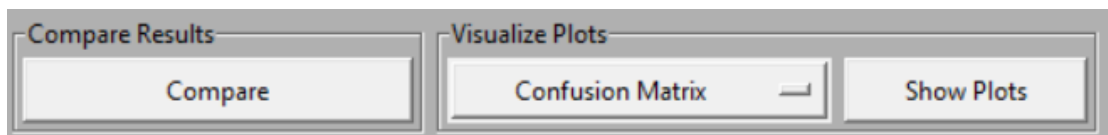


Figura 74 - Comparação de resultados e visualização de gráficos

Após a criação de um novo projeto, para que os dados sejam todos gravados nos ficheiros correspondentes, é necessário fechar por completo a aplicação.