



**Ciências
ULisboa**

Faculdade
de Ciências
da Universidade
de Lisboa

Engenharia Conhecimento
Licenciatura em Engenharia Informática
2023/2024

Modelos de Classificação e Regressão
Thyroid disease Data Set

DEPARTAMENTO DE INFORMÁTICA

Grupo 6:

- Eduardo Proença 57551
- Tiago Oliveira 54979
- Bernardo Lopes 54386

Introdução e Objetivos

Este projeto foi desenvolvido com o intuito de utilizar diferentes técnicas de aprendizagem automática, aprendidas ao longo da unidade curricular, para construir os melhores modelos de classificação e regressão possíveis.

Para tal, foi fornecido um conjunto de dados, uma versão do "thyroid0387" data set, que contém 31 características e 7338 instâncias. Os dados deste conjunto, não se encontram processados, sendo possível encontrar valores categóricos e contínuos, assim como valores em falta. Desta maneira, ao longo do desenvolvimento do projeto irão ser utilizadas técnicas como codificação, normalização e imputação de dados.

Quanto à construção dos modelos serão utilizadas técnicas como seleção de variáveis, *cross validation* e afinamento de hiperparâmetros.

Ao longo do projeto serão testados diferentes modelos como *LinearRegression*, *DecisionTreeClassifier*, *LogisticRegression*, *Naive Bayes*, *KNN* e *SVM*

Objetivo 1 (O1)

- Os alunos devem fornecer os melhores modelos de classificação possíveis usando quaisquer métodos abordados nas aulas
- A classificação deve ser de acordo com 8 classes da variável alvo "diagnoses"

Objetivo (O2)

- Podemos prever com exatidão a idade dos indivíduos, tendo em conta os outros atributos?
- Podemos prever com exatidão o sexo dos indivíduos, tendo em conta os outros atributos?

Objetivo (O3)

- Quais são as características mais significativas dos melhores modelos obtidos acima (O1 e O2)

Processamento dos dados

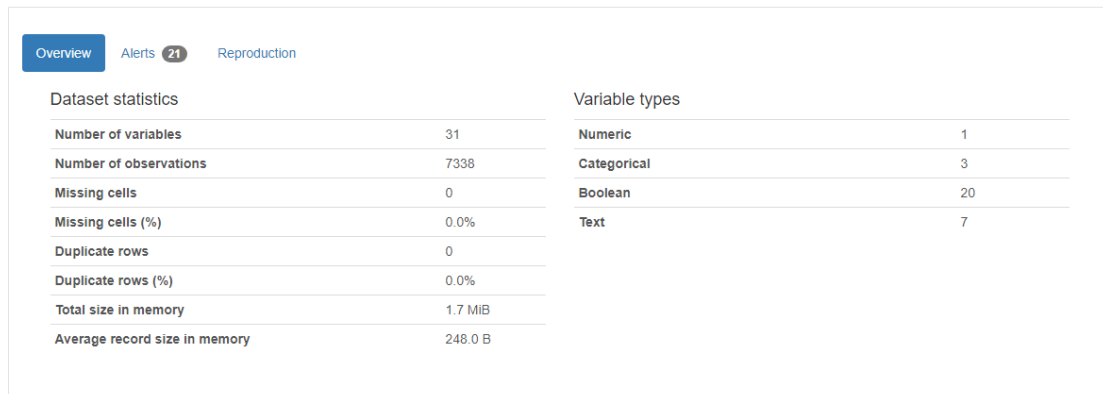
Devido ao facto de terem sido testados vários modelos e embora não seja a melhor prática, foi decido fazer o processamento dos dados antecipadamente, de maneira a reduzir a complexidade da tarefa e por sua vez o tamanho da mesma.

Desta forma, o ideal seria fazer um processamento de dados específico para cada modelo, tendo em conta as suas características. Portanto, é de realçar que os resultados obtidos não são perfeitos, pelo que ainda existe margem para melhoria. No entanto, as técnicas utilizadas neste projeto, foram técnicas presentes na construção de diferentes tipos de modelos, e que na sua maioria permitem bons resultados.

Perfil de dados

Começou-se por fazer um *profiling* dos nossos dados (usando a classe *ProfileReport* do package *ydata_profiling*), para obter uma informação mais detalhada sobre as nossas variáveis.

Overview



The screenshot shows the 'Overview' tab of a ProfileReport. It contains two tables: 'Dataset statistics' and 'Variable types'. The 'Dataset statistics' table lists metrics like number of variables, observations, missing cells, duplicate rows, and memory size. The 'Variable types' table lists the count of variables for each type: Numeric (1), Categorical (3), Boolean (20), and Text (7).

Overview	Alerts 24	Reproduction
Dataset statistics		
Number of variables	31	
Number of observations	7338	
Missing cells	0	
Missing cells (%)	0.0%	
Duplicate rows	0	
Duplicate rows (%)	0.0%	
Total size in memory	1.7 MiB	
Average record size in memory	248.0 B	
Variable types		
Numeric	1	
Categorical	3	
Boolean	20	
Text	7	

Figura 1 - ProfileReport Overview

Devido ao facto dos valores em falta serem representados por “?” e algumas colunas estarem em formato de texto, para além de não ser possível ver os valores em falta no *report*, este também indica a presença de *text variables*, que na verdade são valores numéricos.

Desta forma, olhando para estes resultados e para a explicação fornecida no ficheiro “data.names”, foi possível identificar três tipos de variáveis: numéricas, booleanas e categóricas. Antes de proceder com a codificação dos dados, foi apagada a variável “[record identification]”, pois esta apenas representa um identificador único, o que não é relevante. Foram também substituídos os valores “?” por *NaN* para, posteriormente serem identificados como valores em falta.

Codificação de dados

Devido ao facto, de existir alguns modelos que não conseguem tratar de valores não numéricos, foi feita uma codificação dos dados, da seguinte maneira:

Todas as variáveis booleanas, ou seja, que só têm dois valores possíveis foram codificadas para 0 e 1. A variável categórica “referral source”, com 6 valores possíveis, foi codificada com os valores 0, 1, 2, 3, 4 e 5. A variável alvo “diagnoses” foi codificada de acordo com as 8 classes indicadas:

- “-” codificado para 0
- A, B, C, D codificados para 1
- E, F, G, H codificados para 2
- I, J codificados para 3
- K codificado para 4
- L, M, N codificados para 5
- R codificado para 6
- Restantes valores codificados para 7

Divisão dos dados

Tendo os dados codificados, procedeu-se à divisão dos dados. Para tal, foi utilizado o método *train_test_split()* do *sklearn*, este transforma o nosso conjunto de dados em dois, um utilizado para treinar os modelos e outro utilizado para testar os mesmos. Foi usado o um *test_size=0.2*, de forma que o nosso conjunto de treino fosse 80% do conjunto de dados e o conjunto de teste fosse 20%.

Normalização

Como foram utilizados modelos como o KNN, que dependem da distância entre os dados, foi importante fazer a normalização dos dados. Esta foi realizada com a classe *StandardScaler*, com uma estratégia de média.

Imputação de dados

Foi visto anteriormente que o nosso conjunto de dados possui valores em falta, no entanto, a grande maioria destes valores apenas indicam que o médico não realizou uma certa medição, pelo que não existe valor a registar. Desta forma estes valores não são verdadeiramente valores em falta e, portanto, podem ser importantes. Por estas razões, foi decidido não apagar nenhum valor e proceder a imputação dos mesmos. Esta foi feita utilizando a classe *SimpleImputer*, com mais uma vez uma estratégia de média.

Seleção de variáveis

O processo de seleção de variáveis foi feito através do uso da classe *SequentialFeatureSelector* (SFS) juntamente com modelos regressão, como o *LinearRegression* e também classificação, como o *LogisticRegression*. Foram realizados diferentes testes, com vários números possíveis de variáveis, utilizando esta técnica. Em baixo encontra-se um exemplo dos resultados:

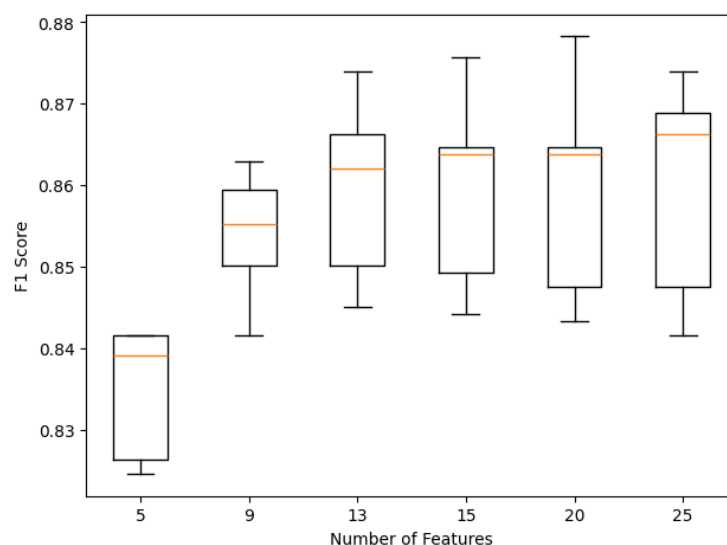


Figura 2 - Seleção de variáveis

Entre os diferentes números de variáveis testados, os que obtiveram melhor resultado, de acordo com o *F1 Score*, foram 13 e 25. Desta maneira, optou-se por utilizar 13 variáveis, pois para além de melhorar os nossos modelos, eliminando dados que possam ser irrelevantes, também ajuda a reduzir o tempo que é necessário para treinar os mesmos.

Resultados dos modelos

Tendo os dados processados e as melhores variáveis seleccionadas, passamos à fase de testar diferentes modelos de classificação. Para avaliar o desempenho de cada modelo foram utilizadas 5 métricas: *Accuracy*, *Precision*, *Recall*, *F1 Score* e *Matthews Correlation Coefficient*. O processo de avaliação dos modelos foi realizado com recurso a uma estratégia de *KFold Cross Validation*.

Model	Accuracy	Precision	Recall	F1 Score	Matthews Correlation Coefficient
DecisionTreeClassifier()	0.9239	0.9234	0.9239	0.9236	0.8274
LogisticRegression()	0.8486	0.8344	0.8486	0.8287	0.6168
GaussianNB()	0.1311	0.7634	0.1312	0.1343	0.1628
KNeighborsClassifier()	0.8649	0.8568	0.8649	0.8523	0.6648
SVC()	0.8435	0.8356	0.8435	0.8169	0.5984

Tabela 1 - Resultados dos modelos de classificação

Rapidamente, foi concluído que o *Naive Bayes* (GaussianNB) não possui um bom desempenho com os nossos dados. Contrariamente o *DecisionTreeClassifier*, demonstrou resultados bastante positivos nas diferentes métricas.

De maneira não perdermos tempo a afinar modelos cujos não se adaptam bem aos nossos dados decidimos seleccionar dos vários modelos testados apenas 3, tendo em conta o MCC (*Matthews Correlation Coefficient*), devido a ser uma métrica bastante confiável. Sendo assim, foram escolhidos para afinamento de hiperparâmetros os modelos *DecisionTreeClassifier*, *KNeighborsClassifier* e *SVC*.

Previsão da idade

Utilizando as mesmas estratégias de processamento de dados e seleção de variáveis, foi treinado um modelo *LinearRegression*, com o objetivo para tentar prever a idade dos indivíduos. Este obteve um valor de RMSE (*Root Mean Squared Error*) igual a 30.3690.

Para obtermos uma melhor visualização deste resultado, construímos uma representação gráfica das previsões. Analisando este gráfico de valores esperados vs. valores previstos, concluiu-se que não é possível prever com exatidão a idade, pois a exatidão entre estas duas variáveis encontra-se muito dispersa.

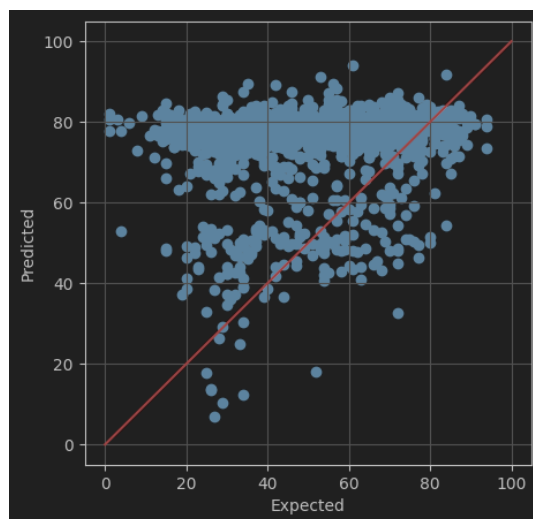


Figura 3 - Resultados esperados vs. Previstos (Idade)

Previsão do sexo

Novamente, utilizando as mesmas estratégias de processamento de dados e seleção de variáveis, treinou-se também os vários modelos de classificação para tentar prever o sexo dos indivíduos. Analisando mais uma vez as métricas, apesar de não serem totalmente negativas, foi obtido um resultado menos positivo face ao anterior. Desta forma, não podemos considerar seguro utilizar estes modelos para prever o sexo, especialmente devido aos valores de MCC obtidos.

Model	Accuracy	Precision	Recall	F1 Score	Matthews Correlation Coefficient
DecisionTreeClassifier()	0.5947	0.6102	0.5947	0.6012	0.0992
LogisticRegression()	0.6946	0.6635	0.6946	0.6251	0.1612
GaussianNB()	0.4136	0.6033	0.4136	0.3829	0.0481
KNeighborsClassifier()	0.6560	0.6318	0.6560	0.6390	0.1462
SVC()	0.6865	0.6448	0.6865	0.5932	0.1016

Tabela 2 - Resultados da previsão do sexo

Afinção dos hiperparâmetros

Tendo escolhido os 3 modelos de classificação que apresentaram melhores métricas. Procedemos à afinção de hiperparâmetros, com o objetivo de melhorar o desempenho destes modelos e assim escolher o melhor entre eles. Foi decidido utilizar uma estratégia de *grid search*, usando a classe *GridSearchCV*. Este processo passou pela definição de listas com diferentes valores possíveis para diferentes parâmetros de cada modelo, as quais foram utilizadas pelo *GridSearchCV*, que testou todas as possíveis combinações. No final, obtemos os melhores scores para cada modelo, assim como os parâmetros utilizados para obter os mesmos.

Melhor modelo:

DecisionTreeClassifier()

Melhores parâmetros:

criterion='entropy'

max_depth=20

min_samples_leaf=3

min_sample_split=10

Model	Best Score
DecisionTreeClassifier()	0.9337
KNeighborsClassifier()	0.8722
SVC()	0.8981

Tabela 3 - Resultados GridSearchCV

Após escolher o melhor modelo, este foi novamente avaliado utilizando a estratégia de avaliação definida anteriormente.

Accuracy	Precision	Recall	F1 Score	Matthews Correlation Coefficient
0.9309	0.9297	0.9309	0.9302	0.8424

Tabela 4 – Avaliação do melhor modelo de classificação

Conclusão

Finalmente, de modo a garantir a qualidade do nosso melhor modelo de classificação, este foi testado com o conjunto de teste composto por 20% dos dados, que foi definido durante a fase processamento de dados, o qual contém dados nunca vistos por este modelo.

Accuracy	Precision	Recall	F1 Score	Matthews Correlation Coefficient
0.9394	0.9382	0.9394	0.9382	0.8603

Tabela 5 – Resultados com o conjunto de teste

Verificou-se que o modelo manteve a sua qualidade e que até houve um pequeno aumento no MCC. Desta maneira, concluímos que é seguro confiar no nosso modelo e que, em princípio não existirão problemas de *overfitting*. Sendo assim, pode-se considerar que o modelo está pronto a fazer novas previsões, quando exposto a novos conjuntos de dados.

Adicionalmente, verificou-se que, das diferentes variáveis contidas no nosso conjunto de dados, as mais importantes para este modelo, foram as variáveis representadas nas colunas: [2 7 14 15 16 17 19 21 24 25 26 27 28] do nosso conjunto de dados.