Tarefa 3: Regredir e Classificar

João Victor Schamne, RA: 2236710

Universidade Tecnológica Federal do Paraná (UTFPR)
Curitiba – PR – Brasil

Resumo. *Este artigo descreve e oferece uma visão geral de como a Tarefa 3: Regredir e Classificar foi desenvolvida, quais foram as abordagens e qual foi o raciocínio utilizado para a solução do problema proposto, quais foram os resultados e o desempenho das técnicas utilizadas código e quais conclusões foram tiradas em relação às soluções desenvolvidas.*

1. Introdução

O problema em questão baseia-se em utilizar técnicas de aprendizado de máquina para reconstituir o cálculo (regressão) que determina a gravidade e a classificação da gravidade a partir do histórico de sinais vitais coletados das vítimas.

Dentro deste problema geral, os subproblemas resolvidos são apresentados nas seguintes subseções:

1.1. Armazenar os dados relevantes do arquivo "sinaisvitais_hist.txt" em uma matriz

Isso foi feito para organizar os dados importantes (pulso, qualidade da pressão, respiração, gravidade e classe) dentro de uma matriz, para depois usar esses dados para treinar e validar os modelos.

1.2. Implementar a validação cruzada k-folds

Foi criada uma função "randomLines()" para separar o dataset em dados para treinar o modelo e dados para validá-lo. Essa função é chamada dentro da função "validacaoCruzada()", que após separar os dados de treinamento e de avaliação, realiza o treino e a validação dos modelos, e obtém a acurácia deles.

1.3. Implementar perceptron multicamadas com scikit-learn

Foi utilizada a biblioteca scikit-learn para implementar um perceptron multicamadas, pois ele é capaz de realizar regressão e classificação

1.4. Implementar árvore de decisão com scikit-learn

Foi utilizada a biblioteca scikit-learn para implementar árvores de decisão. O scikit-learn implementa o algoritmo CART (Classification and Regression Trees), pois o ID3 apenas realiza classificação. O algoritmo CART constrói árvores binárias usando o recurso e o limite que geram o maior ganho de informação em cada nó.

2. Fundamentação Teórica

As técnicas presentes na solução foram implementadas utilizando a biblioteca scikit-learn, pois é uma biblioteca fácil de usar que contém os recursos necessários para resolver os problemas desta tarefa. As descrições das técnicas utilizadas estão descritas nas seguintes subseções:

2.1. Perceptron multicamadas

Foi utilizada a técnica de redes neurais (Multilayer Perceptron), pois ela é capaz de realizar regressão e classificação. Foram criadas duas funções: uma que realiza a regressão usando "sklearn.neural_network.MLPRegressor()", e outra que realiza classificação, implementando "sklearn.neural_network.MLPClassifier()".

2.2. Árvore de decisão

Foram criadas duas funções para implementar a técnica de árvore de decisão usando o recurso "tree.DecisionTreeRegressor()" para a regressão, e "tree.DecisionTreeClassifier()" para classificação. O scikit-learn utiliza o algoritmo CART para árvores de decisão, pois permite realizar regressão e classificação.

3. Metodologia

Para cada uma das técnicas escolhidas, foram testados diversos parâmetros.

3.1. Perceptron multicamadas

Como foi descrito na seção 2.1, foram criadas duas funções, uma para realizar a regressão com o perceptron multicamadas, e outra para realizar a classificação. Dentro dessas funções, o modelo é treinado (usando o método fit()) e validado (usando o método score()). Ambas retornam a acurácia e o modelo treinado.

Os parâmetros que foram testados foram:

- **hidden_layer_sizes:** número de neurônios em cada camada oculta;
- **activation:** função de ativação;
- **solver:** otimizador para fazer os ajustes de peso na rede neural;
- **alpha:** um parâmetro para regularização/penalidade, que combate o overfitting restringindo o tamanho dos pesos;
- **random_state:** determina a geração de números aleatórios para pesos e bias;
- **max_iter:** número máximo de iterações, o otimizador irá iterar até alcançar esse número máximo ou convergir;
- **learning rate init:** taxa de aprendizado inicial para a atualização de pesos;

3.2. Árvore de decisão

Como foi descrito na seção 2.2, foram criadas duas funções, uma para realizar a regressão com a árvore de decisão e outra para realizar a classificação. Dentro dessas funções, o modelo é treinado (usando o método fit()) e validado (usando o método score()). Ambas retornam a acurácia e o modelo treinado.

Os parâmetros testados foram:

- **criterion**: a função para medir a qualidade de uma divisão;
- **max_depth:** profundidade máxima da árvore;
- **max_leaf_nodes:** o número máximo de nós folhas da árvore;
- **min_samples_leaf:** o número mínimo de amostras necessárias em um nó folha;
- **min_weight_fraction_leaf:** a fração ponderada mínima da soma total de pesos necessária para estar em um nó folha;
- **splitter:** a estratégia utilizada para escolher a divisão em cada nó;
- **random_state:** controla a aleatoriedade do estimador;

4. Resultados e análise

A seguir, estão os resultados das técnicas na etapa de avaliação de acordo com diferentes parâmetros testados para cada uma delas e a análise destes resultados.

4.1 Resultados da regressão realizada pelo MLP

- Acurácia: 74,4%
- Raiz quadrada do erro quadrático médio: 8,3
- Parâmetros:
    - 'hidden_layer_sizes': [(100)],
    - 'activation': ['relu'],
    - 'solver': ['lbfgs'],
    - 'alpha': [0.001]
    - ‘max_iter’: [2000]

4.2 Resultados da classificação realizada pelo MLP

- Acurácia: 100%
- Precision: 1
- Recall: 1
- F-score: 1
- Parâmetros:
    - 'hidden_layer_sizes': [(100)],
    - 'activation': ['tanh'],
    - 'solver': ['adam'],

- 'alpha': [0.05],
- ‘max_iter’: [2000]

4.3 Resultados da regressão realizada pela árvore de decisão

- Acurácia: 92%
- Raiz quadrada do erro quadrático médio: 4,75
- Parâmetros:
  - 'criterion':['poisson'],
  - 'max_depth': [8],
  - 'max_leaf_nodes': [80],
  - 'min_samples_leaf': [5],
  - 'min_weight_fraction_leaf': [0.0001],
  - 'splitter': ['best'],
  - 'random_state': [24]

4.4 Resultados da classificação realizada pela árvore de decisão

- Acurácia: 70%
- Precision: 0.566
- Recall: 0.7
- F-score: 0.6
- Parâmetros:
  - 'criterion':['gini'],
  - 'max_depth': [8],
  - 'max_leaf_nodes': [5],
  - 'min_samples_leaf': [5],
  - 'min_weight_fraction_leaf': [0.0001],
  - 'splitter': ['random'],
  - 'random_state': [24]

5. Conclusões

A técnica que apresentou o melhor desempenho para regressão, foi a árvore de decisão, pois apresentou 92% de acurácia e a raiz quadrada do erro quadrático médio foi de 4,75.

A técnica de classificação que apresentou a maior acurácia (100%), maior precisão, recall e f-score. No entanto, isso pode indicar overfitting, o que não é desejado. A árvore de decisão apresentou uma acurácia de 70%, precisão de 0.5, recall de 0.7, f-score de 0.6.

6. Referências bibliográficas

https://scikit-learn.org/

7. Apêndice

O código foi desenvolvido no Windows, para executá-lo, basta abrir a pasta num editor de código, e executar o arquivo “main.py”.