Análise de Dados Técnicas de Aprendizagem Automática

1st Gonçalo Medeiros
Departamento de Engenharia Informática
Instituto Superior de Engenharia do Porto
Porto, Portugal
1211388@isep.ipp.pt

2st Guilherme Rodrigues
Departamento de Engenharia Informática
Instituto Superior de Engenharia do Porto
Porto, Portugal
1211474@isep.ipp.pt

3rd Cátia Remelgado
Departamento de Engenharia Informática
Instituto Superior de Engenharia do Porto
Porto,Portugal
1210787@isep.ipp.pt

I. Introdução

O artigo científico "Análise Exploratória de Dados dos niveis de obesidade em pessoas com idade entre os 14 e os 61 anos" foi escrito como parte da Unidade Curricular (UC) de Análise de Dados em Informática (ANADI). É parte do 6º semestre da Licenciatura em Engenharia Informática (LEI) do Instituto Superior de Engenharia do Porto (ISEP). O artigo segue as várias fases do desenvolvimento de modelos de aprendizagem automática. A linguagem Python foi usada para analisar e processar os dados relacionados à estimativa dos níveis de obesidade em pessoas com idades entre os 14 e os 61 anos e diversos hábitos alimentares e condições físicas.

Objetivos

O objetivo principal deste artigo consiste na aplicação de algoritmos de aprendizagem automática na exploração de dados e respetiva comparação usando os testes estatísticos mais adequados.

De forma a responder de maneira positiva ao problema anteriormente enunciado, foram identificados os seguintes objetivos especıficos:

- Definir a metodologia de trabalho
- Análise e discussão dos resultados com recurso ao Phyton
- Escrita de Artigo Técnico com a Análise de Dados

II. Estado da Arte

Regressão Linear Simples

Uma regressão linear é um modelo estatístico que analisa a relação entre uma resposta variável (geralmente chamada de Y) e uma ou mais variáveis e suas interações (geralmente chamadas de X ou variáveis explicativas ou preditores).

Regressão Linear Múltipla

Quando uma regressão leva em conta dois ou mais preditores para criar o linear regressão, é chamada de regressão linear múltipla (MLR). Para MLR, a variável dependente ou alvo (Y) deve ser contínua/real, mas o preditor ou a variável independente pode ser de forma contínua ou categórica. Cada variável de recurso deve modelar o relacionamento linear com a variável dependente. MLR tenta ajustar uma linha de regressão através de um espaço multidimensional de pontos de dados.

Árvore de Regressão

Uma árvore de regressão é um tipo de árvore de decisão. A árvore de regressão usa a soma dos quadrados e análise da regressão para prever os valores do campo destino.

Rede Neuronal

As redes neurais são modelos de aprendizagem automática inspirados no funcionamento dos neurônios biológicos. Elas consistem em uma rede de neurônios artificiais organizados em camadas, onde cada neurônio realiza operações simples de cálculo e transmissão de sinais. Essas redes podem ser configuradas de várias formas e aprendem ajustando os pesos das conexões entre neurônios. Historicamente, as redes neurais têm sido usadas para uma variedade de aplicações, desde a solução de problemas de classificação e regressão até tarefas mais complexas, como reconhecimento de padrões em imagens e processamento de linguagem natural. Uma das vantagens principais das redes neurais é sua capacidade de lidar com dados complexos e não lineares de forma robusta, devido à natureza não linear das funções de ativação. No entanto, uma desvantagem significativa é a dificuldade de interpretação humana dos modelos resultantes.

Árvore de Decisão

Uma árvore de decisão é uma representação visual de um modelo de aprendizagem automática que toma decisões com base nos valores dos atributos. Ela consiste em nós de decisão que testam atributos específicos e ramos que representam os possíveis resultados desses testes. A árvore começa no nó raiz e desce até chegar a nós folha, onde são feitas as classificações. O objetivo é criar uma estrutura

que separe eficientemente os dados em classes distintas, facilitando a tomada de decisões com base nos atributos fornecidos.

SVM

O Support Vector Machine (SVM) é um algoritmo de aprendizagem supervisionada que classifica dados encontrando uma linha ou hiperplano ótimo que maximiza a distância entre cada classe em um espaço N-dimensional. Este algoritmo é usado para classificação linear ou não linear, regressão e detecção de outliers. As SVMs podem ser aplicadas em tarefas como classificação de texto, imagens, detecção de spam, identificação de escrita à mão, análise de expressão genética, detecção facial e detecção de anomalias. Uma das vantagens principais das SVMs é sua capacidade de lidar eficientemente com dados de alta dimensionalidade e relacionamentos não lineares. No entanto, uma desvantagem é a sensibilidade na escolha dos hiperparâmetros, como a função de kernel e o parâmetro de regularização. SVMs são robustas em relação ao ruído nos dados, mas podem ser computacionalmente caras para grandes conjuntos de dados

KVizinhos-mais-próximos

O algoritmo k-Nearest Neighbor (kNN) é uma técnica de aprendizagem supervisionada utilizada para classificar instâncias de dados com base na similaridade com os seus vizinhos mais próximos. É reconhecido pela sua simplicidade e capacidade de lidar com problemas de classificação e regressão. Sua eficiência computacional pode ser comprometida em conjuntos de dados de grande escala. O kNN é não paramétrico, dependendo da informação local dos pontos de dados vizinhos para fazer previsões. A escolha do parâmetro k é crucial para o desempenho do kNN, exigindo técnicas de validação cruzada para determinar o valor ideal. Em resumo, o kNN é uma ferramenta flexível e intuitiva para classificação e regressão, cuja eficácia depende da gestão cuidadosa dos parâmetros e da complexidade computacional.

K-fold cross validation

A validação cruzada k-fold é uma técnica de validação de modelos que divide o conjunto de dados em k subconjuntos (ou folds) aproximadamente iguais. O modelo é treinado k vezes, cada vez usando k-1 folds como dados de treinamento e 1 fold como dados de validação. Isso permite que cada subconjunto seja usado como dados de validação exatamente uma vez. Ao final das k iterações, as métricas de desempenho são geralmente agregadas para fornecer uma estimativa mais precisa do desempenho do modelo. A validação cruzada k-fold é amplamente utilizada para avaliar a capacidade de generalização de um modelo e ajudar na seleção de hiperparâmetros. Essa abordagem é especialmente útil quando se tem um conjunto de dados limitado e se deseja utilizar cada observação tanto para treinamento quanto para validação.

III. Realização do projeto

A. Análise e Discussão de Resultados

Primeiramente, leram-se os dados do ficheiro que contém uma estimativa dos níveis de obesidade em pessoas com idades entre os 14 e os 61 anos e diversos hábitos alimentares e condições físicas. Foram recolhidos dados de 2111 pessoas, nos quais foram obtidos 17 atributos relacionados com hábitos alimentares e condição física dos participantes

Regressão

Nesta parte do trabalho começamos por derivar um novo atributo, "IMC" usando a informação dos atributos do Peso e Altura. Para isso utilizamos o seguinte algoritmo : IMC = kg/m2.

De seguida, analisamos os atributos do conjunto de dados mais significativos usando gráficos, análises estatísticas. Depois da criação dos gráficos e análises conseguimos revelar alguma informação interessante. Neste primeiro grafico podemos ver a distribuição do IMC, e com a devida a análise podemos perceber que existe um grande conjunto que se encontra num intervalo preocupante, ou seja, acima dos 25 que significa que estao acima do peso aceitável para a sua altura. Tambem podemos verificar um grande pico que representa um conjunto que se encontra dentro do limite saudável que se estende do 18.5 ate 24.9.

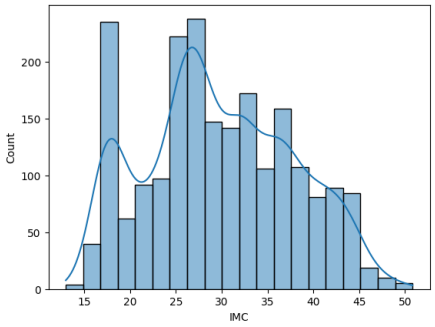

Fig. 1. Gráfico distribuição IMC

Neste gráfico, podemos ver no que toca ao género que enquanto os dois apresentam uma distribuição normal em quase todas as categorias, podemos observar uma grande distinção entre os casos de obesidade severa e mórbida.

Neste ultimo gráfico podemos ver que através de uma simples observação, os individuos que ja tenham um historico de obesidade na familia, ficam com mais propensão a seguir o mesmo caminho.

A matriz de correlação é uma ferramenta estatística que nos permite avaliar as relações entre variáveis. Esta vai exibir os coeficientes de correlação entre diferentes fatores, indicando a força e a direção dessas conexões. Como podemos verificar a matriz de correlaçao gerada através dos valores numéricos

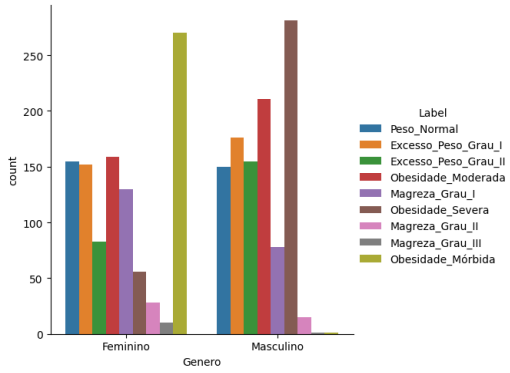

Fig. 2. Comparação entre géneros

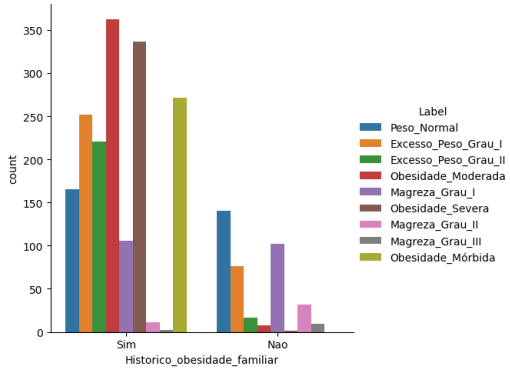

Fig. 3. Comparaçao entre historico familiar

apresenta pouca correlação com exceção de algumas variéveis como o peso e altura.

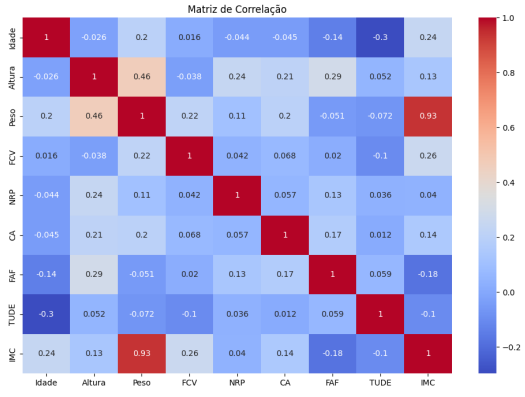

Fig. 4. Matriz de Correlação

Acabamos por criar um modelo de regressão linear simples para a variável “IMC” usando o atributo relativo à “Idade”

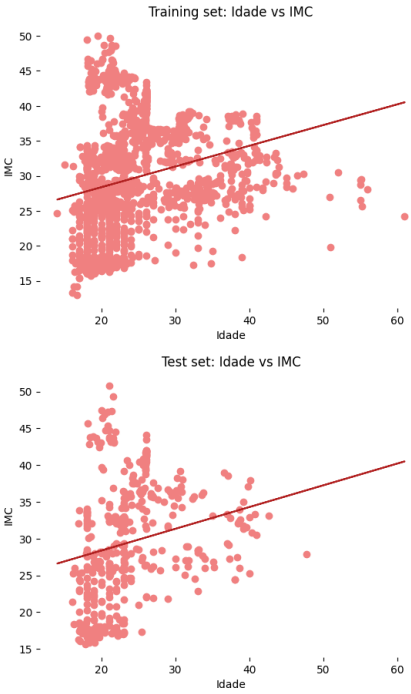

Fig. 5. Idade vs IMC

de cada registo. A funçao linear resultante acabou por ser: IMC = 0.30 * Idade + 22.48. Como podemos ver nos gráficos apresentados a escolha do atributo ”Idade” não foi o melhor para obtermos uma maior precisão e aproximação a linha. Então escolhemos o atributo ”Peso” que resultou numa maior precisao e num maior grau de confiança do modelo. Podemos então concluir que o peso ajuda ao modelo a ter uma melhor precisão devido á sua melhor relção com o IMC.

Regressão linear múltipla

	Actual	Predicted
0	25.269124	25.296631
1	47.718705	49.374647
2	40.870732	40.063927
3	29.146663	28.982068
4	32.873110	31.992046

TABLE I
COMPARAÇÃO ENTRE VALORES REAIS E PREVISTOS

Através da tabela anterior conseguimos perceber que as precisões são razoavelmente precisas com uma média absoluta de erro (MAE) de 0.569 e um erro quadrático médio (MSE) de 0.560. Isso indica-nos que o modelo está a fazer uma boa precisão dos valores do IMC. O R2 score é de 99,13% da variabilidade nos dados.

Árvore de Regressão

Mean Absolute Error Train set: 0.619
Mean Absolute Error Test: 0.775
Mean Absolute Error: 0.775

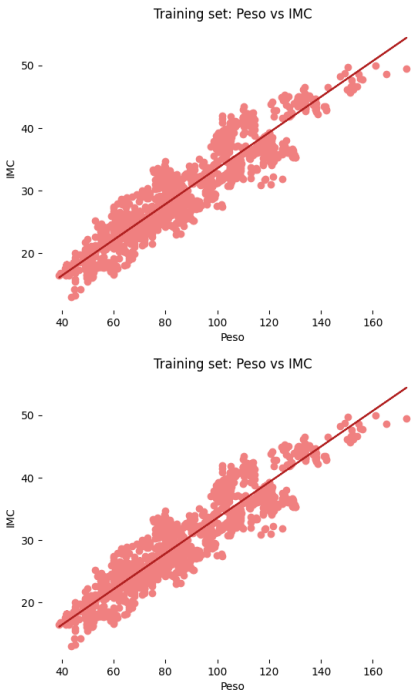

Fig. 6. Peso vs IMC

Root Mean Squared Error: 1.0449

Observando os resultados obtidos conseguimos perceber que os resultados previstos irão estar próximos dos valores reais.

Rede Neuronal

Test set R$\hat{2}$: 0.99696
Test set RMSE: 0.01

Estes resultados permite-nos perceber que o modelo está a obter resultados bastante precisos do IMC. Os resultados obtidos com a rede neuronal superam os resultados da Regressão Linear Múltipla e a Árvore de Regressão.

CLASSIFICAÇÃO

Árvore de Decisão

O código está a implementar um modelo de Árvore de Decisão com validação cruzada K-Fold para o conjunto de dados para prever o risco de obesidade de cada indivíduo utilizando a função `DecisionTreeClassifier`.

Assim, cria um modelo de Árvore de Decisão utilizando a função `DecisionTreeClassifier` com os parâmetros `max_depth` que limita a profundidade máxima da árvore a 12, impedindo que ela se ajuste excessivamente aos dados de treinamento sendo um valor para evitar um modelo muito complexo, `min_samples_leaf` que define que cada nó folha deve ter pelo menos 5 amostras e `random_state`, configurado como 42, assim estes parâmetros previnem overfitting.

Após iterar através de todos os folds, calcula a accuracy de 86.13% e o desvio padrão de 2.54% das precisões.

Outras Arquiteturas

Com um `max_depth` de 12, `min_samples_leaf` de 15 e `random_state` de 42, o código obteve uma *Mean Accuracy* de 84.83269935193013 e *Standard Deviation* de 2.293008180205751.

Com um `max_depth` de 5, `min_samples_leaf` de 5 e `random_state` de 42, o código obteve uma *Mean Accuracy* de 81.6955480417019 e *Standard Deviation* de 1.5839834544360578.

Como podemos observar, outras arquiteturas demonstraram um desempenho pior na accuracy.

SVM

Este modelo define o número de 10 ”folds” para a validação cruzada K-Fold, uma técnica que divide o conjunto de dados em K partes (ou ”folds”), treina o modelo em K-1 dessas partes, e depois testa-o na parte restante. Em seguida, cria um modelo SVM com um kernel linear dado que foi o que obteve o melhor desempenho e define uma grelha de parâmetros para o `GridSearchCV` para encontrar os melhores parâmetros para o modelo e evitar overfitting. Treina o modelo SVM com a grelha de parâmetros usando o `GridSearchCV`. Inicializa listas para armazenar as métricas de cada fold. Itera através de cada fold, treinando o modelo nos dados de treino, fazendo previsões nos dados de teste, e calculando várias métricas (precisão, sensibilidade, especificidade e F1).

Assim concluímos que a melhor arquitetura foi {’C’: 100, ’gamma’: ’scale’}, com uma accuracy de 94.41 e desvio padrão de 0.014718977839040005.

Outras Arquiteturas

Com o kernel `rbf` obtivemos uma *Mean Accuracy* de 84.6521% e *Std Accuracy* de 0.007915673273625039.

Com o kernel `poly` obtivemos uma *Mean Accuracy* de 80.6733% e *Std Accuracy* de 0.009973558381620371.

Redes Neuronais

Este modelo define uma lista de otimizadores e inicializadores de pesos para serem testados no modelo e inicializa listas para armazenar a accuracy e desvio padrão. Itera sobre cada combinação de otimizador e inicializador, construindo um modelo de rede neural para cada combinação. A arquitetura escolhida foi de 12 camadas, sendo a mais complexa com 64 nós na última camada devido aos outputs.

Compila o modelo utilizando a função de perda `sparse_categorical_crossentropy` e a accuracy. Treina o modelo nos dados de treino e avalia o modelo no conjunto de teste, calculando as previsões e as métricas. Por fim, encontra e imprime o resultado com a maior accuracy que foi 94.53%, juntamente com o melhor otimizador Nadam e melhor inicializador normal.

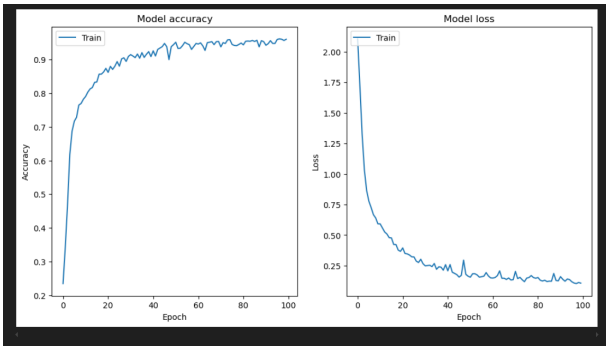

Fig. 7. Enter Caption

Outra Arquitetura

Uma arquitetura com mais nós obteve uma acurácia de 91.72%, com o melhor otimizador sendo o SGD e o inicializador heuniform.

```
for optimizer in optimizers:
    for initializer in initializers:
        model4 = tf.keras.Sequential([
            tf.keras.Input(shape=(17,)),
            tf.keras.layers.Dense(units=128, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=96, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=64, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=64, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=32, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=32, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=32, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=32, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=32, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=32, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=9, activation="softmax", kernel_initializer=initializer)
        ])
```

Fig. 8. Rede com mais nós

Uma arquitetura com mais uma camada obteve uma acurácia de 91.252%, com o melhor otimizador sendo o SGD e o inicializador lecununiform.

```
for initializer in initializers:
        model4 = tf.keras.Sequential([
            tf.keras.Input(shape=(17,)),
            tf.keras.layers.Dense(units=64, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=48, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=32, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=32, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=32, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=16, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=16, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=16, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=16, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=16, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=16, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=16, activation="relu", kernel_initializer=initializer),
            tf.keras.layers.Dense(units=9, activation="softmax", kernel_initializer=initializer)
        ])
```

Fig. 9. Rede com mais uma camada

KNN

O modelo implementa K-Nearest Neighbors (KNN) utilizando a função `KNeighborsClassifier` com iteração manual sobre os parâmetros. Depois de iterar sobre todas as combinações de parâmetros, ajusta o modelo KNN com os melhores parâmetros. Avalia o modelo usando validação cruzada e imprime a accuracy de 82.71%, o desvio padrão de 1.90, os melhores parâmetros {'Nneighbors': 7, 'weights': 'distance', 'metric': 'manhattan', 'p': 1, 'algorithm': 'auto', 'leaf_size': 20} da accuracy. Para além disso foi feito uma versão do método usando `GridSearch` na qual foi obtido o mesmo resultado.

```
Melhores parâmetros: {'n_neighbors': 7, 'weights': 'distance', 'metric': 'manhattan', 'p': 1, 'algorithm': 'auto', 'lea
Média de acurácia: 82.71014039166593
Desvio padrão da acurácia: 1.9052524493296161
Melhor Sensitivity: 0.8767535399877123
Melhor Specificity: 1.0
Melhor F1 Score: 0.8532706052040157
```

Fig. 10. KNN

Verificação de diferença significativa no desempenho dos dois melhores modelos

O código tem como objetivo comparar o desempenho de dois modelos, usando um nível de significância de 5%. Neste caso, os melhores modelos são o SVM com uma accuracy de 94.41 e a Rede Neuronal com accuracy de 94.53. As pontuações da accuracy são extraídas e armazenadas na variáveis e também impressas. Para comparar as pontuações dos dois modelos, é realizado um teste t de duas amostras independentes, utilizando a função `ttest_ind` da biblioteca `scipy.stats`. Esse teste retorna a estatística t e o valor p, que são armazenados nas variáveis `tstat_svm3` e `pval_svm3`, respectivamente. A partir do valor p obtido de 0.000183597807, verificamos que a diferença entre as pontuações de acurácia dos dois modelos é estatisticamente significativa. Como o valor p foi menor que 0.05, conclui-se que há uma diferença significativa no desempenho dos modelos e ao comparar as médias das pontuações de acurácia dos dois modelos, o modelo de redes é considerado superior.

```
Pontuações de acurácia do Modelo SVM: [0.9385342789598109, 0.9241706
Pontuações de acurácia do Modelo 3: [19.621749222278595, 37.58865296
Modelo SVM vs Modelo 3: t statistic: -4.03, p value: 0.000183597807
Há uma diferença significativa no desempenho dos dois modelos.
O Modelo 3 (Rede Neural) tem melhor desempenho.
```

Fig. 11. Diferença 2 melhores modelos

Métricas: Accuracy; Sensitivity; Specificity e F1

Ao comparar os resultados dos modelos anteriores, observamos que o modelo SVM apresentou um excelente desempenho geral em termos de accuracy, com uma média de 94.41% e um desvio padrão de 0.0147. Além disso, o SVM mostrou uma sensibilidade de 0.8428, especificidade de 0.9782 e um F1 score de 0.8430. Porém, a rede neural superou o SVM em termos de accuracy, alcançando uma média de 94.59%, com sensibilidade de 0.8588, especificidade de 0.9499 e um F1 score de 0.8625. O modelo KNN, apesar de ter a melhor sensibilidade de 0.8768 e a melhor especificidade de 1.0, apresentou a pior accuracy média de 82.71% e um desvio padrão de 1.9053, indicando uma maior variabilidade nos seus resultados, com um F1 score de 0.8533. A árvore de decisão teve uma acurácia média de 86.14% e um desvio padrão de 2.5484, com sensibilidade de 0.7827, especificidade

de 0.9135 e o pior desempenho em termos de F1 score 0.7875. Assim, podemos concluir que, embora a Rede Neuronal tenha o melhor desempenho geral, o KNN destaca-se pela sua alta sensibilidade e especificidade, enquanto SVM e a árvore de decisão apresentam desempenhos intermediários.

```
Mean Accuracy: 86.1362355593125
Standard Deviation: 2.54838911500018
Mean Sensitivity: 0.7827384177957944
Mean Specificity: 0.9135336349620337
Mean F1 Score: 0.7874788988131181
```

Fig. 12. Métrica da Árvore

```
Kernel: linear, Best Params: {'C': 100, 'gamma': 'scale'}, Mean Accuracy: 94.4104960057365, Std Accuracy: 0.01471897783
Mean Sensitivity: 0.842750330489853 6
Mean Specificity: 0.9781540024785043
Mean F1 Score: 0.8429529946140623
```

Fig. 13. Métrica SVM

```
Melhor accuracy: 94.56264972686768
Melhor otimizador: Nadam
Melhor inicializador: normal
Sensibilidade: 0.8588638436458402
Especificidade: 0.9499999984166666
F1: 0.8625301603643802
```

Fig. 14. Métrica da Rede

```
Média de accuracy: 82.71014039166593
Desvio padrão da accuracy: 1.9052524493296161
Melhor Sensitivity: 0.8767535399877123
Melhor Specificity: 1.0
Melhor F1 Score: 0.8532706052040157
```

Fig. 15. Métrica KNN

Seleção de atributos

O código realiza a seleção de atributos para melhorar o desempenho dos modelos de classificação.

Com base na avaliação dos modelos desenvolvidos, podemos afirmar que a seleção de atributos não melhorou o desempenho dos modelos de classificação. Os resultados obtidos após a seleção de atributos foram os seguintes:

- Árvore de Decisão: accuracy de 79.20%
- SVM: accuracy de 76.12%
- Rede Neural: Acurácia de 19.62%
- KNN: accuracy de 78.01%

Comparando estes resultados com os valores anteriores dos modelos sem seleção de atributos:

- Árvore de Decisão: accuracy antes de 86.14% e passou para 79.20%
- SVM: accuracy antes de 94.41% e passou para 76.12%
- Redes Neurais antes de 94.56% e passou para 19.62%
- KNN: accuracy antes de 82.71% e passou para 78.01%

Verificamos que a seleção de atributos levou a uma piora no desempenho dos modelos, especialmente o modelo das redes, que como o modelo apresenta uma arquitetura mais complexa, que era necessária para o modelo anterior, faz com que o modelo seja demasiado complexo para o número de atributos. Todos os modelos tiveram desempenhos inferiores após a seleção de atributos.

```
                          Atributo        Score
4                             Peso  1452.245126
0                      Unnamed: 0   832.248399
5   Historico_obesidade_familiar   112.280739
1                         Genero    75.714166
7                            FCV    72.074753
9                           CCER    68.415289
2                          Idade    55.266528
15                           CBA    31.989371
6                          FCCAC    27.949344
8                            NRP    18.417692
accuracy do Modelo 1: 0.7919621749408984
accuracy do Modelo 2: 0.7612293144208038
14/14 [==============================] - 0s 81
accuracy do Modelo 3: 0.19621749222278595
accuracy do Modelo 4: 0.7801418439716312
```

Fig. 16. Seleção de Atributos

Seleção de novos preditores

Este código realiza a criação de novos preditores a partir dos dados disponíveis no conjunto de dados fornecido. Utiliza-se o índice de massa corporal (IMC) e a conjução dos preditorws derivado dos atributos existentes fumador e CBA. Em seguida, treina os modelos SVM e Rede Neural com a inclusão deste novo preditor.

Os resultados são os seguintes: accuracy do Modelo SVM com IMC e FumadorCBA de 95.27% e da Rede Neural 91.72%. A análise estatística realizada através do teste t de uma amostra indica que não há uma diferença significativa no desempenho do modelo SVM com a utilização do IMC (valor p = 0.3068). No entanto, há uma diferença significativa no desempenho do modelo 3 com a inclusão do IMC e FumadorCBA (valor p = 2.94e-18). Esses resultados sugerem que a inclusão dos preditores teve um impacto significativo no desempenho da rede neural, mas não afetou significativamente o desempenho do modelo SVM.

O gráfico mostra que a precisão do modelo aumenta rapidamente nas primeiras 20 epoch e depois estabiliza em torno de 0.9, enquanto a perda diminui drasticamente inicialmente

e depois estabiliza em torno de 0.1. Isso indica que o modelo está aprendendo bem.

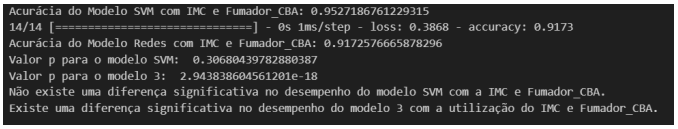

Fig. 17. Novos Preditores

Redes Neuronais atributo Genero

Este modelo define uma lista de otimizadores e inicializadores de pesos para serem testados no modelo e guarda a accuracy e desvio padrão.
O modelo obteve uma accuracy de *93.85%*, o melhor otimizador para o modelo é *RMSprop* e o melhor inicializador *glorot normal* .
Com uma especificidade de *92.85%* e uma sensibilidade de *93.84%*.

SVM atributo Genero

Melhores resultados do modelo SVM:

- Melhor acurácia: 92,99
- Melhor parâmetro C: 10
- Melhor parâmetro gamma: 'auto'
- Sensibilidade: 93,08
- Especificidade: 93,46
- F1-score: 92,97

Estes resultados indicam que o modelo SVM apresentou um bom desempenho, com uma acurácia de quase 93%. A escolha dos parâmetros C=10 e gamma='auto' parece ter sido eficaz para obter este resultado.
A sensibilidade de 93,08% mostra que o modelo tem uma boa capacidade de detetar corretamente os casos positivos. Já a especificidade de 93,46% indica que ele também consegue identificar bem os casos negativos.

Em comparação com outros trabalhos, esses resultados ficam próximos aos obtidos em , onde o SVM alcançou 98,6% de acurácia. Porém, ficam abaixo dos 94,53% reportados no estudo da rede neuronal.
Portanto, pode-se concluir que o modelo SVM com os parâmetros escolhidos apresentou um bom desempenho, com acurácia, sensibilidade e especificidade acima de 93%. Mas ainda há espaço para melhorias para atingir resultados ainda melhores.

CONCLUSÕES

Regressão

Na parte da regressão, começamos por derivar o atributo "IMC" a partir dos dados de Peso e Altura usando a fórmula IMC = kg/m². Em seguida, analisamos os dados de forma gráfica e estatística, revelando que muitos indivíduos têm um IMC acima de 25, indicando obesidade, enquanto outros estão dentro do intervalo saudável de 18.5 a 24.9. Observamos também diferenças significativas na distribuição do IMC entre os gêneros, especialmente nos casos de obesidade severa e mórbida. Notamos que indivíduos com histórico familiar de obesidade são mais propensos a desenvolver a condição. Utilizamos uma matriz de correlação para identificar as relações entre variáveis, mostrando que peso e altura são os mais correlacionados. Finalmente, criamos um modelo de regressão linear simples para prever o IMC com base na Idade, resultando na fórmula IMC = 0.30 * Idade + 22.48. No entanto, a Idade não se mostrou um bom preditor, então usamos o atributo Peso, que forneceu maior precisão e confiança no modelo.

Árvore de Decisão

O modelo atual com `max_depth=12` e `min_samples_leaf=15` tem a maior precisão média (86.14%) entre as três arquiteturas testadas, o que sugere que essa configuração é a mais eficaz para este conjunto de dados em termos de classificação. O desvio padrão dos modelos alternativos sugere que a precisão é mais consistente (menor variabilidade) quando a profundidade da árvore é menor. A configuração do modelo atual é mais complexa (`max_depth=12`), o que pode explicar a maior precisão, mas também pode levar a maior variabilidade. Em contrapartida, os modelos com menor profundidade são mais simples e oferecem resultados mais consistentes, mas com menor precisão.

SVM

A utilização de 10 "folds" para a validação cruzada permitiu que o desempenho fosse avaliado de forma abrangente em diferentes conjuntos de dados. A utilização do `GridSearchCV` foi crucial, garantindo que o modelo final fosse ajustado de forma ótima aos dados de treino e evitando o *overfitting*. Com os parâmetros {`'C': 100, 'gamma': 'scale'`}, o modelo SVM demonstrou um desempenho excepcional, alcançando uma precisão média de 94.41% com um desvio padrão de 0.0147. As outras arquiteturas testadas com os kernels 'rbf' e 'poly' apresentaram menor desempenho, com precisões médias de 84.65% e 80.67%, respectivamente.

Redes Neuronais

O modelo com a maior acurácia alcançada foi de 94.0898%, utilizando o otimizador `Nadam` e o inicializador `normal`. Outras arquiteturas foram testadas, incluindo uma com mais nós e outra com mais camadas. A arquitetura com mais nós obteve uma acurácia de 91.7258%, utilizando o otimizador `SGD` e o inicializador `he_uniform`, enquanto a arquitetura com mais camadas alcançou uma acurácia de 91.2529%, com o otimizador `Nadam` e o inicializador `lecun_uniform`. Se o modelo se tornar demasiado complexo para o conjunto de dados, adicionar mais nós ou camadas pode prejudicar o desempenho. Embora adicionar mais nós ou camadas possa parecer uma maneira intuitiva de melhorar o desempenho de uma rede neural, é importante considerar cuidadosamente a complexidade dos dados e os desafios associados à otimização

e generalização do modelo. Nem sempre mais complexo significa melhor desempenho.

KNN

Os melhores parâmetros identificados para o modelo de classificação foram {'n_neighbors': 7, 'weights': 'distance', 'metric': 'manhattan', 'p': 1, 'algorithm': 'auto', 'leaf_size': 20}. Esses parâmetros foram selecionados com base na precisão do modelo durante a validação cruzada. Encontrar os valores ideais é crucial para evitar *overfitting* ou *underfitting*. O KNN é considerado não paramétrico, o que significa que sua complexidade aumenta com o tamanho do conjunto de dados de treinamento. Além disso, o cálculo das distâncias pode ser computacionalmente intensivo, especialmente para grandes conjuntos de dados.

Verificação de diferença significativa no desempenho dos dois melhores modelos

O valor p obtido foi 0.000183597807, indicando uma diferença significativa entre as pontuações de precisão dos modelos. Com um valor p menor que 0.05, conclui-se que há uma diferença estatisticamente significativa no desempenho dos modelos. Assim, com base na análise estatística, o terceiro modelo é considerado superior ao SVM em termos de desempenho.

Métricas: Accuracy; Sensitivity; Specificity e F1

Ao comparar os resultados dos modelos, observou-se que o SVM teve uma ótima precisão, com média de 94.41% e desvio padrão baixo. A rede neural superou o SVM em precisão média, alcançando 94.59%. O KNN destacou-se pela sensibilidade e especificidade, apesar de ter a pior média de precisão. Tanto o KNN quanto a árvore de decisão mostraram variabilidade nos resultados. No geral, a rede neural teve o melhor desempenho, seguida pelo KNN em termos de sensibilidade e especificidade, enquanto o SVM foi consistente, e a árvore de decisão teve o pior desempenho em termos de F1 score.

Seleção de atributos

A avaliação dos modelos após a seleção de atributos revelou uma queda significativa no desempenho em comparação com os resultados anteriores. Todos os modelos mostraram uma diminuição na acurácia, sugerindo que a seleção de atributos pode ter removido informações relevantes. Isso indica que os modelos originais eram mais eficazes sem a seleção de atributos.

- Árvore de Decisão: de 86.14% para 79.20%
- SVM: de 93.38% para 76.12%
- Rede Neural: de 94.09% para 19.62%
- KNN: de 82.71% para 78.01%

Seleção de novos preditores

Os resultados revelam que o Modelo SVM, com IMC e FumadorCBA, alcançou uma precisão de 95.27%, enquanto a Rede Neural obteve 91.72%. O teste estatístico mostrou que não há diferença significativa no desempenho do SVM com a inclusão do IMC (p-valor = 0.3068). No entanto, houve uma diferença significativa no desempenho do modelo 3 com a inclusão do IMC e FumadorCBA (p-valor = 2.94e-18). Isso sugere que a inclusão desses preditores teve um impacto significativo no desempenho da rede neural, mas não afetou o desempenho do SVM de forma significativa.

Rede Neuronal vs SVM

Os resultados obtidos para as redes neuronais e o modelo SVM mostram que ambos os modelos apresentaram desempenhos razoáveis para o atributo "Genero". A rede neuronal obteve uma acurácia de 93,85%, enquanto o modelo SVM alcançou uma acurácia de 92,99%. Embora o modelo SVM tenha um desempenho menor, ele ainda apresenta uma boa capacidade de detetar casos positivos e negativos. Em resumo, ambos os modelos apresentaram resultados satisfatórios, mas há espaço para melhorias para atingir resultados ainda melhores.

REFERENCES

[1] J. Ross Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco, CA, 1992.

[2] Christopher Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[3] Catarina Silva & Bernardete Ribeiro, *Aprendizagem Computacional em Engenharia*, Imprensa da Universidade de Coimbra, 2018.

[4] Sebastian Raschka, *STAT479 FS18: Intro to Machine Learning*, Fall 2018.

[5] Tom Mitchell, *Machine Learning*, McGraw-Hill, 1997.

[6] Catarina Silva & Bernardete Ribeiro, *Aprendizagem Computacional em Engenharia*, Imprensa da Universidade de Coimbra, 2018.

[7] Chantal D. La and Daniel T. Larose, *Data Science Using Python and R*, John Wiley Sons, Inc., 2019.

[8] What are support vector machines (SVMs) ?, IBM, [Online]. Available: https://www.ibm.com/topics/support-vector-machine.

[9] Catarina Silva & Bernardete Ribeiro, *Aprendizagem Computacional em Engenharia*, Imprensa da Universidade de Coimbra, 2018.

[10] Sebastian Raschka, *STAT479 FS18. L01: Intro to Machine Learning*, Fall 2018.