Instituto Politécnico de Viseu
Escola Superior de Tecnologia e Gestão de Viseu
Mestrado em Engenharia Informática – Sistemas de Informação

Relatório de Projeto Final de Análise Inteligente de Dados

Algoritmos *Supervised Machine Learning* para aprendizagem de modelos de classificação de dados

Viseu, 2022

Instituto Politécnico de Viseu
Escola Superior de Tecnologia e Gestão de Viseu
Mestrado em Engenharia Informática – Sistemas de Informação

Relatório do projeto final de Análise Inteligente de Dados

Algoritmos *Supervised Machine Learning* para aprendizagem de modelos de classificação de dados

23783 – Carolina Ferreira

22382 – Luciano Correia

17025 – Manuel Lopes

Viseu, 2022

Índice

Índice de Figuras

Índice de Tabelas

1 Introdução

Este projeto foi desenvolvido para a unidade curricular de Análise Inteligente de Dados lecionada pelo docente Filipe Pinto, tendo como âmbito o desenvolvimento de uma aplicação que faça a implementação de 4 algoritmos *Supervised Machine Learning* para aprendizagem de modelos de classificação de dados, utilizando dois datasets, disponíveis no site da UCI: *Breast Cancer Coimbra data set* e *Congressional Voting Records data set*.

O projeto consiste no desenvolvimento de uma aplicação que implemente os vários algoritmos *Supervised Machine Learning* sugeridos: *Naive Bayes, Decision Tree, Random Forest* e kNN. Desta forma, torna-se possível tirar algumas conclusões acerca da exatidão de cada algoritmo em situações reais diferentes, neste caso, dois *datasets*.

Os principais objetivos com o desenvolvimento deste trabalho, além de conseguir compreender o modo de funcionamento de cada algoritmo, é perceber e aplicar, na prática, o tratamento dos dados para construção dos modelos de *Machine Learning*. Sendo este tratamento dos dados um passo indispensável e decisivo na hora de construir os modelos, foram testadas várias combinações de codificação dos dados, utilizando *Label Encoding, One Hot Encoding* e *Standard Scaling*, assim como diferentes percentagens de dados de teste (15%, 30% e 50%) para calcular a exatidão e matrizes de confusão dos modelos construídos.

2 Enquadramento

Os algoritmos implementados foram o *Naive Bayes*, *Decision Tree*, *Random Forest* e kNN. Ao longo do relatório será feita a comparação entre a *accuracy* e matriz de confusão obtida a partir dos vários modelos construídos tendo por base cada algoritmo.

Além da comparação direta dos algoritmos também será avaliada a performance dos modelos de *Machine Learning* através do pré processamento que é feito nos dados, nomeadamente a performance nos casos de uso em que são utilizados diferentes *Encoding Methods*: *Label Encoder*, *Label Encoder* com *Standard Scaler*, *Label Encoder* com *One Hot Encoder* e um último caso de uso em que se fez a conjunção entre os três processos mencionados anteriormente.

Para finalizar, foi também testada a *accuracy* e matriz de confusão dos modelos fazendo variar a percentagem dos dados para teste entre 15%, 30% e 50% para análise das diferenças e assim concluir se existe uma grande significância.

3 Desenvolvimento do projeto

A aplicação foi desenhada para ser utilizada de uma forma interativa, dando a opção ao utilizador de escolher como quer tratar os dados do *dataset* antes de construir o modelo, assim como qual a percentagem de dados para teste e o algoritmo a utilizar. Tendo feito a sua escolha são mostrados os resultados relativos à combinação escolhida. Estes resultados englobam tanto a *accuracy* como a matriz de confusão alcançados.

O papel do *Label Encoding* no tratamento de dados é converter valores nominais/categóricos para números. Este passo é essencial em *datasets* com dados que se enquadrem neste tipo. Já em *datasets* constituidos apenas por valores numéricos já não existe essa necessidade de conversão.

Olhando para o *Congressional Voting Records dataset* em que as colunas descritivas são constituídas por valores que variam entre "y", "n" ou "?" é óbvio que é necessário utilizar o *Label Encoder* para converter estes valores para valores numéricos que o modelo possa entender. Desta forma estes valores passaram a ser representados por 0, 1 e 2. Isto levanta um problema que é a distância entre os valores. Se o modelo for construído com os dados nesta disposição terá em conta a distância entre os valores, ou seja, para o modelo será mais diferente os valores 0 e 2, do que o 1 e 2. Para evitar que isto aconteça, excluindo as colunas em que de facto as distâncias entre os valores interessem, como por exemplo nos dados do *Breast Cancer dataset*, pode ser utilizado o *One Hot Encoder* que irá criar N colunas para N variações que existam naquela coluna. Ou seja, recuperando o caso anterior dos valores 0, 1 e 2, a coluna original desmembrar-se-á em 3 novas colunas, cada uma ela com opção de ter o valor 0 ou 1 a indicar se aquele registo corresponde à informação correspondente a cada uma das colunas ou não. Desta forma o problema das distâncias entre valores nominais/categóricos fica resolvido.

O *Standard Scaler*, último processo de *encoding* estudado neste projeto, à semelhança do *One Hot Encoder*, ajuda nas situações em que queremos relacionar os dados de várias colunas utilizando a mesma escala, para não correr o risco de ter "colunas fantasma" no nosso modelo que não têm qualquer influência na predição final dada a sua escala ser bastante reduzida em comparação com a escala de outras colunas do mesmo *dataset*.

Visto isto, em relação ao *Congressional Voting Records dataset*, é esperado que os resultados alcançados sejam melhores quando os dados forem tratados utilizando o *One Hot Encoding*, não tendo qualquer influência o uso do *Standard Scaler Encoding*, uma vez que os dados estarão todos na mesma escala. Já em relação ao *Breast Cancer dataset*, é esperado que se alcancem melhores resultados com a combinação inversa, dado que os dados são numéricos, e a escala destes varia bastante, havendo assim necessidade de uma "*standardization*".

3.1. Algoritmos e Implementação

Naive Bayes

O *Naive Bayes* é um algoritmo *Machine Learning* de classificação probabilística com origem no Teorema de *Bayes*. É simples, mas, ao mesmo tempo, poderoso. O Teorema de *Bayes* é uma fórmula que se traduz na probabilidade condicional de um evento acontecer tendo em conta que outro evento já ocorreu.

A característica principal do algoritmo, sendo o motivo do receber "*naive*" na sua designação é a sua desconsideração total pela correlação entre as variáveis, tratando cada *feature* independentemente. Tendo este fator em conta, se a correlação entre as variáveis for de facto relevante, o *Naive Bayes* tende a falhar na predição de nova informação.

Sendo um algoritmo mais simples e rápido, comparativamente a outros classificadores, o seu desempenho acaba por ser superior, de modo geral. A acrescentar, um número relativamente pequeno de dados de teste é suficiente para classificar com precisão satisfatória e é frequentemente utilizado para classificação de texto, diagnóstico médico e filtro de emails spam.

A aprendizagem é baseada na análise individual de cada atributo e recolha de estatísticas para cada uma das classes.

• BernoulliNB - recolhe com que frequência cada atributo de cada classe é diferente de zero.

• MultinomialNB - leva em consideração o valor médio de cada atributo para cada classe.

• GaussianNB - armazena o valor médio, bem como o desvio padrão de cada atributo para cada classe.

Para realizar uma predição, o novo exemplo será comparado com as estatísticas de cada uma das classes e a melhor classe correspondente é a atribuída.

Neste projeto foi o utilizado o *Gaussian Naive Bayes*. Este classificador é usado quando os valores são de natureza contínua (como acontece no *Breast Cancer Dataset*) e é assumido que seguem uma distribuição de Gauss, Figura 1.

```
def predict_naive_bayes(self):
    classifier = GaussianNB() #Our Model!
    classifier.fit(self.descriptive_train, self.target_train)
    prediction = classifier.predict(self.descriptive_test)
    print(self.print_statistics(self.target_test, prediction))
```

Figura 1 - Função para predição com recurso ao algoritmo Naive Bayes.

Decision Tree

Uma árvore de decisão é uma estrutura semelhante a uma árvore para a tomada de decisão em que cada nó de ramificação representa uma escolha entre uma série de alternativas e cada folha corresponde a uma decisão final. Os modelos *Decision Tree* são amplamente usados para tarefas de classificação e regressão. Essencialmente, eles aprendem uma hierarquia de perguntas se/senão, levando a uma decisão. Esta simplicidade traduz-se em tempos de treino mais reduzidos.

Para construir uma árvore, o algoritmo pesquisa todos os testes possíveis e encontra aquele que é mais informativo sobre a variável de destino. Normalmente, construir uma árvore até que todas as folhas sejam puras leva a modelos que são muito complexos e altamente ajustados aos dados de treino. A presença de folhas puras significa que uma árvore é 100% precisa no conjunto de treino, deste modo deve tentar limitar-se a profundidade da árvore.

As vantagens destes modelos passam pela simplicidade na interpretação e visualização das árvores de decisão, pelo facto de lidarem com dados numéricos e categóricos, a rapidez no treino dos modelos assim como na classificação de novas instâncias. É ainda de salientar que as relações não lineares entre as features não afetam o desempenho da árvore, evitando a standardização ou normalização dos dados. Por outro lado, tem como grande desvantagem a facilidade em cair no *overfitting*, isto é, ajustar-se demasiado bem ao conjunto de dados de treino e, assim, demonstrar incapacidade de prever corretamente novos resultados.

```
def predict_decision_tree(self):
    classifier = DecisionTreeClassifier(criterion="entropy", random_state = 0)
    classifier.fit(self.descriptive_train, self.target_train)
    prediction = classifier.predict(self.descriptive_test)
    print(self.print_statistics(self.target_test, prediction))
```

Figura 2 - Função para predição com recurso ao algoritmo Decision Tree.

Random Forest

O *Random Forest* surge no sentido de colmatar uma das principais desvantagens das árvores de decisão, o ajustamento aos dados de treino (*overfitting*). Estes modelos consistem simplesmente numa coleção de árvores de decisão, onde cada árvore é ligeiramente diferente das outras. Cada árvore pode fazer um trabalho relativamente bom de previsão, mas provavelmente ajustará demasiado parte dos dados de treino. Uma das características mais importantes na construção de cada árvore é a injeção de aleatoriedade para garantir que cada árvore seja, de facto, diferente. Se construirmos muitas árvores, podemos reduzir o *overfitting* e manter a capacidade de generalização, calculando a saída como a média de resultados.

As principais vantagens destes modelos são o melhoramento geral das árvores de decisão, a robustez, precisão e o facto de não necessitarem da normalização dos dados. A principal desvantagem é o facto de serem pouco adequados para *datasets* com muitos atributos (problemas de dimensão elevada) ou de informação esparsa.

```
def predict_random_forest(self):
    classifier = RandomForestClassifier(n_estimators=20, criterion="entropy", random_state=0)
    classifier.fit(self.descriptive_train, self.target_train)
    prediction = classifier.predict(self.descriptive_test)
    print(self.print_statistics(self.target_test, prediction))
```

Figura 3 - Função para predição com recurso ao algoritmo Random Forest.

kNN

O kNN é um algoritmo de aprendizagem fácil capaz de se enquadrar em problemas de classificação como de regressão, apesar de ser mais frequentemente utilizado com eficácia para classificação. É utilizada uma métrica de distância para encontrar as K mais instâncias semelhantes a uma determinada instância e, por fim, seleciona a classe mais comum. Este algoritmo mantém e armazena todos os dados de treino e usa-os para prever o resultado de uma nova instância.

De modo geral, o KNN é um algoritmo intuitivo, poderoso e de fácil implementação, mas, por outro lado, requer demasiados recursos computacionais para prever eficazmente o resultado de uma instância completamente nova. As vantagens da sua utilização passam pela facilidade em interpretar o modelo e pelo facto de funcionar bem considerando um número pequeno de vizinhos (*neighbors*). Por sua vez, as desvantagens centram-se no elevado tempo de previsão quando o conjunto de treino é muito grande (tanto em amostras como atributos) e no facto de, na prática, existir uma dificuldade acrescida em lidar com dados reais de larga escala.

```
def predict_knn(self):
    classifier = KNeighborsClassifier(n_neighbors=5, metric="minkowski", p = 2) #Euclidean Distance
    classifier.fit(self.descriptive_train, self.target_train)
    prediction = classifier.predict(self.descriptive_test)
    print(self.print_statistics(self.target_test, prediction))
```

Figura 4 - Função para predição com recurso ao algoritmo kNN.

4 Resultados Obtidos

Os dados nas tabelas abaixo seguem o modelo proposto pelo professor no enunciado e refletem os resultados obtidos nas diferentes combinações de algoritmos e tratamento de dados, fazendo também variar as percentagens da amostra para teste entre 15%, 30% e 50%.

Além da *accuracy*, como já foi dito anteriormente, é calculada e apresentada a matriz de confusão da predição realizada. Na Figura 5 está um exemplo da matriz de confusão apresentada ao fazer a predição tratando os dados apenas com o *Label Encoder*, 15% de dados de teste e recorrendo ao algoritmo *Naive Bayes*.

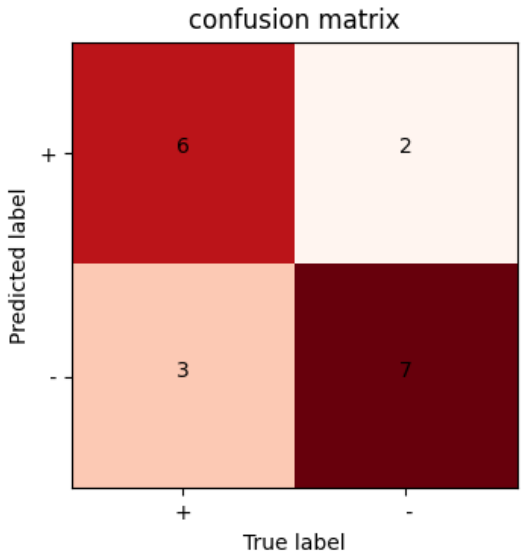

Figura 5 - Exemplo de matriz de confusão calculada.

Fazendo uma breve análise da matriz de confusão é possível ver a distribuição dos dados de teste consoante a sua precisão. Neste caso, existiram 6 registos em que era esperado que o modelo fizesse uma predição do indicador cancerígena positiva, e o modelo acertou, 3 registos que o modelo não detetou nenhum indício cancerígena e devia ter detetado, 2 registos em que o modelo detetou um indício cancerígena quando na realidade não era verdade e mais 7 registos onde o modelo não detetou nenhum indício corretamente.

Com a cor verde, assinala-se em cada uma das tabelas o melhor resultado obtido, refletindo assim a melhor combinação e com a cor vermelha assinalam-se os piores resultados, refletindo a pior combinação. Mais à frente faremos a interpretação mais profunda destes resultados obtidos.

Test size %	Precision	Naive Bayes	Decision Tree	Random Forest	kNN
15	**Label Encoder**	72.22	66.67	66.67	66.67
30		62.86	54.29	65.71	68.57
50		60.34	67.24	50.00	60.34
15	**Label Encoder + Standard Scaler**	72.22	66.67	66.67	44.44
30		62.86	54.29	65.71	54.29
50		60.34	67.24	51.72	62.07
15	**Label Encoder + One Hot Encoder**	55.56	61.11	55.56	66.67
30		51.43	54.29	51.43	60.00
50		53.45	53.49	51.72	63.79
15	**Label Encoder + One Hot Encoder + Standard Scaler**	66.67	61.11	55.56	44.44
30		57.14	54.29	51.43	57.14
50		53.45	53.45	51.72	55.17

Tabela 1 - Resultados obtidos a partir do Breast Cancer data set.

Test size %	Precision	Naive Bayes	Decision Tree	Random Forest	kNN
15	**Label Encoder**	86.36	95.45	95.45	89.39
30		90.08	96.18	96.18	90.08
50		88.07	91.28	96.79	88.99
15	**Label Encoder + Standard Scaler**	86.36	95.45	95.45	89.39
30		90.08	96.18	96.18	90.08
50		88.07	91.28	96.79	89.45
15	**Label Encoder + One Hot Encoder**	95.45	90.91	96.97	90.91
30		94.66	94.66	96.18	90.84
50		89.45	91.28	94.95	90.83
15	**Label Encoder + One Hot Encoder + Standard Scaler**	95.45	90.91	96.97	90.91
30		94.66	94.66	96.18	90.84
50		89.45	91.28	94.95	91.74

Tabela 2 - Resultados obtidos a partir do Congressional Voting Records data set.

Apesar de o *Breast Cancer dataset* ser relativamente pequeno no que diz respeito a amostra de dados, esperávamos obter valores de *accuracy* algo superiores ao verificado. Por esse motivo, decidimos encontrar os valores de correlação entre as variáveis de modo a aferir se esse

fator pode ou não ser impactante nos resultados. As matrizes abaixo refletem os valores de correlação entre as variáveis do *Breast Cancer dataset* e do *Congressional Voting Records dataset*, respetivamente.

Nas duas figuras abaixo dispostas, Figura 6 e Figura 7, estão expostas as matrizes de correlação de cada *dataset*.

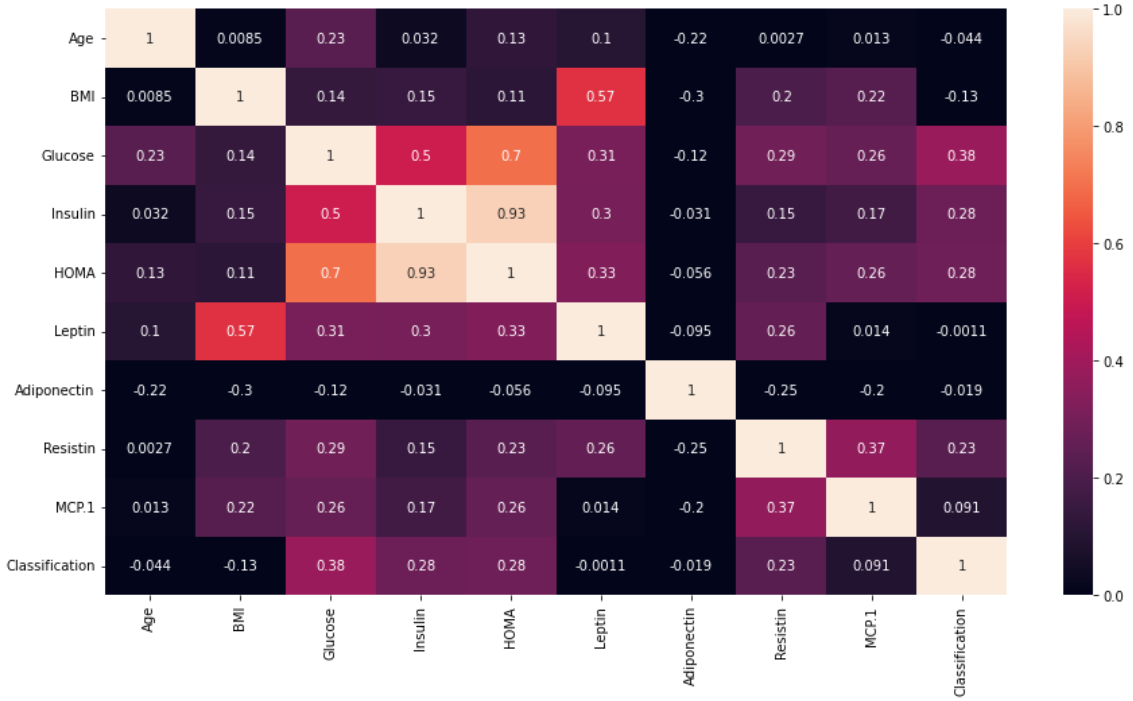

Figura 6 - Matriz de correlação do Breast Cancer Dataset.

À primeira vista notamos que existe uma elevada correlação entre algumas das colunas deste *dataset*, o que aliado ao reduzido número de dados do *dataset* terá sido a causa para as baixas *accuracies* calculadas pelos modelos construídos, Tabela 1. Mais à frente, nas conclusões, serão mais aprofundadas e discutidas as razões que terão levado aos resultados obtidos.

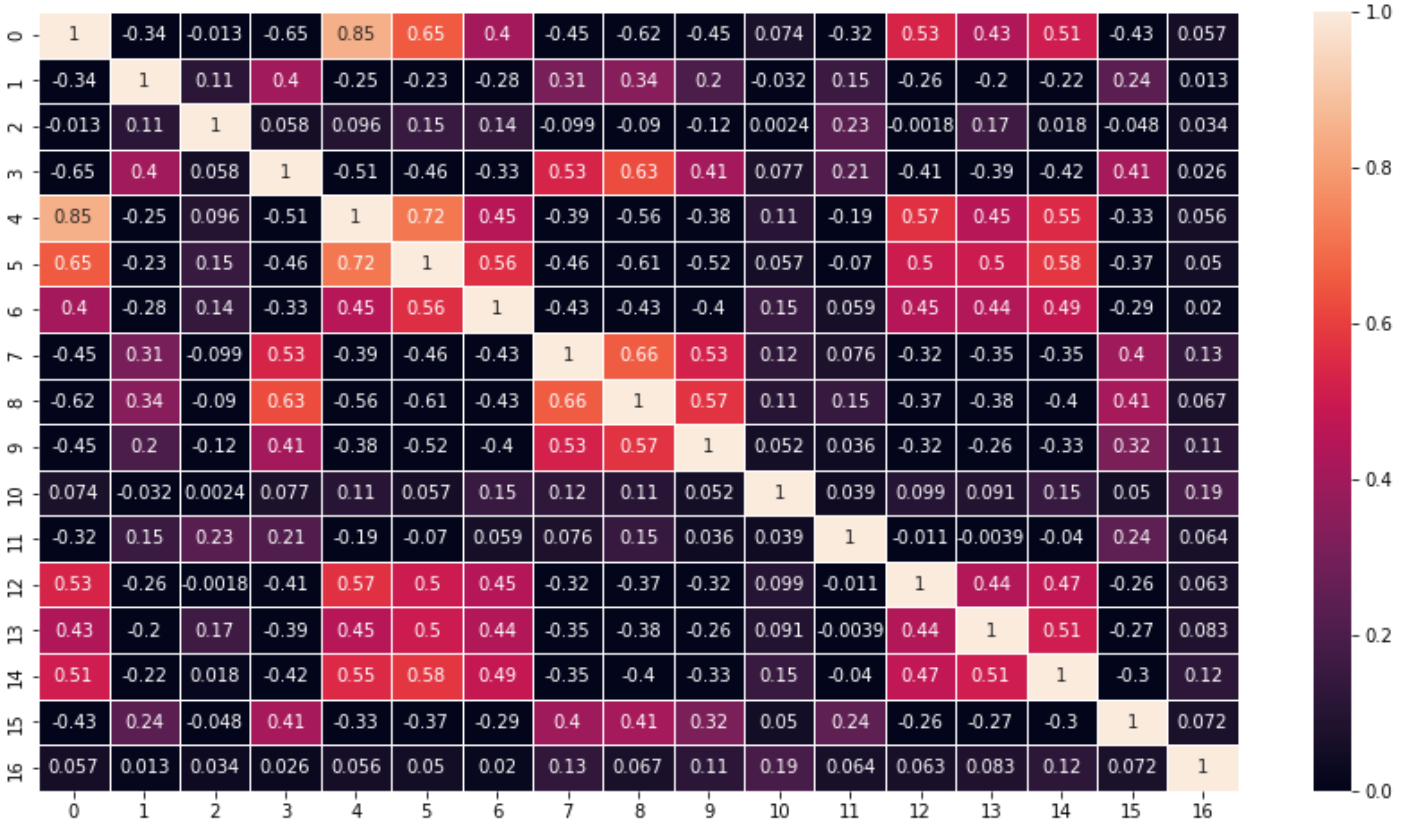

Figura 7 - Matriz de correlação Congressional Voting Records Dataset.

Sob um primeiro olhar, rapidamente verificamos que existem alguns valores elevados na Figura 6 enquanto, por outro lado, encontramos uma grande quantidade de valores de correlação negativos na Figura 7. Valores mais altos estabelecem correlações mais fortes entre duas variáveis e, consequentemente, uma dependência maior entre elas, tornando-se mais difícil perceber o seu peso individual em relação a um target.

Para maior exatidão e conclusões mais rigorosas, procedemos ao cálculo do valor médio de correlação entre variáveis para cada uma das tabelas. O valor médio correspondente ao *Breast Cancer dataset* encontrado foi de aproximadamente 0.1502 e para o *Congressional Voting Records dataset* foi de 0.0337. Esta diferença média das correlações entre as colunas dos *datasets* estudados, podem ajudar a explicar a diferença dos resultados obtidos em cada um. *Datasets* com uma maior correlação média entre as colunas e um número reduzido de registos, faz com que a influência de um registo menos correto seja de maior magnitude, afetando assim as predições calculadas pelo modelo.

5 Conclusões

Olhando para os resultados obtidos utilizando o *Breast Cancer Dataset*, Tabela 1, rapidamente concluímos que nenhum dos classificadores obteve resultados com *accuracy* superior aos 85%, por isso, consideramos que nenhum deles é realmente adequado ao problema começando logo pelo facto deste *dataset* ser demasiado pequeno, reduzindo o número de valores de teste. Como isto acontece, cada resultado de predição errado terá um peso muito grande na *accuracy* final.

À partida, sabendo que o *Naive Bayes* é comumente utilizado para problemas de classificação em contexto de diagnóstico médico, esperávamos obter melhores resultados, no entanto, é possível que isso não tenha acontecido não só pelo *dataset* ser tão reduzido, mas também pelo facto de a correlação entre algumas *features* não poder ser completamente desconsiderada como acontece neste algoritmo. Como exemplo simples temos o caso dos valores da Glucose e da Insulina: se a Insulina estiver em valores não regularizados tendo em conta os valores-padrão, a Glucose, obrigatoriamente, também estará uma vez que a Insulina é responsável por retirar a Glucose do sangue e passá-la para as células. Se o valor da Insulina estiver demasiado baixo, o valor de Glucose no sangue será elevado não devendo ser considerado isoladamente. Regra geral, quando existe uma correlação entre variáveis, isso indica que alterações numa das variáveis induz também a alterações na outra e, logicamente, quanto mais forte for essa correlação, mais difícil será mudar uma variável sem que a outra seja alterada também. Deste modo, pode ser feita uma generalização para os restantes classificadores na medida em que nas *Decision Trees* o *outcome* é afetado já que existe a escolha de uma das variáveis para maximizar o ganho de informação a cada etapa em vez de uma combinação de variáveis. Quanto maior for a correlação entre variáveis, mais difícil será apurar a importância de uma variável isolada para o *target*.

Esta linha de pensamento "teórica" comprovou-se ao analisar os valores de correlação entre as variáveis de ambos os *datasets*. *No Breast Cancer Dataset* observamos que:

- as variáveis das colunas "HOMA" e "Insulin" têm uma correlação de 0,93;
- as variáveis das colunas "HOMA" e "Glucose" têm uma correlação de 0,70;
- as variáveis das colunas "BMI" e "Leptin" têm uma correlação de 0.57.

Tendo em conta que este *dataset* é pequeno e são poucas as suas variáveis, é normal que ao existir uma correlação tão alta entre várias das suas variáveis, o número de predições erradas seja mais alto quando essa correlação é desconsiderada e que cada uma dessas predições tenha um peso muito maior na *accuracy* final, ficando assim com valores abaixo do que seria esperado inicialmente.

Apesar de encontrarmos alguns valores de correlação também elevados entre variáveis do *Congressional Voting Records dataset*, Figura 7, o erro é mais baixo do que no primeiro já que o *dataset* é muito maior e tem um número de variáveis também muito superior. No global, o valor médio de correlação é 0.033 sendo, assim, praticamente irrelevante.

Quanto ao *Congressional Voting Records dataSet*, Tabela 2, os melhores valores de *accuracy* foram obtidos utilizando o algoritmo *Random Forest*, atingindo sempre os 96,97% quando a percentagem do teste se encontrava nos 15%, e tratando os dados com *One Hot Encoder*, como tínhamos previsto uma vez que os dados do *dataset* eram dados nominais e com todos os possíveis valores eram igualmente relevantes, sendo assim importante eliminar a distância entre estes quando os dados foram tratados pelo *Label Encoder*. Os valores não diferiram muito entre o *Random Forest* e o *Decision Tree*, ainda assim, existe a possibilidade de que o *Random Forest* tenha feito a correção de algum *overfitting* do *Decision Tree*, explicando assim a diferença de alguns pontos percentuais entre ambos os classificadores. Tendo em conta que este *dataset* tem como possíveis valores "yes", "no" ou "?", para atribuir um valor a cada variável, era expectável que os melhores valores fossem obtidos nestes dois classificadores.

Os piores resultados foram obtidos com o *Naive Bayes* e kNN, mas, ainda assim, mantiveram-se acima dos 85%. O algoritmo kNN, terá tido resultados inferiores possivelmente por utilizar o número de vizinhos como critério. Ora, o problema é que os dados são relativos a votos acerca de temas que geram alguma discussão e alguns poderão até ser bastante ambíguos entre os membros de um partido, o que torna a divisão dos membros de cada partido mais difícil, aumentando assim a imprevisibilidade da predição. Além desta ambiguidade entre os temas que foram a votos e as ambições de cada pessoa do partido, uma das razões para o algoritmo *Naive Bayes* não se ter adaptado tão bem aos dados como os restantes, foi certamente o facto deste algoritmo funcionar melhor em dados contínuos, o que não se verifica no *dataset* em questão.

De modo geral, é possível concluir também que os melhores resultados foram obtidos quando a divisão dos dados da amostra entre treino e teste foi de 85 e 15 porcento respetivamente. Isto terá acontecido devido ao facto de utilizando uma maior percentagem de testes, a amostra de dados de treino do modelo ser reduzida, diminuindo também, consequentemente o treino do mesmo, o que levava a predições mais falíveis.

Referências

Austria, Yolanda & Goh, Marie & Jr, Lorenzo & Lalata, Jay-Ar & Goh, Joselito & Vicente, Heintjie. (2019). Comparison of Machine Learning Algorithms in Breast Cancer Prediction Using the Coimbra Dataset. International journal of simulation: systems, science & technology. 10.5013/IJSSST.a.20.S2.23.

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[Patricio, 2018] Patrício, M., Pereira, J., Crisóstomo, J., Matafome, P., Gomes, M., Seiça, R., & Caramelo, F. (2018). Using Resistin, glucose, age and BMI to predict the presence of breast cancer. BMC Cancer, 18(1).

Xia, Y. (12 de Janeiro de 2021). Why Naive Bayes fails to tell if you have Covid-19. Obtido de Data Driven Investor: https://medium.datadriveninvestor.com/why-naive-bayes-fails-to-tell-if-you-have-covid-19-54aae712a590

Badr, W. (18 de Janeiro de 2019). towards data sciente. Obtido de Why Feature Correlation Matters …. A Lot!: https://towardsdatascience.com/why-feature-correlation-matters-a-lot-847e8ba439c4