Título do Trabalho : Previsão de sucesso/insucesso de alunos de LOP

Introdução

O relatório em questão trata de tentar prever, através de um algoritmo de aprendizado de máquina, se um aluno regularmente matriculado na disciplina de Lógica de Programação da Escola de Ciências e Tecnologia pode ser aprovado ou não, o mais rápido possível, isto é, no máximo até a metade de um semestre regular poder dar uma estimativa da chance de um aluno qualquer obter sucesso ou insucesso na disciplina. O trabalho é composto pelos alunos Tobias Aguiar e Leandro de Souza. A motivação se da em razão das turmas no curso de ciências e tecnologia apresentarem quantidades muito grandes de alunos e nem sempre é possível atender a todos aqueles que estão com dificuldades, o que acaba sendo um problema para estes alunos que acabam sendo desmotivados, o que os levam ao insucesso ou a desistência da disciplina. Para solucionar isso e conseguir atingir esses alunos o mais rápido possível, desenvolveu-se um algoritmo de aprendizado de máquina (Multi-Layer Perceptron, será explicado mais adiante) para dar maior suporte a um maior número de alunos que passam por dificuldades. Para isso, dispõe-se de uma base de dados onde se contém parâmetros de vários estudantes, a exemplo de questões realizadas durante uma semana, quantidade de questões submetidas, dentre outros.

Metodologia

O método usado para previsão do desempenho do aluno na matéria Lógica de programação foi o multilayer perceptron (MLP). Perceptron Multicamadas (PMC ou MLP — Multi Layer Perceptron) é uma rede neural com uma ou mais camadas ocultas com um número indeterminado de neurônios. A camada oculta possui esse nome porque não é possível prever a saída desejada nas camadas intermediárias. Tal método possui 4 passos: inicialização, ativação, treinamento dos pesos e iteração.

Antes de aplicar algoritmos de Machine Learning a qualquer problema, é necessário sempre dividir os dados históricos nestes 2 grupos: treino e teste. Assim, treinamos o algoritmo com um grande volume de dados de treino; depois pode-se validar o resultado deste algoritmo com os dados de teste; e só então poderemos colocar nosso algoritmo em produção, com confiança de que ele realmente

consegue prever dados reais. O treinamento foi composto por 80% do total e o restante foram dados do teste.

Foram selecionados os seguintes atributos, conforme a tabela abaixo. Através desse atributos, espera que o aluno que possua essas características bem sucedidas seja aprovado, ou seja, o aluno que está estudando na lista, no portal LOP e está sendo eficaz.

Tabela 01- Vetor de entrada

Nome	Significado	Motivação
totalsub	total de submissões feitas	Mede se o aluno está praticando
igualACeml123	total de submissões que acertou 100%	Mede se o aluno é eficaz
igualACeml123	total de submissões que acertou 100%	Mede se o aluno é eficaz
subListaExer23	quantidade de submissão na lista de exercício	Identifica se as listas estão sendo feitas
subListaExer45	quantidade de submissão na lista de exercício	Identifica se as listas estão sendo feitas
subListaExer67	quantidade de submissão na lista de exercício	Identifica se as listas estão sendo feitas

Códigos

A primeira parte consiste na definição da biblioteca e endereçamento do banco de dados. Logo após, é escolhida o vetor de entrada e a variável de saída, conforme a figura 01. Para isso, foi usado a função “dataset.iloc”.

Figura 01- Escolha do vetor de entrada e variável de saída

```
X = dataset.iloc[:,[ 19, 20, 21, 46, 47, 48]].values
y = dataset.iloc[:, 11].values

print(X[0:6,:])
```

Na obtenção do conjunto de dados do treinamento e teste (figura 02), foi usado a função “train_test_split”, em que ela pode retornar as variáveis devidas. Os parâmetros usados foram os conjunto de entrada e saída definido anteriormente, o “test_size” e o “random_state”. O “test_size” define a proporção da quantidade de teste comparado com o total e o “random_state” define o tipo de sorteio.

Figura 02- Código da definição de dados de treinamento e teste e suas escalas.

```
[5]  # Dividindo o conjunto de dados no conjunto de treinamento e no conjunto de teste
     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
     #"test_size" refere ao tamanho do teste comparado ao total; random faz o sorteio da proporção; c
```

```
[6]  # Feature Scaling
     from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()  #escala do dados, normalizaçao
     X_train = sc.fit_transform(X_train)   #atualização de escala , agr todas estao de 0-1
     X_test = sc.transform(X_test)
```

Outra parte importante da implementação do MultiLayer Perceptron é definição de camadas de operação(figura 03), no caso, foi a de entrada, a segunda e da saída. Na figura 04, pode-se ver a implementação do predict, função que classificar a saída dos dados em 1 ou 0, gerando duas classes distintas. Por último, ainda na figura 04, é criada a matriz de confusão para poder visualizar o desempenho do algoritmo.

Figura 03- Código dos ajustes das camadas

```
[9]  # Adding the input layer and the first hidden layer
     classifier.add(Dense( activation = 'relu', input_dim = 10 , units = 16, kernel_initializer = 'un


     # Adding the second hidden layer
     classifier.add(Dense( activation = 'relu', units = 20, kernel_initializer = 'uniform' ))


     # Adding the output layer
     classifier.add(Dense( activation = 'sigmoid', units = 1, kernel_initializer = 'uniform'))
```

Figura 04 - Código da definição do predict e obtenção da matriz de confusão.

```
[12] # Part 3 - Making the predictions and evaluating the model

     # Predicting the Test set results
     y_pred = classifier.predict(X_test)
     print(y_pred[0:10])

     y_pred = (y_pred > 0.5)
     print(y_pred[0:10])

     # Making the Confusion Matrix
     from sklearn.metrics import confusion_matrix
     cm = confusion_matrix(y_test, y_pred)
```

Experimentos

Só foi avaliado um parâmetro, que no caso em questão foi a situação final do aluno (aprovado ou reprovado), através do uso de atributos que considerou-se os que mais pesaram para o resultado final total de submissões feitas, taxa de acerto e quantidade de submissão na listas de exercícios .

Com isso , os resultados indicam uma taxa de acerto de 71.42 %, onde também é indicado erros e acertos através da matriz de confusão, o que é razoável e mostra a possibilidade de dar maior suporte a alunos com dificuldades. Espera que se for adicionados atributos como frequência do aluno na aula, na monitoria, experiência prévia aumente o acerto e possa identificar os alunos que têm mais possibilidade de reprovar.