Mineração de Dados Educacionais
Profa. Patricia A. Jaques Maillard
CDTEC – PPGC
UFPEL

Aula 5 – Exercício de Programação
Regressão

Esse exercício de programação envolve modificar o código do exercício de programação da Aula 3 para tornar esse um problema de regressão. Basicamente, no lugar de predizer se o aluno foi aprovado, o modelo deve predizer qual será a nota final do aluno. E tem ainda alguns pré-processamentos adicionais que vamos fazer.

Suponha que você foi contratado para ajudar a escola 'EscolaLegal' a melhorar a taxa de aprovação em uma disciplina específica, que tem apresentado resultados abaixo do esperado nos últimos anos. A escola 'EscolaLegal' disponibilizou uma base de dados aberta que contém informações dos alunos que se matricularam na disciplina nos últimos 5 anos, incluindo informações sobre suas notas nas edições anteriores, gênero, idade, carga horária semanal de estudo, entre outras.

Você decide usar uma técnica de mineração de dados para construir um modelo que possa prever, com base nas informações disponíveis, qual vai ser a nota final do aluno na disciplina em questão. Para isso, você vai utilizar uma técnica de regressão.

Vamos utilizar a base de dados "*student-performance*" disponível no repositório *UCI Machine Learning*. Essa base de dados contém informações sobre alunos nas disciplinas de matemática e português de uma escola secundária em Portugal. Vamos criar um modelo que preveja se um aluno terá sucesso ou não na disciplina de matemática ou português (você escolhe) com base em algumas variáveis como gênero, idade, número de reprovações, carga horária semanal de estudo, entre outras.

Na página seguinte segue algumas orientações para o desenvolvimento do trabalho.

No entanto, eu indico que você tente **primeiramente resolver o problema por sua conta e apenas após leia as orientações**. Dessa forma, a sua leitura já vai ser mais engajada pelas dificuldades experimentadas e você vai se dar conta melhor do que você fez diferentemente e que novos comandos são úteis.

1) Entendendo o problema

A partir do enunciado você sabe que precisa prever a nota final do aluno na disciplina de matemática ou português a partir de notas parciais e de outros dados.

1.1) Você sabe que é um problema de regressão. Você precisa predizer a nota final do aluno na disciplina.

1.2) A nossa variável de saída (output label) é G3, ou seja, ela contém a nota a ser predita.

Vamos importar então os dados e interpretá-los para tentar entender melhor o problema.

2) Importando os dados:

Primeiramente, você deve baixar a base de dados do UCI Machine Learning que se encontra em https://archive.ics.uci.edu/ml/datasets/Student+Performance. Referência:

- P. Cortez and A. Silva. Using Data Mining to Predict Secondary School Student Performance. In A. Brito and J. Teixeira Eds., Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) pp. 5-12, Porto, Portugal, April, 2008, EUROSIS, ISBN 978-9077381-39-7.

2.1) Você pode fazer o download dos dados e descompactar o arquivo zip. Abra o arquivo `student-mat.csv` (para a disciplina de matemática) ou `student-por.csv` (para a disciplina de português) no Google Colab. Esses são os arquivos que contêm as notas dos alunos em matemática e português, além de outros dados. Após, use o comando `df = pd.read_csv(filepath, delimiter=';')` (onde `df` pode ser qualquer nome que você escolheu para o dataframe que receberá os dados importados) do Pandas, `filepath` é o caminho do arquivo, e `delimiter=';'` é o delimitador csv dos dados (geralmente é , ou ;).

```
df = pd.read_csv(filepath, delimiter=';')
```

2.2) Você pode escolher apenas um dos arquivos para ler, já que você vai criar um modelo para apenas uma das disciplinas: matemática OU português.

3) Entendendo os dados a partir de uma análise descritiva:

Vamos começar fazendo uma análise descritiva dos dados. Essa análise vai nos ajudar a entender melhor os dados para decidir que tipo de ajustes seria necessário realizar.

3.1) Primeiramente, você pode usar o comando `df.head(numero_linhas)` para ter uma visualização rápida dos dados. Ele mostra as primeiras `numero_linhas` dos dados importados.

- Essa visualização vai lhe ajudar a ter uma ideia dos dados. Você também pode olhar o arquivo `student.txt` que veio junto no arquivo zipado para ter uma descrição dos dados. Ex:

```
df.head()
```

3.2) Você pode também usar `df.info()` para obter uma descrição rápida dos dados. Ele mostra o número de colunas, os nomes das colunas, número de entradas não-null para cada coluna e os tipos dos dados nas colunas. Você vai ver que o Pandas salvou as strings com o tipo `object` e os números como inteiros. Isso não vai ser um problema

para nós, mas se desejar você pode usar o parâmetro `dtype` do comando `read_csv()` para configurar os tipos desejados para as colunas.

- Com esses comandos, você vai ter mais informações complementares sobre os dados que podem lhe ajudar a decidir que variáveis você quer utilizar.
- Ex:

```
df.info()
```

3.3) Você também pode usar o comando `df.describe()` para visualizar uma descrição qualitativa dos dados. Esse comando vai retornar os valores máximo e mínimo, média, desvio padrão e percentis para cada uma das variáveis. Também retorna a quantidade de dados em cada coluna. Talvez você queira usar o comando `pandas.set_option('display.precision', 2)` para configurar a precisão dos números, o que vai mudar a visualização, como a seguir.

- Ex:

```
pd.set_option('display.precision', 2)
df.describe()
```

- Esses comandos ajudam a ver os valores dos dados e a sua distribuição. Podem lhe ajudar posteriormente no processamento. Por exemplo, você vai perceber que o valor máximo no G3 é 20, enquanto no G1 e no G2 é 19. Como G1, G2 e G3 representam as notas dos alunos em determinados períodos, possivelmente todas elas têm o mesmo valor máximo que é 20. Assim, você pode usar o valor 20 para configurar a normalização dos dados. Voltaremos a esse ponto mais adiante.
- Aqui também podemos ver se temos valores faltantes (nulos) pela linha `count`. Todas as colunas têm 395 células, assim não temos dados faltantes.

4) Pré-processando os dados

Agora que você já visualizou os dados, vamos trabalhar neles, começando pelo pré-processamento.

4.1) Vamos primeiramente começar removendo as instâncias com dados faltantes com o comando `dropna` (videoaula 2.2). Como vimos que nossa base de dados não tem dados faltantes (item 3.3 acima), esse é um comando opcional para esse problema. Mas você pode querer inseri-lo mesmo assim caso no futuro deseje treinar o mesmo modelo com outra base de dados que você não tem certeza se vai ter ou não dados faltantes. Vamos fazer isso antes de qualquer outro pré-processamento para não considerar dados removidos nas próximas fases.

4.2) Você pode ainda inserir o código para remover as instâncias duplicadas, caso existam. Da mesma forma, vamos fazer isso antes das próximas fases para não considerar dados removidos.

4.3) Como a nossa variável de saída é numérica, ela vai ter que processada também. Assim, vamos começar criando a nossa variável de saída que vai ser um dataframe chamado y, com uma única coluna, que é a variável G3.

4.4) Nós vamos começar criando um dataframe que vai conter todas as colunas do dataframe lido do arquivo csv que contenham dados que queremos explorar, ou seja, testar se eles levam a um melhor modelo. Vamos chamar esse novo dataframe de `X_possiveis`. Assim, quando criarmos o nosso dataframe X de variáveis de entrada, nós vamos criar a partir de `X_possiveis`, que já vai ter nossos dados processados. Veja novamente os resultados dos passos 2 e a partir deles escolha variáveis para considerar. Obs.: coloque em `X_possiveis` todas as variáveis que ache coerente considerar como

entrada para esse problema. Nos próximos passos, vamos escolher apenas alguma delas para começar, mas a ideia aqui é pegar todas que façam sentido. Você só não pode incluir como variável de entrada G2 e G3, porque a ideia é tentar prever a nota final do estudante no meio do ano letivo.

4.5) Agora vamos criar o nosso dataframe X com as variáveis de entrada que queremos treinar o modelo. Comece com algumas poucas variáveis. Se desejar, pode repetir as etapas a partir desse item (5.5) para testar com mais variáveis. Por enquanto, vamos fazer isso manualmente, mas nas próximas aulas vamos aprender métodos para automatizar esse processo, entre outros.

4.6) Agora você pode verificar se você tem variáveis categóricas e realizar o processo de *one-hot-encoding*, como explicado nas videoaulas 2.2 e 2.3, como mostrado a seguir, onde `coluna1` e `coluna2` são os nomes das colunas nas quais você quer normalizar. Forneça apenas os nomes das colunas que queira aplicar o one-hot-encoding.

```
X = pd.get_dummies(X, columns=['coluna1','coluna2'])
X.head()
```

4.7) A primeira coisa que você tem que fazer antes de escalar os dados é separar o conjunto de treinamento do conjunto de teste. Isso porque o treinamento não pode ser contaminado indiretamente pelo processamento que você vai realizar. Por exemplo, se você considerar todo o conjunto de dados para a normalização, você está usando informação dos dados de teste para o treinamento, o que pode fazer com que seu algoritmo tenha um desempenho melhor do que o real. Embora esse seja um tipo de vazamento de dados (do inglês *data leakage*) indireto e menos perigoso que o *data leakage* direto (quando usamos dados do teste no treinamento), ainda assim deve ser evitado e uma boa prática é separar os dados antes de escalá-los ou realizar qualquer tipo de pré-processamento que forneça informações. Você pode usar o comando `train_test_split` visto na videoaula 2.3.

4.8) Agora, você pode normalizar todos os dados numéricos usando o objeto `MinMaxScaler()` e seus métodos. Observe que embora a padronização em si vai ser realizada em todos os dados, você deve ajustar os dados considerando apenas os dados de treinamento (o ajuste pega o valor máximo e mínimo e faz os cálculos necessários). Para isso, você vai usar os comandos `fit()` e `transform()` separadamente. O `fit()` vai receber apenas os dados de treinamento e o `transform()` deve ser realizado nos dados de treinamento e teste separadamente[1]. Aqui, também observe que G1 e G2 possuem como maior valor 19 (essa foi a maior nota que um aluno tirou), mas que seu valor máximo real é 20. Você pode ajustar manualmente o valor máximo do `MinMaxScaler()`. Olhe a documentação da classe para descobrir como fazer:

```
scaler = MinMaxScaler()
scaler.fit(X_train) #o ajuste considera apenas o X de treinamento
X_train=scaler.transform(X_train) #escalando os dados de treinamento
X_test=scaler.transform(X_test) #escalando os dados de teste apenas
```

- OBS1: Você pode padronizar todos os dados (não é necessário selecionar colunas), porque a essa altura todos os nossos dados já são numéricos.

- [1] Para saber mais de porque fazer ajuste e transformação das escalas separadamente nos conjuntos de treinamento e teste, sugiro: https://machinelearningmastery.com/data-preparation-without-data-leakage/

- OBS2: Caso você, precise padronizar os dados de saída (y), como o ajuste deve ser realizado para todas as variáveis, sugiro concatenar o dataframe y ao dataframe X e depois escalar.

5) Treinando o modelo

5.1) Agora você está pronto para treinar o seu modelo usando os algoritmos vistos em aula: Regressão Linear e Árvore de Decisão para Regressão.

5.2) Teste ambos os algoritmos fazendo a escala os dados e não fazendo. O que acontece?

- Observe que a Regressão Linear precise que os dados sejam padronizados, mas não a árvore de decisão. Mesmo assim, teste ambos os algoritmos nos dois cenários (com dados normalizados e com dados não normalizados) para verificar o que acontece.

6) Avaliando o seu modelo

Agora você pode avaliar o seu modelo. Não esqueça de treinar os modelos e avaliar para os diferentes cenários de pré-processamento.

- Insira os resultados encontrados com uma descrição em texto no seu código.
- Obs: O tipo de avaliação que estamos fazendo ainda não é a ideal. Mais adiante, aprenderemos a trabalhar com cross-validação e falaremos também de conjunto hold-out e outras técnicas para melhorar nossas avaliações de modelos criados.