



ADDETC – Área Departamental de Engenharia Eletrónica e Telecomunicações
e de Computadores

MEIM - Mestrado Engenharia informática e multimédia

Aprendizagem e Mineração de Dados

Projeto final

Turma:

MEIM-11D

Trabalho realizado por:

Miguel Távora N°45102

Duarte Domingues N°45140

Docente:

Artur Ferreira

Data: 20/07/2021

Índice

1. INTRODUÇÃO	1
2. DESENVOLVIMENTO	2
CRIAÇÃO DO MODELO DE BASE DE DADOS.....	2
CRIAÇÃO AUTOMÁTICA DA BASE DE DADOS E EXPORTAÇÃO DOS DADOS ..	4
CRIAÇÃO DE MODELOS DE CLASSIFICAÇÃO DE DADOS	5
CLASSIFICAÇÃO DOS DADOS	5
CLASSIFICADOR 1R.....	6
CLASSIFICADOR ID3	7
CLASSIFICADOR NAÏVE BAYES	8
CODIFICAÇÃO DOS DADOS	8
IMPLEMENTAÇÃO DOS CLASSIFICADORES NAIVE BAYES E ID3	9
AVALIAÇÃO DO DESEMPENHO DOS DIVERSOS CLASSIFICADORES.....	9
CLASSIFICAR CONJUNTO DE DADOS NÃO CLASSIFICADOS	10
TRANSFORMAÇÃO DO DATASET (CSV) PARA O DEVIDO FORMATO (TAB) ..	10
APLICAR O CLASSIFICADOR 1R À BASE DE DADOS DE COGUMELOS (INÍCIO PROJECT A1)	11
UTILIZAÇÃO DO ORANGE PARA O CONJUNTO DE DADOS	12
3. CONCLUSÕES	14
4. BIBLIOGRAFIA	15

Índice de ilustrações

Figura 1 - Modelo conceptual da base de dados	1
Figura 2 - conjunto de dados com o formato Three-Row Header	4
Figura 3 - repartição dos dados	5
Figura 4 - Desempenho dos diversos classificadores.....	9
Figura 5 - ficheiro csv original.....	10
Figura 6 - ficheiro csv transformado.....	10
Figura 7 - Métricas de desempenho do classificador.....	11
Figura 8 - Matriz de confusão	11
Figura 9 - Modelo Orange.....	12
Figura 10 - Matriz confusão de Random Forest	Erro! Marcador não definido.
Figura 11 - Matriz confusão de Tree.....	Erro! Marcador não definido.
Figura 12 - Métricas de desempenho dos classificadores	13

1. Introdução

Este projeto baseia-se no processamento da informação de um conjunto de dados recolhidos por uma equipa de um centro médico. Em relação aos dados recolhidos são aplicadas técnicas de aprendizagem automática, de forma a extrair informação relevante para atingir uma solução para um problema.

O centro médico em questão é chamado “*MedKnown*” e tem uma equipa que se especializa na prescrição de lentes com base nos diagnósticos dos seus pacientes. A prescrição das lentes é baseada nos seguintes atributos: idade, prescrição, astigmático e taxa de lágrima. Baseado nestes quatro atributos um dos doutores da equipa da “*MedKnown*” tem que decidir o tipo de lentes que tem que prescrever a um paciente. Os quatro atributos podem assumir os seguintes valores:

- Idade (jovem, presbiópico, pré- presbiópico)
- Prescrição (miópe ou hipermetrope)
- Astigmático (sim ou não)
- Taxa de lágrima (normal ou reduzida)

As lentes prescritas podem ser duras, suaves ou nenhuma.

O desenvolvimento deste projeto envolveu diferentes etapas que envolveram assuntos como a criação de base dados em SQL, extração de informação de dados e classificação através de modelos de aprendizagem automática. O projeto dividiu-se nas seguintes partes:

- Criação de uma base de dados SQL baseada no modelo entidade relacionamento, de forma a suportar toda a informação associada a uma visita.
- Criação de scripts para automaticamente criar a base de dados, as restrições associadas e automaticamente povoar a base de dados.
- Utilização da *Orange data-mining framework* para analisar e visualizar informação sobre os dados.
- Utilização da linguagem de programação Python para processamento de dados e criação de modelos baseados em técnicas de classificação dos atributos. Foram criados modelos de classificação baseado nos seguintes métodos de aprendizagem automática: método 1R, árvores de decisão com algoritmo ID3 e métodos de *Naive Bayes*.

- Avaliação dos modelos criados a partir de métricas de desempenho e comparação dos resultados entre os diferentes modelos.
- Criação de scripts Python para utilizar os modelos para classificação de conjunto de dados em que os atributos não têm uma classe associada.

Na segunda etapa do projeto é realizada a análise e classificação de um conjunto de dados de alta dimensão, permitindo testar com melhor certeza o modelo de classificação implementado. Os dados foram recolhidos por um instituto chamado “FungiData”, este conjunto de dados baseia-se na comestibilidade de cogumelos, se são venenosos ou não.

2. Desenvolvimento

Criação do modelo de base de dados

Como foi referido na introdução, o primeiro passo do projeto foi a criação de uma base de dados, baseada no modelo entidade relacionamento, para suportar a informação relacionada com uma visita.

As tabelas “PATIENT” e “DOCTOR” são bastantes simples, tendo apenas o identificador e informação básica da pessoa, como por exemplo o nome.

A tabela “DISEASE” é utilizada para descrever certas doenças que o paciente pode ter, nomeadamente “myope”, “hypermetrope” e “astigmatic”.

A tabela “VISIT” guarda a data, o paciente e o doutor presentes numa visita. Esta tabela também guarda o diagnóstico para os atributos “tear rate” e “age” e o tipo de lentes prescritas ao paciente.

Por fim como numa visita um paciente pode ter mais que uma doença, foi criada uma tabela adicional chamada “PATIENT_DISEASE” que associa a tabela “DISEASE” com a tabela “VISIT”, desta forma zero ou mais doenças podem estar associadas a uma visita.

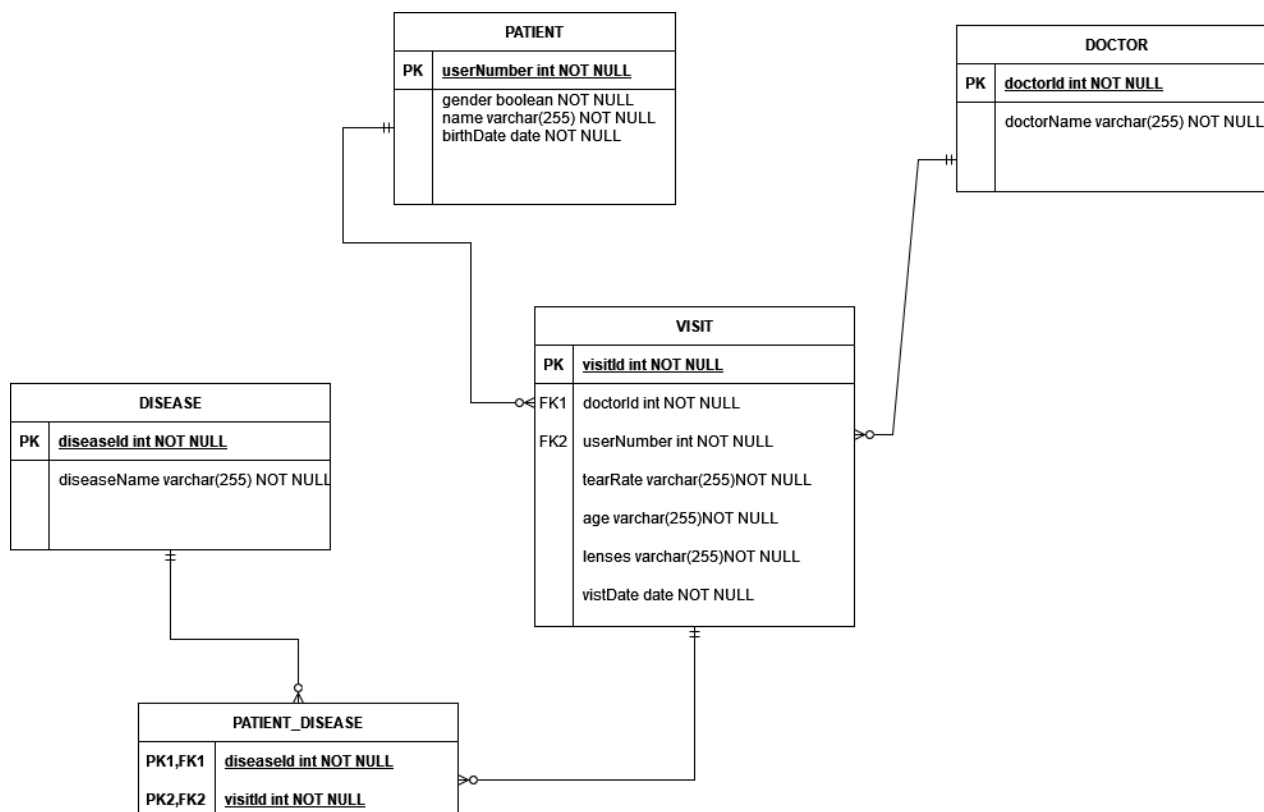


Figura 1 - Modelo conceitual da base de dados

Criação automática da base de dados e exportação dos dados

De forma a facilitar o processo de criação da base de dados foram criados *scripts* que permitem criar automaticamente a base de dados, criar as restrições e povoar a base de dados. De forma a armazenar e gerir os dados foi utilizado PostgreSQL. O PostgreSQL tem a vantagem de oferecer múltiplos recursos para facilitar a gestão e integridade dos dados.

De modo a conseguir exportar os dados no formato correto para poderem ser utilizados no *Orange data-mining framework*, foi necessário primeiro organizar os dados no formato “Three-Row Header”. Este formato é caracterizado por ter três linhas de cabeçalho (*header*). A primeira linha contém os nomes dos atributos, a segunda linha define os tipos dos atributos (neste projeto todos os atributos são do tipo discretos), por fim a última linha define um papel opcional dos atributos, neste caso o único valor presente na terceira linha é “*class*” para a coluna do atributo “*lenses*”, pois esta é a classe dos dados. Por fim os dados são exportados para um ficheiro “.csv” de forma a poderem ser usados para a criação de modelos de aprendizagem automática numa fase seguinte.

Na seguinte figura, pode-se observar um exemplo dos dados exportados, no formato “Three-Row Header”.

	A	B	C	D	E	F	G	H
1	age	prescript	astigmatic	tear_rate	lenses			
2	discrete	discrete	discrete	discrete	discrete			
3					class			
4	young	myope	yes	normal	hard			
5	young	hypermetr	no	reduced	soft			
6	young	hypermetr	yes	reduced	none			
7	young	hypermetr	no	reduced	none			
8	presbyopic	myope	yes	reduced	none			
9	presbyopic	hypermetr	yes	reduced	none			
10	presbyopic	hypermetr	no	reduced	none			

Figura 2 - conjunto de dados com o formato Three-Row Header

Criação de modelos de classificação de dados

Tendo os dados no formato correto para poderem ser utilizados, foram criados modelos em Python de classificação de dados recorrendo a métodos de aprendizagem automática.

De forma a conseguir processar os dados recorreu-se ao uso da biblioteca de mineração de dados do Orange para Python. Esta biblioteca oferece mecanismos que permitem realizar operações na base de dados, como por exemplo baralhar o conjunto de dados, aceder aos valores dos diferentes atributos e separar os dados em categorias e alvo.

Classificação dos dados

No processo de classificação pretende-se, consoante os valores dos dados para as características (*age*, *prescription*, *astigmatic*, *tear rate*) conseguir prever com a máxima precisão o alvo (*lenses*). Os modelos utilizados neste projeto são modelos de aprendizagem supervisionada. Em aprendizagem supervisionada cada instância do conjunto de treino contém uma *label* que indica qual é o seu *output*, desta forma há conhecimento prévio do *output* que os dados devem ter. O algoritmo destes métodos mede a sua precisão através de uma função de erro, ajustando-se até o erro ser minimizado suficientemente. Em contraste, algoritmos de aprendizagem não supervisionada não contêm *labels* para as diferentes instâncias do conjunto de dados, desta forma não há conhecimento prévio do *output* que os dados devem ter.

De forma a poder treinar e testar os modelos é necessário dividir os dados em conjunto de treino e conjunto de teste. O conjunto de treino é responsável por treinar o modelo, e o conjunto de testes é utilizado para avaliar a precisão e o desempenho do modelo. Por defeito em todos os modelos realizados foi realizada uma separação de treino/ teste de 70/30, sendo o conjunto de dados previamente baralhado.



Figura 3 - repartição dos dados

Classificador 1R

O método de classificação 1R baseia-se em determinar qual dos atributos do conjunto de dados fornece a melhor previsão de cada valor da classe.

No algoritmo do classificador 1R inicialmente é necessário para cada atributo:

- Por cada valor, calcular a frequência (*freq*) com que se relaciona a cada valor da classe. Para isto é usado um método para calcular a matriz de contingência, retornando o número de ocorrência do valor de uma classe, consoante o valor de um atributo, para todos os valores do atributo.
- Calcular o erro para cada valor consoante a classe. Isto é realizado através de uma matriz de erro, retornando o erro do valor de uma classe, consoante o valor de um atributo, para todos os valores do atributo. O Erro é igual a $1 - P(\text{valor-classe} \mid \text{valor-atributo})$.
- Escolher os pares (valor-atributo, valor-classe) com menor erro.
- Calcular o erro total do atributo através da soma dos erros dos pares escolhidos.

De seguida tendo o erro total de todos os atributos, escolher o atributo com menor erro e retornar os pares (valor-atributo, valor-classe) associados.

O modelo do classificador 1R foi implementado através de uma classe chamada `ModelOneR`. Esta classe recebe como parâmetros uma *string*, com o nome do ficheiro que contém o conjunto de dados e um *float*, com um valor entre $[0,1]$, com a percentagem de dados para serem usados no conjunto de teste. Os métodos mais importantes desta classe são os seguintes:

- `__splitDataset(self, dataset, testsize)` – Método privado que permite baralhar aleatoriamente um conjunto de dados e dividi-lo em conjunto de treino e teste.
- `fitOneR(self)` – Permite realizar o algoritmo 1R no conjunto de dados, retornando uma lista com os pares (valor-atributo, valor-classe) com menor erro, associados a atributo com menor erro. O método retorna uma lista de listas, uma lista para cada valor do atributo com menor erro, que contém a seguinte informação: [atributo, valorAtributo, valorClasse, percentagemErro, erro, total], sendo o total o número de instâncias para o tuplo (atributo, valorAtributo, valorClasse) e erro, número de instâncias erradas para o mesmo tuplo.

- `predict(self, dic, dataset)` – Permite consoante um conjunto de dados não classificados prever a classe das instâncias, recebendo como argumento o modelo 1R treinado.
- `saveModel(self, model)` – Permite guardar o modelo 1R treinado para um ficheiro Pickle, para ser possível ser utilizado previamente sem ter que estar a treinar o modelo outra vez.

Os resultados obtidos para o classificador 1R foram avaliados através de múltiplas métricas de desempenho para o conjunto de testes, que vão ser apresentadas mais à frente no relatório.

Classificador ID3

O classificador ID3 é um classificador do tipo árvore de decisão. Em árvores de decisão o algoritmo é baseado na criação de uma estrutura em árvore que permite classificar a classe de uma instância a partir de condições efetuadas aos valores dos seus atributos. As árvores de decisão são estruturas que contêm nós e setas, cada nó é utilizado para realizar uma decisão ou representar um resultado, cada seta representa uma questão sim / não.

O algoritmo ID3 para a criação de uma árvore de decisão usa uma abordagem *greedy*, selecionando o melhor atributo que produz máximo ganho de informação ou menor entropia. Entropia é uma medida que mede o grau de incerteza no conjunto de dados.

Para a implementação do classificador ID3 foi utilizada a biblioteca `sklearn`. A implementação do classificador no `sklearn` pode ser feita da seguinte forma:

- **`model=tree.DecisionTreeClassifier(critirion='entropy')`**: Permite criar um modelo de classificador de árvore de decisão ID3, ao definir o *critirion* como *entropy* a árvore de decisão é construída com base em entropia e ganho de informação.

Classificador Naïve Bayes

Classificadores Naïve Bayes são uma coleção de algoritmos de classificação baseados no teorema de Bayes. O teorema de Bayes baseia-se na probabilidade de um evento ocorrer consoante a probabilidade de outro evento já ter ocorrido. Nos algoritmos de classificação Naïve Bayes cada característica a ser classificada é independente das outras.

Os classificadores Naïve Bayes utilizados no projeto foram do tipo Gaussiano e Categórico. Em Naïve Bayes Gaussiano, os valores de cada característica são distribuídos consoante uma distribuição gaussiana. Naïve Bayes Categórico é utilizado para classificação com características discretas que estão distribuídas categoricamente, portanto aplica-se bem ao contexto do projeto.

De forma a criar os classificador utilizaram-se as classes: `sklearn.naive_bayes.GaussianNB` e `sklearn.naive_bayes.CategoricalNB` do `sklearn`.

Codificação dos dados

Para os dados poderem ser utilizados nos classificadores do `sklearn`, não podem estar em formato de *string*. Para contornar este problema, foi necessário realizar um processo codificação dos dados antes da classificação.

Para a codificação das categorias utilizou-se *ordinal encoding* e para as classes *label encoding*. Em *ordinal encoding* cada categoria única é atribuída um valor inteiro. O *label encoding* é utilizado para codificar apenas classes e não múltiplas categorias, codificando as categorias em valores inteiros entre 0 e o número de classes-1. Para os processos de codificação utilizou-se o *ordinal encoder* e o *label encoder* do `sklearn`.

A utilização de *ordinal encoding* pode por vezes ter o problema de forçar uma relação de ordem nos dados. Devido a esta relação de ordem, os modelos de classificação podem assumir uma relação natural de ordem de hierarquia de importância entre os dados o que pode possivelmente afetar o desempenho dos modelos. Uma alternativa para resolver este problema seria a utilização de *One-Hot Encoding*, neste tipo de codificação é criada uma coluna binária para cada categoria e retorna uma matriz esparsa. No conjunto de dados cada atributo não tem uma gama de possíveis valores elevada, portanto o uso de *ordinal encoding* foi suficiente.

Implementação dos classificadores Naive Bayes e ID3

Os modelos dos classificadores Naive Bayes e ID3 foram implementado a partir de uma classe chamada `DeployableModel`. A classe recebe como parâmetro o tipo de classificador a ser utilizado e permite criar e treinar o modelo a partir da utilização de funções do `sklearn`. A classe oferece também métodos para dividir os dados em teste e treino, codificar os dados, avaliar o desempenho dos classificadores e guardar um modelo classificado num ficheiro `pickle`.

Avaliação do desempenho dos diversos classificadores

Os diferentes modelos de classificação referidos anteriormente forem avaliados a partir de diferentes métricas de desempenho. De seguida pode-se observar os resultados obtidos para os diferentes classificadores, com conjunto de teste igual a 30 por cento e conjunto de treino igual a 70 por cento do conjunto de dados. Como os valores obtidos variam bastante consoante o parâmetro *randomstate* do `sklearn train_test_split` utilizado para baralhar os dados, foram realizadas 100 iterações com valores aleatórios para este parâmetro e foi medida a média das métricas de avaliação para as 100 iterações. O parâmetro de *average* usado para *recall*, *precision* e *f1 score* foi do tipo *weighed*.

Os valores obtidos foram os seguintes:

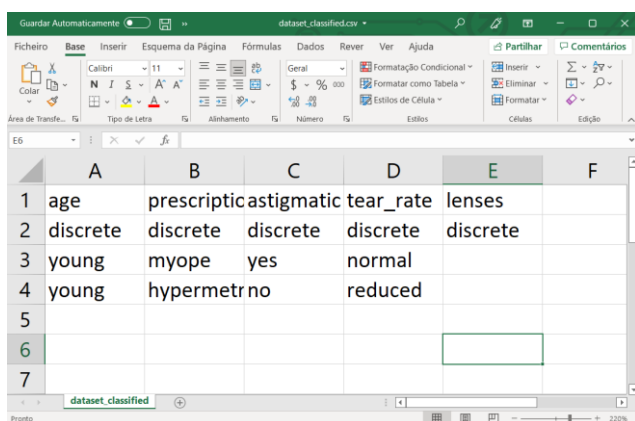
Model	Accuracy	Recall Score	Precision Score	F1 score
OneR	0.42	0.42	0.42	0.38
CategoricalNb	0.72	0.72	0.75	0.71
GaussianNb	0.76	0.76	0.8	0.74
ID3	0.79	0.79	0.84	0.78

Figura 4 - Desempenho dos diversos classificadores

Como se pode observar, o classificador que teve melhor desempenho foi o classificador ID3, tendo a maior taxa de acerto em relação aos outros.

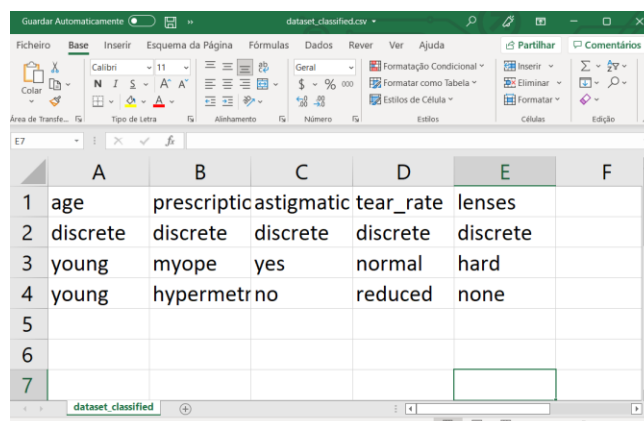
Classificar conjunto de dados não classificados

De forma a poder utilizar os modelos treinados, foi realizado um script em Python que permite classificar, a partir de um modelo treinado, um conjunto de dados não classificados de um ficheiro e retornar um ficheiro csv com todos os dados classificados. Este script utiliza o ficheiro pickle com os modelos previamente treinados para realizar o processo de classificação.



	A	B	C	D	E	F
1	age	prescriptic	astigmatic	tear_rate	lenses	
2	discrete	discrete	discrete	discrete	discrete	
3	young	myope	yes	normal		
4	young	hypermetr	no	reduced		
5						
6						
7						

Figura 5 - ficheiro csv original



	A	B	C	D	E	F
1	age	prescriptic	astigmatic	tear_rate	lenses	
2	discrete	discrete	discrete	discrete	discrete	
3	young	myope	yes	normal	hard	
4	young	hypermetr	no	reduced	none	
5						
6						
7						

Figura 6 - ficheiro csv transformado

Transformação do dataset (csv) para o devido formato (tab)

O conjunto de dados fornecido originalmente encontra-se num ficheiro do tipo csv, em que os dados não têm o formato “*Three-Row Header*”, tendo apenas uma linha de cabeçalho a definir os nomes dos atributos e da classe. Portanto foi necessário converter os dados para um ficheiro do tipo tab com o formato correto.

Para realizar a conversão foram criadas duas funções, a função `createArrayWithThreeHeadedRow(fileName)` e a função `writeToTab(rows, newFile)`. Ambas as funções utilizam o módulo `csv` do Python, para escrever ou para ler do ficheiro.

A função `createArrayWithThreeHeadedRow(fileName)` recebe como argumento o nome do ficheiro csv original e retorna uma lista com todas as linhas do ficheiro csv, com 2 linhas adicionais para o cabeçalho, a função tem o seguinte funcionamento:

1. Abrir o ficheiro csv original em modo de leitura.
2. Criar uma lista vazia para guardar futuramente as linhas do ficheiro csv.

3. Percorrer cada linha do ficheiro original e adicionar à lista.
4. Adicionar 2 linhas adicionais, na segunda e terceira posição da lista, uma para definir o tipo dos atributos e outra a classe dos atributos.
5. Retornar a lista com todas as linhas no formato correto, para poder ser utilizada para escrita de um ficheiro tab.
6. Remover a última linha do ficheiro.

A função `writeToTab(rows, newFile)`, recebe como argumento a lista retornada da função anterior e o nome pretendido para o novo ficheiro tab. Esta função abre um novo ficheiro em modo de escrita e escreve cada linha da lista para o novo ficheiro com o formato específico utilizado em ficheiros tab.

Aplicar o classificador 1R à base de dados de cogumelos (início Project A1)

De forma a classificar os dados da base de dados dos cogumelos foi aplicado o modelo 1R ao conjunto de dados. Foi avaliado o desempenho do classificador para um conjunto de treino e de teste de 70/30. Os resultados obtidos foram os seguintes:

Model	Accuracy	Recall Score	Precision Score	F1 score	Missclassified samples
OneR	0.99	0.986	0.986	0.986	34

Figura 7 - Métricas de desempenho do classificador

	EDIBLE	POISONOUS
EDIBLE	1315	0
POISONOUS	34	1176

Figura 8 - Matriz de confusão

Como se pode observar pelas métricas de desempenho, o algoritmo teve um desempenho muito positivo tendo uma precisão de 0.99. Porém foram obtidos 34 falsos positivos, o que não é algo positivo num conjunto de dados que se baseia na ingestão ou não de uma substância

venenosa. O ideal seria ter um número de falsos negativos superior a falso positivos.

Aplicando o algoritmo 1R ao conjunto de dados sem separar em treino e teste, com o seguinte formato: (attr, valorAttrb, valorClass) : (erro, total), obteve-se o seguinte resultado:

```
( odor, ALMOND, EDIBLE ) : (0, 400)
( odor, ANISE, EDIBLE ) : (0, 400)
( odor, CREOSOTE, POISONOUS ) : (0, 192)
( odor, FISHY, POISONOUS ) : (0, 576)
( odor, FOUL, POISONOUS ) : (0, 2160)
( odor, MUSTY, POISONOUS ) : (0, 48)
( odor, NONE, EDIBLE ) : (120, 3688)
( odor, PUNGENT, POISONOUS ) : (0, 256)
( odor, SPICY, POISONOUS ) : (0, 576)
```

Como se pode observar o atributo escolhido pelo 1R foi o “odor”.

Utilização do Orange para o conjunto de dados

O modelo Orange criado foi o seguinte:

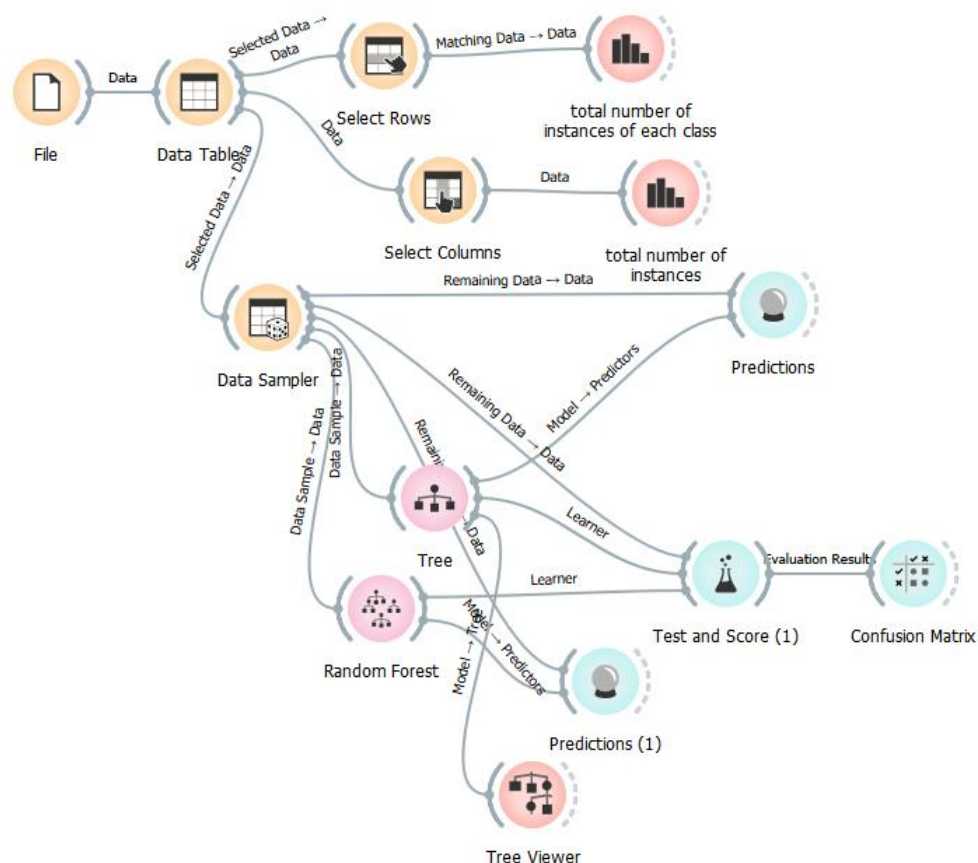


Figura 9 - Modelo Orange

De modo a repartir os dados utiliza-se um operador Data Sampler com *cross-validation*, que reparte 70% dos dados. Para classificação utilizaram-se os operadores *Tree* e *Random Forest*, que representam dois métodos de classificação.

De seguida de forma a avaliar os resultados foi calculada a matriz de confusão, curva ROC, foram realizadas predições e mediu-se métricas de desempenho utilizando o operador *Test and Score*.

		Predicted		Σ
		EDIBLE	POISONOUS	
Actual	EDIBLE	1300	0	1300
	POISONOUS	46	1184	1230
Σ		1346	1184	2530

Figura 11 - Matriz de confusão Tree

		Predicted		Σ
		EDIBLE	POISONOUS	
Actual	EDIBLE	1300	0	1300
	POISONOUS	3	1227	1230
Σ		1303	1227	2530

Figura 10 - Matriz de confusão Forrest

Model	AUC	CA	F1	Precision	Recall
Tree	0.977	0.982	0.982	0.982	0.982
Random Forest	1.000	0.999	0.999	0.999	0.999

Figura 12 - Métricas de desempenho dos classificadores

Como se pode observar pelas métricas de avaliação, o classificador que obteve melhor desempenho foi o *Random Forest*.

3. Conclusões

Em suma neste projeto foram utilizadas e consolidadas diferentes técnicas de gestão de dados, processamento de dados e por fim implementação e teste de algoritmos de classificação.

A partir dos modelos de classificação criados para o conjunto de dados da base de dados “*MedKnown*”, conclui-se que o classificador mais aplicado para o conjunto de dados era o classificador ID3. Uma crítica possível de ser feita em relação aos resultados obtidos é o facto do conjunto de dados utilizado ter uma dimensão muito baixa, tendo apenas 16 instâncias. A partir de conjunto de dados tão pequenos é muito difícil ter uma noção correta do desempenho dos modelos de classificação.

Em contraste, o conjunto de dados fornecido pelo instituto “*FungiData*”, eram de elevada dimensão, permitindo avaliar com maior certeza os diferentes modelos de classificação utilizados. O classificador que teve melhores resultados para o conjunto de dados do “*FungiData*” foi o *Random Forest*, neste tipo de classificadores, na implementação do Orange, os valores em omissão são substituídos pela variável com maior ocorrência, o que é uma grande vantagem.

4. Bibliografia

Paulo Trigo Silva, Aprendizagem e Mineração de Dados, 02 _extracaoDeConhecimento, b01_algoritmosMetodosEssenciais.pdf

Paulo Trigo Silva, Aprendizagem e Mineração de Dados, 02 _extracaoDeConhecimento, b03_avaliacaoDaAprendizagem.pdf

Paulo Trigo Silva, Aprendizagem e Mineração de Dados, 02_ extracaoDeConhecimento, b04en_algorithmsStatisticalSupport.pdf

Paulo Trigo Silva, Aprendizagem e Mineração de Dados, 02_ extracaoDeConhecimento, b05_algoritmosInducaoArvoresDecisao.pdf

One-Hot Encoding vs. Label Encoding using Scikit-Learn:

<https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/>

ID3 Decision Tree Classifier from scratch in Python:

<https://towardsdatascience.com/id3-decision-tree-classifier-from-scratch-in-python-b38ef145fd90>