UNIVERSIDADE DO MINHO
ESCOLA DE ENGENHARIA

Preparação e análise das condições de funcionários de uma empresa

Estudo do dataset utilizando scikit-learn

Ana Marta Santos Ribeiro A82474
Jéssica Andreia Fernandes Lemos A82061
Tiago Dias de Sousa A81922

MIEI - 4º Ano - Aprendizagem e Extração de Conhecimento

Braga, 24 de Outubro de 2020

Conteúdo

1 Introdução

Este relatório surge no âmbito do trabalho prático da unidade curricular de Aprendizagem e Extração de Conhecimento que se encontra inserida no perfil de Sistemas Inteligentes. Neste pretende-se que seja desenvolvido um modelo classificador utilizando o ambiente de desenvolvimento Python/Sklearn.

O problema apresentado consiste em prever a ausência de um funcionário tendo em conta as suas condições. Assim sendo, é necessário começar por observar e analisar as *features* do *dataset* fornecido. De seguida tem de se realizar o tratamento dos dados, para melhorar a performance de classificação do modelo. Por fim, é preciso treinar e validar cada um dos modelos lecionados nas aulas. Como extra, devem ser implementadas técnicas *hyperparameterization optimization* por forma a otimizar os modelos.

2 Aquisição dos dados

Como todos os datasets necessários para este problema nos foram fornecidos, não tivemos de realizar a aquisição dos dados. Como já foi referido anteriormente, o problema em questão neste projeto é a previsão da ausência de um funcionário tendo em conta várias informações sobre ele, pelo que o dataset disponível é relativo à condição dos mesmos. Destas informações que nos são dadas, reparamos que algumas estão diretamente ligadas ao seu emprego e que indutivamente nos parecem mais prováveis de estar mais relacionadas com a ausência, como a carga de trabalho diária do funcionário, a indisciplina e a distância da residência ao local de trabalho.

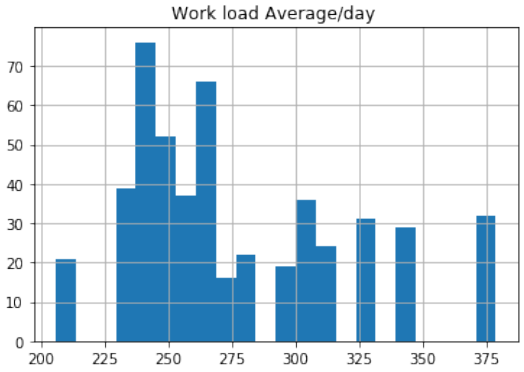
Existem outras features que pelo contrário, antes da análise de dados, nos pareceram mais prováveis de ser irrelevantes no contexto da previsão, como por exemplo a altura, peso ou o número de animais que o funcionário possui, pois, com o nosso conhecimento prévio do problema, parece-nos que estarão pouco relacionadas com a ausência dos funcionários.

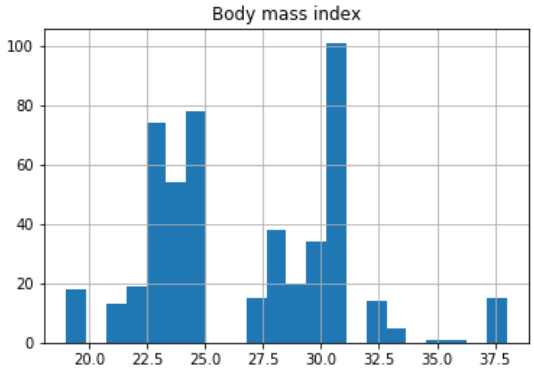
O principal objetivo consiste em tratar todos os dados de forma a ser possível relacioná-los com a variável *target*, neste caso o absent, que adquire o valor um caso o funcionário vá trabalhar e zero caso contrário.

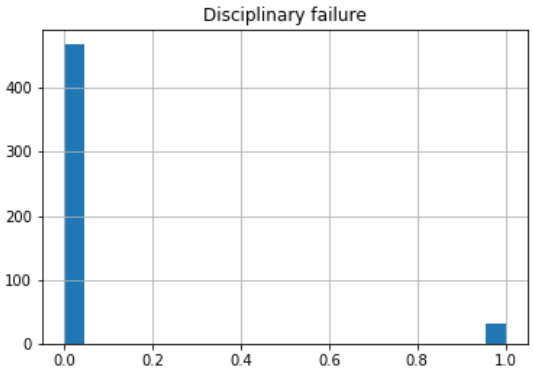
3 Visualização dos dados

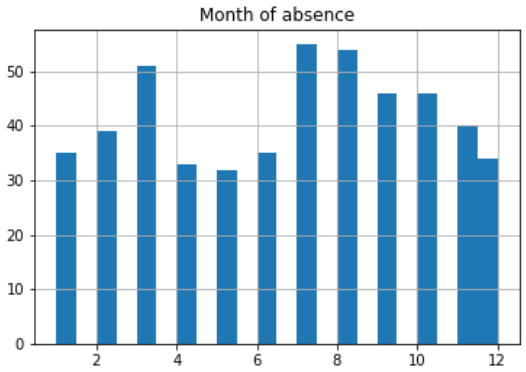
3.1 Histogramas

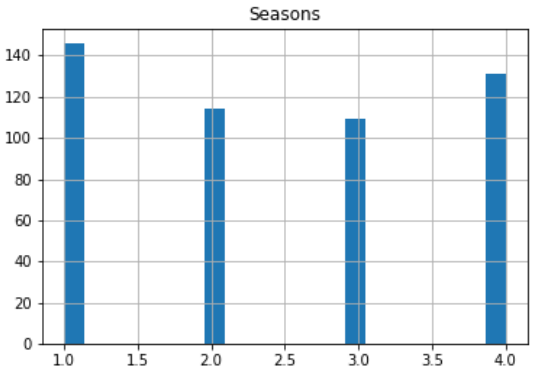
Nesta secção serão apresentados histogramas para visualização da ocorrência das features do *dataset*, de modo a tentar encontrar alguns padrões que auxiliem a interpretação dos dados e previsão dos resultados.

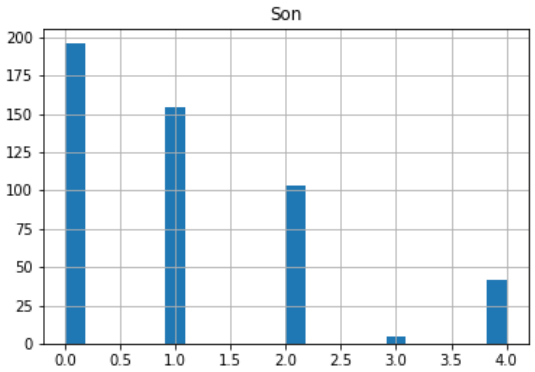


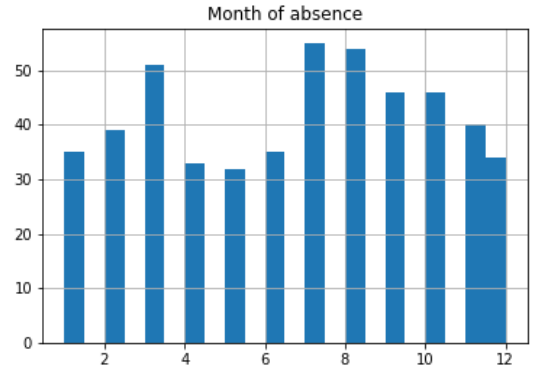


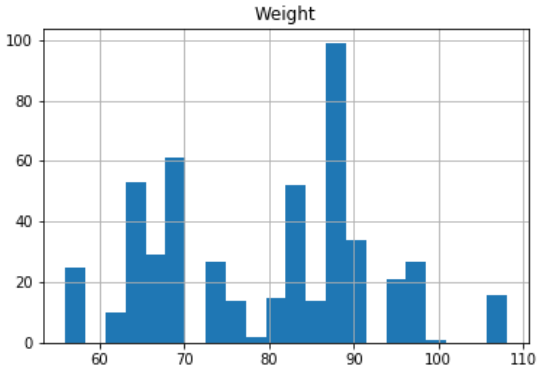


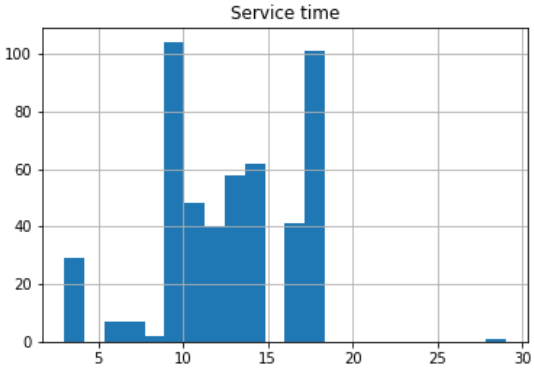


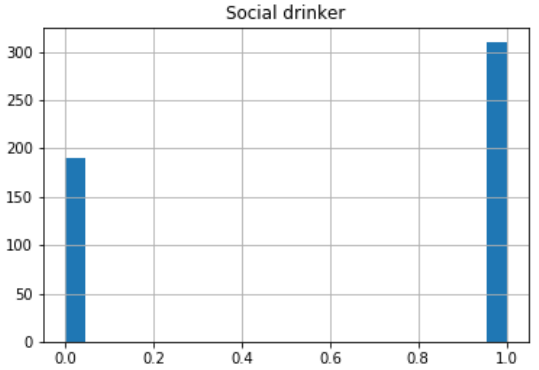


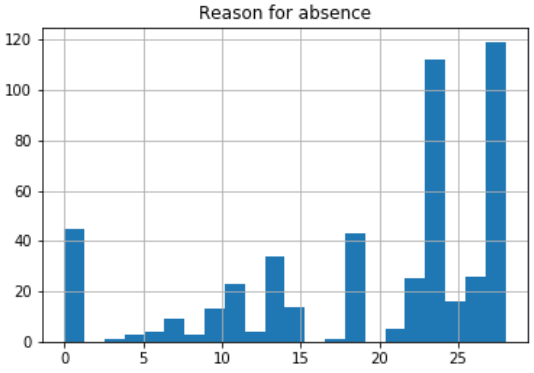


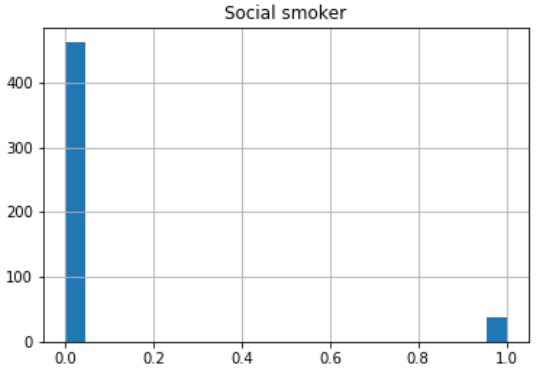

Month of absence
0
10
20
30
40
50
2
4
6
8
10
12

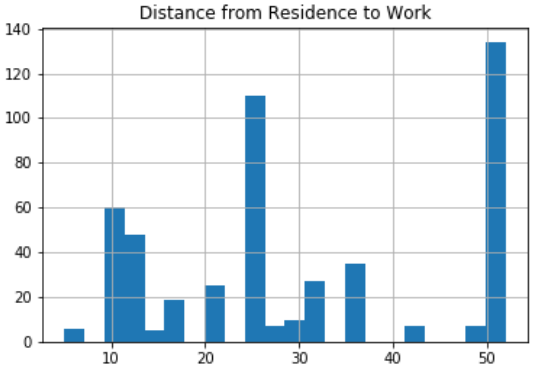

Weight
0
20
40
60
80
100
60
70
80
90
100
110

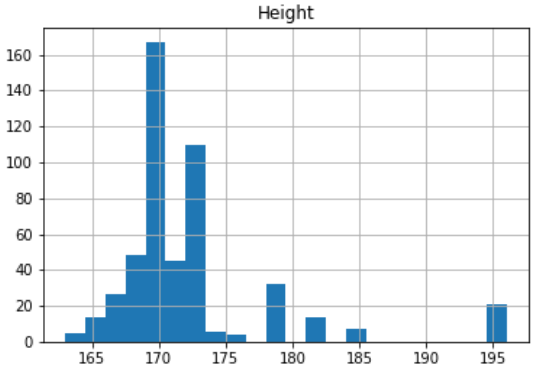

Service time
0
20
40
60
80
100
5
10
15
20
25
30

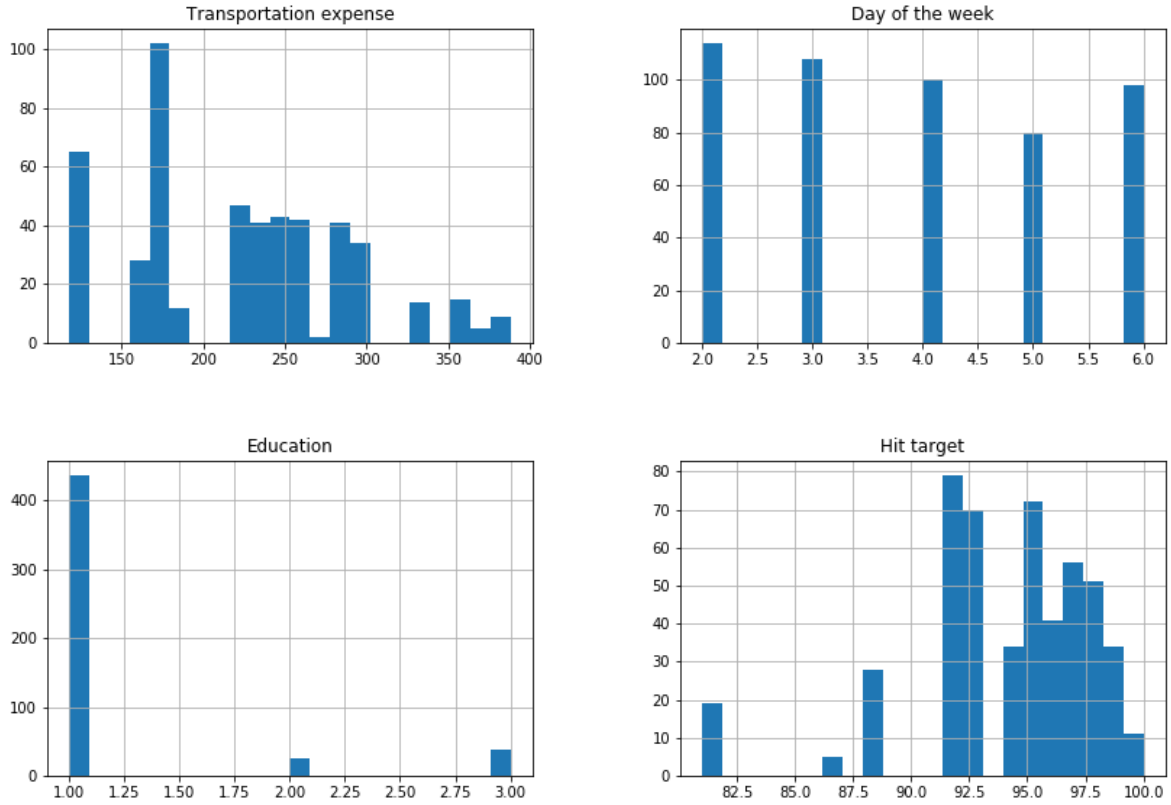

Social drinker
0
50
100
150
200
250
300
0.0
0.2
0.4
0.6
0.8
1.0


Reason for absence
0
20
40
60
80
100
120
0
5
10
15
20
25


Social smoker
0
100
200
300
400
0.0
0.2
0.4
0.6
0.8
1.0


Distance from Residence to Work
0
20
40
60
80
100
120
140
10
20
30
40
50


Height
0
20
40
60
80
100
120
140
160
165
170
175
180
185
190
195



3.2 Medidas de tendência e dispersão

De modo a ter uma noção mais percetível dos dados do problema, utilizamos o método *describe* para o nosso *dataset*, que devolve indicadores estatísticos como a média, desvio padrão, mínimo, máximo e os percentis 25, 50 e 75.

Estes indicadores surgem não só como forma de possibilitar uma análise numérica dos dados, mas também como forma de apoio à análise de outliers feita de seguida.

	Reason for absence	Month of absence	Day of the week	Seasons	Transportation expense	Distance from Residence to Work	Service time	Age	Work load Average/day	Hit target
count	500.000000	500.000000	500.00000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000
mean	19.288000	6.614000	3.88000	2.450000	223.640000	29.978000	12.650000	36.660000	276.792768	94.168000
std	8.543245	3.343555	1.43587	1.165425	67.323155	15.068498	4.036345	6.137731	43.422723	3.912338
min	0.000000	1.000000	2.00000	1.000000	118.000000	5.000000	3.000000	27.000000	205.917000	81.000000
25%	13.000000	3.750000	3.00000	1.000000	179.000000	16.000000	10.000000	33.000000	244.387000	92.000000
50%	23.000000	7.000000	4.00000	2.000000	225.000000	26.000000	13.000000	37.000000	265.017000	95.000000
75%	26.000000	9.000000	5.00000	4.000000	260.000000	50.000000	16.000000	40.000000	306.345000	97.000000
max	28.000000	12.000000	6.00000	4.000000	388.000000	52.000000	29.000000	58.000000	378.884000	100.000000

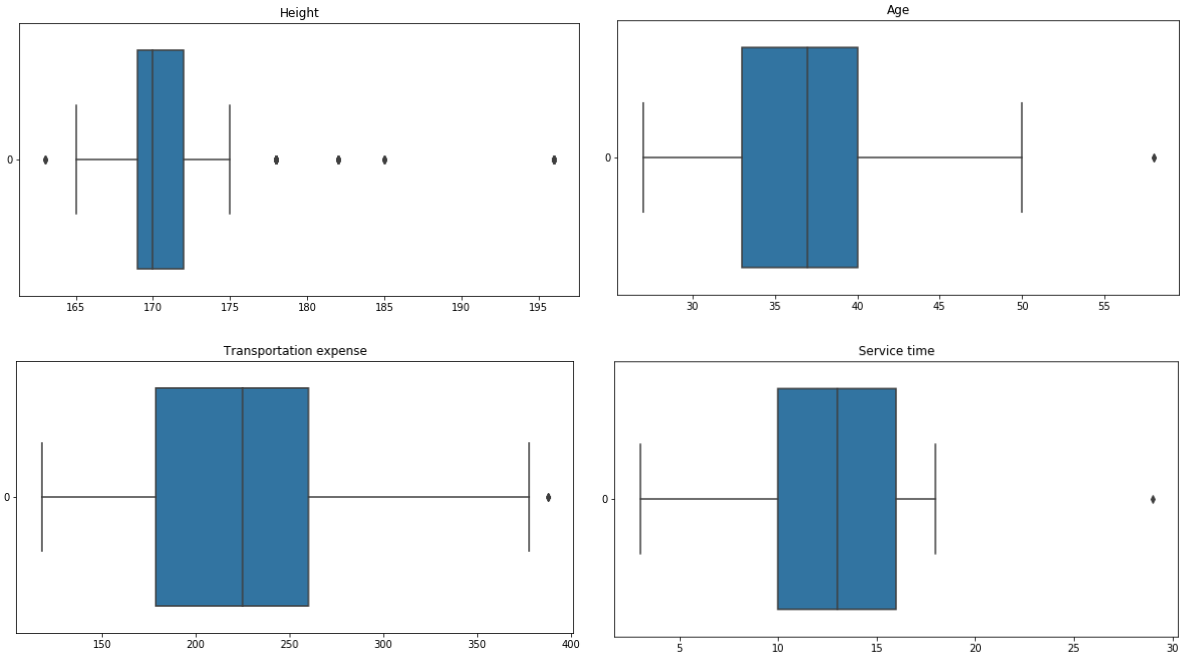
Figura 1. Indicadores Estatísticos do Dataset

	Disciplinary failure	Education	Son	Social drinker	Social smoker	Pet	Weight	Height	Body mass index	Absent
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000
mean	0.064000	1.204000	1.086000	0.620000	0.076000	0.628000	79.698000	172.098000	26.870000	0.790000
std	0.244998	0.561261	1.178721	0.485873	0.265264	1.071406	12.605101	6.234913	4.151092	0.407716
min	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	56.000000	163.000000	19.000000	0.000000
25%	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	69.000000	169.000000	24.000000	1.000000
50%	0.000000	1.000000	1.000000	1.000000	0.000000	0.000000	83.000000	170.000000	25.000000	1.000000
75%	0.000000	1.000000	2.000000	1.000000	0.000000	1.000000	89.000000	172.000000	31.000000	1.000000
max	1.000000	3.000000	4.000000	1.000000	1.000000	5.000000	108.000000	196.000000	38.000000	1.000000

Figura 2. Indicadores Estatísticos do Dataset

3.3 Outliers

Através da visualização dos gráficos anteriores e da análise das medidas de tendências e dispersão foi possível verificar a ocorrência de um ou dois valores muito maiores aos que têm sido registados noutras instâncias, como tal torna-se essencial percebermos de uma forma mais clara as features com outliers. Passando ao estudo dos mesmos foi possível então perceber quais as features que continham, sendo apresentado de seguida alguns exemplos nos quais é bastante percetível a sua visualização.


Height
0
165
170
175
180
185
190
195
Age
0
30
35
40
45
50
55
Transportation expense
0
150
200
250
300
350
400
Service time
0
5
10
15
20
25
30

4 Data Preprocessing

4.1 Tratamento do Dataset

Inicialmente, analisamos as entradas do dataset de modo a tentar encontrar valores em falta ou incorretamente inseridos. Neste contexto percebemos que não poderíamos analisar a feature Workload Average/day porque esta se encontrava em forma de String, para resolver isto foi efetuada a esta coluna a transformação seguinte:

```
data['Work load Average/day '] = [x.replace(',', '.') for x in
data['Work load Average/day ']]

data['Work load Average/day '] = data['Work load Average/day '].astype(float)
```

4.2 Missing Values Analysis

Analisamos o *dataset* e verificamos se existiriam valores em falta. Através da seguinte linha de código conseguimos constatar que não existia informação em falta:

```
pd.DataFrame(data.isnull().sum())
```

4.3 Feature Selection

Existem 3 classes gerais de algoritmos de seleção de features que são:

- Filter methods
- Wrapper methods
- Embedded methods

4.3.1 Filter methods

Neste método é aplicada uma medida estatística para atribuir uma pontuação a cada feature. Assim, estas são classificadas pela pontuação e como tal selecionadas para serem mantidas ou não no conjunto de dados. Para aplicação deste algoritmo, optámos por utilizar a função *SelectKBest* (*Unvariate Selection*). Neste método optamos por usar *Chi-Squared* que é um teste estatístico para valores não negativos para selecionar as top k features que pretendemos. Usamos ainda outro método, o *VarianceThreshold*.

```
[6.65228472e+01 4.29279354e+00 3.59177992e-02 4.96026620e+00
 1.35411719e+01 1.52429226e+01 2.26486502e+00 6.88538472e+00
 1.07531580e+01 1.32254231e+00 1.20380952e+02 2.49960449e-02
 2.85246832e+00 1.94442824e-04 1.65032835e-01 7.04967692e-01
 9.20682086e+00 3.30162263e-02 2.84634477e+00]

[ True False False False False False False False False False  True False
 False False False False False False]

[ 0 10]
```

Figura 3. SelectKBest

```
[7.28410560e+01 1.11570040e+01 2.05760000e+00 1.35550000e+00
 4.52334240e+03 2.26605516e+02 1.62595000e+01 3.75964000e+01
 1.88176180e+03 1.52757760e+01 5.99040000e-02 3.14384000e-01
 1.38660400e+00 2.35600000e-01 7.02240000e-02 1.14561600e+00
 1.58570796e+02 3.87963960e+01 1.71971000e+01]
```

Figura 4. VarianceThreshold

4.3.2 Wrapper methods

Os métodos de wrapper consideram a seleção de um conjunto de recursos como um problema de pesquisa, onde diferentes combinações são preparadas, avaliadas e comparadas com outras combinações. Este processo de pesquisa pode ser metódico ou estocástico. O algoritmo que optamos por utilizar foi o *Recursive Feature Elimination* que cria modelos repetidamente e mantém o recurso que permite o melhor e pior desempanho em cada iteração sendo classificado os recursos tendo em conta a ordem da sua eliminação.

```
[False False False False False False False False False False  True False
  True False False False False False]

[ 7 13 12  2 17 14 18  6 16  8  1  5  1  9  3  4 11 15 10]

[10 12]
```

Figura 5. Recursive Feature Elimination

4.3.3 Embedded methods

Os métodos incorporados aprendem quais os recursos que contribuem para a precisão do modelo enquanto este é criado. Decidimos implementar o *Principal Component Analysis* que utiliza álgebra linear para transformar o conjunto de dados. E também o *Feature Importance* uma vez que árvores de decisão como *Random Forest* e *Extra Trees*

podem ser usadas para estimar a importância dos recursos.

```
[6.51593468e-01 2.69387514e-01 3.14339170e-02 2.34969706e-02
 9.75036750e-03 6.22503399e-03 3.14907071e-03 2.58662848e-03
 9.03237523e-04 7.24113323e-04 2.81593635e-04 1.57909067e-04
 1.32742345e-04 1.08405996e-04 3.79887690e-05 1.19459097e-05
 8.73174996e-06 5.53314513e-06 4.82854982e-06]
```

Figura 6. Principal Component Analysis

```
[7.28410560e+01 1.11570040e+01 2.05760000e+00 1.35550000e+00
 4.52334240e+03 2.26605516e+02 1.62595000e+01 3.75964000e+01
 1.88176180e+03 1.52757760e+01 5.99040000e-02 3.14384000e-01
 1.38660400e+00 2.35600000e-01 7.02240000e-02 1.14561600e+00
 1.58570796e+02 3.87963960e+01 1.71971000e+01]
```

Figura 7. Feature Importance

As features selecionadas pelos diferentes métodos variam um pouco, pelo que optamos por verificar quais tinham um maior impacto nos resultados. Após obtermos os diferentes resultados verificamos que alguns eram bastante semelhantes optando por selecionarmos duas features: Reason for absence e Disciplinary failure.

4.4 Normalization/Standardization

Com o intuito de obter os melhores resultados possíveis optamos por reduzir a escala dos dados, dado que existiam valores bastantes diferentes como pudemos observar na secção anterior. Para tal, recorremos a técnicas de *normalization* e *standardization*.

No processo de normalização os dados são convertidos de forma a que a sua norma, l1 ou l2 conforme especificado, seja 1. De modo a escolher a norma mais adequada ao problema, optamos por calcular a *accuracy* para os modelos, com uma *feature selection* obtida do *SelectKBest* com um $k = 2$. Com isto pretendemos verificar aquele que é mais benéfico. Tendo em conta os resultados apresentados na figura seguinte, foi possível verificar que o desempenho era o mesmo tanto com o l1 como com l2.

```
---------Normalization l1---------
Random Forest 0.804167
Nearest Neighbors 0.808333
SVM 0.808333
LogisticRegression 0.804167
Decision Tree 0.808333
AdaBoost 0.804167
Naive Bayes 0.804167
---------Normalization l2---------
Random Forest 0.804167
Nearest Neighbors 0.808333
SVM 0.808333
LogisticRegression 0.804167
Decision Tree 0.808333
AdaBoost 0.804167
Naive Bayes 0.804167
```

Figura 8. Normalização

Assim sendo, optamos por recorrer à normalização do tipo l2. O processo efetuado encontra-se representado de seguida:

```
normalizer = preprocessing.Normalizer(norm='l2')
values_normalized = normalizer.transform(data.values)
data = pd.DataFrame(values_normalized, columns=data.columns)
```

Quanto ao processo de standardization optamos por utilizar o *MinMaxScaler*, que é o mais comum, para um intervalo entre 0 e 1. Assim, de seguida encontra-se ilustrado este processo:

```
scaler = preprocessing.MinMaxScaler(feature_range=(0, 1))
values_standardized = scaler.fit_transform(data.values)
data = pd.DataFrame(values_standardized, columns=data.columns)
```

5 Model Selection, Model Training and Validation

Como já foram realizados todos os processos relativos ao tratamento de dados na secção anterior, nomeadamente a seleção das *features* e a *normalization* e *standardization*, passamos à escolha do modelo mais adequado. Para tal, começamos por aplicar diversos modelos distintos:

- Random Forest Regressor
- K-Nearest Neighbors
- Support Vector Machine
- Logistic Regression
- Decision Tree Regressor
- AdaBoost Classifier
- Gaussian Naive Bayes

Optamos por testar estes modelos aprendidos nas aulas através da *normalização (l2)* e da *standardization (MinMaxScaller)*. Para além disso, tivemos em consideração na seleção das *features* o método indicado anteriormente, o *SelectKBest* com $k = 2$, ou seja, as features que foram utilizadas tanto no *treino* como no *teste* foram as *Reason for absence* e *Disciplinary failure*. Deste modo, é possível verificar de seguida os resultados obtivos para os diferentes modelos tanto recorrendo à *normalization* como à *standardization*.

```
---------Normalization---------
Random Forest 0.804167
Nearest Neighbors 0.808333
SVM 0.808333
LogisticRegression 0.804167
Decision Tree 0.808333
AdaBoost 0.804167
Naive Bayes 0.804167
--------Standardization--------
Random Forest 0.808333
Nearest Neighbors 0.808333
SVM 0.808333
LogisticRegression 0.808333
Decision Tree 0.808333
AdaBoost 0.804167
Naive Bayes 0.808333
```

Figura 9. Resultado dos modelos

Tendo em conta a figura apresentada, podemos concluir que os resultados são bastante similares, pelo que escolher o modelo mais adequado se torna mais difícil. No entanto optamos por escolher o modelo SVM, dado que apresenta capacidade de previsão sem casas decimais o que permite evitar os arredondamentos. Para além disso verificamos que quando aumentamos o k no processo de seleção de features este modelo apresenta melhor desempenho. Basicamente, para o valor selecionado existem alguns modelos com os quais podemos obter um bom resultado, como podemos constatar, contudo isso deve-se ao facto deste número de features selecionadas ser reduzido. Com um valor superior poderiamos verificar uma maior variação dos resultados nos diferentes modelos.

6 Hyperparameter Optimization

Como verificamos na secção anterior o método que selecionamos foi o *Support Vector Machine*. Uma vez que os resultados que alcançamos foram através dos valores dos parâmetros por defeito, passamos então a verificar o seu comportamento com hiper-parametrização desses parâmetros. Após realizamos todo o processo como na secção anterior, efetuamos um estudo de todos os parâmetros verificando quais destes dariamos mais importância. Para realizarmos a hiper-parametrização construimos então um dicionário em que para cada parâmetro indicamos os valores que estes podem tomar. Para este processo optamos por seguir a estratégia *Grid Search*. Como resultado obtemos que o parâmetro indicado como melhor é o seguinte:

```
Best parameters set found on:
{'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
```

Figura 10. Parâmetros indicados pelo Grid Search

Como podemos ver pela coluna *f1-score* os resultados melhoraram, passando do valor obtido anteriormente que era 0.808333 para 0.90.

	precision	recall	f1-score	support
1	0.82	1.00	0.90	196
micro avg	0.82	1.00	0.90	196
macro avg	0.82	1.00	0.90	196
weighted avg	0.82	1.00	0.90	196

Figura 11. Resultados obtidos

7 Conclusão

Com a finalização deste projeto, consideramos que este foi um bom projeto uma vez que fizemos uma boa análise do *dataset* e aplicamos diversas técnicas de modo a verificar os resultados provenientes da seleção das features. Contudo, os resultados obtidos não alcançaram as nossas expectativas e deparamo-nos com resultados que não eram expectáveis, inicialmente, para determinados modelos como é o caso das *Support Vector Machines* em que na seleção de diferentes features obtivemos diversas vezes os mesmos valores de *accuracy*.

Em suma, ao longo do desenvolvimento do projeto foi notório que as fases iniciais são de extrema importância para a conceção do projeto uma vez que estas permitem uma análise cuidada do *dataset* e definir as melhores estratégias de tratamento do mesmo.