Prospeção de Dados: Classificação de Dados de Vinho e Indígenas

Francisco Passinho[1], Luís Cavique[2]

[1] Universidade Aberta de Portugal, Lisboa, Portugal, francisco.passinho@gmail.com
[2] Universidade Aberta de Portugal, Lisboa, Portugal, lcavique@gmail.com

Resumo

O data mining é uma área multidisciplinar que tem como objetivo extrair e descobrir padrões em grandes grupos de dados através de algoritmos de inteligentes. Neste artigo, foi realizado um estudo de classificação de dois datasets amplamente usados na literatura: dados de reconhecimento de vinhos italianos e dados de diabetes de indígenas Pima. Foi realizada uma análise estatística dos dados e a respetiva classificação com vários algoritmos de classificação. Foram obtidos resultados semelhantes e em alguns casos superiores aos reportados na literatura. Os melhores classificadores foram as Florestas Aleatórias e as Redes Neuronais com valores de exatidão acima dos 80%.

Palavras-chave: classificação, prospeção de dados.

Title: Data Mining: Classification of Wine and Pima Indians Datasets.

Abstract: Data mining is a multidisciplinary area that aims to extract and find patterns in large groups of data using intelligent algorithms. In this paper, we performed a classification study of two widely used datasets in the literature: Italian wine recognition data and Pima indigenous diabetes data. A statistical analysis of the data and the respective classification was carried out with several classification algorithms. Similar results and, in some cases, better results were obtained when compared to those reported in the literature. The best classifiers were Random Forests and Neural Networks with accuracy values above 80%.

Keywords: classification, data mining.

1. Introdução

O *data mining* é uma área multidisciplinar da ciência da computação que tem como objetivo extrair e descobrir padrões em grandes grupos de dados através de algoritmos de inteligentes (J. Han, 2011). O *data mining* está inserido na área de inteligência artificial (*machine learning)*, *data science* e *deep learning*.

O *machine learning* consiste em programas que "aprendem" automaticamente a reconhecer padrões e a tomar decisões. O *machine learning* pode dividir-se em aprendizagem ativa, semi-supervisionada, aprendizagem não supervisionada, e supervisionada (J. Han, 2011). Na aprendizagem ativa o utilizador tem um papel ativo na identificação da classe dos dados. A aprendizagem semi-supervisionada utiliza ambos os métodos de aprendizagem não supervisionada e supervisionada. A aprendizagem não supervisionada, normalmente definida por *clustering*, procura padrões com base nos dados sem ter conhecimento prévio de como estes se dividem. Por sua vez, na aprendizagem supervisionada, os algoritmos têm esta informação. A classificação é definida como um método de aprendizagem supervisionada em que um modelo é construído tendo em conta a que classe os dados pertencem. Estas classes são categóricas (discretas e sem ordem) e podem ter vários significados dependendo da aplicação.

Este trabalho consistiu em estudar uma área do *data mining*, em particular, a área de classificação aplicada a duas bases de dados: reconhecimento de vinhos italianos e dados de diabetes de indígenas Pima. Pretendeu-se explorar a utilização de diferentes algoritmos de classificação e a comparação dos resultados dos dois datasets.

O dataset "Wine" foi disponibilizado pelo Institute of Pharmaceutical and Food Analysis and Technologies (Forina, Leardi, C, & Lanteri, 1998). O dataset inclui dados da análise química de vinhos cultivados em Itália, mas derivados de três cultivos diferentes. Este dataset contém um total de 12KB e dados de análise química de 174 vinhos italianos de três cultivos conhecidos. A análise química determinou as quantidades de 13 constituintes diferentes, atributos já pré-calculados (álcool, ácido málico, cinzas, magnésio, tonalidade, prolina, etc). O objetivo da utilização deste dataset foi classificar os 174 vinhos num dos três cultivos diferentes, com base na sua análise química.

O dataset "PimaIndians" foi disponibilizado pelo National Institute of Diabetes and Digestive and Kidney Diseases (PeterH Bennett, 1970; National Institutes of Health, 2020) e foi criado a partir de dados de mulheres de origem indígena "Pima". O objetivo da utilização deste dataset é prever se uma dada pessoa tem ou não diabetes com base nas medidas de oito fatores de risco. Este dataset contém um total de 24KB e inclui dados de 768 mulheres com pelo menos 21 anos de idade. O atributo a classificar (classe) foi a ausência/presença de diabetes e os atributos incluem informação como o número de gravidezes, concentração de glucose, tensão arterial, espessura da prega de pele, idade, etc.

Numa primeira fase, foi efetuada uma pesquisa detalhada do que é normalmente utilizado para analisar este tipo de dados. De seguida, foi efetuada uma análise dos dados de forma a perceber quais serão os melhores atributos e os melhores classificadores a adotar. Posteriormente, foi feita a implementação destes classificadores em Python utilizando a biblioteca do Scikit-Learn (Scikit-Learn, 2021). Depois de treinados os modelos com os classificadores escolhidos, o desempenho dos modelos foi avaliado utilizando métricas de desempenho.

2. Estado da arte de classificação dos datasets escolhidos

Tendo em conta o ano em que os datasets em estudo foram criados (cerca de 20 e 50 anos para o dataset "Wine" e "PimaIndians", respetivamente), existem já muitos estudos que

os abordaram, quer para resolver o problema principal e apresentar as propostas dos autores, quer para validar novos algoritmos em conjunto com outros datasets.

Os algoritmos Naïve Bayes e redes neuronais foram utilizados para classificar ambos os datasets (Kontkanen, 1999) e obtiveram valores de exatidão de 97,8% e 98,9% para o dataset "Wine" e 74,5% e 76,0% para o dataset "PimaIndians", respetivamente a cada algoritmo. Outras abordagens com redes neuronais associadas com o algoritmo K-Vizinhos Mais Próximos mostraram resultados máximos de exatidão de 96,0% e 75,6% com o dataset "Wine" e "PimaIndians", respetivamente (Jiang, 2004). Árvores de Decisão foram também consideradas para classificar os dois datasets com a exatidão chegando aos 91,7% e 78,8% para o dataset "Wine" e "PimaIndians", respetivamente (Sagi, 2020). Vários tipos de Florestas Aleatórias (ou grupos de Árvores de Decisão), permitiram obter valores de exatidão até 98,9% para o dataset "Wine" (Liu, 2005) e 79,1% para o dataset "PimaIndians" (Banfield, 2004) (Liu, 2005) (Sagi, 2020).

Alguns algoritmos, como é o caso do Support Vector Machines, têm limitações em classificar datasets com mais de uma classe. Num estudo de 2002, um artigo utilizou Support Vector Machines para classificar o dataset "Wine" utilizando a metodologia "um-contra-um" e "um-contra-todos". Na metodologia "um-contra-um", os autores efetuaram a classificação utilizando apenas duas classes de cada vez, obtendo 99,4% de exatidão, enquanto que, utilizando "um-contra-todos", uma das classes era considerada como a classe positiva e as restantes como classe negativa, obtendo 98,9% de exatidão (Chih-Wei Hsu, 2002). Métodos de seleção de atributos também permitiram melhorar a classificação com o dataset "PimaIndians" por 7% (de 72% para 79%) com redes RBF (Brauer, 1997).

Em conclusão, os resultados dos vários artigos publicados com os dois datasets mostram que a classificação do dataset "PimaIndians" é mais desafiante que a classificação do dataset "Wine". Não existem diferenças significativas de classificação utilizando os diferentes classificadores, apesar das Florestas Aleatórias e as redes neuronais terem mostrado melhores desempenhos.

3. Metodologia

Nesta secção será apresentada a metodologia geral de classificação utilizada para classificar os dois datasets escolhidos: "Wine" e "PimaIndians".

3.1. Classificadores

Para avaliar ambos os datasets "Wine" e "PimaIndians" foram utilizados cinco classificadores: Naïve Bayes, K-Vizinhos Mais Próximos, Árvores de Decisão, Florestas Aleatórias e Redes Neuronais. Para a implementação dos classificadores foram utilizadas as bibliotecas Python representadas no **Error! Reference source not found.**.

Nas seguintes secções são apresentados os fundamentos teóricos de alguns dos classificadores mencionados na secção acima e que serão utilizados neste trabalho.

```
1.  import pandas as pd # gerir os dados da tabela
2.  import numpy as np # cálculos numéricos
3.  import matplotlib.pyplot as plt # gráficos
4.  from sklearn.preprocessing import StandardScaler # normalizar features
5.  from sklearn.model_selection import train_test_split # separar treino/teste
6.  from sklearn.metrics import accuracy_score, balanced_accuracy_score,
matthews_corrcoef, cohen_kappa_score, roc_curve, auc
7.  from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay # calculo das
metricas
8.  from sklearn.model_selection import cross_val_score, cross_val_predict # cross-
validation e obter resultados
9.  from sklearn.naive_bayes import GaussianNB # naive bayes
10. from sklearn.model_selection import GridSearchCV # optimização de parametros
11. from sklearn.neighbors import KNeighborsClassifier # k-vizinhos mais proximos
12. from sklearn.ensemble import RandomForestClassifier # florestas aleatorias
13. from sklearn.neural_network import MLPClassifier # redes neuronais
14. from sklearn import tree # arvores de decisao
15. from sklearn.multiclass import OneVsRestClassifier # classificação "um-contra-todos"
16. from sklearn.preprocessing import label_binarize # binarizar classes para a
classificação "um-contra-todos"
```

Bloco de código 1. Importação de bibliotecas Python comum a ambos os datasets.

3.1.1. Naïve Bayes

O algoritmo Naïve Bayes é um algoritmo probabilístico que assume que os atributos do dataset são independentes. O algoritmo não requer datasets de grandes dimensões.

Este algoritmo é baseado no teorema de Bayes (Tamais, 2019). Considerando um caso com n atributos, $X = x_1, \dots, x_n$ é o conjunto de atributos, y é a label e $\hat{y}$ é a label prevista, o algoritmo pode ser representado da seguinte forma:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n|y)}{P(x_1, \dots, x_n)}$$

em que $P(y \mid x_1, \dots, x_n)$ é a probabilidade condicionada de acontecer y dado dos atributos $x_1, \dots, x_n$. Uma vez que o algoritmo assume que os atributos do dataset são independentes, $P(x_1, \dots, x_n|y) = P(x_1|y) \times \dots \times P(x_n|y)$. $P(x_1, \dots, x_n)$ é constante e, portanto, pode ser considerado que:

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

O classificador atribui uma dada classe a uma observação, maximizando o valor de $P(y \mid x_1, \dots, x_n)$:

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Este classificador foi implementado em Python com a função GaussianNB().

3.1.2. K-Vizinhos Mais Próximos (K-Nearest Neighbors)

O algoritmo K-Vizinhos Mais Próximos classifica os dados tendo em conta a distância entre o ponto a considerar e os dados utilizados para treinar o algoritmo. A classe atribuída a cada ponto resulta da maioria das classes atribuídas aos K-Vizinhos mais próximos (R. Nisbet, 2009).

Este algoritmo tem dois principais parâmetros que podem ser otimizados: o número de vizinhos K a considerar e a função da distância usada para calcular a distância entre os dados. Esta distância pode ser a distância euclidiana (em linha recta), distância Manhattan ou "city-block" (em linha horizontal e vertical) ou a de Chebyshev. Estas distâncias podem ser calculadas entre dois pontos k e l, seguindo as seguintes fórmulas:

$$\text{Distância euclideana} = \sqrt{\sum_{i=1}^{d} (x_{ik} - x_{il})^2}$$

$$\text{Distância Manhattan} = \sum_{i=1}^{d} | x_{ik} - x_{il} |$$

$$\text{Distância Chebyshev} = \max_{i}(| x_{ik} - x_{il} |)$$

O classificador K-Vizinhos Mais Próximos foi implementado com a função KNeighborsClassifier () e foram otimizados o número de vizinhos (considerando de 1 a 10) e o tipo de distâncias (euclidiana, Manhattan e Chebyshev).

3.1.3. Árvores de decisão (Decision Trees)

As Árvores de Decisão criam modelos com uma estrutura de árvore, dividindo o conjunto de treino em problemas de aprendizagem mais pequenos. Uma árvore de decisão geralmente começa com um único nó/folha, que se divide em vários nós/folhas. Cada um desses nós leva a nós adicionais, ramificando-se noutras possibilidades. As árvores de decisão trabalham diretamente de uma só preparação de tabela de dados, não exigindo qualquer preparação prévia dos dados. Também não são afetadas pela falta de valores nos dados ou dados discrepantes. Incluem uma análise exaustiva, mas com recursos nada complicados para o seu desenvolvimento, ao contrário dos dois classificadores falados acima.

A escolha dos nós/folhas pode ser determinada por dois tipos de critérios: entropia ou índice de gini. Ambos verificam como os dados estão distribuídos de acordo com a classe a prever.

$$gini = \sum_{i=1}^{C} p_i (1 - p_i)$$

$$\boldsymbol{entropia} = -\sum_{\boldsymbol{i=1}}^{\boldsymbol{C}} \boldsymbol{p_i} \log_2 \boldsymbol{p_i}$$

onde p_i é a probabilidade dos dados pertencerem à classe i. O objetivo na construção da árvore é minimizar a entropia ou o índice de gini.

As Árvores de Decisão foram implementadas com a função DecisionTreeClassifier (). Existem alguns parâmetros que podem ser otimizados nas árvores de decisão que definem como a árvore é criada e como é feita a decisão de que nós/folhas da árvore são redundantes e podem ser excluídas da árvore. Foram otimizados o critério a utilizar (gini ou entropia), a profundidade máxima (níveis) da árvore (1 a 10), o número mínimo de observações em cada folha (2 a 15), o número mínimo de observações para separar os dados numa nova folha (1 a 15) e o máximo de folhas (2 a 15).

3.1.4. Florestas Aleatórias (Random Forests)

As florestas aleatórias são uma combinação das árvores de decisão que permitem generalizar os resultados quando comparados com uma só árvore de decisão. As florestas aleatórias criam várias árvores de decisão utilizando uma seleção aleatória dos atributos do dataset para cada árvore (Breiman, 2001). Tal como no caso do algoritmo K-Vizinhos Mais Próximos, o resultado da classificação resulta da maioria dos resultados de todas as árvores.

As Florestas Aleatórias com a função RandomForestClassifier() e a sua otimização de parâmetros inclui os parâmetros das árvores de decisão e o número de árvores (5, 50, 100 e 150).

3.1.5. Redes Neuronais

Uma rede neuronal é constituída por elementos chamados neurónios que podem ser organizados em várias camadas. Um Perceptrão, é uma rede neuronal simples, ou de camada única, que simboliza um neurónio. Um só neurónio é constituído por p sinais de entrada (C_1, C_2, …, C_P), que representam os atributos, por pesos (w_1, w_2, …, w_p); e por um *bias*, uma entrada de valor "1" associada a um peso "w" em cada neurónio.

O valor de saída desta rede neuronal é calculado a partir da soma ponderada dos sinais de entrada ($\boldsymbol{v_k}$) e da aplicação de uma função de ativação e de uma função de limitação.

O valor de $\boldsymbol{v_k}$ é calculado como $\boldsymbol{w_1 C_1 + w_2 C_2 + \ldots + w_p C_p}$. Um exemplo da função de ativação é a função sigmóide onde o valor resultante se encontra compreendido entre 0 e 1:

$$\boldsymbol{f(a)} = \frac{\boldsymbol{1}}{\boldsymbol{(1 + e^{-a})}}$$

A função de limitação permite binarizar o valor tendo em conta um dado limiar (por exemplo 0,5). Assim, os valores de saída serão valores booleanos (0 ou 1):

$$y_k = 1, \text{ se } f(a) >= \text{limiar usado}$$

ou

$$y_k = 0, \text{ se } f(a) < \text{limiar usado}$$

A aprendizagem da rede neuronal implica a alteração dos pesos a cada iteração, consoante o erro obtido e a partir dos pesos antes utilizados. Para o cálculo destes novos pesos são necessários parâmetros como a taxa de aprendizagem.

As Redes Neuronais foram implementadas em Python com a função MLPClassifier(). Na otimização de parâmetros foram variados o número de camadas e neurónios da rede (3 camadas com 50 neurónios cada; 3 camadas com 50, 100 e 50 neurónios respetivamente; 1 camada com 100 neurónios; 1 camada com 50 neurónios), a função de ativação (sigmóide, tangente hiperbólica, unidade linear retificada), o método de propagação de erros, a taxa de aprendizagem (constante ou adaptativa) e o máximo de iterações (50, 100, 150). Estes parâmetros foram utilizados com base em exemplos na literatura.

3.2. Classificação

Num primeiro passo, os dados foram separados em duas variáveis (x e y), um grupo de atributos/*features* (características representativas dos dados) e um atributo discriminante/*label* (atributo que define a que classe pertence). Os dados de cada atributo (x) foram normalizados com média 0 e desvio padrão 1, como apresentado no **Error! Reference source not found.**.

```
1.  dados = pd.read_csv('wine.csv')
2.  x = dados.iloc[:,1:]
3.  y = dados["Class"]
4.  std_scaler = StandardScaler()
5.  x = pd.DataFrame(std_scaler.fit_transform(x), columns=x.columns)
```

Bloco de código 2. Exemplo de leitura de dados e normalização de atributos com o dataset "Wine". O código para o dataset "PimaIndians" é semelhante.

O conjunto de dados foi dividido em conjunto de treino e teste na proporção 75%/25%, como apresentado no bloco de código 3. O classificador cria um modelo ao aprender padrões a partir do conjunto de treino. Depois de criado o modelo, este é testado utilizando um conjunto de teste que recebe apenas o grupo de atributos das observações respetivas. O grupo de *labels* previstas é posteriormente comparado com as *labels* originais associadas ao grupo de teste e utilizado para calcular métricas de desempenho.

```
1.    # Dividir em treino e teste
2.   X_train, X_test, y_train, y_test = train_test_split(x, y, random_state=0)
```

Bloco de código 3. Exemplo de código para dividir o conjunto de dados em treino e teste.

Os algoritmos foram treinados utilizando o método Validação Cruzada por Camadas Estratificadas (com um valor de K=5) no conjunto de treino. Este método consiste em treinar e testar o classificador várias vezes, dividindo os dados em diferentes subconjuntos de treino e teste em cada iteração. O classificador é treinado com K-1 subconjuntos e testado pelo subconjunto que falta. O melhor modelo é escolhido tendo em conta a média dos resultados de classificação de cada iteração. No caso da Validação Cruzada por Camadas Estratificadas é assegurada que a proporção das classes em cada subconjunto é mantida. O melhor modelo foi encontrado avaliando as métricas de desempenho como a exatidão, exatidão ponderada, o coeficiente de correlação Matthews, a matriz confusão e a curva ROC, calculadas utilizando o conjunto de teste. O bloco de código 4 apresenta um exemplo desta metodologia para o classificador K-Vizinhos Mais Próximos, incluindo a divisão em treino/teste, otimização de parâmetros com Validação Cruzada por Camadas Estratificadas e cálculo das métricas.

Um dataset pode ter dados pertencentes apenas a duas classes (caso binário) ou a mais classes. No caso binário, a condição negativa é frequentemente representada pela *label* 0 e a condição positiva é representada pela *label* 1.

```
3.   # k-Vizinhos Mais Próximos
4.   parameters = {
5.                 'n_neighbors':np.arange(1,10),
6.                 'metric':["euclidean","manhattan","chebyshev"]
7.                }
8.   neigh = KNeighborsClassifier()
9.   # Otimização de parâmetros, Validação Cruzada por Camadas Estratificada (pré-definido)
com 75% dos dados
10.  clf = GridSearchCV(neigh, parameters, cv=5)
11.  clf.fit(X_train, y_train)
12.  # Teste do melhor modelo com 25% dos dados
13.  y_pred_test_bin_KNN = clf.best_estimator_.predict(X_test)
14.  y_pred_test_prob_KNN = clf.best_estimator_.predict_proba(X_test)
15.  # Métricas
16.  # https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score
17.  print('Accuracy:' + str(accuracy_score(y_test, y_pred_test_bin_KNN)))
18.  # https://scikit-learn.org/stable/modules/model_evaluation.html#balanced-accuracy-
score
19.  print('Balanced accuracy:' + str(balanced_accuracy_score(y_test,
y_pred_test_bin_KNN)))
20.  # https://scikit-learn.org/stable/modules/model_evaluation.html#matthews-correlation-
coefficient
```

```
print('Matthews Correlation Coefficient:' + str(matthews_corrcoef(y_test,
y_pred_test_bin_KNN)))
# Matriz Confusão
cm = confusion_matrix(y_test, y_pred_test_bin_KNN)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
```

Bloco de código 4. Exemplo de classificação e cálculo de métricas com o algoritmo K-Vizinhos Mais Próximos que é comum para os dois datasets.

3.3. Métricas de desempenho

Existem várias métricas que podem ser utilizadas para avaliar o desempenho dos algoritmos de classificação.

A exatidão representa a fração do número de observações corretamente classificados e é calculada da seguinte forma:

$$\frac{1}{n_{obs}} \sum_{i=1}^{n_{obs}} (\hat{y}_i - y_i) = \frac{\sum VP + \sum VN}{n_{obs}}$$

onde n_{obs} corresponde ao número de observações, $\hat{y}_i$ à classe prevista e y_i à classe real. VP é o número de Verdadeiros Positivos (observações com classe positiva e classificadas como positivas) e VN é o número de Verdadeiros Negativos (observações com classe negativa e classificadas como negativas).

O número de VP e VN, tal como o número de Falsos Positivos (FP - observações com classe negativa e classificadas como positivas) e Falsos Negativos (FN - observações com classe positiva e classificadas como negativas) podem ser analisados através da matriz confusão (figura 1). A matriz confusão é calculada a partir dos resultados de classificação binários obtidos pelo classificador.

			Classe Real	
			Condição **positiva** 1	Condição **negativa** 0
Classificação	Previsão **Positiva**	1	Verdadeiros Positivos (VP)	Falsos Positivos (FP)
	Previsão **negativa**	0	Falsos Negativos (FN)	Verdadeiros Negativos (VN)

Figura 1. Estrutura da matriz confusão.

Quando não existe a mesma proporção de observações para cada classe, a exatidão mencionada acima pode não ser a mais representativa do desempenho dos algoritmos. Nestes casos, podem ser utilizadas duas outras métricas: a exatidão ponderada e o coeficiente de correlação Matthews.

A exatidão ponderada tem em conta a proporção de falsos positivos e falsos negativos, e é calculada da seguinte forma:

$$\frac{1}{2}\left(\frac{VP}{VP+FN}+\frac{VN}{VN+FP}\right)$$

O coeficiente de correlação Matthews tem também em conta as métricas da matriz confusão e é calculada da seguinte forma:

$$\frac{VP \times VN - FP \times FN}{\sqrt{(VP+FP)(VP+FN)(VN+FP)(VN+FN)}}$$

Este coeficiente está compreendido entre -1 e +1, sendo que +1 representa uma classificação perfeita, 0 representa uma classificação aleatória e -1 representa uma classificação inversa. Permite inferir sobre o desempenho do classificador tendo como referência a classificação "à sorte".

O desempenho dos algoritmos pode ser comparado utilizando a curva ROC. A curva ROC é uma representação gráfica que ilustra o desempenho de classificador em função da Taxa de Falsos Positivos (TFP) e a Taxa de Verdadeiros Positivos (TVP), que são calculados da seguinte forma:

$$TFP = \frac{FP}{FP+VN}$$

$$TVP = \frac{VP}{VP+FN}$$

A curva ROC (figura 2) é obtida ao variar o limiar que transforma os resultados de classificação em forma de probabilidades em forma binária. Para cada valor do limiar, pode-se obter um valor de VP e FN que permitirá traçar a curva ROC. A partir das curvas ROC, pode-se calcular a área de baixo da curva, normalmente representado por AUC (do inglês, Area Under the Curve). Um classificador perfeito terá uma curva que se aproxima do canto superior esquerdo (TFP=0 e TVP=1), onde a AUC será igual a 1. Um classificador é considerado bom/razoável se a sua curva estiver acima da diagonal que resulta numa AUC igual a 0.5. Abaixo dessa curva, o classificador é considerado mau.

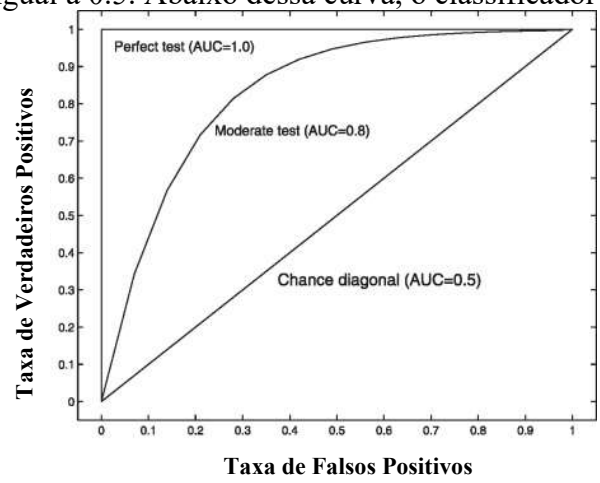


Figura 2. Exemplo de três curvas ROC (Andy.K Devos, 2007).

4. Dataset de vinhos italianos

Nesta secção será apresentada a descrição dos atributos do dataset de vinhos italianos, a análise estatística descritiva, resultados e respetiva discussão da classificação.

4.1. Análise dos Dados

A tabela 1 apresenta a descrição dos atributos do dataset "Wine". Existem 13 atributos que serão utilizados para classificar a classe do tipo de cultivo. Existem 174 observações no total, 59 observações/vinhos da classe 1, 71 da classe 2 e 48 da classe 3.

Tabela 1. Descrição dos atributos do dataset "Wine".

#	Nome de atributo	Tipo	Descrição
1	**Id**	Uid	Identificador único do registo/exemplo
2	**Class**	Int	Número do cultivo de vinhos (1, 2 ou 3)
3	**Alcohol**	double	Teor alcoólico (% v/v)
4	**MalicAcid**	double	Ácido málico
5	**Ash**	double	Cinza
6	**AlcalinityAsh**	double	Alcalinidade das cinzas
7	**Magnesium**	Int	Magnésio
8	**PhenolsTotal**	double	Fenóis totais
9	**Flavanoids**	double	Flavanóides
10	**NonflavanoidPhenols**	double	Fenóis inflavonóides
11	**Proanthocyanins**	double	Proantocianidinas
12	**Color**	double	Intensidade da cor
13	**Hue**	double	Tonalidade
14	**OD280/OD315**	double	OD280 / OD315 de vinhos diluídos
15	**Proline**	Int	Prolina

Nos diagramas de caixas de dados originais de cada atributo/feature do dataset "Wine" verificou-se que existe variabilidade entre os valores de todos os atributos, nomeadamente no seu valor médio. O atributo Prolina e o Magnésio têm valores bastante diferentes dos restantes. Por isso, é importante fazer a normalização dos atributos, o que resulta no gráfico da figura 3. A partir deste novo gráfico, é possível comparar as distribuições que, apesar de terem diferenças, estas não são significativas.

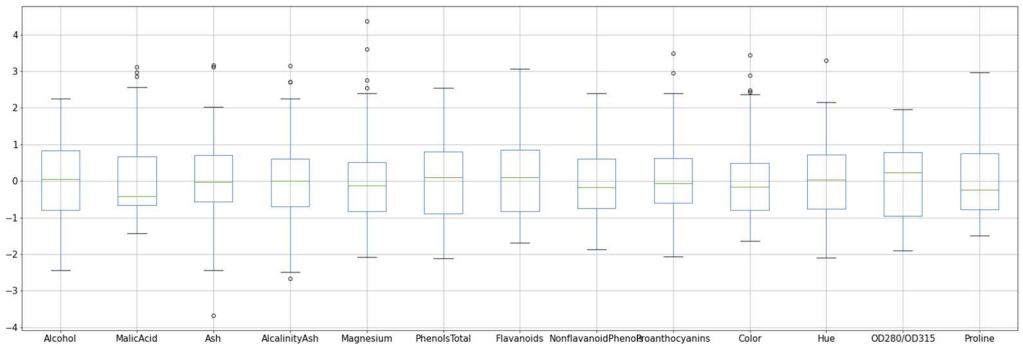


Figura 3. Diagrama de caixas dos dados de cada atributo/feature do dataset "Wine" normalizados com média 0 e desvio padrão 1.

Na figura 4 estão representados os diagramas de caixas de cada atributo agrupados por classe, o que permite ver as diferenças de cada atributo por cada tipo de vinho e aferir quais atributos permitirão distingui-los. Por exemplo, no caso dos Fenóis e Flavanóides a classe 1 e 3 são bastante distinguíveis, uma vez que as suas distribuições não se cruzam. Já no caso do atributo Álcool, a distribuição da classe 2 distingue-se mais das distribuições das restantes classes.

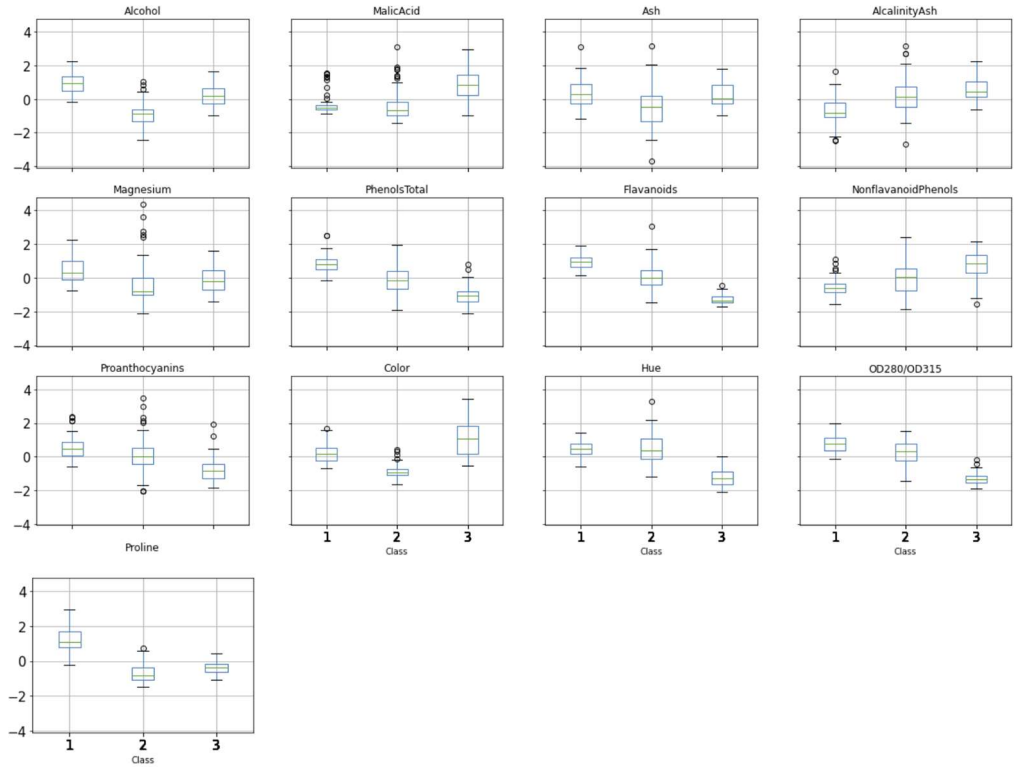


Figura 4. Diagrama de caixas dos dados de cada atributo/feature do dataset "Wine" normalizados com média 0 e desvio padrão 1, agrupados pelas três classes de cultivo.

4.2. Resultados e Discussão da Classificação

Como descrito na secção 3, o conjunto de dados foi dividido em conjunto de treino e teste, o que resultou em 133 observações para treino e 45 observações para teste. O conjunto de treino foi utilizado para treinar e otimizar os parâmetros do modelo, enquanto que o desempenho do modelo foi avaliado no conjunto de teste.

Nas tabelas 2 e 3 são apresentados os resultados de classificação do conjunto de teste do dataset "Wine" correspondentes à avaliação das três classes, que correspondem a três cultivos. Os melhores resultados de classificação foram obtidos com as Florestas Aleatórias e as Redes Neuronais, com exatidão de 100%. Nos restantes classificadores, existem diferenças entre a exatidão e a exatidão ponderada, resultantes da proporção entre classes do dataset. No entanto, o coeficiente de correlação Matthews mostra que a classificação é satisfatória, com valores acima dos 0,9.

Tabela 2. Resultados de classificação do dataset "Wine" (parte 1).

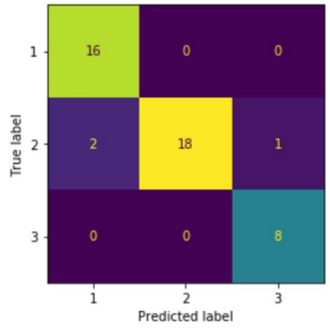
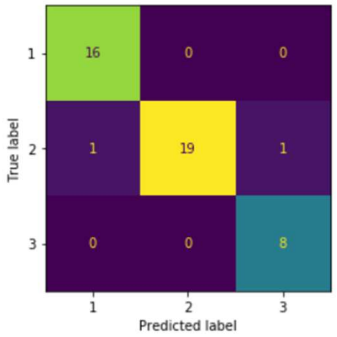
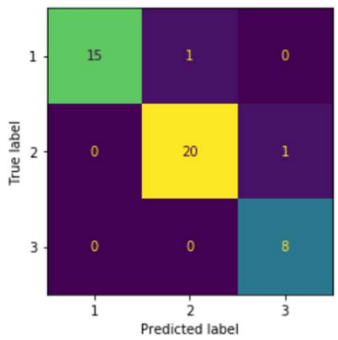
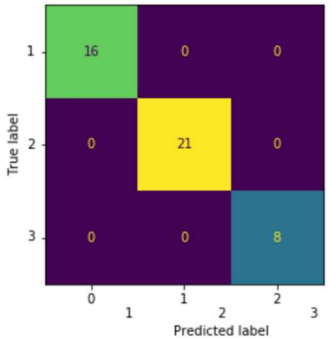
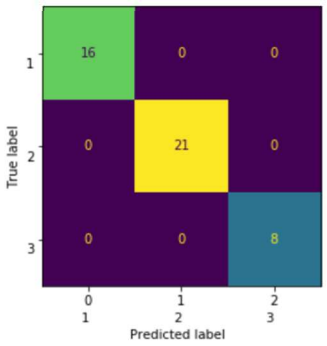
Classificadores	Exatidão	Exatidão Ponderada	Coeficiente de correlação Matthews	Matriz Confusão
Naïve Bayes	0,9333	0,9523	0,9000	
K-Vizinhos Mais Próximos	0,9555	0,9683	0,9320	
Árvores de Decisão	0,9556	0,9633	0,9300	
Florestas Aleatórias	1,000	1,000	1,000	

Tabela 3. Resultados de classificação do dataset "Wine" (parte 2).

Classificadores	Exatidão	Exatidão Ponderada	Coeficiente de correlação Matthews	Matriz Confusão
Redes Neuronais	1,000	1,000	1,000	

A representação gráfica da árvore de decisão obtida para este dataset foi obtida com o bloco de código 5 e encontra-se apresentada na figura 5. Apenas os atributos Flavanóides, Cor e Prolina são utilizados para classificar os dados, o que é uma indicação de que estes serão os atributos com maior importância na classificação. O atributo Flavanóides já tinha sido mencionado na secção 4.1 durante a análise dos diagramas de caixas, pois mostrava uma diferença maior na distribuição agrupada por classes. Seguindo esta árvore, podem ser criadas 5 regras que permitam a uma pessoa ou a um sistema classificar um novo conjunto de dados de análises químicas de vinhos:

- 1º Regra: Se Flavanóides <= 1,575 e Cor <= 3,825 então o vinho é do tipo de cultivo (classe) 2.
- 2º Regra: Se Flavanóides <= 1,575 e Cor > 3,825 então o vinho é do tipo de cultivo (classe) 3.
- 3º Regra: Se Flavanóides > 1,575 e Prolina <= 724,5 então o vinho é do tipo de cultivo (classe) 2.
- 4º Regra: Se Flavanóides > 1,575 e Prolina > 724,5 e Cor <= 3,46 então o vinho é do tipo de cultivo (classe) 2.
- 5º Regra: Se Flavanóides > 1,575 e Prolina > 724,5 e Cor > 3,46 então o vinho é do tipo de cultivo (classe) 1.

```
1.  fig = clf.best_estimator_.fit(x,y)
2.  plt.figure(figsize=(15,12))
3.  tree.plot_tree(fig, fontsize=10, feature_names=x.columns.values,
    class_names=["1","2","3"])
4.  plt.show()
```

Bloco de código 5. Criação da figura com a árvore de decisão obtida (“clf.best_estimator_”).

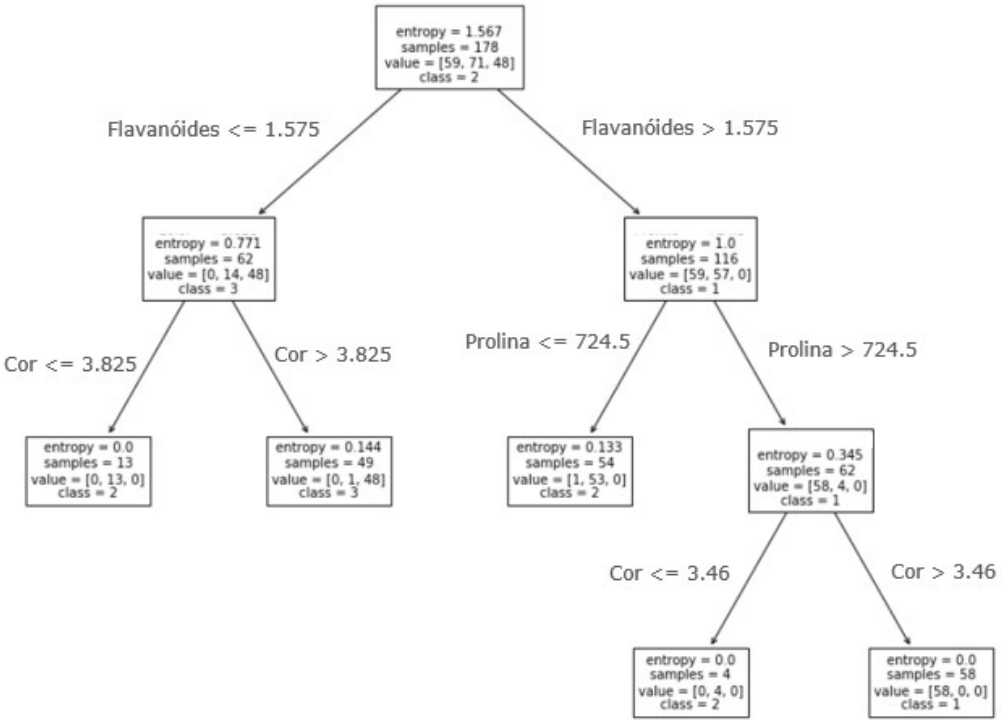


Figura 5. Representação gráfica da árvore de decisão obtida com o dataset “Wine”.

O dataset “Wine” tem três classes e por isso, para calcular a curva ROC, é necessário fazer classificação aos pares. Nesse sentido foi utilizada a função OneVsRestClassifier() que permite classificar os dados num método “um-contra-todos”, em que uma classe é considerada como condição positiva e as restantes como condição negativa. Assim, os resultados do dataset “Wine” tem três curvas ROC por classificador, correspondente a cada uma das iterações do método “um-contra-todos”. O bloco de código 6 apresenta um exemplo deste método utilizando o algoritmo K-Vizinhos Mais Próximos.

Nas tabelas 4 e 5 estão apresentadas as curvas ROC de cada classificador na metodologia “um-contra-todos” e a respetiva área debaixo da curva. Os classificadores Naïve Bayes e Florestas Aleatórias são os que têm melhores resultados da curva ROC. Estes resultados mostram que a classificação “um-contra-todos” obtém melhores resultados do que a classificação em que as três classes são utilizadas.

```
 # k-Vizinhos Mais Próximos "um-contra-todos"
parameters = {
              'estimator__n_neighbors':np.arange(1,10),
              'estimator__metric':["euclidean","manhattan","chebyshev"]
             }
clf_temp = OneVsRestClassifier(KNeighborsClassifier())
# Otimização de parâmetros Validação Cruzada por Camadas Estratificadas
clf = GridSearchCV(clf_temp, parameters, cv=5)
clf.fit(X_train, y_train)
y_pred_test_KNN = clf.best_estimator_.predict_proba(X_test)
# Calcular curva ROC curve e área da curva para cada classe
fpr = dict()
tpr = dict()
roc_auc = dict()
y_test_new = label_binarize(y_test, classes=[1, 2, 3])
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_new[:,i], y_pred_test_prob_KNN[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
plt.figure()
plt.plot(fpr[0], tpr[0],
         lw=2, label='Class 1 (area = %0.2f)' % roc_auc[0])
plt.plot(fpr[1], tpr[1],
         lw=2, label='Class 2 (area = %0.2f)' % roc_auc[1])
plt.plot(fpr[2], tpr[2],
         lw=2, label='Class 3 (area = %0.2f)' % roc_auc[2])
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc="lower right")
plt.show()
```

Bloco de código 6. Exemplo de classificação “um-contra-todos” com o algoritmo K-Vizinhos Mais Próximos para cálculo da curva ROC para o dataset “Wine”.

Tabela 4. Resultados da curva ROC do dataset "Wine" (parte 1).

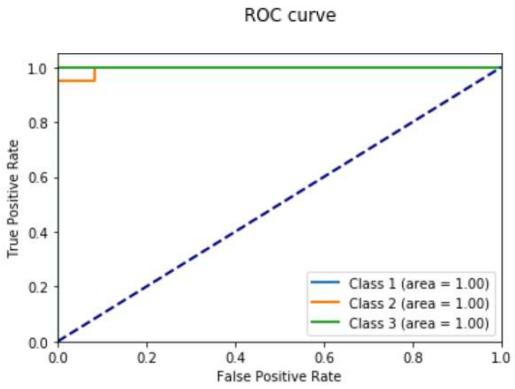
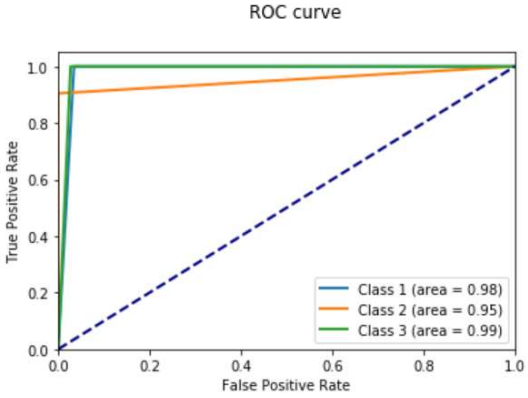
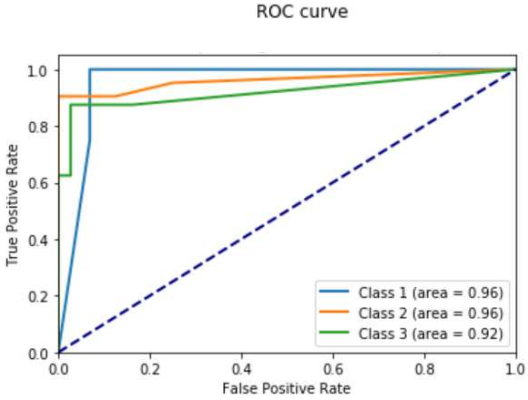
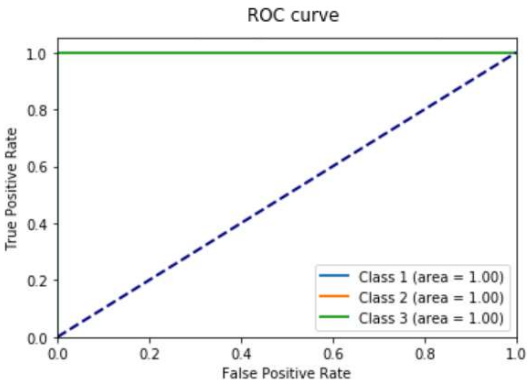
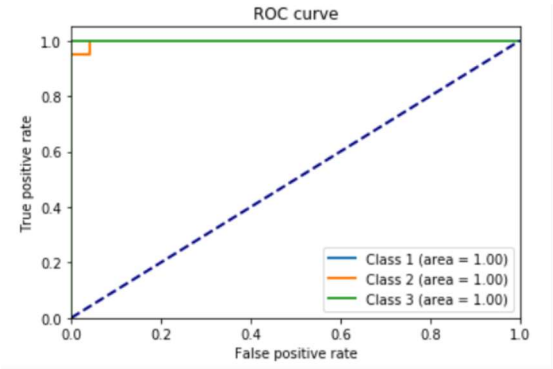
Classificadores	**Curva ROC e respetiva área (AUC)**
Naïve Bayes	
K-Vizinhos Mais Próximos	
Árvores de Decisão	

Tabela 5. Resultados da curva ROC do dataset "Wine" (parte 2).

Classificadores	Curva ROC e respetiva área (AUC)
Florestas Aleatórias	
Redes Neuronais	

Com o algoritmo das Florestas Aleatórias conseguimos analisar a importância dos atributos para a classificação. Na figura 6 estão representadas as importâncias de cada um dos atributos, obtidas utilizando o Bloco de código 7.

```
importances = clf.best_estimator_.feature_importances_
std = np.std([tree.feature_importances_ for tree in
clf.best_estimator_.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

# Plot the impurity-based feature importances of the forest
plt.figure()
plt.title("Feature importances")
plt.bar(range(x.shape[1]), importances[indices],
        color="r", yerr=std[indices], align="center")
plt.xticks(range(x.shape[1]), indices)
plt.xlim([-1, x.shape[1]])
plt.show()
```

Bloco de código 7. Estimativa da importância dos atributos a partir do classificador Florestas Aleatórias e criação de gráfico.

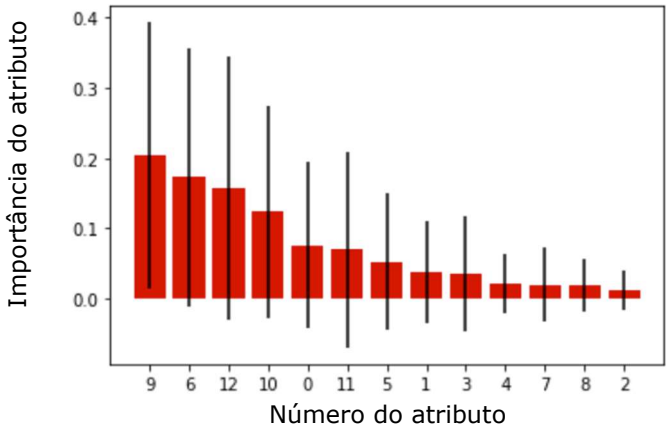


Figura 6. Gráfico da importância dos atributos no dataset "Wine". Os números no eixo horizontal correspondem aos seguintes atributos: 9=Color; 6=Flavanoids; 12=Proline; 10=Hue; 0=Alcohol; 11=OD280/OD315; 5=PhenolsTotal; 1=MalicAcid; 3=AlcalinityAsh; 4=Magnesium; 7=NonflavanoidPhenols; 8=Proanthocyanins 2=Ash.

O desempenho dos classificadores tendo em conta os atributos com maior importância foi avaliado, selecionando os atributos a estudar como apresentado no Bloco de código 8.

```
1.  x = dados.iloc[:,1:]
2.  std_scaler = StandardScaler()
3.  x = pd.DataFrame(std_scaler.fit_transform(x), columns=x.columns)
4.  x = x.iloc[:,[9,6,12,10]]
```

Bloco de código 8. Exemplo de código para selecionar atributos 9, 6, 12 e 10 que correspondem aos atributos de maior importância do dataset "Wine".

Na tabela 6 estão representados os resultados para estes diferentes testes. Como o número de dados em cada uma das classes é diferente, a exatidão ponderada foi a métrica utilizada para apresentar estes resultados. À exceção das árvores de decisão, quando se utilizam menos atributos para a classificação, o desempenho dos classificadores piora, apesar destes atributos serem aqueles com maior importância para a classificação (como observado na figura 6). O desempenho das árvores de decisão quando apenas os 3 atributos com maior importância são utilizados é idêntico ao desempenho das Florestas Aleatórias e Redes Neuronais.

Tabela 6. Resultados de exatidão ponderada para vários testes com diferentes atributos do dataset "Wine".

	Todos os atributos 9=Color 6=Flavanoids 12=Proline 10=Hue 0=Alcohol 11=OD280/OD315 5=PhenolsTotal 1=MalicAcid 3=AlcalinityAsh 4=Magnesium 7=NonflavanoidPhenols 8=Proanthocyanins 2=Ash	**4 atributos de maior importância** 9=Color 6=Flavanoids 12=Proline 10=Hue	**3 atributos de maior importância** 9=Color 6=Flavanoids 12=Proline	**2 atributos de maior importância** 9=Color 6=Flavanoids	**1 atributo de maior importância** 9=Color
Naïve Bayes	0,952	0,937	0,916	0,863	0,754
K-Vizinhos Mais Próximos	0,968	0,952	0,937	0,889	0,717
Árvores de Decisão	0,963	0,968	0,984	0,905	0,696
Florestas Aleatórias	0,984	0,984	0,895	0,884	0,696
Redes Neuronais	0,984	0,958	0,911	0,863	0,696

4.3. Conclusão

Os atributos mais representativos do dataset "Wine" são a intensidade da Cor, a quantidade de Flavanóides e a quantidade de Prolina. A árvore de decisão obtida (figura 5) permite retirar regras que podem ser utilizadas por uma pessoa ou um sistema para avaliar a que tipo de cultivo pertence um determinado vinho, utilizando apenas estes três atributos. Neste caso, significa que os três cultivos utilizados para esta base de dados se distinguem pela composição química de quantidade de Flavanóides e a Prolina e a intensidade da Cor. A partir da árvore de decisão é possível tirar as seguintes regras gerais:

- **Vinho do tipo de cultivo 1:** se a quantidade de flavanóides > 1,575, quantidade de prolina > 724,5 e intensidade de cor > 3,46.
- **Vinho do tipo de cultivo 3:** se a quantidade de flavanóides <= 1,575 e intensidade de cor > 3,825.

- **Vinho do tipo de cultivo 2:** caso contrário, ou seja, se a quantidade de flavanóides <= 1,575 e intensidade de cor <= 3,825 **ou** se a quantidade de flavanóides > 1,575 e a quantidade de prolina <= 724,5 **ou** se a quantidade de flavanóides > 1,575, a quantidade de prolina > 724,5 e intensidade de cor <= 3,46.

No entanto verificou-se que a utilização de todos os atributos para a classificação, melhora o desempenho dos algoritmos, permitindo uma exatidão de 100% e exatidão ponderada de 98,4% em alguns destes. Nestes casos, o modelo criado em Python ou noutra linguagem teria de ser utilizado visto que o modelo não é facilmente reprodutível graficamente. Os resultados estão dentro da mesma gama de valores dos reportados por outros autores (secção 2). Tanto neste estudo como nos estudos reportados, os melhores classificadores são as florestas aleatórias e as redes neuronais.

Estes resultados permitem concluir que os tipos de cultivo dos vinhos são distinguíveis a partir da sua análise química. Para utilizar um destes modelos perante um novo vinho, o utilizador teria que fazer o mesmo tipo de análise química apresentado neste dataset, ou apenas utilizando os três atributos mencionados acima e normalizar os seus valores tendo em conta os limites utilizados na criação deste modelo.

5. Dataset dos Indígenas Pima

Nesta secção será apresentada a descrição dos atributos do dataset, a análise estatística descritiva, resultados e respetiva discussão da classificação do dataset dos Indígenas Pima.

5.1. Análise dos Dados

A tabela 7 apresenta a descrição dos atributos do dataset “PimaIndians”. Existem 8 atributos que serão utilizados para classificar a classe do tipo de ausência/presença de diabetes, através de codificação binaria (0/1). Existem 768 observações no total, 500 observações da classe 0 (ausência de diabetes) e 268 da classe 1 (presença de diabetes).

No diagrama de caixas dos dados originais de cada atributo/feature do dataset “PimaIndians” verificou-se que o atributo SerumInsulin tem uma amplitude maior que os restantes. Mais uma vez, é feita a normalização dos atributos, o que resulta no gráfico da figura 7. Neste caso, as diferenças também não são significativas.

Tabela 7. Descrição dos atributos do dataset "PimaIndians".

#	Nome de atributo	Tipo	Descrição
1	**Id**	Uid	Identificador único do registo
2	**NuPregnancy**	Int	Número de gravidezes
3	**Glucose**	Int	Concentração plasmática de glicose por 2 horas em teste oral de tolerância à glicose
4	**DiastolicBP**	Int	Pressão arterial diastólica (mm Hg)
5	**TricepsSkinFoldThickness**	Int	Espessura da dobra da pele do tríceps (mm)
6	**SerumInsulin**	Int	Nível de Insulina em soro de 2 horas (mu U/ml)
7	**BodyMassIndex**	double	Índice de massa corporal (peso em kg / (altura em m)^2)
8	**DiabetesPedigree**	double	Função de pedigree de diabetes (probabilidade de desenvolver diabetes com base no histórico familiar)
9	**Age**	Int	Idade (em anos)
10	**Class**	Int	Ausência/presença de diabetes 0: ausência de diabetes 1: presença de diabetes

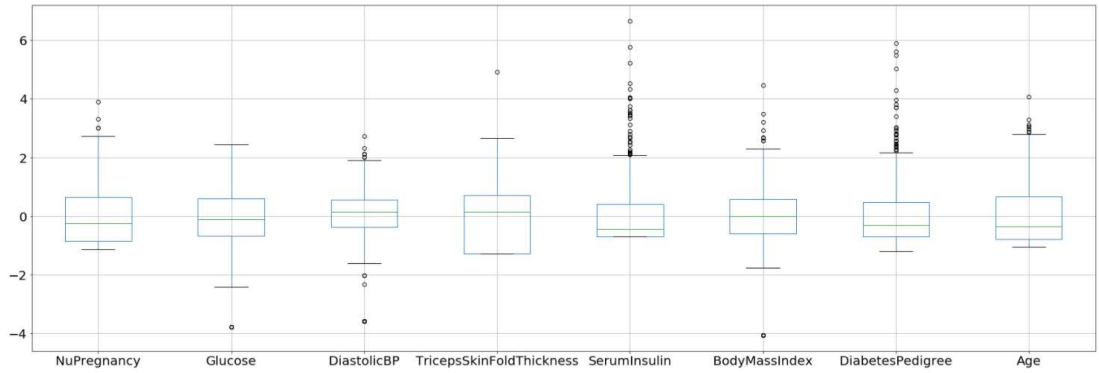


Figura 7. Diagrama de caixas dos dados de cada atributo/feature do dataset "PimaIndians" normalizados com média 0 e desvio padrão 1.

Na figura 8 estão representados os diagramas de caixa dos atributos agrupados por classe. Ao contrário do que acontecia no dataset "Wine", neste caso as distribuições de cada atributo não são visualmente diferenciáveis. Por exemplo, no caso do número de gravidezes, a amplitude de valores é maior para a classe 1 (com diabetes), no entanto as distribuições não apresentam muitas diferenças entre si. Isto explica o facto dos resultados apresentados pelos outros autores (mencionados na secção 2) mostrarem que a classificação do dataset "PimaIndians" tem piores resultados de classificação que o dataset "Wine".

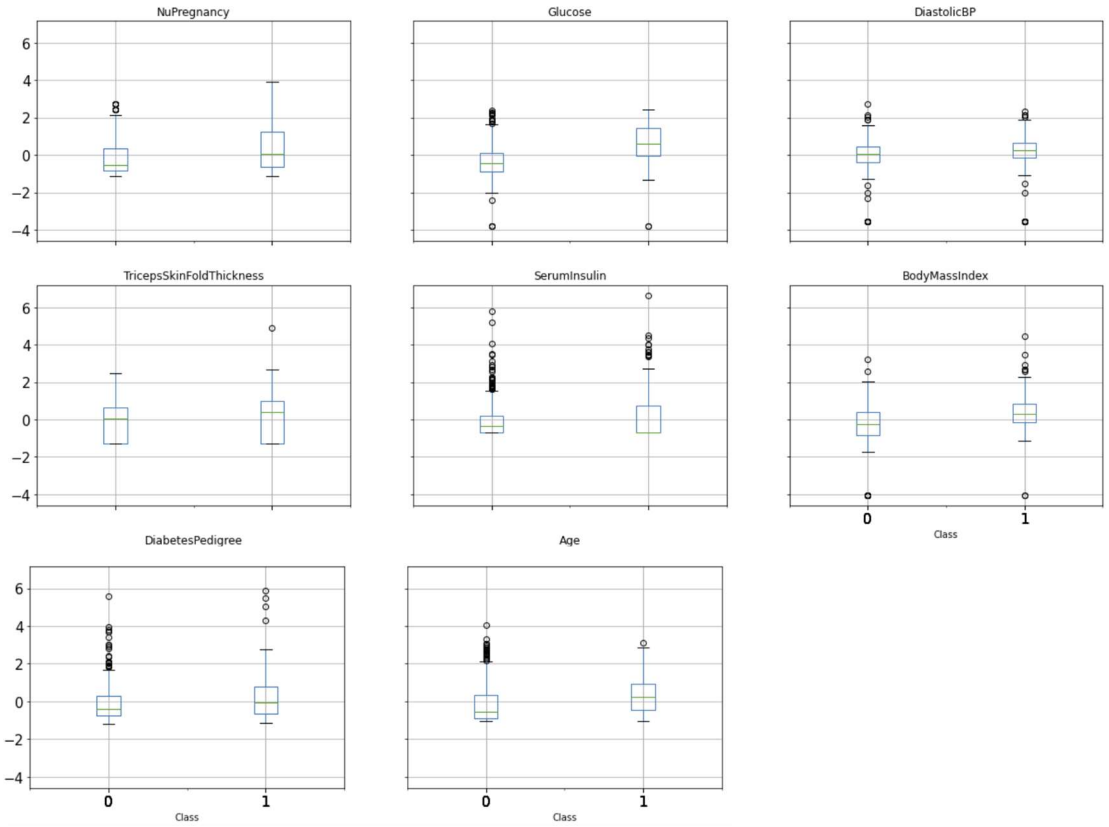


Figura 8. Diagrama de caixas dos dados de cada atributo/feature do dataset "PimaIndians" normalizados com média 0 e desvio padrão 1, agrupados pelas duas classes (ausência/presença de diabetes).

5.2. Resultados e Discussão da Classificação

Como descrito na secção 3, o conjunto de dados foi dividido em conjunto de treino e teste, o que resultou em 576 observações para treino e 192 observações para teste. O conjunto de treino foi utilizado para treinar e otimizar os parâmetros do modelo, enquanto que o desempenho do modelo foi avaliado no conjunto de teste.

Nas tabelas 8 e 9 são apresentados os resultados de classificação do conjunto de teste do dataset "PimaIndians" correspondentes à avaliação das duas classes (ausência/presença de diabetes). Os melhores resultados de classificação foram obtidos com as Redes Neuronais, não ultrapassando os 80,2% de exatidão. Em qualquer um dos classificadores, existem mais Falsos Negativos do que Falsos Positivos. Este resultado pode ser afetado pelo facto de existirem mais dados da condição negativa do que da condição positiva. Tal como no caso do dataset "Wine", existem diferenças entre a exatidão e a exatidão ponderada, resultantes da proporção entre classes do dataset. Neste caso, o coeficiente de correlação Matthews mostra que a classificação é pior, com valores de cerca de 0.4. No entanto, mostra que os classificadores têm melhor desempenho do que a sorte.

Tabela 8. Resultados de classificação do dataset "PimaIndians" (parte 1).

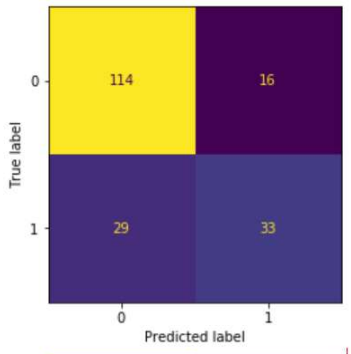
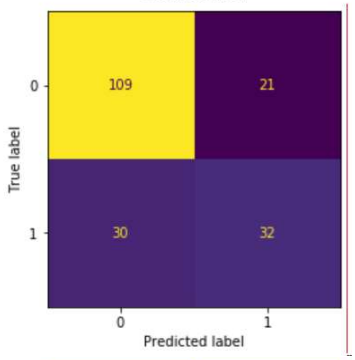
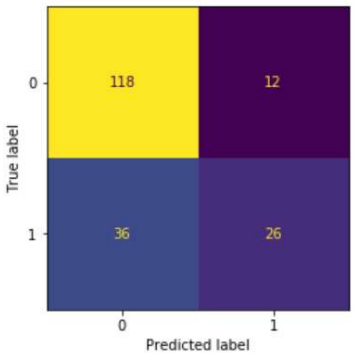
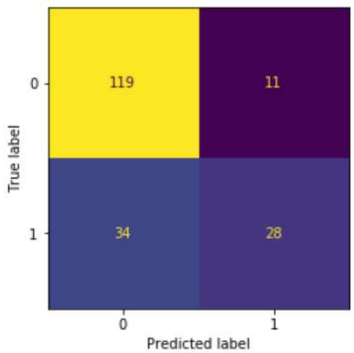
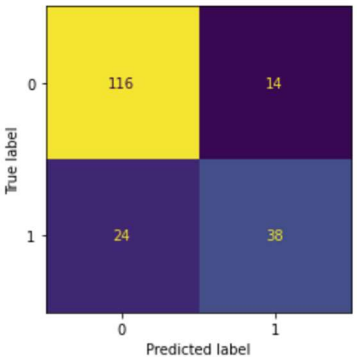
Classificadores	Exatidão	Exatidão Ponderada	Coeficiente de correlação Matthews	Matriz Confusão
Naïve Bayes	0,7656	0,7046	0,4389	
K-Vizinhos Mais Próximos	0,7344	0,6773	0,3709	
Árvores de Decisão	0,7500	0,6635	0,3838	
Florestas Aleatórias	0,7396	0,6896	0,3901	

Tabela 9. Resultados de classificação do dataset "PimaIndians" (parte 2).

Classificadores	Exatidão	Exatidão Ponderada	Coeficiente de correlação Matthews	Matriz Confusão
Redes Neuronais	0,8020	0,7530	0,5320	

A representação gráfica da árvore de decisão obtida está representada na figura 9, tendo sido obtida pelo Bloco de código 9. A partir desta árvore é possível verificar que apenas os atributos Glucose, Idade e Índice de Massa Corporal são utilizados para classificar os dados. Esta é uma indicação para a importância da avaliação da importância dos atributos. Seguindo esta árvore, podem ser criadas 5 regras que permitam a uma pessoa ou a um sistema classificar um novo conjunto de dados de fatores de risco de uma mulher ter ou não diabetes:

- 1º Regra: Se Glucose <= 127,5 e Idade <= 28,5 então não tem diabetes (classe 0).
- 2º Regra: Se Glucose <= 127,5, Idade > 28,5 e Índice de Massa Corporal <= 26,35 então não tem diabetes (classe 0).
- 3º Regra: Se Glucose <= 127,5, Idade > 28,5 e Índice de Massa Corporal > 26,35 então não tem diabetes (classe 0).
- 4º Regra: Se Glucose > 127,5 e Índice de Massa Corporal <= 29,95 então não tem diabetes (classe 0).
- 5º Regra: Se Glucose > 127,5 e Índice de Massa Corporal > 29,95 então tem diabetes (classe 1).

Ou seja, se a 5ªregra não se verificar, então, segundo estes três atributos, a mulher não terá diabetes.

```
1. fig = clf.best_estimator_.fit(x,y)
2. plt.figure(figsize=(15,12))
3. tree.plot_tree(fig, fontsize=10, feature_names=x.columns.values, class_names=["0","1"])
4. plt.show()
```

Bloco de código 9. Criação da figura com a árvore de decisão obtida ("clf.best_estimator_").

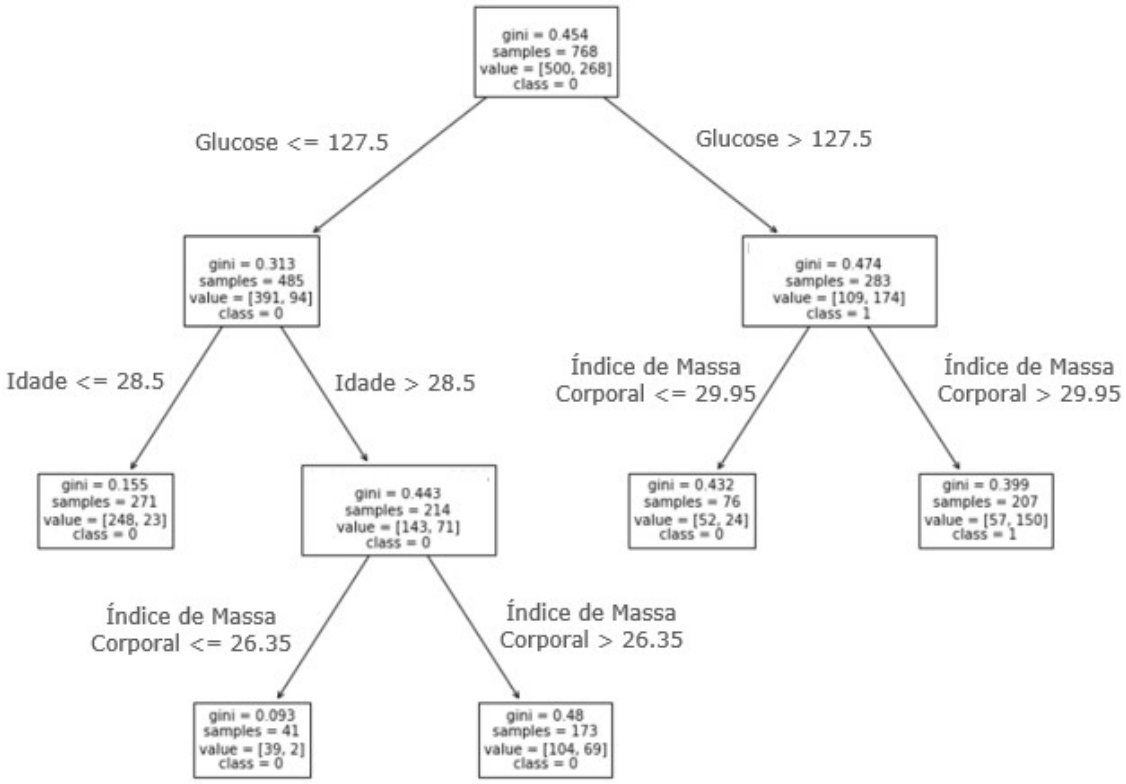


Figura 9. Representação gráfica da árvore de decisão obtida para o dataset "PimaIndians".

O dataset "PimaIndians" tem apenas duas classes e, portanto, a análise da curva ROC é direta, tal como é apresentado no bloco de código 10.

```
1.  plt.figure(1)
2.  plt.plot([0, 1], [0, 1], 'k--')
3.  fpr, tpr, _ = roc_curve(y_test, y_pred_test_prob_NB[:,1])
4.  roc_auc = auc(fpr, tpr)
5.  plt.plot(fpr, tpr, label='NB (area = %0.2f)' % roc_auc)
6.  fpr, tpr, _ = roc_curve(y_test, y_pred_test_prob_KNN[:,1])
7.  roc_auc = auc(fpr, tpr)
8.  plt.plot(fpr, tpr, label='KNN (area = %0.2f)' % roc_auc)
9.  fpr, tpr, _ = roc_curve(y_test, y_pred_test_prob_DT[:,1])
10. roc_auc = auc(fpr, tpr)
11. plt.plot(fpr, tpr, label='DT (area = %0.2f)' % roc_auc)
12. fpr, tpr, _ = roc_curve(y_test, y_pred_test_prob_RF[:,1])
13. roc_auc = auc(fpr, tpr)
```

```
14. plt.plot(fpr, tpr, label='RF (area = %0.2f)' % roc_auc)
15. fpr, tpr, _ = roc_curve(y_test, y_pred_test_prob_NN[:,1])
16. roc_auc = auc(fpr, tpr)
17. plt.plot(fpr, tpr, label='NN (area = %0.2f)' % roc_auc)
18. plt.xlabel('False positive rate')
19. plt.ylabel('True positive rate')
20. plt.title('ROC curve')
21. plt.legend(loc='best')
22. plt.show()
```

Bloco de código 10. Cálculo da curva ROC para o dataset "PimaIndians".

Na figura 10 estão apresentadas as curvas ROC de cada classificador e a respetiva área debaixo da curva (AUC). Os classificadores Naïve Bayes, Florestas Aleatórias e Redes Neuronais representados a azul, vermelho e roxo respetivamente, são os que têm melhores resultados da curva ROC, com maior AUC e que se aproximam do canto superior esquerdo (Taxa de Verdadeiros Positivos = 1 e Taxa de Falsos Positivos = 0).

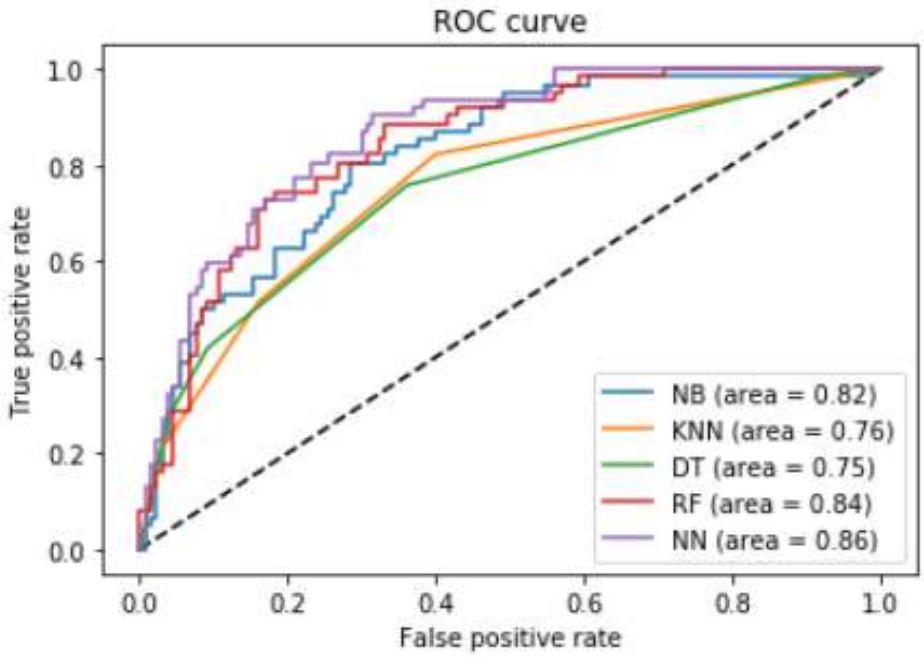


Figura 10. Curva ROC do dataset "PimaIndians". NB = Naïve Bayes; KNN = K-Vizinhos Mais Próximos; DT = Árvores de Decisão; RF = Florestas Aleatórias; NN = Redes Neuronais.

Também para este dataset, com o algoritmo das Florestas Aleatórias conseguimos analisar a importância dos atributos para a classificação, utilizando o Bloco de código 11. Na figura 11 estão representadas as importâncias de cada um dos atributos.

```
importances = clf.best_estimator_.feature_importances_
std = np.std([tree.feature_importances_ for tree in
clf.best_estimator_.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

# Plot the impurity-based feature importances of the forest
plt.figure()
plt.title("Feature importances")
plt.bar(range(x.shape[1]), importances[indices],
        color="r", yerr=std[indices], align="center")
plt.xticks(range(x.shape[1]), indices)
plt.xlim([-1, x.shape[1]])
plt.show()
```

Bloco de código 11. Estimativa de importância dos atributos a partir do classificador Florestas Aleatórias e criação de gráfico.

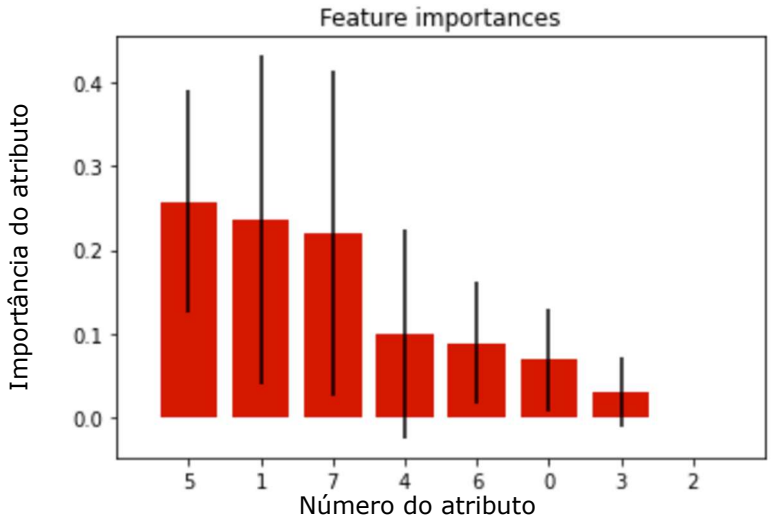


Figura 11. Gráfico de importância dos atributos do dataset "PimaIndians". Os números no eixo horizontal correspondem aos seguintes atributos: 5=BodyMasssIndex; 1=Glucose; 7=Age; 4=Seruminsulin; 6=DiabetesPedigree; 0=NuPregnacy; 3=TricepsSkinFoldThickness; 2=DiastolicBP.

O desempenho dos classificadores tendo em conta os atributos com maior importância foi avaliado, selecionando os atributos a estudar como apresentado no Bloco de código 12.

```
x = dados.iloc[:,0:-1]
std_scaler = StandardScaler()
x = pd.DataFrame(std_scaler.fit_transform(x), columns=x.columns)
x = x.iloc[:,[5,1,7,4]]
```

Bloco de código 12. Exemplo de código para selecionar atributos 5, 1, 7 e 4 que correspondem aos atributos de maior importância do dataset "PimaIndians".

Na tabela 10 estão representados os resultados da métrica exatidão ponderada para testes com diferentes conjuntos de atributos. Tal como explicado no caso do dataset "Wine", a exatidão ponderada é também a métrica mais indicada uma vez que não existe o mesmo

número de dados para cada uma das classes deste dataset. No caso dos algoritmos Naïve Bayes, K-Vizinhos Mais Próximos e Árvores de Decisão, quando se utilizam menos atributos (especialmente apenas os 2 atributos de maior importância), o seu desempenho melhora, com um aumento de exatidão ponderada entre 0,8 e 4,1%. Nos restantes casos, quando se utilizam menos atributos o desempenho dos classificadores piora em comparação ao caso em que todos os atributos são utilizados, apesar destes atributos serem os atributos com maior importância para a classificação (como observado na figura 11). Neste dataset, os melhores resultados são obtidos com as Redes Neuronais e utilizando todos os atributos.

Tabela 10. Resultados de exatidão ponderada para vários testes com diferentes atributos do dataset "PimaIndians".

	Todos os atributos 5=BodyMassIndex 1=Glucose 7=Age 4=Serum Insulin 6=Diabetes Pedigree 0=NuPregnacy 3=TricepsSkin FoldThickness 2=DiastolicBP	**4 atributos de maior importância** 5=BodyMass Index 1=Glucose 7=Age 4=Serum Insulin	**3 atributos de maior importância** 5=BodyMass Index 1=Glucose 7=Age	**2 atributos de maior importância** 5=BodyMass Index 1=Glucose	**1 atributo de maior importância** 5=BodyMass Index
Naïve Bayes	0,705	0,696	0,700	0,724	0,575
K-Vizinhos Mais Próximos	0,677	0,704	0,708	0,685	0,579
Árvores de Decisão	0,664	0,655	0,663	0,705	0,512
Florestas Aleatórias	0,690	0,684	0,684	0,680	0,533
Redes Neuronais	0,753	0,704	0,708	0,705	0,567

5.3. Conclusão

Os atributos mais representativos do dataset PimaIndians são o Índice de Massa Corporal, a Glucose e a Idade. No entanto, na árvore de decisão obtida (figura 9) apenas dois deles são relevantes para a tomada de decisão. Esta árvore permite retirar regras que podem ser utilizadas por uma pessoa ou um sistema para avaliar se, perante estes três atributos uma mulher tem ou não diabetes. A partir da árvore de decisão é possível tirar as seguintes conclusões gerais:

- **Mulher com diabetes:** se a quantidade de glucose > 127,5 e o Índice de Massa Corporal > 29,95.
- **Mulher sem diabetes:** caso contrário, ou seja, se a quantidade de glucose <= 127,5 **ou** se a quantidade de glucose > 127,5 e o Índice de Massa Corporal <= 29,95.

No entanto verificou-se que o melhor resultado de classificação é obtido com as Redes Neuronais utilizando todos os atributos, atingindo 80,2% de exatidão e 75,3% de exatidão ponderada. Neste caso, à semelhança do que acontecia para o dataset "Wine", o modelo criado em Python ou noutra linguagem teria de ser utilizado visto que o modelo não é facilmente reprodutível graficamente. Também neste dataset, os resultados estão dentro da mesma gama de valores dos reportados por outros autores (secção 2). Os melhores classificadores foram também as Florestas Aleatórias e as Redes Neuronais, sendo que a Rede Neuronal criada neste trabalho teve valores de exatidão ligeiramente superiores aos reportados em (Kontkanen, 1999), possivelmente devido aos parâmetros utilizados.

Estes resultados permitem concluir que a quantidade de glucose e o índice de massa corporal são fatores determinantes para medir a probabilidade das mulheres indígenas Pima terem diabetes. É importante destacar que é necessário fazer um paralelo para o mundo ocidental, visto que este dataset foi criado por um grupo específico de mulheres há vários anos atrás. No entanto, ainda hoje se verifica que estes dois fatores são fatores importantes. A diabetes é definida pelo facto do pâncreas deixar de produzir insulina, a substância responsável por processar a glucose. A acumulação de glucose no sangue é um indicador da diabetes e o excesso de peso pode ser também um fator influenciador do desenvolvimento da doença. O modelo aqui criado é mais simples que o modelo criado para os vinhos. Para avaliar os fatores de risco para uma pessoa desenvolver diabetes, um utilizador teria de medir o peso e altura de forma a calcular o Índice de Massa Corporal e medir a quantidade de glucose no sangue. Os valores teriam também de ser normalizados tendo em conta os limites utilizados na criação deste modelo.

6. Conclusões

Neste trabalho foi desenvolvido um programa de raiz na linguagem Python capaz de analisar e proceder à classificação de dois datasets com características diferentes. O dataset "Wine" com três classes e dados de análises químicas de vinhos italianos e o dataset "PimaIndians" com duas classes e dados de fatores de risco de ter a doença diabetes.

Os resultados de exatidão do dataset “PimaIndians” são inferiores aos do dataset “Wine”. Uma possível explicação é a diferença pequena entre as distribuições dos atributos quando agrupados por classe no dataset “PimaIndians”.

O melhor classificador para o dataset “Wine” foi o classificador Florestas Aleatórias e as Redes Neuronais com valor de exatidão 100%, tal como representado pela curva ROC. O melhor classificador para o dataset “PimaIndians” foi também as Redes Neuronais, seguidas do classificador Naïve Bayes, com valor de exatidão 80,2% e 76,6%, respetivamente. O mesmo pode ser observado pela curva ROC.
Foi possível encontrar regras que permitam a um utilizador classificar o tipo de cultivo dos vinhos italianos ou se uma mulher tem diabetes, utilizando as árvores de decisão. Apesar dos valores de desempenho serem diferentes entre os datasets, não houve diferenças significativas entre os diferentes classificadores para os dois datasets.

No futuro, poderão testar-se novos algoritmos de classificação. No entanto, os resultados reportados por outros autores mostram que os valores de exatidão são semelhantes aos obtidos neste trabalho.

REFERÊNCIAS

Andy K. Devos., S. v. (2007). Chapter 11 - Classification of Brain Tumours by Pattern Recognition of Magnetic Resonance Imaging and Spectroscopic Data. Em A. C. Azzam F.G. Taktak (Ed.), *Outcome Prediction in Cancer* (pp. 285-318). Elsevier.

Banfield R. E. (2004). A comparison of ensemble creation techniques. *International Workshop on Multiple Classifier Systems* (pp. 223-232). Cagliari, Italy: Springer, Berlin, Heidelberg.

Brauer S. S. (1997). *Feature selection by means of a feature weighting approach.* Forschungsberichte Kunstliche Intelligenz, Institut fur Informatik, Technische Universitat Munchen.

Breiman L. (2001). Random Forests. *Machine Learning, 45*, 5-32.

Castillo G. (26 de Abril de 2021). *Lesson 3: Evaluation and Comparison of Supervised Learning Algorithms.* Obtido de Slideshare: https://pt.slideshare.net/gladysCJ/evaluation-and-comparison-of-supervised-learning-algorithms

Chih-Wei Hsu, &. C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks, 13*(2), 415-425.

Forina M., Leardi, R., C, A., & Lanteri, S. (1998). *PARVUS - An Extendible Package for Data Exploration, Classification and Correlation.* Genoa, Italy: Institute of Pharmaceutical and Food Analysis and Technologies.

Han J. P. (2011). *Data mining: Concepts and techniques (The Morgan Kaufmann Series in Data Management Systems).* Waltham, MA: Elsevier.

Jiang Y. &. (2004). Editing training data for kNN classifiers with neural network ensemble. *International symposium on neural networks* (pp. 356-361). Dalian, China: Springer, Berlin, Heidelberg.

Kontkanen P. L. (1999). Using Bayesian networks for visualizing highdimensional data. *Proceedings of Pre- and Post-processing in Machine Learning and Data Mining: Theoretical Aspects and Applications.* Chania, Greece.

Kulin M., Kazaz, T., De Poorter, E., & Moerman, I. (2021). A Survey on Machine Learning-Based Performance Improvement of Wireless Networks: PHY, MAC and Network Layer. *Electronics, 10*(3).

Liu F. T. (2005). Maximizing tree diversity by building complete-random decision trees. . *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 605-610). Hanoi, Vietnam: Springer, Berlin, Heidelberg.

National Institutes of Health. (3 de janeiro de 2020). *National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK)*. Obtido em 23 de abril de 2021, de National Institutes of Health: https://www.nih.gov/about-nih/what-we-do/nih-almanac/national-institute-diabetes-digestive-kidney-diseases-niddk

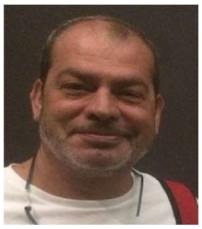
PeterH Bennett T. B. (1970). DIABETES MELLITUS IN AMERICAN (PIMA) INDIANS. *The Lancet, 298*(7716), 125-128.

R. Nisbet J. E. (2009). *Handbook of statistical analysis and data mining applications.* Boston, MA: Academic Press.

Sagi O. &. (2020). Explainable decision forest: Transforming a decision forest into an interpretable tree. *Information Fusion, 61*, 124-138.

Scikit-Learn. (janeiro de 2021). *Scikit-Learn - Machine Learning in Python*. Obtido em 27 de março de 2021, de Scikit-Learn - Machine Learning in Python: https://scikit-learn.org/stable/

Tamais A. L. (25 de agosto de 2019). *Modelos de Predição | Naive Bayes*. Obtido em 25 de abril de 2021, de Medium: https://medium.com/turing-talks/turing-talks-16-modelo-de-predi%C3%A7%C3%A3o-naive-bayes-6a3e744e7986



Francisco Passinho, é Licenciado em Engenharia Informática (2021) pela Universidade Aberta. Atualmente desempenha funções de administração, operação e manutenção de sistemas aplicacionais e administração e operação de redes informáticas num Instituto Público. Tem como áreas de interesse, a intersecção da Informática com a Inteligência Artificial, designadamente a área de "Data Mining".

Luís Cavique, Professor Auxiliar no Departamento de Ciências e Tecnologia (DCeT), Secção de Informática, Física e Tecnologia (SIFT). Licenciado em Engenharia Informática em 1988 pela FCT-UNL. Obteve o grau Mestre em Investigação Operacional e Eng. Sistemas pelo IST-UTL em 1994. Obteve o grau de Doutor em Eng. Sistemas pelo IST-UTL em 2002. Tem como áreas de interesse, a intersecção da Informática com a Engenharia de Sistemas designadamente a área de "Data Mining".

(esta página par está propositadamente em branco)