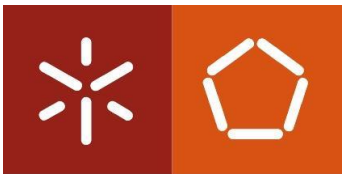

Universidade do Minho

Departamento de Informática

Trabalho Prático 2
Aprendizagem e Extração de Conhecimento

Bruno Martins (a80410) Catarina Machado (a81047)
Filipe Monteiro (a80229)

18 de Janeiro de 2020

Conteúdo

Resumo

No presente documento começamos por analisar, preparar e estruturar as *features* e *target variables* para o treino de modelos de Extração de Conhecimento, utilizando técnicas como *Feature Selection* e *Feature Extraction* e, em seguida, procedemos ao treino e validação de alguns modelos preditivos de *Machine Learning*, tais como Regressão Logística, KNN, SVM, Redes Neuronais, *Decision Trees* e *Random Forest*, sendo feita uma análise e otimização da performance dos modelos preditivos desenvolvidos.

1 Introdução

Os modelos preditivos de *Machine Learning* utilizam dados e algoritmos estatísticos para identificar a probabilidade de acontecimentos futuros a partir de dados históricos. A utilização destes modelos permite resolver problemas difíceis e descobrir novas oportunidades, sendo cada vez mais utilizados em diversas áreas como a deteção de fraudes, a otimização de campanhas de marketing, melhoria de operações e redução de riscos.

Uma das fases cruciais na resolução de problemas de classificação é a etapa de análise e tratamento do *dataset*. Desta forma, é importante visualizar a distribuição das *features* do conjunto de dados de modo a interpretar a sua relação com o *target variable*, percebendo assim quais são as features mais relevantes para um dado problema. Para um bom tratamento dos dados e, consequentemente, se obter uma melhor performance de classificação do modelo preditivo utilizado, é necessário a aplicação de técnicas de tratamento de dados como a verificação de *outliers*, a normalização de dados, a categorização de *features*, *feature extraction*, entre várias outras.

Ao longo deste processo de tratamento de dados, os diferentes modelos preditivos deverão ser treinados, percebendo através de métricas de avaliação de performance (como, por exemplo, a métrica de *score Accuracy*) se o modelo se encontra validado.

Através do presente relatório pretendemos demonstrar o resultado da investigação e aplicação de conhecimentos na implementação de um modelo de aprendizagem, focando em especial a parte da extração de conhecimento.

1.1 Caso de Estudo

O problema consiste na preparação e análise de um *dataset* relativo às condições de funcionários de uma empresa, como forma de prever a sua ausência. O conjunto de dados possui 740 casos de estudo com 20 *features* distintas (que podem ou não estar relacionados à ausência do trabalho) adquiridas de 36 funcionários diferentes.

As *features* do caso de estudo são as seguintes:

- ID – identificador único do funcionário;
- *Reason for absence* – número identificador associado à razão da ausência estruturadas pelo Código Internacional de Doenças (CID);
- *Month of absence* – mês do caso de estudo;
- *Day of the week* – dia da semana do caso de estudo (segunda-feira (2), terça-feira (3), quarta-feira (4), quinta-feira (5), sexta-feira (6));
- *Seasons* – estação do ano do caso de estudo;
- *Transportation expense* – gastos financeiros associados ao transporte do funcionário;
- *Distance from Residence to Work* – distancia entre o local de trabalho e a residência do funcionário (quilómetros);
- *Service time* – tempo de serviço anual que o funcionário apresenta na empresa; Age – idade do funcionário;

- *Work load Average/day* – carga de trabalho médio que o funcionário apresenta por dia;
- *Hit target* – carga de trabalho percentual concluído pelo funcionário;
- *Disciplinary failure* – define se o funcionário já apresenta pelo menos uma falha disciplinar dentro da empresa (sim=1; não=0);
- *Education* – grau de escolaridade do funcionário (ensino médio (1), licenciatura (2), pós-graduação (3), mestrado e doutorado (4));
- *Son* – número de filhos que o funcionário apresenta;
- *Pet* – número de animais de estimação que o funcionário apresenta;
- *Social drinker* – define se o funcionário é bebedor social (sim=1; não=0);
- *Social smoker* – define se o funcionário é fumador (sim=1; não=0);
- *Weight* – peso do funcionário;
- *Height* – altura do funcionário;
- *Body mass index* – índice de massa corporal do funcionário.

Através do conjunto de dados, o *target variable* a prever é apresentado pelo seguinte dado:

- *Absent* – apresenta se o funcionário irá ausentar ao trabalho (sim=1; não=0).

Para o desenvolvimento do modelo de classificação do problema em questão, foi utilizado o ambiente de desenvolvimento *Python/Sklearn*.

1.2 Estrutura do Relatório

O presente relatório está dividido em 4 diferentes capítulos.

No capítulo 1, **Introdução**, é feito um enquadramento e contextualização do trabalho prático e, em seguida, é feita uma descrição do problema a desenvolver, assim como o objetivo do projeto.

No capítulo 2, **Análise e Tratamento dos Dados**, é feita uma análise dos dados e é feita a descrição da abordagem utilizada, nomeadamente na construção de novos *datasets* com os dados devidamente tratados.

Em seguida, no capítulo 3, **Especificação e Treino dos Modelos** são apresentados os vários modelos preditivos utilizados, assim como as diferentes técnicas utilizadas com o intuito de melhorar a performance dos mesmos.

Por fim, no capítulo 4, **Conclusão**, termina-se o relatório com uma síntese do trabalho realizado.

2 Análise e Tratamento dos Dados

Como método de calcular o melhor modelo possível para a resolução do nosso problema, decidimos criar vários *datasets* (várias análises do *dataset*) e aplicar os diferentes modelos a estes de modo a perceber as suas relações e para encontrar aquele mais otimizado.

Após então da inspeção do documento contendo a informação sobre o significados das diferentes *features* do *dataset* fornecido juntamente com estes, começamos por detectar aquelas que, a nosso ver poderiam ter grande peso na previsão do modelo. Entre eles encontram-se:

- *Disciplinary failure:* assumimos que, pela lógica, se um indivíduo tivesse aviso, não iria faltar tanto;
- *Reason for absence:* as diferentes razões podem mostrar se o indivíduo faltou ou não;
- *Seasons*: existem alturas do ano com mais índice de faltas do que outros;
- *Son*: a existência de crianças na família pode provocar uma falta por parte do trabalhador;
- *Age*: quanto mais novo mais probabilidade tem de faltar (mentalidades ?);
- Entre outros.

Um aspeto importante deste *dataset* é a coluna *Disciplinary failure*. Em primeiro lugar, 93,3% dos casos tinham valor 0 e destes, 85% faltavam (*absence*=1), sendo que os restantes 6,6% dos casos tinham valor 1 e destes 100% faltavam. Devido à enorme discrepância, decidimos que seria melhor retirar esta coluna por favorecer, e muito, a possibilidade de faltar. Nesta figura e na próxima fica bastante bem evidenciado a questão da *feature* **Disciplinary failure** que detém 50% do peso de cálculo da nossa varíavel de decisão.

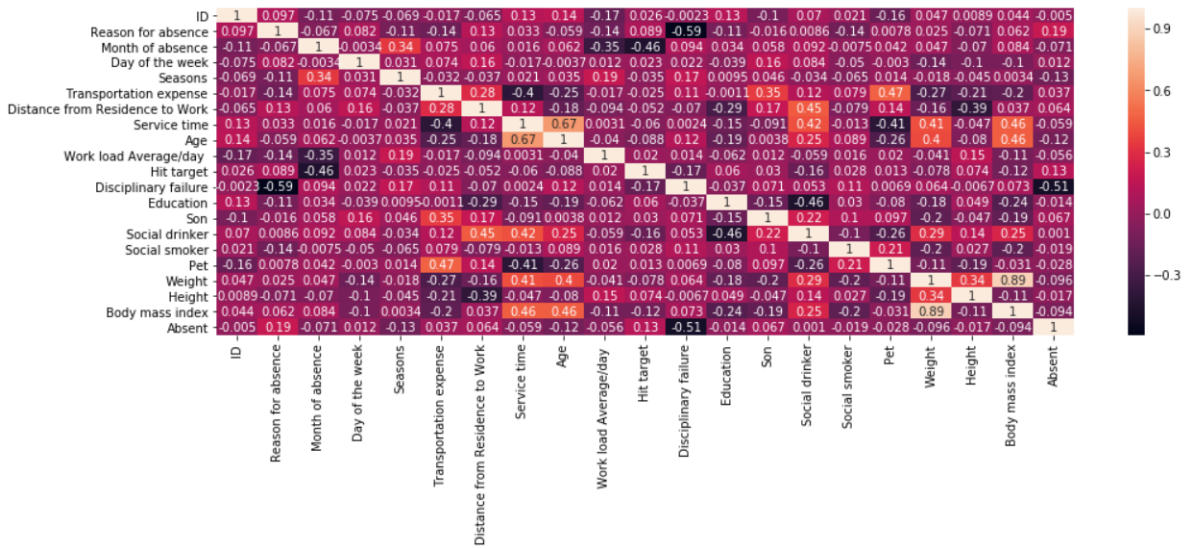

Figura 1: Correlações entre todas as *features* do *dataset*, calculadas através da funcção *corr* fornecida pelo *pandas*.

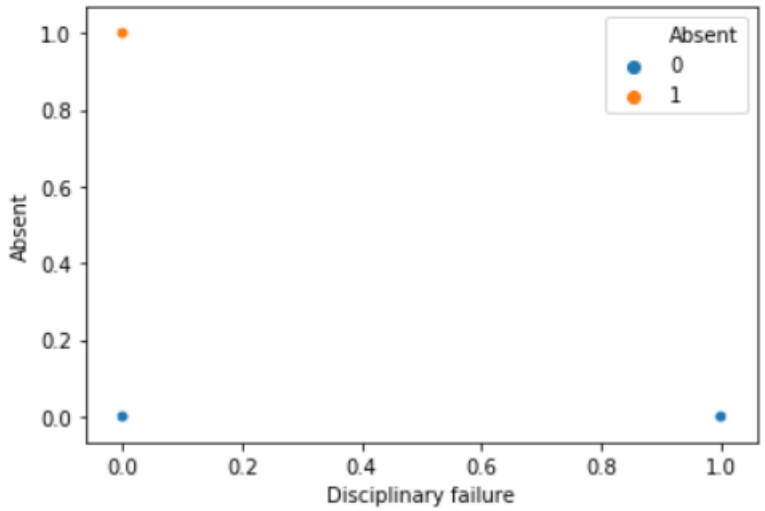

Figura 2: *Pairplot* entre **Disciplinary failure** e **Absent** - a nossa variável de decisão.

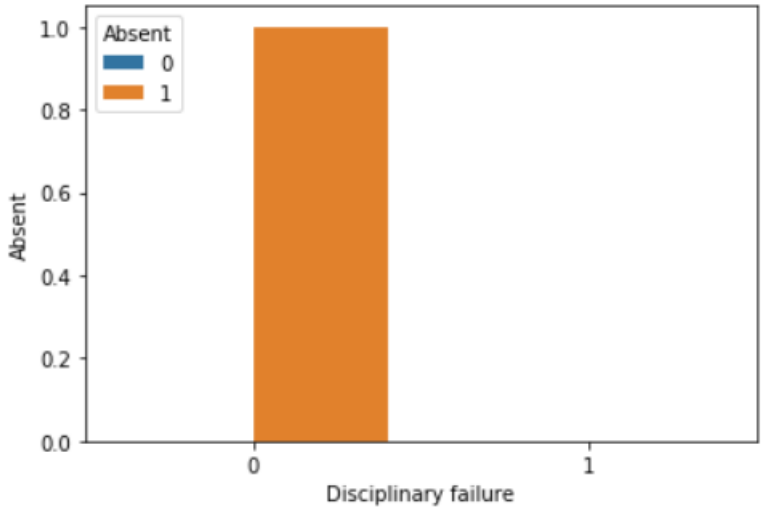

Figura 3: *Barplot* entre **Disciplinary failure** e **Absent** - a nossa variável de decisão.

Para além desta foi retirado o ID também de todos os *datasets* pois não era de todo relevante. Em relação a *outliers*, com a remoção de algumas das colunas e cortes dos dados, detetamos apenas uma entrada que achámos valer a pena:

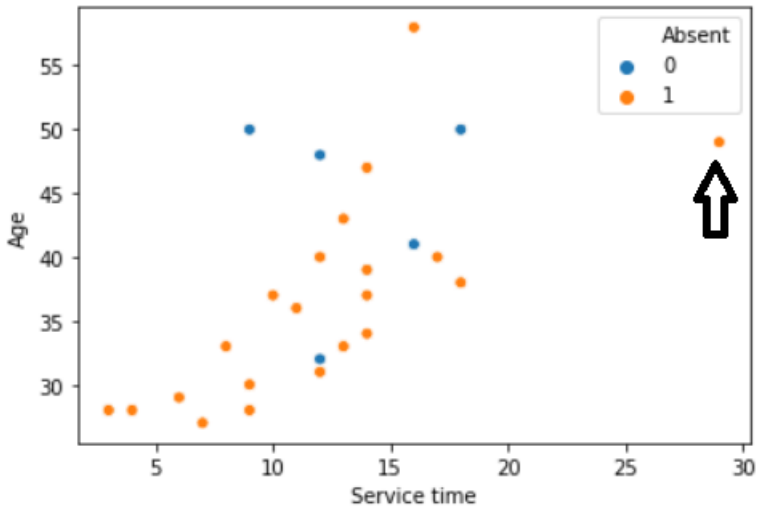

Figura 4: Entrada única que não parece seguir a curva desenhada pelo restos dos dados.

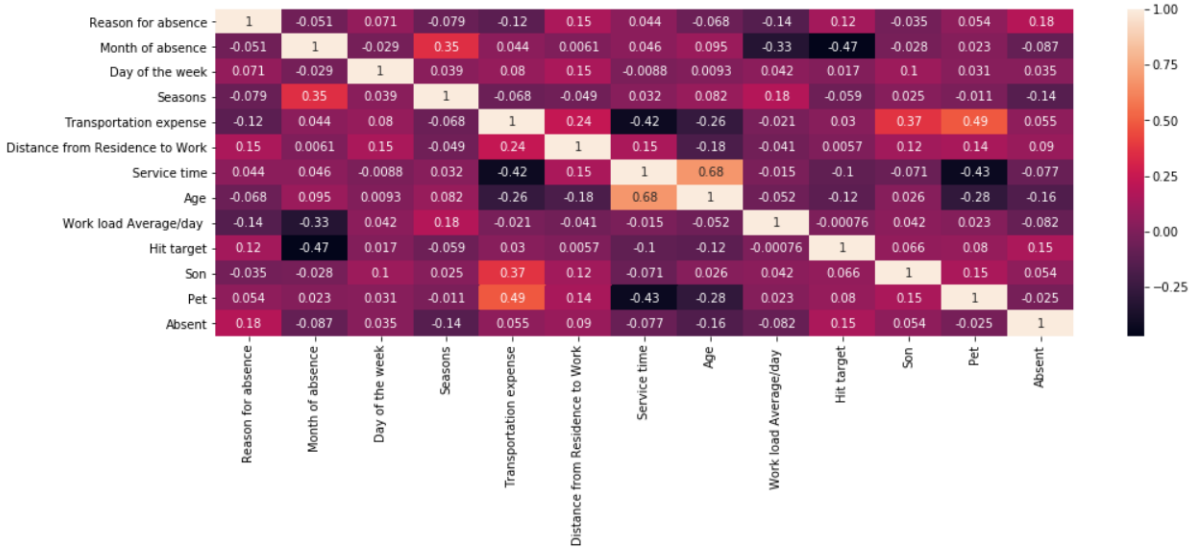

Figura 5: Correlações entre todas as *features* do *dataset* exceto algumas definidas para o primeiro *dataset*, calculadas através da funcção *corr* fornecida pelo *pandas*.

Por isso, o primeiro *dataset* foi criado usando aquele já alterado até agora mas ainda removendo todas as *features* que achávamos irrelevantes pela análise dos gráficos das relações entre estas, nomeadamente: **Weight, Height, Body mass index e Month**. *Month* foi retirado por uma questão de normalização dos valores, visto que na coluna *Seasons* já se encontram, em parte, algum conhecimento de *Month*.

O segundo *dataset* foi criado usando o *dataset* base e removendo *features* aconselhadas por um algoritmo de classificação que também evidencia as características mais importantes na sua classificação. Este algoritmo é a regressão logística implementada pelo *XGBoost*, um algoritmo que explora a otimização do gradiente fazendo com que o processamento possa ser distribuído e desta forma altamente eficiente. As que calculou foram: **Body mass index, Transportation expense, Service time, Pet, Son, Day of the week, Education, Weight, Height, Month of absence.**

Visto haver um grande desequilíbrio entre as duas classes da variável de decisão (79/21),

geramos 2 outros *datasets*, baseados no anteriormente descrito, onde fazíamos oversampling e undersampling deste, respetivamente, para obter um *ratio* de 50/50.

Os diferentes modelos usados então para resolver o problema, discutidos de seguida, foram:

- Regressão Logística Múltipla;
- *Support Vector Machine* (SVM);
- *Neural Networks*;
- *K-Nearest Neighbour* (KNN);
- *Decision Trees*;
- *Random Forest*.

3 Especificação e Treino dos Modelos

3.1 Regressão Logística

Para classificar os dados através deste método utilizamos uma biblioteca chamada *XGBoost* (*eXtreme Gradient Boost*) que nos permite obter resultados melhores que uma regressão mais básica. Este modelo foi aplicado primeiramente a todos os dados de treino exceptuando o ID e os outliers. Como resultado da aplicação deste método observamos uma accuracy de 75% mas uma grande ocorrência de falsos negativos e falsos positivos. Este modelo providencia também um gráfico contendo a importância que deu a cada uma das características do conjunto de dados, oferecendo assim a possibilidade de uma análise crítica dos resultados obtidos.

```
[[  6  38]
 [ 25 171]]
              precision    recall  f1-score   support

           0       0.19      0.14      0.16        44
           1       0.82      0.87      0.84       196

    accuracy                           0.74       240
   macro avg       0.51      0.50      0.50       240
weighted avg       0.70      0.74      0.72       240

<matplotlib.axes._subplots.AxesSubplot at 0x7f6c1f6b4208>
```

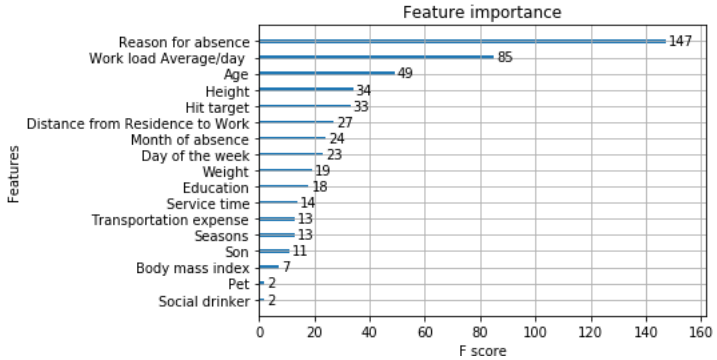

Figura 6: *Confusion Matrix* e respetiva *Classification Report* do *dataset 0*.

Para o primeiro dataset com um tratamento de dados mais cuidadoso não foi obtida qualquer mudança na accuracy mas, após uma análise mais aprofundada verificamos que, na matriz de confusão, há um pequeno balancear dos resultados em tendência da resposta a que não irá estar presente. Podemos assim verificar que para este modelo, as mudanças efetuadas possam conduzir ao overfitting se continuarem a ser aplicadas.

```
[[  5  39]
 [ 23 173]]
              precision    recall  f1-score   support

           0       0.18      0.11      0.14        44
           1       0.82      0.88      0.85       196

    accuracy                           0.74       240
   macro avg       0.50      0.50      0.49       240
weighted avg       0.70      0.74      0.72       240

<matplotlib.axes._subplots.AxesSubplot at 0x7f76b2ab7dd8>
```

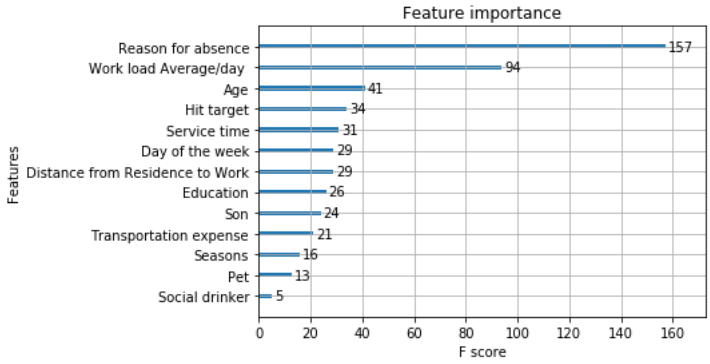

Figura 7: *Confusion Matrix* e respetiva *Classification Report* do *dataset 1*.

Para o segundo dataset, como os dados foram tratados de acordo com o importância dada às características mais importantes utilizadas no dataset com pouco tratamento de dados vemos uma clara melhoria nos resultados. A *accuracy* sobe para os 78% e não se verifica a ocorrência de tantos falsos negativos.

```
[[  6  38]
 [ 16 180]]
              precision    recall  f1-score   support

           0       0.27      0.14      0.18        44
           1       0.83      0.92      0.87       196

    accuracy                           0.78       240
   macro avg       0.55      0.53      0.53       240
weighted avg       0.72      0.78      0.74       240

<matplotlib.axes._subplots.AxesSubplot at 0x7f76b1b7e278>
```

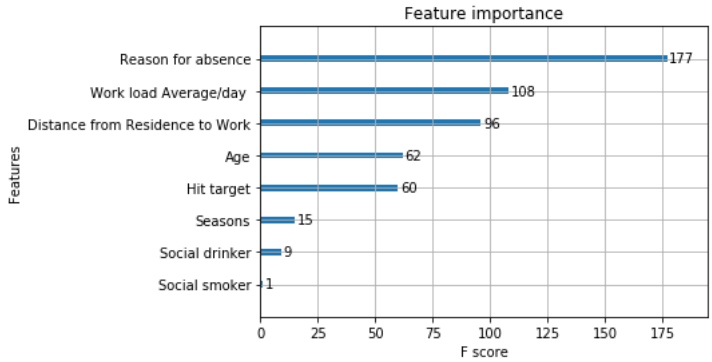

Figura 8: *Confusion Matrix* e respetiva *Classification Report* do *dataset 2*.

Nos datasets seguintes em que foram testadas opções de tratamento de dados mais agressivas, vemos uma clara redução na accuracy. No terceiro dataset o valor dos falsos positivos aumenta

exponencialmente, tornando o modelo mau na decisão. No quarto dataset existe uma melhoria em relação ao terceiro mas, mesmo assim, não é um bom resultado devido ainda ao grande número de falsos negativos.

```
[[ 25  19]
 [ 94 102]]
              precision    recall  f1-score   support

           0       0.21      0.57      0.31        44
           1       0.84      0.52      0.64       196

    accuracy                           0.53       240
   macro avg       0.53      0.54      0.48       240
weighted avg       0.73      0.53      0.58       240

<matplotlib.axes._subplots.AxesSubplot at 0x7f76b2b271d0>
```

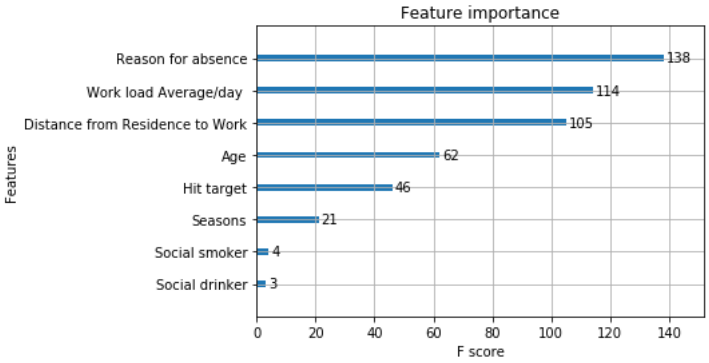

Figura 9: *Confusion Matrix* e respetiva *Classification Report* do *dataset 3*.

```
[[ 15  29]
 [ 63 133]]
              precision    recall  f1-score   support

           0       0.19      0.34      0.25        44
           1       0.82      0.68      0.74       196

    accuracy                           0.62       240
   macro avg       0.51      0.51      0.49       240
weighted avg       0.71      0.62      0.65       240

<matplotlib.axes._subplots.AxesSubplot at 0x7f76b2c0cd30>
```

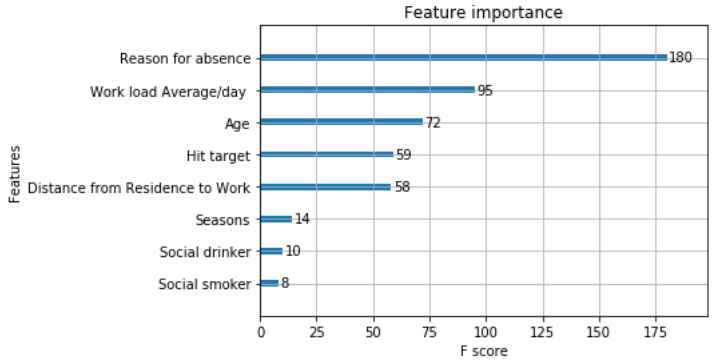

Figura 10: *Confusion Matrix* e respetiva *Classification Report* do *dataset 4*.

3.2 *K-Nearest Neighbors*

Para tentar resolver o problema proposto criou-se um modelo *K-Nearest Neighbors*. Este modelo guarda todos os dados de treino para que, quando necessário, faça comparações não necessitando de realizar aprendizagem. Numa primeira fase foi testado o modelo com todos os atributos do *dataset* exceto o ID. Verificamos um *accuracy* de 76,25%, que não é um mau resultado, mas olhando para os resultados da matriz de confusão podemos observar que o modelo não se está a comportar de acordo com o esperado visto que temos um grande número de falsos negativos e falsos positivos. Podemos observar que, em proporção os mais preocupantes são os falsos positivos pois estão em maioria em relação aos verdadeiros negativos. O K escolhido nesta simulação do modelo foi 1, visto que como a classificação é binária é o que nos garante melhores resultados sem overfitting. Este facto é apoiado pelos testes realizados. Podemos ver isso na figura X.

```
0.7416666666666667
[[  3  41]
 [ 21 175]]
              precision    recall  f1-score   support

           0       0.12      0.07      0.09        44
           1       0.81      0.89      0.85       196

    accuracy                           0.74       240
   macro avg       0.47      0.48      0.47       240
weighted avg       0.68      0.74      0.71       240
```

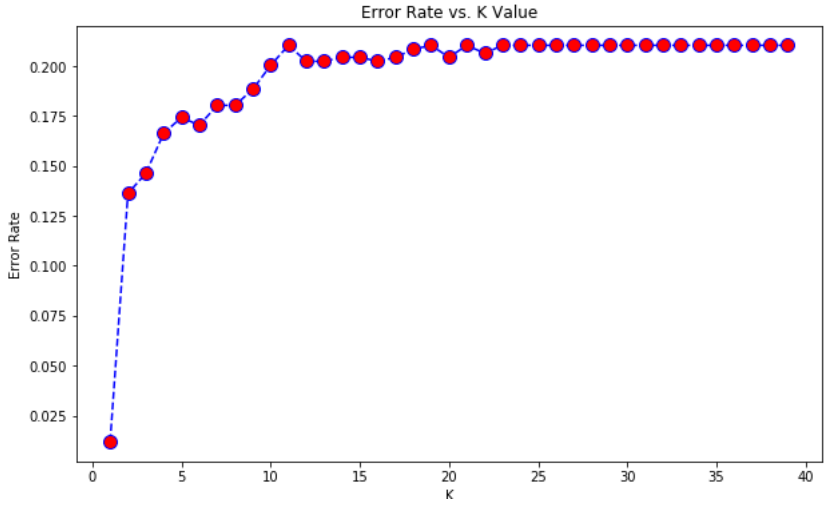

Figura 11: *Confusion Matrix* e respetiva *Classification Report* do *dataset 0*, incluindo escolha do K.

Para os *datasets* seguintes (um e dois), não existiram quaisquer alterações aos resultados. Isto pode-se dever à ainda grande quantidade de colunas existentes nos dados e que, este algoritmo, não suporta eficazmente.

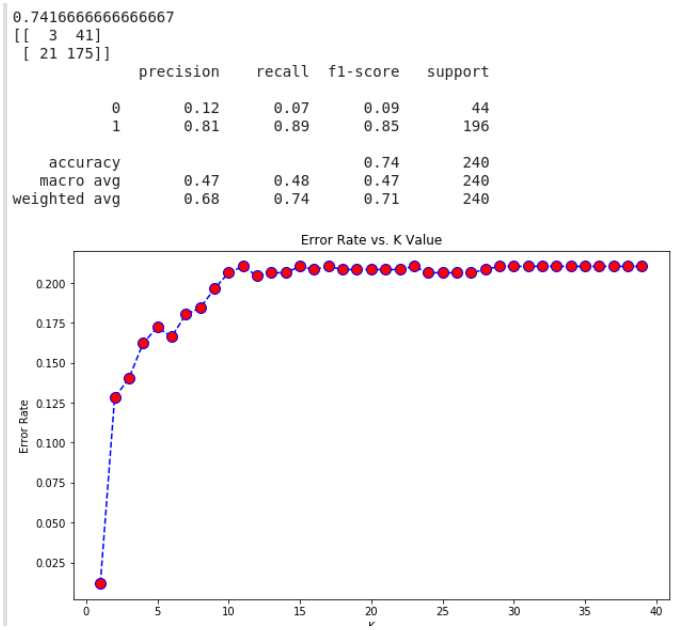

Figura 12: *Confusion Matrix* e respetiva *Classification Report* do *dataset 1*, incluindo escolha do K.

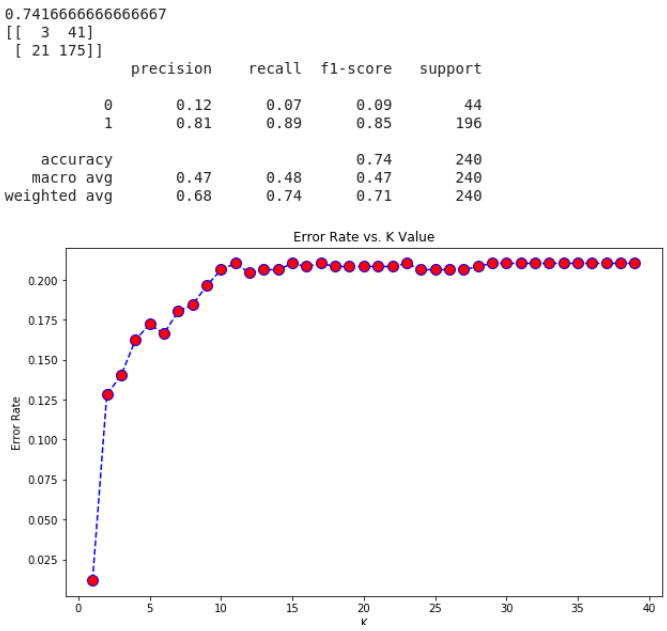

Figura 13: *Confusion Matrix* e respetiva *Classification Report* do *dataset 2*, incluindo escolha do K.

Para o dataset três e quatro, existe uma redução significativa na accuracy, 45% e 63% respectivamente, e um aumento drástico nos falsos positivos. Sem fazer uma análise mais aprofundada retiramos logo que este modelo não é adequado para este tipo de dados.

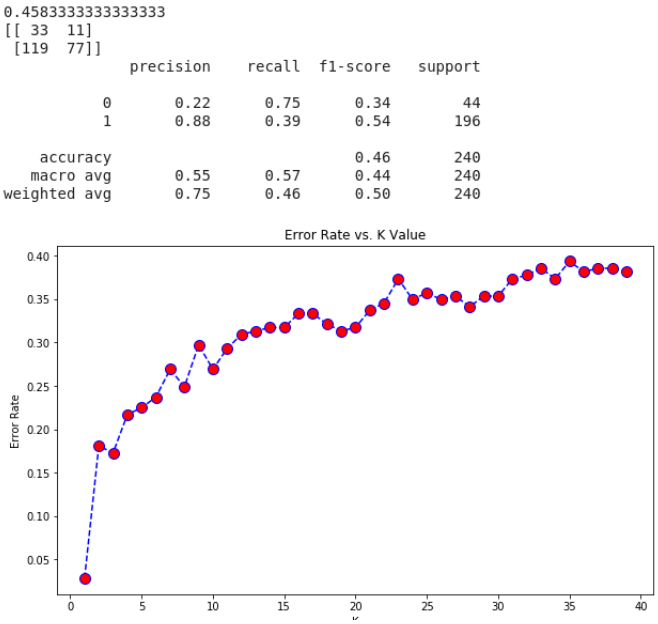

Figura 14: *Confusion Matrix* e respetiva *Classification Report* do *dataset 3*, incluindo escolha do K.

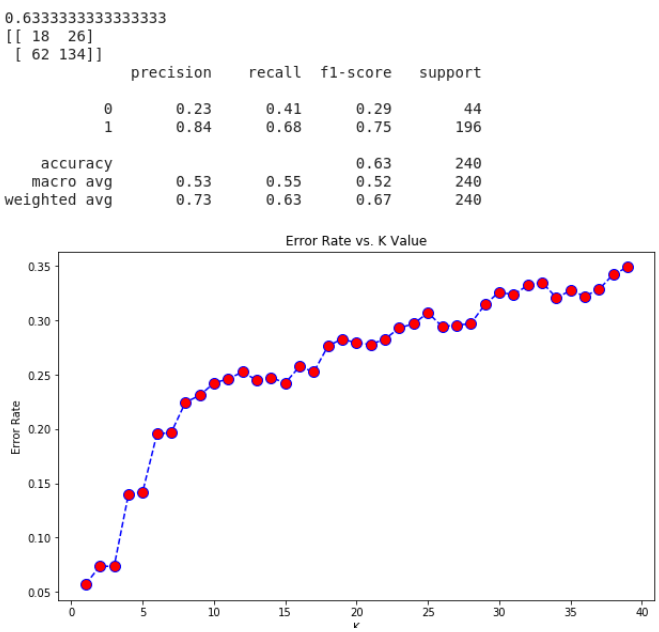

Figura 15: *Confusion Matrix* e respetiva *Classification Report* do *dataset 4*, incluindo escolha do K.

3.3 *Support Vector Machines*

O modelo *Support Vector Machines* (SVM) é um dos mais utilizados em problemas de classificação de dados, supervisionada, ou seja, nós fornecemos ao modelo as respostas para uma aprendizagem correta. Como dito anteriormente, estamos a trabalhar com 5 *datasets* diferentes e por isso iremos averiguar a performance deste modelo nos mencionados. Sendo este modelo bastante útil quando possuímos um grande número de *features*, achámos que no *dataset* 0 teria um melhor desempenho que os outros, mas infelizmente não foi o caso. Mesmo com ainda mais 2 iterações onde fazíamos *oversampling* e *undersampling*, usando *K-cross validation*, e normalizando também os dados, tornando-os valores entre 0 e 1, também nada fizeram, sendo que apesar de haver uma *accuracy* de cerca de 82%, a *confusion matrix* destas eram sempre iguais: acertava os positivos mas falhava todos os negativos. Ou seja, aparentava estar *overfitted* e não tinha aprendido de todo.

```
[[  0  44]
 [  0 196]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        44
           1       0.82      1.00      0.90       196

    accuracy                           0.82       240
   macro avg       0.41      0.50      0.45       240
weighted avg       0.67      0.82      0.73       240
```

Figura 16: *Confusion Matrix* e respetiva *Classification Report*. Igual em todos os datasets de treino

Na parametrização deste modelo foi usado o *GridSearch* pertencente ao *sklearn* para encontrar os parâmetros mais adequados para o modelo, resultando:

- C: 1;
- Gamma: 10
- Kernel: *Radial basis function* (rbf);

3.4 Redes Neuronais

Apesar de não termos trabalhado com redes neuronais nas aulas (apenas foram apresentados trabalhos sobre o modo do seu funcionamento) decidimos incorporá-las devido à facilidade de implementação usando as bibliotecas *python* usadas até agora e por também serem um modelo bastante utilizado em problemas de classificação. O grande defeito da utilização deste é o baixo volume de dados de treino, que pode causar *overfitting* (verificado mais à frente).

Esta possui uma camada de entrada com **n** neurônios para **n** *features* do *dataset*, duas camadas ocultas com 64 neurônios cada e uma função de ativação *relu* e uma camada final de saída com apenas um neurônio e função de ativação *sigmoid* (gera uma probabilidade entre 0 ou 1, entre duas classes). Usamos a função *binary crossentropy* para calcular a perda e o método de otimização *adam* (estes parâmetros foram retirados de livros sobre modelos de classificação usando redes neuronais com algumas tentativas de outros parâmetros aconselhados). O treino foi feito com 4 diferentes *datasets*, em 50 *epochs* com *batch size* de 10 (número de entradas processadas necessárias para atualizar o modelo).

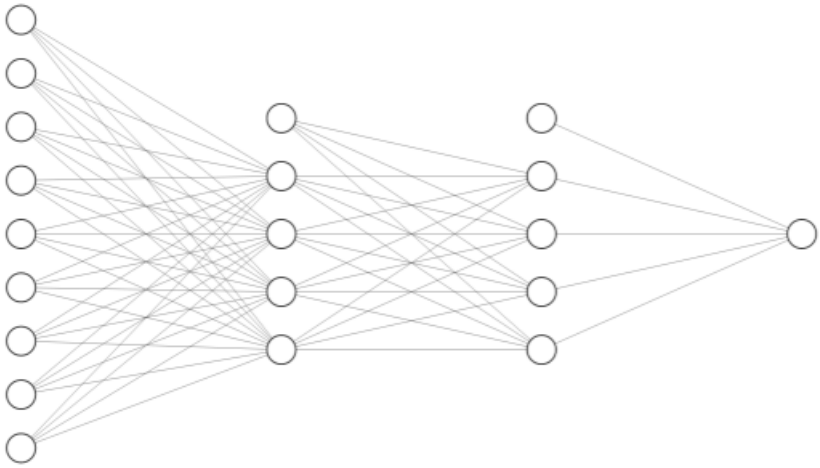

Figura 17: Arquitetura base da rede neuronal. As camadas intermédias não se encontram realmente iguais às utilizadas devido a não caberem na imagem.

Este modelo acabou por seguir o mesmo caminho do anterior, SVM, mas teve um problema ainda maior: depois de um treino, por vezes aprendia a apostar sempre em positivos e outra vezes, noutra repetição do treino, sempre em negativos. Mesmo com normalizações e os diferentes *datasets*, o resultado era sempre um dos dois referidos.

```
[[ 44   0]
 [196   0]]
              precision    recall  f1-score   support

           0       0.18      1.00      0.31        44
           1       0.00      0.00      0.00       196

    accuracy                           0.18       240
   macro avg       0.09      0.50      0.15       240
weighted avg       0.03      0.18      0.06       240
```

Figura 18: *Confusion Matrix* e respetiva *Classification Report* numa sessão do treino com o *dataset* 0, só com negativos.

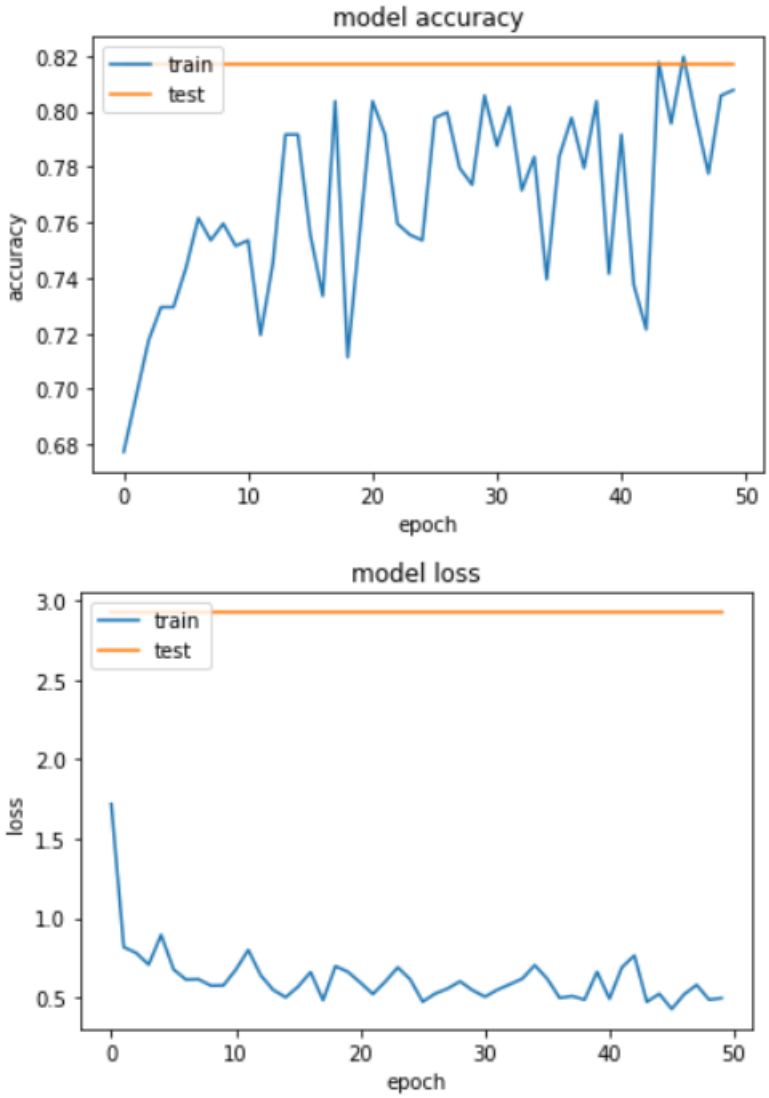

Figura 19: Gráfico da *performance* do modelo na aprendizagem e nos testes no caso 1.

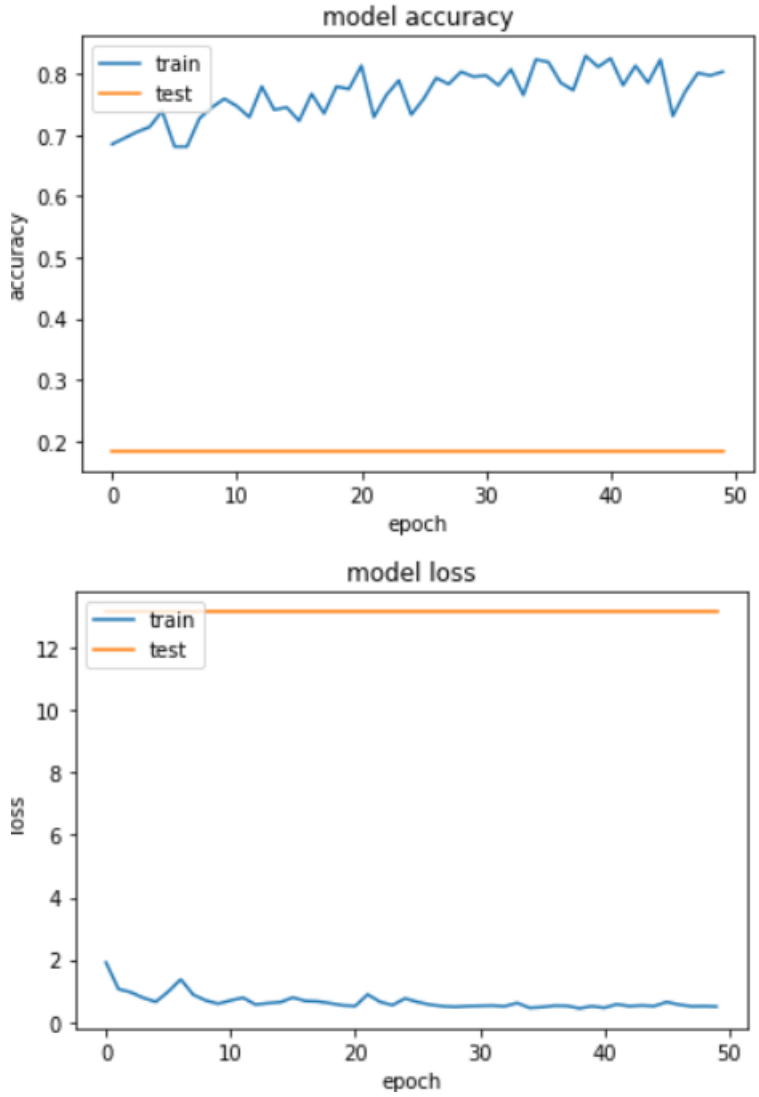

Figura 20: Gráfico da *performance* do modelo na aprendizagem e nos testes no caso 2.

Nota-se que apesar do treino aparentar uma correta aprendizagem, os treinos demonstram que de facto não aprendeu corretamente. Se usarmos o *dataset* de treino para testar, verificamos que a *accuracy* é muito alta e ele acerta 90% dos casos. Este comportamento realça no *overfitting* dos dados, sendo estranho os dois possíveis casos ocorridos.

Nós achavamos que com o aumento dos dados (no *dataset* com *oversampling*) iríamos ter melhor resultados, mas não foi o verificado.

3.5 *Decision Trees*

As Árvores de Decisão são outra alternativa para a resolução de problemas de classificação.

Numa primeira tentativa, como variáveis independentes foram selecionadas todas as colunas do *dataset*, com exceção da coluna ID. Depois de criado o modelo da *Decision Tree*, utilizando *Scikit-learn*, a performance do modelo foi avaliada através da métrica *accuracy*. O valor obtido desta métrica foi 62.5%, o que é um valor significativo, mas com o devido tratamento dos dados tem potencial para ser bastante melhor. Analisando a precisão do modelo, através de um *classification report* e da respetiva *confusion matrix*, percebemos que a precisão do modelo no caso de valores de *absent* iguais a 0 é de apenas 21% de acerto, enquanto que para valores de *absent* iguais a 1 é de 83%.

```
Accuracy: 0.625
              precision    recall  f1-score   support

           0       0.21      0.36      0.26        44
           1       0.83      0.68      0.75       196

    accuracy                           0.62       240
   macro avg       0.52      0.52      0.51       240
weighted avg       0.71      0.62      0.66       240

[[ 16  28]
 [ 62 134]]
```

Figura 21: Métricas do *dataset* 0 (*Decision Tree*).

Na Figura 22 é possível visualizar a árvore de decisão resultante. Tal como se pode ver, a árvore sofre de *overfitting*, pois é bastante larga o que a torna bastante específica, não conseguindo generalizar bem, principalmente para os valores negativos de *absent*.

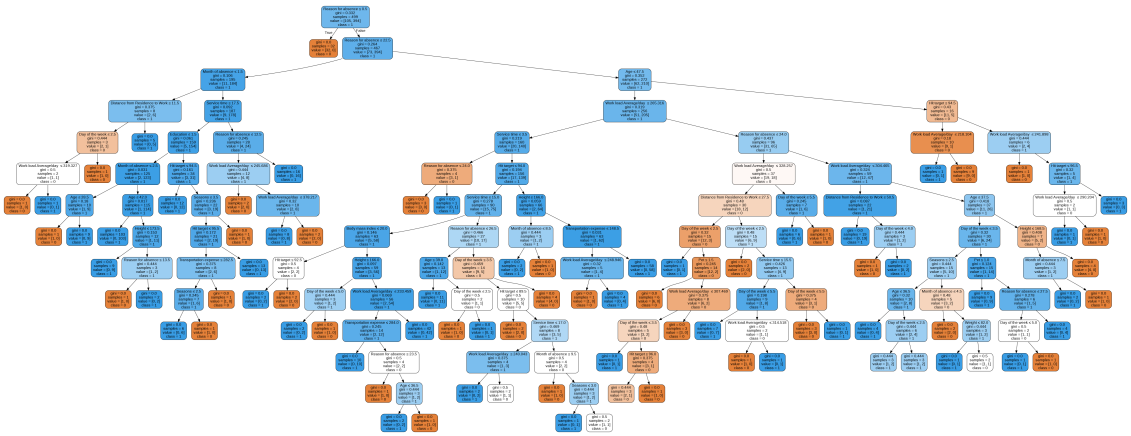

Figura 22: Árvore de Decisão do *dataset* 0.

No entanto, é possível otimizar a performance da *Decision Tree* alterando os parâmetros de ***criterion*** (escolher a medida de seleção dos atributos, que poderá ser *gini*, se quisermos utilizar o *gini index*, ou *entropy*, se preferirmos utilizar *information gain*), ***splitter*** (estratégia de *split* dos dados, que poderá ser *best* ou *random*) e ***max_depth*** (profundidade máxima da árvore). Alterando o valor de *criterion* para *entropy*, *splitter* para *random* e limitando a profundidade da árvore para 3, os resultados melhoraram substancialmente.

```
Accuracy: 0.8083333333333333
              precision    recall  f1-score   support

           0       0.38      0.07      0.12        44
           1       0.82      0.97      0.89       196

    accuracy                           0.81       240
   macro avg       0.60      0.52      0.50       240
weighted avg       0.74      0.81      0.75       240

[[  3  41]
 [  5 191]]
```

Figura 23: Métricas do *dataset* 0 (*Decision Tree* otimizada).

O *accuracy* obtido foi superior, atingindo os 80.83%. No entanto, a precisão obtida não foi a mais satisfatória, aumentado apenas para 38%. Como se pode ver através da *confusion matrix*, apenas 3 em 44 negativos estão a ser corretamente identificados. Já no caso dos positivos, 191 em 194 estão a ser corretamente classificados, o que é um resultado bastante positivo. Isto significa que o modelo acaba por não aprender bem o que é um negativo.

Tal como se pode ver na Figura 24, a árvore resultante tem somente uma profundidade igual a 4 e já não sofre de *overfitting*.

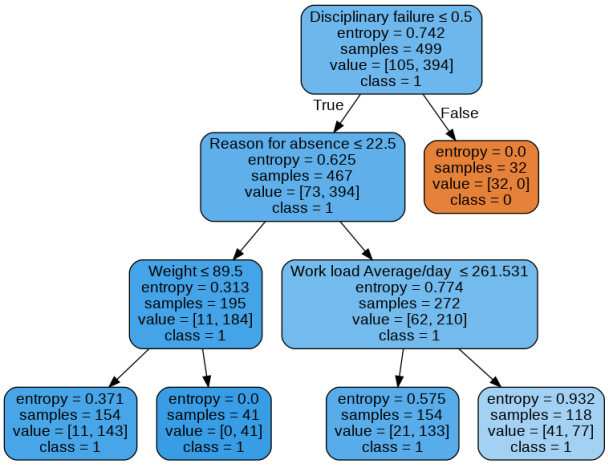

Figura 24: Árvore de Decisão otimizada do *dataset* 0.

Numa segunda tentativa, optamos por reduzir o número de colunas, removendo todas as *features* que achávamos irrelevantes pela análise dos gráficos das relações entre estas, nomeadamente: *Weight*, *Height*, *Body mass index* e *Month* (primeiro *dataset*). No entanto, a *accuracy* obtida foi semelhante à obtido anteriormente, assim como as restantes métricas. Otimizando o modelo, os resultados também foram idênticos aos obtidos no *dataset* anterior.

Em seguida, optamos por também reduzir o número de colunas, mas desta vez removendo *features* aconselhadas por um algoritmo de classificação, considerando assim apenas as seguintes *features*: *Body mass index*, *Transportation expense*, *Service time*, *Pet*, *Son*, *Day of the week*, *Education*, *Weight*, *Height*, *Month of absence* (segundo *dataset*). Os resultados obtidos foram novamente semelhantes aos já obtidos, sendo até ligeiramente piores.

```
Accuracy: 0.5625
              precision    recall  f1-score   support

           0       0.21      0.52      0.30        44
           1       0.84      0.57      0.68       196

    accuracy                           0.56       240
   macro avg       0.53      0.55      0.49       240
weighted avg       0.73      0.56      0.61       240

[[ 23  21]
 [ 84 112]]
```

```
Accuracy: 0.8041666666666667
              precision    recall  f1-score   support

           0       0.36      0.09      0.15        44
           1       0.83      0.96      0.89       196

    accuracy                           0.80       240
   macro avg       0.59      0.53      0.52       240
weighted avg       0.74      0.80      0.75       240

[[  4  40]
 [  7 189]]
```

Figura 25: Métricas do *dataset* 2 (*Decision Tree* não otimizada e otimizada, respetivamente).

Desta vez, decidimos utilizar o *dataset* de treino balanceado, visto haver um grande desequilíbrio entre as duas classes da variável de decisão (79/21), e poderá ser esta a solução para se colmatar o excesso de falsos positivos (40 em 44). Usando a estratégia de *oversampling* para obter um ratio de 50/50 os resultantes foram também semelhantes aos já obtidos, mas usando a estratégia de *undersampling* os valores mudam ligeiramente, porém, são menos satisfatórios.

```
Accuracy: 0.725
              precision    recall  f1-score   support

           0       0.24      0.23      0.23        44
           1       0.83      0.84      0.83       196

    accuracy                           0.73       240
   macro avg       0.53      0.53      0.53       240
weighted avg       0.72      0.72      0.72       240

[[ 10  34]
 [ 32 164]]
```

Figura 26: Métricas do *dataset* 4 (*Decision Tree* otimizada).

3.6 *Random Forest*

Com o objetivo de melhorar os resultados obtidos, recorremos também às *Random Forests* para ajustar (*fit*) os dados e prever (*predict*) os resultados. As *Random Forests* são também um algoritmo de aprendizagem supervisionada, que cria e combina várias árvores de decisão.

Treinando para o *dataset* número zero, os resultados obtidos encontram-se na figura seguinte. Estes resultados são idênticos aos obtidos apenas com apenas uma *Decision Tree* otimizada (*accuracy* = 80.83%).

```
Accuracy: 0.8083333333333333
              precision    recall  f1-score   support

           0       0.38      0.07      0.12        44
           1       0.82      0.97      0.89       196

    accuracy                           0.81       240
   macro avg       0.60      0.52      0.50       240
weighted avg       0.74      0.81      0.75       240

[[  3  41]
 [  5 191]]
```

Figura 27: Métricas do *dataset* 0 (*Random Forest*).

Para o primeiro *dataset*, os resultados já são ligeiramente diferentes, diminuindo o número de falsos positivos. No entanto, o número de falsos negativos aumentou e, no total, a *accuracy* foi inferior (78.3%).

```
Accuracy: 0.7833333333333333
              precision    recall  f1-score   support

           0       0.32      0.16      0.21        44
           1       0.83      0.92      0.87       196

    accuracy                           0.78       240
   macro avg       0.57      0.54      0.54       240
weighted avg       0.74      0.78      0.75       240

[[  7  37]
 [ 15 181]]
```

Figura 28: Métricas do *dataset* 1 (*Random Forest*).

Já no caso do segundo *dataset*, com a seleção das *features* com maior importância, a *accuracy* voltou aos 80.83%, mas a precisão tanto para *absent* igual a 0 como para igual a 1 foi superior, de 38% para 42% e de 82% para 83%, respetivamente.

```
Accuracy: 0.8083333333333333
              precision    recall  f1-score   support

           0       0.42      0.11      0.18        44
           1       0.83      0.96      0.89       196

    accuracy                           0.81       240
   macro avg       0.62      0.54      0.54       240
weighted avg       0.75      0.81      0.76       240

[[  5  39]
 [  7 189]]
```

Figura 29: Métricas do *dataset* 2 (*Random Forest*).

Para o terceiro e quarto *datasets*, os resultados foram menos satisfatórios visto que a *accuracy* diminuiu bastante, tal como se pode ver na Figura 30.

```
Accuracy: 0.6291666666666667
              precision    recall  f1-score   support

           0       0.20      0.34      0.25        44
           1       0.82      0.69      0.75       196

    accuracy                           0.63       240
   macro avg       0.51      0.52      0.50       240
weighted avg       0.71      0.63      0.66       240

[[ 15  29]
 [ 60 136]]
```

```
Accuracy: 0.625
              precision    recall  f1-score   support

           0       0.18      0.30      0.22        44
           1       0.82      0.70      0.75       196

    accuracy                           0.62       240
   macro avg       0.50      0.50      0.49       240
weighted avg       0.70      0.62      0.66       240

[[ 13  31]
 [ 59 137]]
```

Figura 30: Métricas do *dataset* 3 e 4, respetivamente (*Random Forest*).

Normalmente, a *Random Forest* obtém um desempenho global melhor que uma única *Decision Tree*. Como estamos a atribuir maior importância à métrica *accuracy*, o valor máximo da *accuracy* obtido em cada um destes modelos foi igual (80.83%).

4 Conclusão

Apesar do modelo SVM ser, em teoria, o mais adequado a problemas de classificação de *datasets* de grandes dimensões, isto não foi verificado pelo que houve um enorme *overfitting* dos dados e o modelo apenas aprendia a identificar os positivos. As redes neuronais caíram no mesmo, sendo ainda mais estranho o seu comportamento de aprendizagem variar. Apesar dos 81,67% de *accuracy* no geral, ele não se adequava para as previsões necessárias.

No caso das *Decision Trees*, os resultados obtidos comparativamente aos outros modelos preditivos são bastante satisfatórios. Se atribuirmos maior importância à métrica *accuracy*, o máximo valor obtido foi de 80,83%, no entanto, revela-se imperfeito devido ao excesso de falsos positivos. O recurso a uma *Random Forest* não se mostrou tão apelativo como esperado, visto que os resultados obtidos foram apenas ligeiramente superiores.

No caso do KNN podemos verificar que este algoritmo não é adequado para a classificação binária deste conjunto de dados visto que o *dataset* contém muitas características. Isso leva a que os dados tivessem de crescer exponencialmente para que a assertividade do modelo fosse mais alta que 74%, o mais alto que foi conseguido.

A regressão logística após a análise de todos os resultados foi dos algoritmos que melhores resultados obteve. Isto deve-se ao facto de não só ter sido utilizada uma regressão mas também foi aplicado um método de *gradient boosting* que normalmente, e aqui se verificou, leva a melhores resultados porque utiliza métodos que diminuem o *overfitting* sendo este um algoritmo mais regular. Dito isto, este modelo foi o que obteve a *accuracy* mais alta no valor de 78%, no entanto os resultados continuam a não ser ideais devido ao ainda número considerável de falsos negativos e falsos positivos.

Posto isto, depois de uma análise extensa dos resultados, concluímos que, apesar de todos os modelos terem sofrido de *overfitting*, nunca conseguindo aprender corretamente a distinção das duas classes, o que melhor desempenho teve foi o de *Random Forest*, usando o *dataset* 2, com uma *accuracy* geral de 81%, este foi o que com mais certeza conseguiu prever corretamente as duas classes, sendo o que contém menos falsos positivos e negativos.