Trabalho Prático
Aprendizagem e Extração de Conhecimento

Ricardo Ferreira[A82568] Hugo Faria[A81283]
Rodolfo Silva[A81716] Bruno Veloso[A78352]

Universidade do Minho, Departamento de Informática

Resumo

O presente relatório tem como objectivo a preparação e análise de um *dataset* relativo às características de funcionários de múltiplas empresas de modo a prever o nível salarial anual do indivíduo. Inicialmente será feita uma análise exploratória dos dados do *dataset*. Seguidamente, serão feitas tentativas de classificação para dados de teste no qual serão usados diferentes modelos para avaliar os resultados de previsão.

Keywords: Análise exploratória, classificação, modelo, dados

Conteúdo

1 Introdução

Este relatório é referente ao segundo trabalho prático da unidade curricular de Análise e Extração de Conhecimento do perfil de Sistemas Inteligentes. Numa primeira fase será feita uma visualização do que os dados representam, bem como a sua análise exploratória. Em seguida, passaremos à preparação do *dataset* de forma a este poder ser usado para resolver os problemas de classificação. Seguidamente, serão aplicados diferentes modelos com a tentativa de classificar os dados de teste que nos foram fornecidos.

Depois de todo este processo, o objetivo passa por avaliar quais os melhores modelos de classificação em que se dará uma explicação da nossa escolha para o que é um bom modelo e uma conclusão sobre todo o trabalho realizado.

1.1 Estrutura do relatório

O relatório encontra-se estruturado da seguinte forma:

Inicialmente, na *secção 1* será feita uma breve introdução e contextualização do problema do caso em estudo, a *secção 2* apresenta a Analise e Pré-Transformação dos dados, de seguida, na *secção 3* serão apresentadas os vários modelos e os respectivos resultados obtidos, *secção 4* apresenta a análise dos resultados obtidos nos modelos presentes na *secção 3*. Por fim o relatório termina com conclusões na *secção 5*, onde é também apresentada uma análise crítica aos resultados obtidos.

2 Preparação dos dados

2.1 Visualização das colunas

Com o objectivo de analisar melhor o *dataset* a estudar foi necessário entender e interpretar cada uma das informações fornecidas e a relação das mesmas com o salário de um indivíduo.

Inicialmente foi importante perceber quais as características e tipos de dados existentes no *dataset*.

Informação das características

- ***Age*:** Idade do indivíduo;
- ***Workclass*:** Classe de trabalho do indivíduo, por exemplo: State-gov, Private, Federal-gov, etc;
- ***Fnlwgt*:** Número estimado de indivíduos que este caso representa;
- ***Education*:** Grau de escolaridade do indivíduo, por exemplo: Bachelors, Masters, 11th, etc;
- ***Education-num*:** Grau de escolaridade do indivíduo como valor numérico;
- ***Marital-status*:** Estado civil do indivíduo;

- ***Occupation*:** Trabalho do indivíduo.
- ***Relationship*:** Estado civil atual;
- ***Race*:** Raça do indivíduo;
- ***Sex*:** Género do indivíduo;
- ***Capital-Gain*:** Lucros de venda de um ativo;
- ***Capital-Loss*:** Perda devido ao ativo ter desvalorizado;
- ***Hours-per-Week*:** Número de horas de trabalho por semana;
- ***Native-Country*:** País de origem;
- ***Salary-Classification*:** Classificação do salário ganho pelo indivíduo, pode ser <=50K ou >50K.

2.2 Análise exploratória dos dados

Será abordada uma análise exploratória dos dados com o objetivo de perceber a quantidade de dados que temos, bem como a relação entre as determinadas características (colunas) e também perceber a distribuição dos dados.

Heat Map

Primeiro, construímos um *Heat Map*, de forma a perceber a correlação entre as várias características (colunas) do *dataset*.

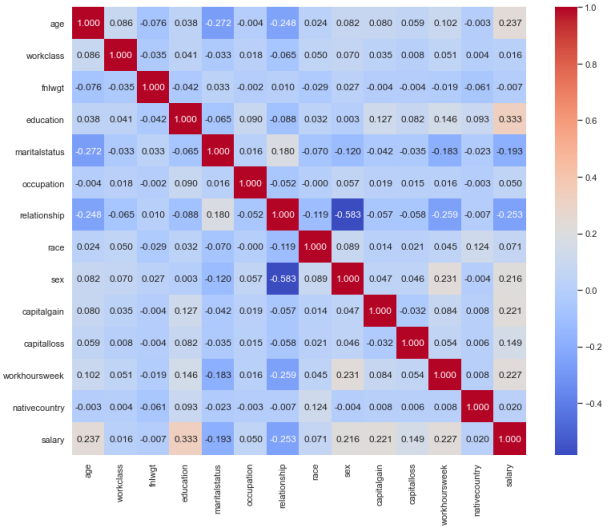

Figura 1. Heat Map que mostra a correlação entre colunas.

Depois da visualização do *Heat Map* consegue-se perceber quais as colunas que mais influenciam o *Salary-Classification* (*salary* na imagem). Uma correlação positiva, quando comparado com *Salary-Classification*, significa que conforme x característica aumenta espera-se que *Salary-Classification* aumente (mais propício a calhar na categoria de >50k de salário), e uma correlação negativa é precisamente o contrário à positiva, no qual se espera que o aumento de uma característica diminua o *Salary-Classification*.

Considerando uma correlação alta como valores acima de 0.20, consegui-mos ver que *Education-num* (*education* na imagem) tem uma correlação positiva alta com o *Salary-Classification*. Também *age*, *sex*, *capital-gain*, *capital-loss* e *hours-per-week* (*workshoursweek* na imagem) têm uma correlação positiva com *Salary-Classification*. *Relationship* tem uma correlação negativa alta com *salary*.

Distribuição do sexo

De seguida, apresenta-se a distribuição de gênero presente no *dataset*.

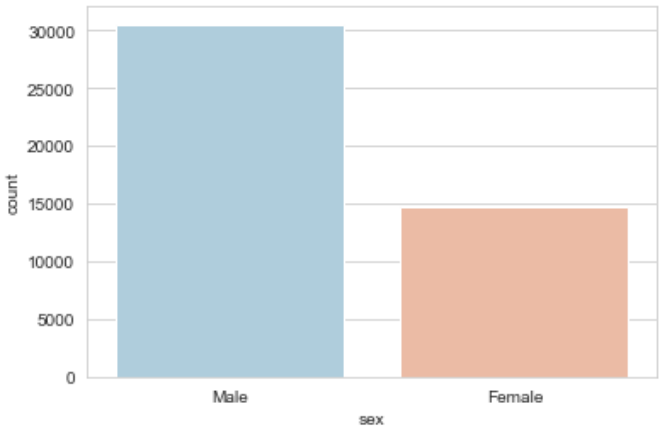

Figura 2. Distribuição do sexo.

Relação entre salário e educação

Para percebermos qual a relação entre a educação e os salários foi necessário a criação de um gráfico que representasse a média do tipo de salário que cada idade recebe, como é possível verificar na seguinte imagem:

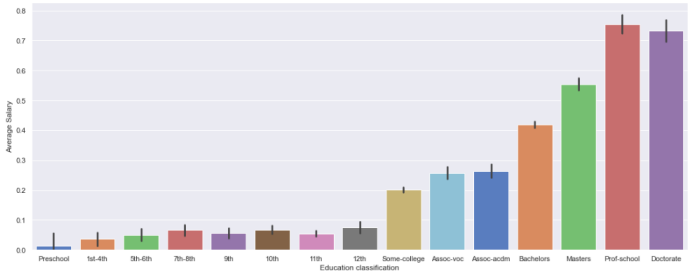

Figura 3. Relação entre salário e educação.

Com isto foi possível verificar que quase de forma linear, quanto maior a educação, maior a percentagem de salários mais elevados, o que faz sentido porque na generalidade, quantos mais estudos uma pessoa tem maior valor tem o seu trabalho.

Salários por idade

Para entender melhor a distribuição das idades com o salário foi feito um gráfico que representa para cada idade quantas pessoas têm que tipo de salário (0 representa salário <=50k e 1 representa salário >50k).

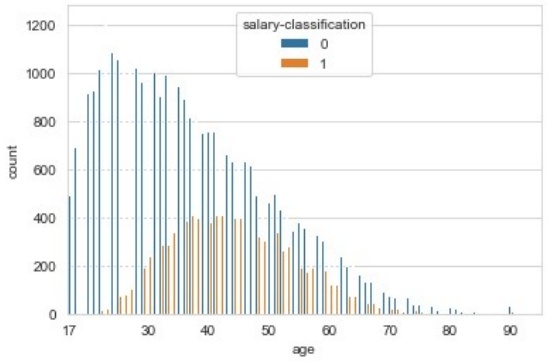

Figura 4. Salários por idade.

Com isto foi possível concluir que a maioria dos salários superiores a 50k situam-se entre os 40 e 50 anos, o que faz sentido, visto que é a suposta idade em que a maioria das pessoas está estabilizada e com um bom trabalho. É possível concluir também, que a maioria dos salários menores a 50k são de pessoas até aos 35 anos, que na generalidade costumam ser pessoas que estão a iniciar a sua carreira(com cerca de 20 anos), ou pessoas que ainda estão a ganhar experiência e portanto não têm um salário elevado.

Contagem do tipo de salário no dataset

Com o objectivo de verificar a diferença entre o número de casos inferiores e superiores a 50k de salário foi produzido o seguinte gráfico (0 representa salário <=50k e 1 representa salário >50k):

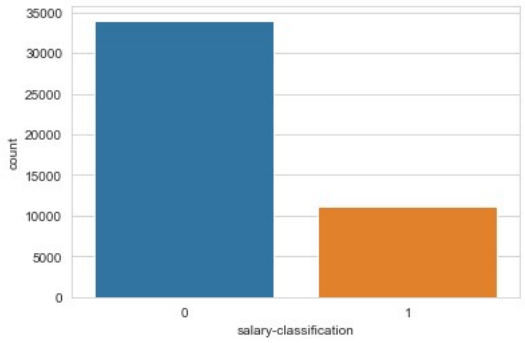

Figura 5. Contagem do tipo de salários.

O que permitiu verificar o que na maioria das vezes acontece numa sociedade, que existe uma disparidade enorme na quantidade de pessoas que recebem salários elevados e salários reduzidos.

2.3 Transformação dos dados

Foram fornecidos dois ficheiros: um chamado *training.csv* que contém os dados de treino de modelos e outro chamado *test.csv* que contém os dados para teste de modelos. Quando falamos em *dataset* referimo-nos à junção dos dados dos dois ficheiros.

O grupo reparou que no *dataset* existe duas *features* com a informação repetida, sendo estas *education* e *education-num* ao que o grupo decidiu remover a coluna *education*.

De forma a adequar a informação que o *dataset* contém, aos modelos de aprendizagem que foram utilizados, foi necessário proceder ao seu pré-processamento. Para isso foi feita uma análise de existência de valores nulos no *dataset*, como é possível verificar na seguinte figura:

```
age                0
workclass       2799
fnlwgt             0
education          0
maritalstatus      0
occupation      2809
relationship       0
race               0
sex                0
capitalgain        0
capitalloss        0
workhoursweek      0
nativecountry    857
salary             0
dtype: int64
```

Figura 6. Verificação da existência de valores nulos (training+test).

Muitas vezes, os dados recolhidos do mundo real apresentam valores em falta ou contêm inconsistências. Posto isto, e após verificar na figura anterior que o *dataset* continha valores nulos em várias características, foram implementados 3 tipos de pré-processamento para tratar de tal:

- Eliminação dos valores nulos presentes no *dataset* (*training+test*);
- Valores nulos no ficheiro de *training* substituídos pela moda da *feature* do *dataset* e valores nulos removidos do ficheiro de *test*;
- O uso do *KnnImputer* que substitui os valores nulos presentes em *training* pela média dos *K_nearest_neighbors*, e valores nulos removidos do ficheiro de *test*.

Tendo em conta estes 3 tipos de pré-processamento para tratar de valores nulos, foram criados diferentes ficheiros com os diferentes dados.

A **primeira abordagem** foi remover todos os nulos do *dataset*, apagando anteriormente a coluna *education*, e convertendo o resto dos dados das *features* para valores inteiros, criando assim dois ficheiros (*training_0null.csv* e *test_0null.csv*).

A **segunda abordagem** passou por converter os valores nulos para a moda da *feature* onde estes se encontram mas só do lado do *training*, removendo os valores nulos do *test*, também removendo a coluna *education*, e no fim, converter o resto dos dados das *features* para valores inteiros, criando assim dois ficheiros (*training_mode.csv* e *test_mode.csv*).

A **terceira abordagem** passou por converter todos os valores nulos presentes em *training* pela média dos *K_nearest_neighbors* mais perto do valor, removendo os valores nulos do lado de *test*, também se removeu a coluna *education*, e no fim, converteu-se os dados das *features* para valores inteiros. Criou-se assim dois ficheiros que são *training_knn.csv* e *test_knn.csv*.

Uma **quarta abordagem**, e última, passou por remover todos os nulos presentes no *dataset*, removendo a coluna *education*, e por fim, fez-se um escaloneamento dos dados das *features* usando a função *StandardScaler* da biblioteca *sklearn*.

O grupo também considerou uma **quinta abordagem**, que seria a aplicação de *Principal Component Analysis*, uma técnica derivada de álgebra linear que permite redução dimensional de um *dataset*, preservando a informação relevante e mantendo a variância dos dados. Depois da aplicação desta técnica os resultados que se obteram não apresentavam uma diferença de *performance*, comparada com as quatro abordagens discutidas anteriormente, significativa o suficiente para decidir a implementação completa desta técnica.

2.4 Balanceamento dos Dados

Os dados presentes no *dataset* eram demasiado desbalanceados, como se pode ver na figura 5, logo procurou-se algoritmos que permitissem o balanceamento desses mesmos dados, tendo-se decidido usar quatro, sendo eles:

RandomOverSampling é a técnica de *Over Sampling* mais simples que se pode utilizar, esta consiste em escolher instâncias da classe minoritária aleatoriamente, duplicando-as no novo *dataset* balanceado.

O uso desta técnica pode provocar erros de *overfitting* no modelo originado, o que torna esta pouco utilizada.

RandomUnderSampling representa a técnica de *Under Sampling* mais simples disponível, a redução é atingida através da seleção aleatória de instâncias da classe maioritária, de modo a serem removidas do *dataset*.

Apesar de esta técnica providenciar uma forma fácil de reduzir o *dataset*, isto leva à perda de muita informação.

SMOTE, Synthetic Minority Oversampling Technique, possivelmente a técnica de *Over Sampling* mais usada para sintetizar novos exemplos. De forma breve, SMOTE começa por escolher aleatoriamente uma instância A da classe minoritária, passando por encontrar os seus *K_nearest_neighbors*. Uma nova instância é depois feita através da seleção aleatória de um dos *K_nearest_neighbors* B. Por fim, uma linha é traçada entre A e B e o novo ponto a inserir no *dataset* é tirado de um ponto algures da linha.

ENN, Edited Nearest Neighbours corresponde a outro algoritmo usado para *Under Sampling* do *dataset* onde para cada instância A, os seus 3 vizinhos mais próximos vão ser analisados. Se A for da classe maioritária e for mal classificado pelos vizinhos, este é removido. No entanto, se A for da classe minoritária e for mal classificado pelos vizinhos, os que pertencem à classe maioritária desses mesmos vizinhos são removidos.

3 Avaliação de Modelos

Depois de se ter estudado e aplicado algumas técnicas de pré-processamento de dados, chegou a altura de avaliar os *datasets* obtidos com algoritmos de aprendizagem supervisionada. Porém, o grupo utilizou um algoritmo de aprendizagem não supervisionado para aumentar o seu conhecimento. Foi usado para implementação dos modelos a linguagem *Python* e também a biblioteca *sklearn*.

3.1 Logistic Regression

Quando falamos em problemas de classificação, o primeiro modelo de aprendizagem que vem à cabeça é *Logistic Regression*(2). Posto isto, o grupo optou por desenvolver *Logistic Regression* para cada uma das quatro(4) abordagens referidas na *secção 2.3*. Após uma simples aplicação do modelo a estas abordagens, foi também aplicado o modelo de *Logistic Regression* mas usando *Over/Under Sampling*(1). Por fim, foi também usado, como termo de comparação, o *SMOTEENN*(1).

De seguida, apresentam-se os diferentes resultados obtidos:

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos	0.796	0.82	0.94	0.65	0.36
Nulos substituídos pela moda	0.794	0.80	0.97	0.73	0.26
KNN Imputer para nulos	0.784	0.80	0.95	0.64	0.28
Nulos removidos e *Standard Scaling*	0.819	0.84	0.94	0.70	0.46

Tabela 1. Resultados *Logistic Regression*.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos	0.781	0.87	0.84	0.55	0.60
Nulos substituídos pela moda	0.744	0.82	0.85	0.48	0.42
KNN Imputer para nulos	0.747	0.82	0.85	0.48	0.41
Nulos removidos e *Standard Scaling*	0.794	0.88	0.84	0.57	0.65

Tabela 2. Resultados de *Logistic Regression* com *Under/Over Sampling*.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos	0.741	0.88	0.76	0.48	0.70
Nulos substituídos pela moda	0.748	0.91	0.74	0.49	0.77
KNN Imputer para nulos	0.744	0.91	0.74	0.49	0.77
Nulos removidos e *Standard Scaling*	0.737	0.92	0.72	0.48	0.80

Tabela 3. Resultados de *Logistic Regression* com *SMOTE* e *ENN*.

3.2 K-Nearest Neighbors

Seguidamente, o grupo decidiu aplicar *K-Nearest Neighbors*(3) na tentativa de uma melhoria dos resultados em comparação com *Logistic Regression*. Para a escolha do K mais apropriado a ser usado, foi decidido o K a usar através do cálculo do erro, escolhendo o K que apresentasse menor erro. Este erro foi calculado para valores de K que vão de 1 a 40, seguido de um gráfico que nos ajudasse a decidir. De seguida, é apresentado um dos gráficos de exemplo. Para os valores deste gráfico foram usados os dados para quando se remove os valores nulos e se usa *Standard Scaling*.

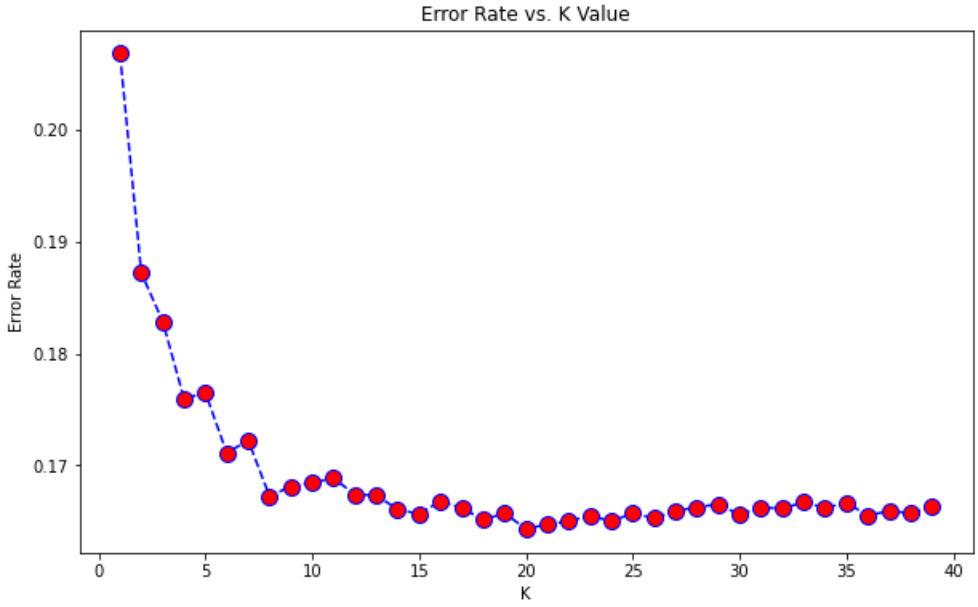

Figura 7. Erros obtidos usando *K-Nearest Neighbors*.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos K=25	0.796	0.79	0.99	0.84	0.34
Nulos substituídos pela moda K=20	0.795	0.79	0.99	0.84	0.21
KNN Imputer para nulos K=20	0.795	0.79	0.99	0.84	0.21
Nulos removidos e *Standard Scaling* K=20	0.835	0.86	0.93	0.71	0.55

Tabela 4. Resultados de *K-Nearest Neighbors*.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos K=25	0.751	0.80	0.89	0.49	0.33
Nulos substituídos pela moda K=20	0.752	0.80	0.89	0.49	0.33
KNN Imputer para nulos K=20	0.753	0.80	0.89	0.50	0.33
Nulos removidos e *Standard Scaling* K=10	0.814	0.90	0.84	0.60	0.72

Tabela 5. Resultados de *K-Nearest Neighbors* com *Under/Over Sampling*.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos K=25	0.608	0.82	0.62	0.33	0.58
Nulos substituídos pela moda K=20	0.779	0.93	0.76	0.53	0.83
KNN Imputer para nulos K=20	0.766	0.94	0.74	0.51	0.85
Nulos removidos e *Standard Scaling* K=20	0.767	0.94	0.74	0.52	0.85

Tabela 6. Resultados de *K-Nearest Neighbors* com *SMOTE* e *ENN*.

3.3 K-Means Clustering

Foi implementado *K-Means Clustering*(4) pela simples razão de que o grupo queria saber mais de como este funcionava e ver os resultados que se obtia com o *dataset* em mão. Este algoritmo é não supervisionado, ou seja, é mais usado para quando não há *labels* para os dados (o nosso *dataset* apresenta *labels*) e este tenta dividir os dados em grupos (*clusters*). Posto isto, foi então implementado mais por uma questão de expandir conhecimento e não na procura de resultados. De seguida, encontram-se os resultados obtidos para *K-Means Clustering* usando 2 *clusters*.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos	0.614	0.75	0.73	0.24	0.25
Nulos substituídos pela moda	0.618	0.75	0.74	0.23	0.25
KNN Imputer para nulos	0.618	0.75	0.74	0.23	0.25
Nulos removidos e *Standard Scaling*	0.593	0.92	0.50	0.36	0.86

Tabela 7. Resultados *K-Means Clustering*.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos	0.532	0.58	0.73	0.39	0.25
Nulos substituídos pela moda	0.536	0.58	0.74	0.40	0.25
KNN Imputer para nulos	0.534	0.58	0.74	0.39	0.25
Nulos removidos e *Standard Scaling*	0.657	0.84	0.51	0.55	0.86

Tabela 8. Resultados *K-Means Clustering* com *RandomUnderSampling* e *RandomOverSampling*.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos	0.614	0.75	0.73	0.24	0.27
Nulos substituídos pela moda	0.617	0.75	0.73	0.24	0.26
KNN Imputer para nulos	0.617	0.75	0.73	0.24	0.26
Nulos removidos e *Standard Scaling*	0.760	0.76	1.00	1.00	0.02

Tabela 9. Resultados *K-Means Clustering* com *SMOTE* e *ENN*

3.4 Support Vector Machines

No caso do modelo de *Support Vector Machines*(5), estes apresentaram ser um dos modelos de maior confiança para o *dataset* que foi estudado. Com isso em mente, decidiu-se fazer um estudo a ver quais os parâmetros com melhores resultados.

De entre os kernels que foram aplicados, apenas o RBF demonstrou ter um bom desempenho nesta situação, assim, têm-se que se acabou por se escolher os parâmetros de C=10, Kernel='rbf' e gamma='scale' para o modelo.

De seguida, têm-se as tabelas representativas dos resultados obtidos com o uso de SVM nos *datasets* desenvolvidos.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos	0.791	0.78	1.00	0.97	0.15
Nulos substituídos pela moda	0.791	0.78	1.00	0.96	0.16
KNN Imputer para nulos	0.781	0.78	1.00	0.99	0.11
Nulos removidos e *Standard Scaling*	0.845	0.86	0.94	0.76	0.54

Tabela 10. Resultados *SVM*.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos	0.791	0.78	1.00	0.97	0.15
Nulos substituídos pela moda	0.791	0.78	1.00	0.96	0.16
KNN Imputer para nulos	0.781	0.78	1.00	0.99	0.11
Nulos removidos e *Standard Scaling*	0.82	0.92	0.83	0.60	0.79

Tabela 11. Resultados *SVM* com *RandomUnderSampling* e *RandomOverSampling*.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos	0.792	0.79	0.99	0.83	0.19
Nulos substituídos pela moda	0.791	0.79	0.99	0.82	0.19
KNN Imputer para nulos	0.791	0.79	0.99	0.82	0.19
Nulos removidos e *Standard Scaling*	0.781	0.95	0.75	0.53	0.87

Tabela 12. Resultados *SVM* com *SMOTE* e *ENN*.

3.5 Naive Bayes

O grupo optou também por usar o modelo de *Naive Bayes*(6) pois ainda não se possuía um modelo fundamentado em classificadores probabilísticos. Vários testes foram realizados para determinar qual a versão de *Naive Bayes* a ser empregue, tendo-se obtido os melhores resultados com a variante de *Gaussian Naive Bayes*. Assim apresenta-se nas seguintes tabelas os resultados obtidos com os *datasets*.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos	0.789	0.81	0.95	0.65	0.31
Nulos substituídos pela moda	0.789	0.81	0.95	0.65	0.31
KNN Imputer para nulos	0.789	0.81	0.95	0.65	0.31
Nulos removidos e *Standard Scaling*	0.796	0.81	0.95	0.68	0.32

Tabela 13. Resultados *Naive Bayes*.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos	0.786	0.81	0.94	0.63	0.31
Nulos substituídos pela moda	0.788	0.81	0.95	0.65	0.31
KNN Imputer para nulos	0.788	0.81	0.95	0.65	0.31
Nulos removidos e *Standard Scaling*	0.806	0.83	0.94	0.68	0.40

Tabela 14. Resultados *Naive Bayes* com *RandomUnderSampling* e *RandomOverSampling*.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos	0.781	0.81	0.94	0.61	0.31
Nulos substituídos pela moda	0.781	0.81	0.94	0.61	0.31
KNN Imputer para nulos	0.781	0.81	0.94	0.61	0.31
Nulos removidos e *Standard Scaling*	0.805	0.84	0.92	0.65	0.44

Tabela 15. Resultados *Naive Bayes* com *SMOTE* e *ENN*.

3.6 Random Forest

Por fim, foi usado *Random Forest*(7), um modelo baseado em *Ensemble Learning*. A *accuracy* apresentada pelo modelo foi a melhor dentro do grupo de modelos aplicados. Este foi atingido usando uma *max depth* de 15 e apresentou o menor *Error Rate* quando se testou o modelo com valores entre 1 e 25, como pode ser visualizado na figura 8 anexada a seguir.

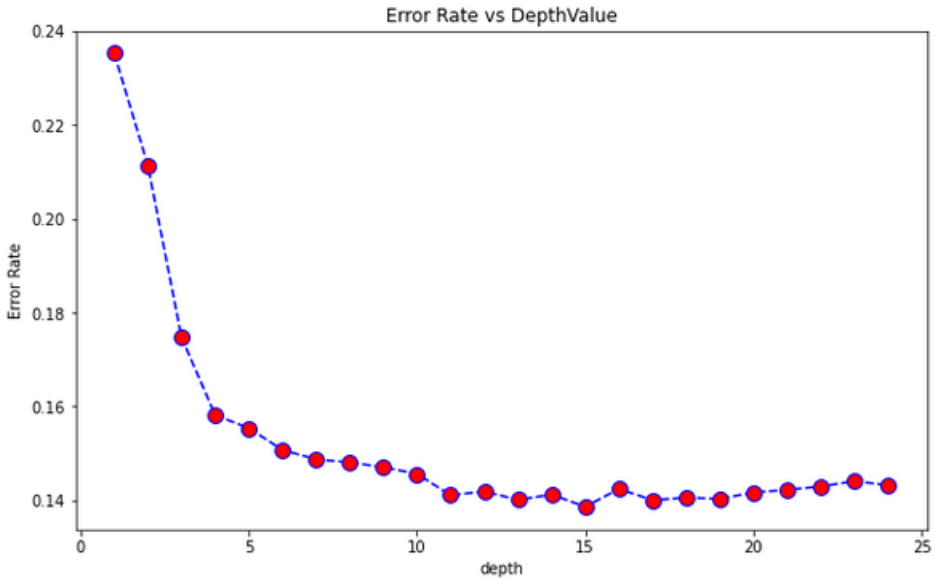

Figura 8. Erros obtidos usando *Random Forest*.

Apresenta-se de seguida as tabelas de resultados para o *dataset* usando *Random Forest*:

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos	0.861	0.88	0.94	0.78	0.61
Nulos substituídos pela moda	0.861	0.88	0.94	0.78	0.61
KNN Imputer para nulos	0.861	0.88	0.94	0.77	0.61
Nulos removidos e *Standard Scaling*	0.861	0.88	0.94	0.78	0.61

Tabela 16. Resultados *Random Forest*.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos	0.839	0.92	0.86	0.64	0.77
Nulos substituídos pela moda	0.841	0.92	0.86	0.64	0.78
KNN Imputer para nulos	0.836	0.93	0.85	0.63	0.80
Nulos removidos e *Standard Scaling*	0.842	0.92	0.86	0.65	0.79

Tabela 17. Resultados *Random Forest* com *RandomUnderSampling* e *RandomOverSampling*.

Dataset Treino	Accuracy	Precision 0	Recall 0	Precision 1	Recall 1
Nulos removidos	0.813	0.92	0.82	0.59	0.78
Nulos substituídos pela moda	0.810	0.92	0.82	0.59	0.78
KNN Imputer para nulos	0.831	0.90	0.87	0.64	0.71
Nulos removidos e *Standard Scaling*	0.811	0.94	0.80	0.58	0.84

Tabela 18. Resultados *Random Forest* com *SMOTE* e *ENN*.

4 Análise de resultados obtidos

Nesta secção analisaremos todos os dados obtidos usando as 4 abordagens anteriormente explicadas, bem como os dados quando usamos estas 4 abordagens depois de aplicado *Under/Over Sampling* e *SMOTE/ENN*.

Como explicado na secção 3.3, não será necessário explicar os dados relativamente ao uso de ***K-Means Clustering*** pois este foi só usado para aumentar o conhecimento do grupo e não para tentar obter resultados bons.

Uma vez que ***Logistic Regression*** foi o primeiro modelo de aprendizagem a ser aplicado, este foi o nosso ponto de partida no que toca a comparações de valores com via a saber se diferentes modelos melhoram ou não. Posto isto, é de notar que a melhor abordagem para ***Logistic Regression***, quando não usado qualquer balanceamento, foi a de *Standard Scaling* com aproximadamente 82% de acerto. Quando aplicado balanceamento com *Under/Over Sampling*, a abordagem de *Standard Scaling* continua a ser a melhor mas com uma redução de acerto para 79%. Por fim, quando aplicado *SMOTE* e *ENN*, todos os valores pioram imenso, pelo que se descartou este tipo de balanceamento para ***Logistic Regression***.

Em relação a ***K-Nearest Neighbors***, quando este não é aplicado balanceamento podemos ver que, mais uma vez, *Standard Scaling* é a melhor abordagem com 84% de acerto. Já com o uso de *Under/Over Sampling*, este *Standard Scaling* continua a ser a melhor abordagem mas com uma redução para 81% de acerto. Também de reparar que o valor de K deste, passa de 20 para 10, uma vez que é quando apresenta menores erros. Uso de *SMOTE* e *ENN* não permitiu qualquer melhoria, até reduziu e bastante a percentagem de *accuracy*. Este modelo de aprendizagem obteve resultados bem melhores que ***Logistic Regression***.

Passando agora para ***Support Vector Machines (SVM)***, sem balanceamento conseguiu atingir uma *accuracy* de 85% quando usado *Standard Scaling*, o que é uma melhoria em comparação com os 84% de ***K-Nearest Neighbors***. Com o uso de *Under/Over Sampling*, não houve uma melhoria comparado com quando não se tem balanceamento, atingindo no máximo uma percentagem de 82%, e usando *SMOTE* e *ENN* também não houve melhoria, atingindo no máximo 79% de *accuracy*.

Os resultados obtidos por ***Naive Bayes*** foram fracos, não passando de 80% quando não aplicado balanceamento. No entanto, pela primeira vez, notou-se uma melhoria quando aplicado *Under/Over Sampling*, chegando a atingir 81% de *accuracy* quando usado *Standard Scaling* e esta melhoria para a mesma abordagem também se notou quando aplicado *SMOTE* e *ENN*.

Por fim, aplicou-se ***Random Forest*** e os resultados melhoraram bastante. Em que, sem balanceamento, independentemente da abordagem, se obteve 86.1% de *accuracy* sendo este o nosso melhor resultado obtido quando comparado com todos os modelos com/sem balanceamento anteriormente aplicados. Quando aplicado balanceamento em ***Random Forest*** não se obteve qualquer melhoria.

Conclui-se então que, para o *dataset* em mão, a melhor abordagem era usar *Standard Scaling*, sem qualquer balanceamento, para ***Random Forest***.

5 Conclusão

A realização deste trabalho permitiu a análise detalhada das diferentes fases de desenvolvimento de um modelo para previsão sobre um conjunto de dados, desde o pré-processamento à aplicação de modelos que melhor se adequa ao *dataset*.

O *dataset* fornecido tinha um desbalanceamento significativo entre as classes, o que dificultou, de certa forma, a capacidade de atingir resultados favoráveis. Neste caso, possuía-se 37155 instâncias onde o salário era <=50k e 11687 instâncias >50k. Devido à existência desta disparidade a *accuracy* demonstrada pelos modelos podia ser ilusória, com isso em mente o grupo procurou maximizar tanto a *accuracy* como o *recall* de >50k. Assim, pôde-se concluir que para o *dataset* em questão, a melhor *accuracy* obtida foi através de *Random Forest*, 86.1%, sem o uso de *Over/Under Sampling*, no entanto a aplicação de *SMOTEENN* permitiu a obtenção de valores bastante satisfatórios para o *recall* de >50k, principalmente em *Support Vector Machines*, atingindindo-se o valor de 87% no melhor caso.

Como extensão ao trabalho descrito, podia ainda ter sido aplicado *fine tuning* dos parâmetros dos algoritmos, através do uso de *GridSearchCV* e ainda, outras técnicas de *Over/Under sampling*, com vista a melhorar ainda mais os resultados obtidos.

Bibliografia

[1] Informação sobre as diferentes formas de se fazer pré-processamento dos dados, https://machinelearningmastery.com/, last visited 28 December 2020

[2] LogisticRegression, sklearn website, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html, last visited 23 December 2020.

[3] KNN, sklearn website, https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html, last visited 23 December 2020.

[4] K-Means Clustering, sklearn website, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html, last visited 23 December 2020.

[5] SVM, sklearn website, https://scikit-learn.org/stable/modules/svm.html, last visited 23 December 2020.

[6] Naive Bayes, sklearn website, https://scikit-learn.org/stable/modules/naive_bayes.html, last visited 23 December 2020.

[7] Random Forest, sklearn website, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, last visited 23 December 2020.