Aprendizagem e Extração de Conhecimento

Ana Gil, Beatriz Rocha, Hugo Matias e João Abreu

Universidade do Minho, Departamento de Informática, 4710-057 Braga, Portugal
e-mail: {a85266,a84003,a85370,a84802}@alunos.uminho.pt

Resumo O presente trabalho prático tem como objetivo a análise e preparação de um conjunto de dados relativo às características de funcionários de múltiplas empresas, o desenvolvimento de modelos preditivos como forma de prever o nível salarial anual do indivíduo e a validação e otimização da *performance* dos modelos preditivos desenvolvidos. Para isto será utilizado o ambiente de desenvolvimento *Python*, aplicando as funcionalidades disponíveis na biblioteca *Sklearn*.

Keywords: Sistemas de Aprendizagem · Extração de Conhecimento · *Machine Learning* · *Data Science* · *Data Mining*

1 Introdução

Os modelos preditivos aplicam técnicas de *Machine Learning* sobre conjuntos de dados, de forma a prever a probabilidade de acontecimentos futuros. Estes modelos permitem solucionar problemas de grande dimensão, sendo utilizados em várias áreas, nomeadamente no diagnóstico médico, na deteção de *spam* e na previsão meteorológica.

O problema que nos foi apresentado consiste em prever o nível salarial anual de um indivíduo, tendo em conta as suas características (como a sua idade, a sua raça, *etc.*). Para a resolução do mesmo, iremos começar por analisar, preparar e visualizar a distribuição do conjunto de dados, de modo a interpretar a sua relação com a nossa *target variable* (i.e., o nível salarial anual). A partir daqui, iremos desenvolver diferentes modelos preditivos e validar a sua *performance* para, no fim, selecionarmos aquele que apresenta melhor desempenho, baseando-nos, para isso, na acurácia do modelo.

2 *Data Acquisition*

Visto que os conjuntos de dados necessários para a resolução deste problema nos foram fornecidos, pudemos passar esta fase à frente. Tal como foi mencionado na secção anterior, o problema em causa consiste na previsão do nível salarial anual de um indivíduo, tendo em conta as suas características, portanto os *datasets* em questão dizem respeito a essas características.

Desde já, podemos fazer uma análise prévia e enumerar aquelas que nos parecem estar menos/mais correlacionadas com a variável a prever. Existem alguns atributos que nos parecem influenciar bastante o nível salarial anual de um funcionário, nomeadamente as horas de trabalho por semana e a ocupação do funcionário. Por outro lado, à primeira vista, o atributo `fnlwgt`, por exemplo, não parece estar muito correlacionado com a *target variable* em questão.

3 *Data Visualization*

3.1 Informação das características

De forma a analisar mais detalhadamente os conjuntos de dados que nos foram fornecidos, primeiro é crucial entender que características estão presentes nos mesmos.

Tabela 1. Descrição dos *datasets*

	Descrição
age	idade do funcionário
workclass	entidade para a qual o funcionário trabalha
fnlwgt	número de pessoas que a entrada representa
education	habilitação literária do funcionário
education-num	representação numérica da característica anterior
marital-status	estado civil do funcionário
occupation	ocupação (emprego) do funcionário
relationship	relação do funcionário
race	raça do funcionário
sex	género do funcionário
capital-gain	ganho de capital do funcionário
capital-loss	perda de capital do funcionário
hours-per-week	número de horas que o funcionário trabalha por semana
native-country	país de nascimento do funcionário
salary-classification	nível salarial anual do funcionário (variável a prever)

3.2 Características numéricas

Nesta secção, iremos apresentar várias tabelas, gráficos e histogramas para que seja possível analisar as características numéricas à exceção de `education-num` por já se encontrar discretizada e de `fnlwgt` por não influenciar a variável a prever.

3.2.1 Medidas de tendência e dispersão

De maneira a possibilitar uma análise numérica dos dados, obtivemos alguns indicadores estatísticos, como o número de ocorrências, a média, o desvio padrão, o valor mínimo, o 25.º, 50.º e 75.º percentis e o valor máximo.

Tabela 2. Estatísticas descritivas do conjunto de dados

	hours-per-week	capital-loss	capital-gain	age
count	30162.000000	30162.000000	30162.000000	30162.000000
mean	40.931238	88.372489	1092.007858	38.437902
std	11.979984	404.298370	7406.346497	13.134665
min	1.000000	0.000000	0.000000	17.000000
25%	40.000000	0.000000	0.000000	28.000000
50%	40.000000	0.000000	0.000000	37.000000
75%	45.000000	0.000000	0.000000	47.000000
max	99.000000	4356.000000	99999.000000	90.000000

3.2.2 *Heat map*

Para visualizarmos a correlação existente entre estas características e a *target variable*, decidimos construir um *heat map*.

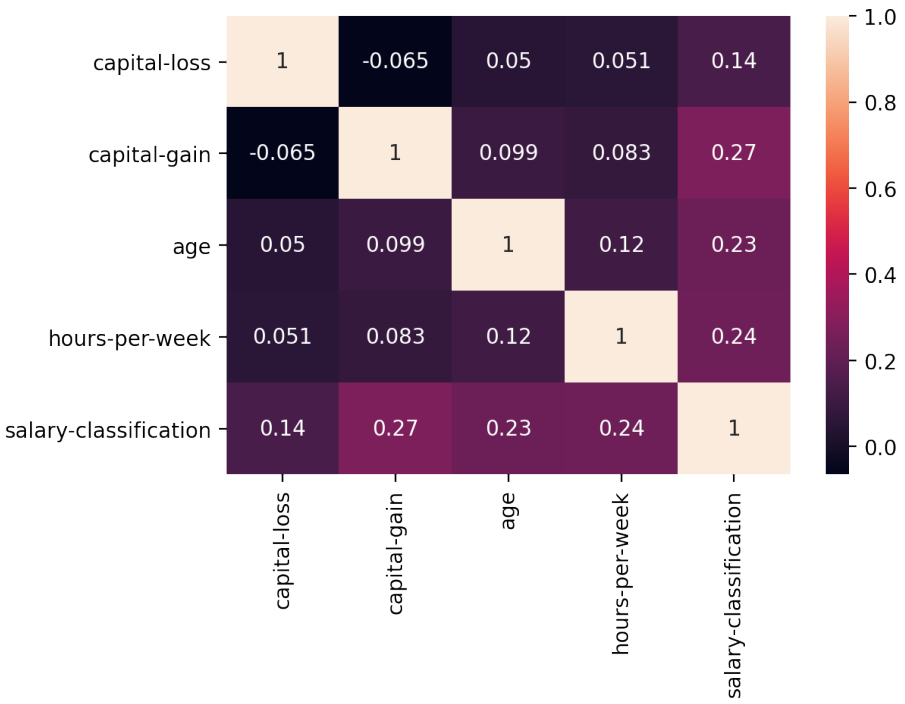

Figura 1. Correlação entre as características numéricas e a *target variable*

3.2.3 Histogramas

De forma a tentar identificar alguns padrões que auxiliem a compreensão dos dados e previsão dos resultados, construímos vários histogramas para observar a ocorrência das características numéricas.

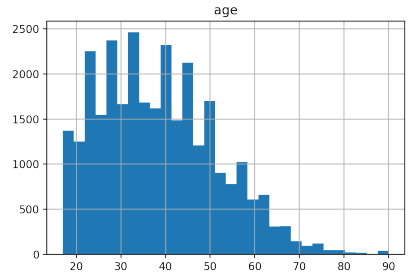

Figura 2. Atributo `age`

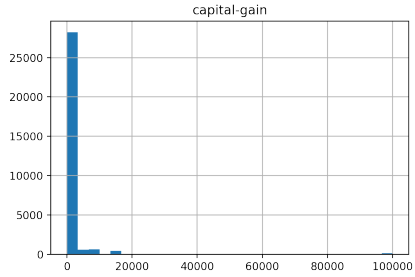

Figura 3. Atributo `capital-gain`

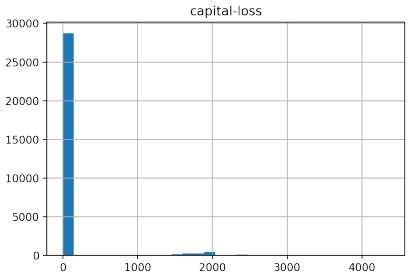

Figura 4. Atributo `capital-loss`

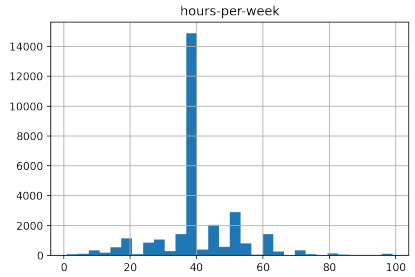

Figura 5. Atributo `hours-per-week`

3.2.4 *Outliers*

Após a observação dos histogramas e das medidas de tendência e dispersão, constatámos que existem alguns valores muito maiores ou muito menores do que os restantes e, deste modo, torna-se fundamental perceber com clareza quais as características que possuem *outliers*. Feito isto, pudemos concluir que `age` é uma delas.

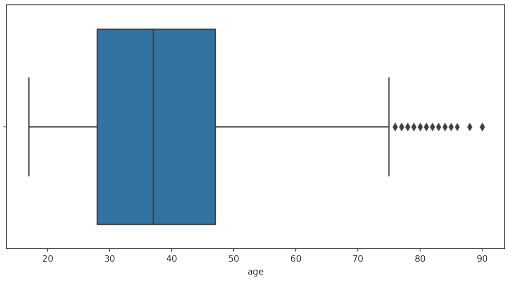

Figura 6. Atributo `age`

3.2.5 Outros gráficos relevantes

Uma vez que a idade de um funcionário e o número de horas que este trabalha por semana podem ter influência, criámos o seguinte gráfico para apoiar a nossa afirmação. Efetivamente, verifica-se que a maior parte da população que tem um nível salarial anual inferior/igual a 50K é mais jovem e trabalha menos horas, enquanto a maior parte da população que tem um nível salarial anual superior a 50K é mais velho e trabalha mais horas.

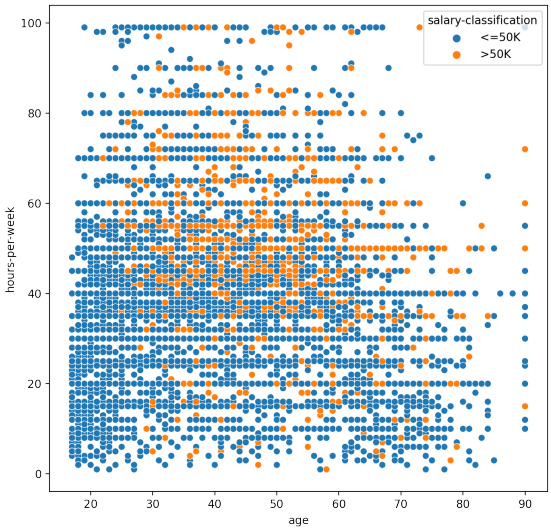

Figura 7. Nível salarial anual na relação idade e horas de trabalho por semana

3.3 Características categóricas

Nesta secção, iremos apresentar várias tabelas e gráficos para que seja possível analisar as características categóricas.

3.3.1 Tabelas de frequência

Para começar, imprimimos tabelas de frequência absoluta para cada uma destas características, para que pudéssemos ver a distribuição dos seus respetivos atributos. De seguida, apresentamos algumas dessas tabelas.

Tabela 3. Frequência absoluta dos vários atributos da característica `education`

	education
HS-grad	9840
Some-college	6678
Bachelors	5044
Masters	1627
Assoc-voc	1307
11th	1048
Assoc-acdm	1008
10th	820
7th-8th	557
Prof-school	542
9th	455
12th	377
Doctorate	375
5th-6th	288
1st-4th	151
Preschool	45

Tabela 4. Frequência absoluta dos vários atributos da característica `marital-status`

	marital-status
Married-civ-spouse	14065
Never-married	9726
Divorced	4214
Separated	939
Widowed	827
Married-spouse-absent	370
Married-AF-spouse	21

Tabela 5. Frequência absoluta dos vários atributos da característica `native-country`

	native-country
United-States	27504
Mexico	610
Philippines	188
Germany	128
Puerto-Rico	109
Canada	107
El-Salvador	100
India	100
Cuba	92
England	86
Jamaica	80
South	71
Italy	68
China	68
Dominican-Republic	67
Vietnam	64
Guatemala	63
Japan	59
Columbia	56
Poland	56
Iran	42
Haiti	42
Taiwan	42
Portugal	34
Nicaragua	33
Peru	30
Greece	29
France	27
Ecuador	27
Ireland	24
Hong	19
Trinadad&Tobago	18
Cambodia	18
Thailand	17
Laos	17
Yugoslavia	16
Outlying-US(Guam-USVI-etc)	14
Hungary	13
Honduras	12
Scotland	11
Holand-Netherlands	1

Tabela 6. Frequência absoluta dos vários atributos da característica `occupation`

	occupation
Prof-specialty	4038
Craft-repair	4030
Exec-managerial	3992
Adm-clerical	3721
Sales	3584
Other-service	3212
Machine-op-inspct	1966
Transport-moving	1572
Handlers-cleaners	1350
Farming-fishing	989
Tech-support	912
Protective-serv	644
Priv-house-serv	143
Armed-Forces	9

Tabela 7. Frequência absoluta dos vários atributos da característica `workclass`

	workclass
Private	22286
Self-emp-not-inc	2499
Local-gov	2067
State-gov	1279
Self-emp-inc	1074
Federal-gov	943
Without-pay	14

3.3.2 Gráficos circulares

Nesta secção, apresentam-se gráficos circulares com a frequência relativa dos atributos das restantes características categóricas para que possamos ver novamente a sua distribuição.

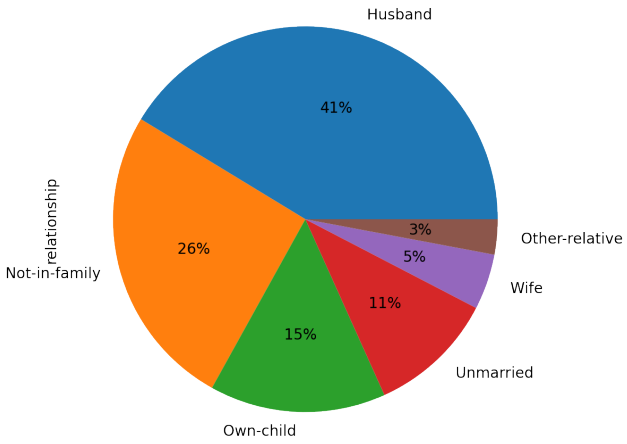

Figura 8. Frequência relativa dos vários atributos da característica `relationship`

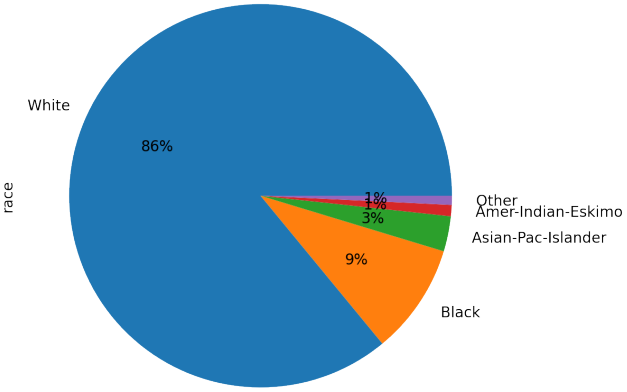

Figura 9. Frequência relativa dos vários atributos da característica `race`

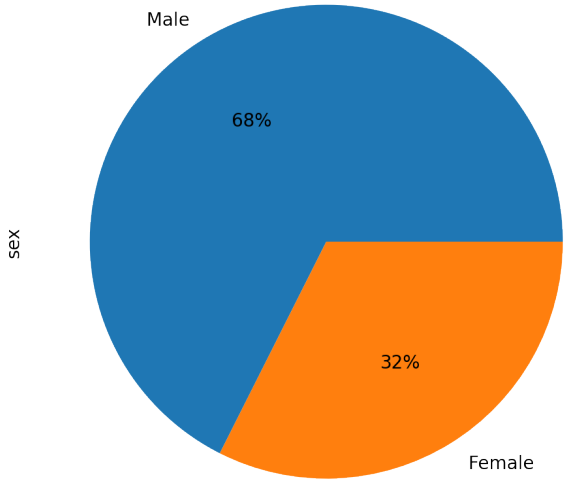

Figura 10. Frequência relativa dos vários atributos da característica `sex`

4 *Data Preprocessing*

De maneira a ajustar os dados contidos em ambos os *datasets* aos modelos de extração de conhecimento implementados, foi necessário proceder ao seu pré-processamento.

4.1 *Missing values analysis*

Em primeiro lugar, reparámos que existiam alguns campos com valor desconhecido ("?"), então optámos por trocar esses caracteres por "NaN".

```
for data in training:
    training[data].replace(' ?', np.nan, inplace=True)

for data in test:
    test[data].replace(' ?', np.nan, inplace=True)
```

Os valores desconhecidos são bastante comuns e podem ter origens variadas, como o facto de não terem sido registados ou o facto de estarem corrompidos. Uma das estratégias para tratar este tipo de casos consiste em descartar as linhas que possuem, pelo menos, um valor "NaN". Contudo, para sabermos se esse seria o procedimento mais correto, calculámos a dimensão que esta falta de valores tinha no *dataset* de treino.

```
nulos = training.isnull().any(axis=1).sum()
total, _ = training.shape
print(f"Percentage of NaN values in the dataset ->
{round(nulos/total* 100, 2)}%")
```

Posto isto, verificámos que o número de linhas que possuem, pelo menos, um valor "NaN", corresponde a apenas cerca de 7% de todos os registos do *dataset* e, portanto, resolvemos eliminar essas linhas.

```
training.dropna(inplace=True)
test.dropna(inplace=True)
```

4.2 *Dataset treatment*

Posto isto, analisámos as várias colunas dos *datasets* e percebemos que existia uma coluna redundante (atributo `education`), visto que já existia uma coluna com a mesma informação, mas com valores numéricos (atributo `education-num`).

```
training = training.drop("education", 1)
test = test.drop("education", 1)
```

Visto que características como `hours-per-week`, `age`, `occupation`, `workclass`, `marital-status`, `race` e `native-country` parecem ser bastante influenciadoras no que toca à previsão do salário, decidimos discretizá-las em categorias para realçar este efeito.

Inicialmente, criámos uma função que discretiza uma característica de acordo com os grupos passados como parâmetro.

```
def discretizer(data, var, bins, group_names):
    bin_value = bins
    group = group_names
    data[var] = pd.cut(data[var], bin_value, labels=group)
```

De seguida, usámos essa função para discretizar a característica `hours-per-week`. Na secção anterior, pudemos constatar que 40 horas é o valor mais frequente, o que faz sentido, visto que corresponde a 8 horas por dia. Assim sendo, vamos discretizar esta característica em torno desse valor.

```
discretizer(training, 'hours-per-week', [0,35,40,60,100], ['Low',
'Medium', 'High','VeryHigh'])
discretizer(test, 'hours-per-week', [0,35,40,60,100], ['Low',
'Medium', 'High','VeryHigh'])
```

Da mesma forma, pudemos discretizar a característica `age`. O valor mínimo desta característica no *dataset* de treino é 17 e o valor máximo é 90, logo podemos discretizá-la em `Young`, `Middle_aged` e `Old`.

```
discretizer(training, 'age', [16,30,55,90], ['Young', 'Middle_aged'
, 'Old'])
discretizer(test, 'age', [16,30,55,90], ['Young', 'Middle_aged'
, 'Old'])
```

No que toca à característica `occupation`, decidimos considerar todos os empregos que contivessem a palavra `managerial` ou `specialty` como `Highskill` e os restantes como `Lowskill`.

```
def occupation_discretizer(x):
    if re.search('managerial', x):
        return 'Highskill'
    elif re.search('specialty',x):
        return 'Highskill'
    else:
        return 'Lowskill'

training['occupation']=training.occupation.apply(lambda x:
x.strip()).apply(lambda x: occupation_discretizer(x))
test['occupation']=test.occupation.apply(lambda x:
x.strip()).apply(lambda x: occupation_discretizer(x))
```

Quanto à característica `workclass`, considerámos todas as classes de trabalho que contivessem a palavra `Private` como `Private`, todas as que contivessem a palavra `Self` como `selfempl`, todas as que contivessem a palavra `gov` como `gov` e as restantes como `others`.

```
def workclass_discretizer(x):
    if re.search('Private', x):
        return 'Private'
    elif re.search('Self', x):
        return 'selfempl'
    elif re.search('gov', x):
        return 'gov'
    else:
        return 'others'


training['workclass']=training.workclass.apply(lambda x:
x.strip()).apply(lambda x: workclass_discretizer(x))
test['workclass']=test.workclass.apply(lambda x:
x.strip()).apply(lambda x: workclass_discretizer(x))
```

Todas as instâncias da característica `marital-status` que começassem com a palavra `Married` passaram a fazer parte do grupo `Married` e as restantes passaram a fazer parte do grupo `Single`.

```
training['marital-status']=training['marital-status'].apply(lambda
x: 'Married' if x.startswith('Married',1) else 'Single')
test['marital-status']=test['marital-status'].apply(lambda
x: 'Married' if x.startswith('Married',1) else 'Single')
```

Já todas as instâncias da característica `race` que contivessem a palavra `White` passaram a fazer parte do grupo `White` e as restantes passaram a fazer parte do grupo `Other`.

```
training['race']=training['race'].apply(lambda x: 'White' if x==
' White' else 'Other')
test['race']=test['race'].apply(lambda x: 'White' if x==
' White' else 'Other')
```

Por fim, as instâncias da característica `race` que contivessem a palavra `United-States` passaram a fazer parte do grupo `Native` e as restantes passaram a fazer parte do grupo `Immigrant`.

```
training['native-country']=training['native-country'].apply(lambda
x: 'Native' if x==' United-States' else 'Immigrant')
test['native-country']=test['native-country'].apply(lambda
x: 'Native' if x==' United-States' else 'Immigrant')
```

Uma vez que os modelos que iremos aplicar mais à frente não conseguem lidar com características categóricas, teremos de as codificar como numéricas. Para aquelas que se tratam de características ordinais ou binárias, optámos por substituir os respetivos atributos um a um pelo valor desejado.

```
encoding = {"age": {"Young": 0, "Middle_aged": 1, "Old": 2},
            "hours-per-week": {"Low": 0, "Medium": 1, "High": 2,
            "VeryHigh": 3},
            "marital-status": {"Married": 0, "Single": 1},
            "race": {"White": 0, "Other": 1},
            "occupation": {"Lowskill": 0, "Highskill": 1},
            "sex": {" Male": 0, " Female": 1},
            "native-country": {"Native": 0, "Immigrant":1},
            "salary-classification": {" <=50K": 0, " >50K": 1}}

training = training.replace(encoding)
test = test.replace(encoding)
```

Nas restantes características, optámos por obter variáveis binárias para cada um dos respetivos atributos.

```
training = pd.get_dummies(training, columns=['workclass',
'relationship'])
test = pd.get_dummies(test, columns=['workclass',
'relationship'])
```

4.3 *Dataset balancing*

Tal como podemos observar na Figura 11, o *dataset* de treino encontra-se desbalanceado.

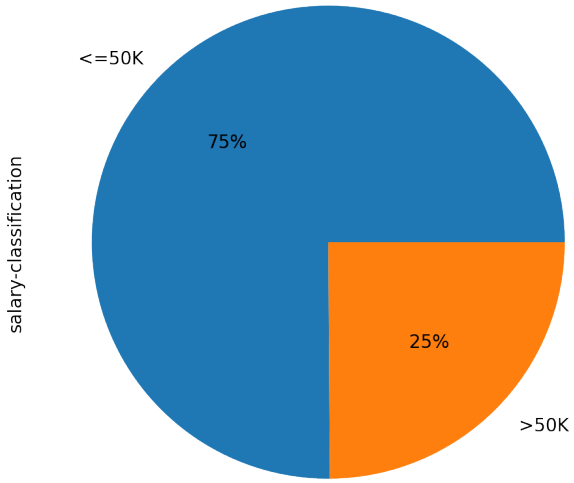

Figura 11. Percentagem de registos por valor da *target variable*

Das 30162 observações que compõem o *dataset* de treino, apenas 7508 representam situações em que um funcionário recebe um salário anual superior a 50K. Se nada for feito em contrário, ao treinar um modelo com este conjunto de dados, este tenderá a classificar todas as observações futuras como pertencendo à classe maioritária (<=50K). Assim sendo, achámos por bem proceder ao balanceamento do *dataset*.

As técnicas de balanceamento que optámos por aplicar, podem ser vistas na figura que se segue.

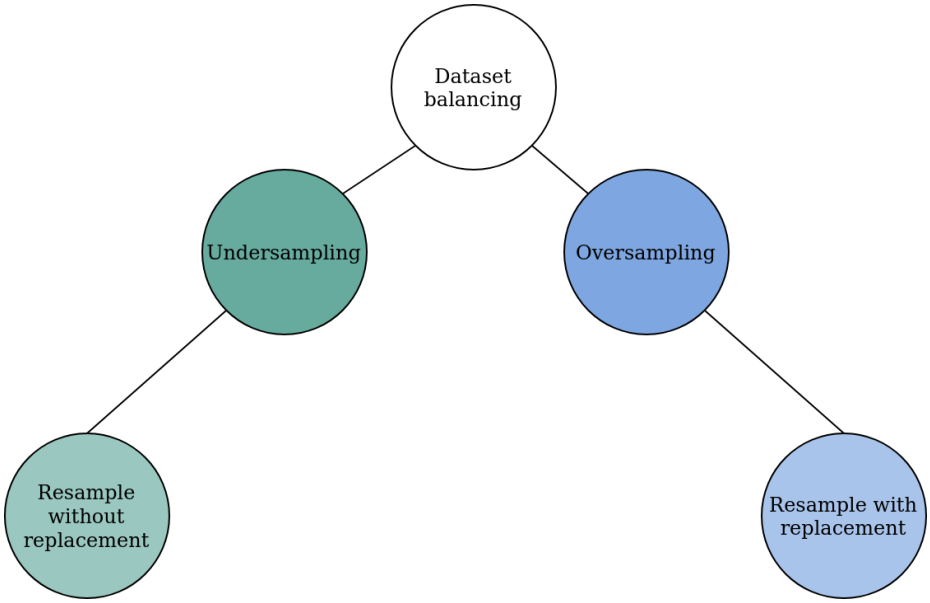

Figura 12. Técnicas de balanceamento aplicadas

4.3.1 *Undersampling*

A técnica de *undersampling* consiste em reduzir o número de observações da classe maioritária ao número de observações da classe minoritária. Neste contexto, a forma mais simples de balancear um *dataset* passa por selecionar exemplos aleatórios da classe maioritária (*resample without replacement*).

```
def underSampler(X_train, y_train):
    rus = RandomUnderSampler(random_state=0)
    X_balanced, y_train = rus.fit_sample(X_train, y_train)
    X_balanced, y_train = shuffle(X_balanced, y_train,
    random_state=0)
    return X_balanced, y_train;
```

4.3.2 *Oversampling*

A técnica de *oversampling* consiste em igualar o número de observações da classe minoritária ao número de observações da classe maioritária. Neste contexto, a forma mais simples de balancear um *dataset* passa por repetir múltiplas vezes as observações da classe minoritária (*resample with replacement*).

```
def overSampler(X_train, y_train):
    ros = RandomOverSampler(random_state=0)
```

```
X_balanced, y_train = ros.fit_sample(X_train, y_train)
X_balanced, y_train = shuffle(X_balanced, y_train,
random_state=0)
return X_balanced, y_train;
```

Na tabela que se segue podem ser vistos os resultados obtidos pelas diferentes técnicas.

Tabela 8. Resultados obtidos pelas diferentes técnicas de balanceamento

	SVM	KNN	Naive Bayes	AdaBoost	Bagging	Stacking
Sem balanceamento	0.79	0.77	0.79	0.86	0.79	0.81
Oversampling	0.79	0.63	0.79	0.8	0.79	0.81
Undersampling	0.79	0.61	0.79	0.81	0.79	0.82

Visto que obtivemos resultados muito semelhantes para ambas as técnicas, optámos por *undersampling*, uma vez que demora menos tempo a executar e não introduz dados falsos no *dataset*.

4.4 *Feature selection*

Feature selection, como o próprio nome indica, consiste em selecionar as *features* que têm um maior impacto na *target variable* em questão e eliminar aquelas que são menos relevantes e se revelam prejudiciais no que toca ao desempenho de um modelo. Alguns dos benefícios de proceder a esta técnica passam por:

- Redução de *overfitting*;
- Melhoria na precisão;
- Redução do tempo de treino.

Na figura que se segue, podem ser vistos os vários métodos de *feature selection* que existem e foram aplicados.

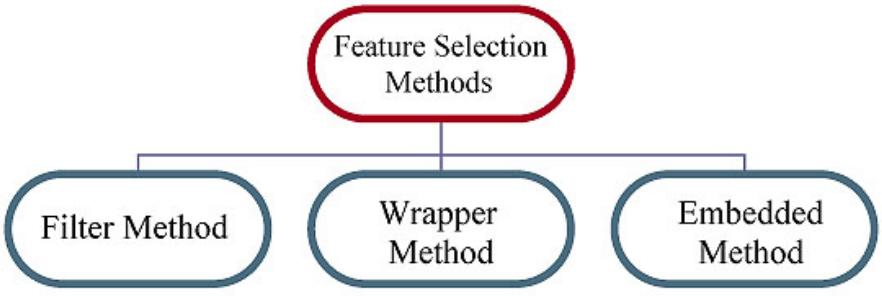

Figura 13. Diferentes métodos de *feature selection*

4.4.1 *Filter method*

Este método avalia a relevância das características independentemente dos modelos de previsão e, consequentemente, os últimos consideram apenas aquelas que obedecem a um certo critério.

Figura 14. *Filter method*

4.4.1.1 *Variance Threshold*

Este método consiste numa abordagem simples no que toca à *feature selection*. Em suma, este remove todas as características cuja variação não atinge o limite passado como parâmetro. Por omissão, o limite passado como parâmetro é 0 e, portanto, são eliminadas todas as características com esta variação. O resultado da aplicação deste método pode ser visto de seguida e, daqui, podemos retirar que, uma vez que não foram encontradas variações nulas entre características, todas se mantiveram.

```
[3.52729416e-01, 1.44055700e+06, 6.73852724e+00, 2.38703153e-01,
2.24836271e-01, 1.08825049e-01, 1.95080065e-01, 9.99990000e+04,
3.90000000e+03, 5.75279633e-01, 7.50358515e-02, 2.05433348e-01,
1.28875976e-01, 3.32867282e-04, 1.17931456e-01, 2.49172327e-01,
1.65654163e-01, 2.13662344e-02, 9.04472097e-02, 7.13395278e-02,
5.80981246e-02]
```

4.4.1.2 *SelectKBest*

Este método remove todas as características, exceto as k características com melhor pontuação. Para isso, realizámos uma pesquisa do valor de k que poderia contribuir para melhores resultados a nível de *accuracy* com o modelo *SVC*.

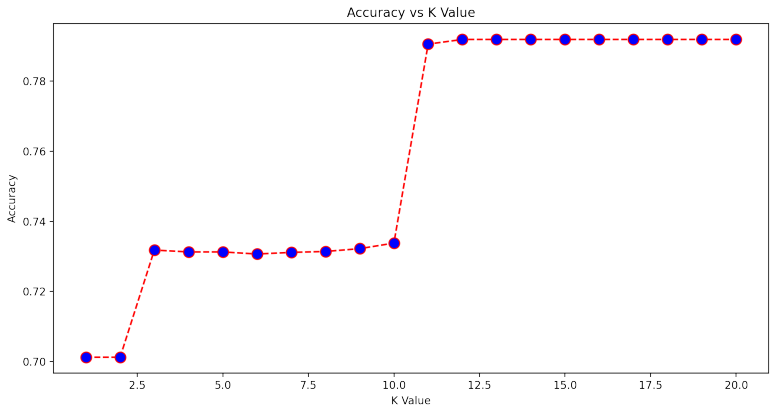

Figura 15. Pesquisa do melhor valor de k

Através do gráfico é possível verificar que o valor de k igual a 12 é aquele que representa um melhor desempenho a nível de *accuracy* (cerca de 79,18%). Este valor de k resulta na seleção das características que a seguir se apresentam.

Tabela 9. Características selecionadas pelo método *SelectKBest*

Feature
age
education-num
marital-status
occupation
sex
capital-gain
capital-loss
hours-per-week
relationship_ Husband
relationship_ Not-in-family
relationship_ Own-child
relationship_ Unmarried

4.4.2 *Wrapper method*

Este método usa combinações de características com o objetivo de encontrar a combinação ideal que resulta num melhor poder de previsão. Este testa cada

característica em relação aos modelos de teste criados com as mesmas, de maneira a avaliar os resultados.

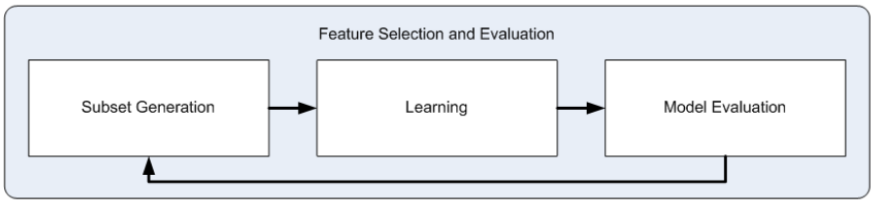

Figura 16. *Wrapper method*

4.4.2.1 *Recursive Feature Elimination*

Dado um estimador externo que atribui pesos a características, o método *Recursive Feature Elimination (RFE)*, como o próprio nome indica, seleciona recursivamente conjuntos de características cada vez menores. Inicialmente, o estimador é treinado com todas as características. Depois, as características menos importantes são removidas desse conjunto e o processo é repetido recursivamente até que o número de características desejadas seja atingido. No nosso caso, optámos por deixar o método escolher as *features* ideais (que podem ser vistas na tabela que se segue) e aplicámos o mesmo com o estimador *Logistic Regression*.

Tabela 10. Características selecionadas pelo método *RFE*

Feature
age
marital-status
occupation
sex
hours-per-week
workclass_others
relationship_ Husband
relationship_ Other-relative
relationship_ Own-child
relationship_ Wife

4.4.3 *Embedded method*

Este método completa o processo de *feature selection* com a construção do próprio algoritmo de aprendizagem. Noutras palavras, este seleciona as carac-

terísticas durante a fase de treino, sendo esse o motivo de se chamar *embedded method*. Um algoritmo de aprendizagem tira proveito do seu próprio processo de seleção de características e procede à classificação das mesmas simultaneamente.

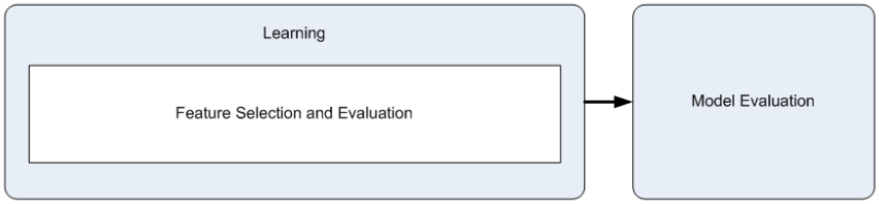

Figura 17. *Embedded method*

4.4.3.1 *Principal Component Analysis*

Principal Component Analysis (PCA) é um método que recorre a álgebra linear para transformar o *dataset* num *dataset* mais pequeno. Este método destaca-se pelo facto de ser possível escolher o número de dimensões, bem como o componente principal no resultado transformado. Este consiste na criação de novas características (componentes principais), que se tratam de composições lineares das originais e os valores das primeiras são conhecidos como pontuações dos componentes principais (sendo que entre 0.3 a 0.49 é considerado "fraco", entre 0.5 a 0.75 é considerado "moderado"e superior a 0.75 é considerado "forte"). É importante mencionar que o número máximo de novas características é equivalente ao número de características originais.

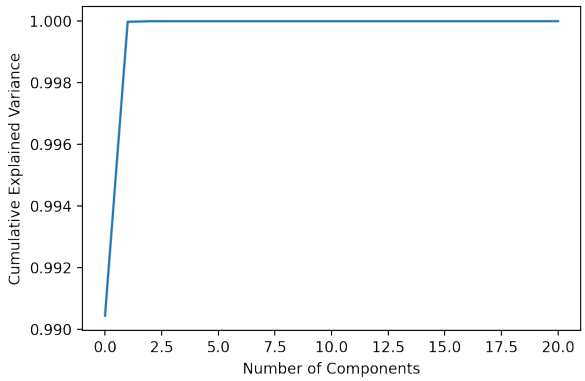

Figura 18. Pesquisa do número de componentes necessários para descrever os dados

Uma parte vital de usar este método é a capacidade de estimar quantos componentes são necessários para descrever os dados. Isto pode ser determinado olhando para a taxa de variação explicada acumulada como uma função do número de componentes. Esta função (que pode ser vista na figura acima) quantifica quanto da variação total das 21 características está contida nos primeiros N componentes. Daqui, podemos observar que precisamos de 19 componentes para reter 100% da variação e na tabela seguinte podemos ver quais são esses 19 componentes.

Tabela 11. Características selecionadas pelo método *PCA*

Feature
age
fnlwgt
education-num
marital-status
occupation
race
sex
capital-gain
capital-loss
hours-per-week
native-country
workclass_Private
workclass_gov
workclass_others
workclass_selfempl
relationship_ Husband
relationship_ Not-in-family
relationship_ Other-relative
relationship_ Own-child

4.4.3.2 *Feature Importance*

Este método refere-se a uma classe de técnicas para atribuir pontuações às características passadas como argumento a um modelo preditivo que indica a importância relativa de cada característica quando se faz uma previsão. Para isso, resolvemos usar o modelo *Extra Trees Classifier* e os resultados podem ser vistos de seguida.

Tabela 12. Características selecionadas pelo método *Feature Importance*

	Importance
fnlwgt	0.330851
education-num	0.115385
marital-status	0.097309
relationship_ Husband	0.065748
capital-gain	0.064775
occupation	0.064027
age	0.052612
hours-per-week	0.048835
relationship_ Not-in-family	0.030828
relationship_ Own-child	0.026898
capital-loss	0.021018
relationship_ Wife	0.017708
sex	0.015783
race	0.010138
relationship_ Unmarried	0.009578
native-country	0.009058
workclass_Private	0.006268
workclass_selfempl	0.004944
workclass_gov	0.004864
relationship_ Other-relative	0.003096
workclass_others	0.000276

As *features* selecionadas pelos diferentes métodos variam um pouco, pelo que optámos por verificar quais tinham um maior impacto nos resultados. Assim sendo, fomos retirando as características uma a uma e imprimindo uma matriz de classificação para todos os modelos, deixando, então, ficar apenas as características que resultam em melhor *accuracy*. Posto isto e visualizando a tabela que se apresenta em baixo, podemos concluir que devem ser selecionadas todas as *features* exceto `fnlwgt`, `relationship_ Not-in-family` e `workclass_selfempl`.

Tabela 13. Valores de *accuracy* de acordo com as *features* eliminadas

	SVM	KNN	Naive Bayes	AdaBoost	Bagging	Stacking
	0.79	0.61	0.79	0.81	0.79	0.82
fnlwgt	0.79	0.82	0.81	0.81	0.79	0.81
fnlwgt, capital-gain	0.77	0.77	0.71	0.78	0.77	0.72
fnlwgt, capital-loss	0.79	0.81	0.82	0.8	0.79	0.81
fnlwgt, race	0.79	0.82	0.82	0.8	0.79	0.81
fnlwgt, relationship_ Not-in-family	0.79	0.82	0.82	0.81	0.79	0.81
fnlwgt, relationship_ Not-in-family, relationship_ Unmarried	0.79	0.82	0.81	0.81	0.79	0.81
fnlwgt, relationship_ Not-in-family, workclass_selfempl	0.79	0.82	0.82	0.81	0.79	0.82
fnlwgt, relationship_ Not-in-family, workclass_selfempl, workclass_Private	0.79	0.81	0.82	0.81	0.79	0.82
fnlwgt, relationship_ Not-in-family, workclass_selfempl, workclass_gov	0.79	0.81	0.82	0.81	0.79	0.81
fnlwgt, relationship_ Not-in-family, workclass_selfempl, native-country	0.79	0.82	0.82	0.8	0.79	0.81
fnlwgt, relationship_ Not-in-family, workclass_selfempl, education-num	0.79	0.77	0.81	0.78	0.79	0.79
fnlwgt, relationship_ Not-in-family, workclass_selfempl, sex	0.79	0.81	0.82	0.8	0.79	0.82
fnlwgt, relationship_ Not-in-family, workclass_selfempl, marital-status	0.79	0.82	0.81	0.8	0.79	0.82
fnlwgt, relationship_ Not-in-family, workclass_selfempl, occupation	0.79	0.82	0.81	0.8	0.79	0.8
fnlwgt, relationship_ Not-in-family, workclass_selfempl, relationship_ Wife	0.79	0.82	0.82	0.8	0.79	0.81
fnlwgt, relationship_ Not-in-family, workclass_selfempl, hours-per-week	0.79	0.79	0.81	0.79	0.79	0.81
fnlwgt, relationship_ Not-in-family, workclass_selfempl, workclass_others	0.79	0.82	0.82	0.81	0.79	0.82
fnlwgt, relationship_ Not-in-family, workclass_selfempl, relationship_ Husband	0.79	0.81	0.82	0.81	0.79	0.82
fnlwgt, relationship_ Not-in-family, workclass_selfempl, relationship_ Other-relative	0.79	0.82	0.82	0.81	0.79	0.82
fnlwgt, relationship_ Not-in-family, workclass_selfempl, relationship_ Own-child	0.79	0.82	0.82	0.81	0.79	0.82
fnlwgt, relationship_ Not-in-family, workclass_selfempl, age	0.79	0.79	0.82	0.81	0.79	0.82

5 *Seleção do Modelo e Otimização*

Findo todo o processo de tratamento de dados, chegou a hora de escolher o modelo de previsão mais adequado. Como pode ser visto na Tabela 13, os diferentes modelos aplicados foram:

- *Support Vector Machines*;
- *K-Nearest Neighbors*;
- *Gaussian Naive Bayes*;
- *AdaBoost Classifier*;
- *Bagging Classifier*;
- *Stacking Classifier*.

Os resultados obtidos com estes modelos de previsão podem ser vistos na seguinte tabela.

Tabela 14. Resultados da aplicação dos diferentes modelos de previsão

	Accuracy
SVM	0.79
KNN	0.82
Naive Bayes	0.82
AdaBoost	0.81
Bagging	0.79
Stacking	0.82

Tendo em conta estes resultados, verificamos que se torna difícil escolher o modelo mais adequado, visto que os valores são muito semelhantes. Contudo,

optámos por escolher o modelo *K-Nearest Neighbors*, uma vez que se trata de um algoritmo simples, é de rápida execução, não faz suposições (ao contrário de outros modelos, como *Linear Regression*), adapta-se facilmente a novos dados de treino, só existe um hiperparâmetro a definir (e quanto menos parâmetros, menos propenso a *overfitting* está o modelo) e é fácil de implementar e otimizar.

Visto que os resultados obtidos com o modelo *KNN* foram alcançados com os valores dos seus parâmetros por defeito, optámos por verificar se trocar o número de vizinhos alteraria o desempenho do mesmo. Para isso, optámos por averiguar qual o número de vizinhos que minimiza o erro. Posto isto, obtivemos o gráfico que a seguir se apresenta e verificámos que, de facto, diminuir o número de vizinhos de 5 para 2 reduz o erro e, consequentemente, melhora o desempenho do modelo (o seu valor de acurácia aumenta de 82% para 83%).

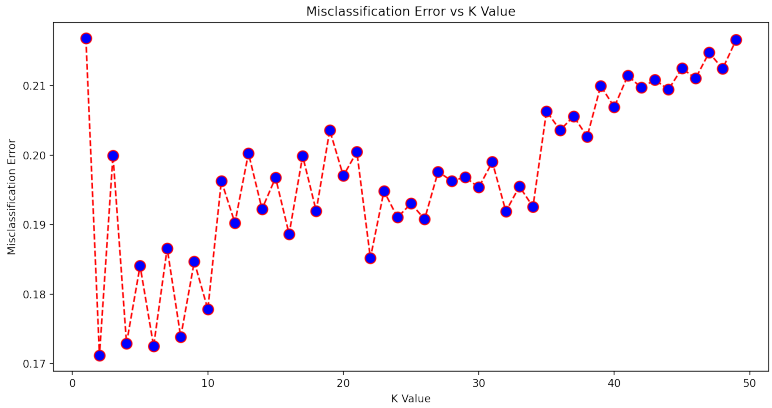

Figura 19. Pesquisa do número ótimo de vizinhos

6 Conclusão

Com a realização deste trabalho foi possível pôr em prática as diferentes fases de desenvolvimento de um modelo de previsão, desde a visualização dos dados até à otimização do mesmo.

Um dos grandes obstáculos com que nos deparámos foi o facto de o conjunto de dados estar desbalanceado, uma vez que os valores de *accuracy* estavam a ser erróneos. Efetivamente, o conjunto de dados de teste era composto por 15060 observações das quais 11360 pertenciam à classe <=50K, logo um modelo que classificasse todos os registos como pertencentes a esta classe teria uma acurácia de, aproximadamente, 0.754. Posto isto, foi necessário explorar diversas formas

de balancear o *dataset*, entre as quais optámos por escolher a técnica de *under-sampling*.

A necessidade de selecionar as características mais relevantes, também nos permitiu explorar diversos métodos, nomeadamente *filter method*, *wrapper method* e *embedded method*.

Por fim, a fase de seleção do modelo também se tornou desafiante, uma vez que os vários modelos de previsão testados tinham valores de *accuracy* muito semelhantes. Acabámos por selecionar o modelo *KNN* que, após a sua otimização, conseguiu atingir uma *accuracy* de 83%.