Aula 3 - Descrição da atividade

Suponha que você foi contratado para ajudar a escola 'X' a melhorar a taxa de aprovação em uma disciplina específica, que tem apresentado resultados abaixo do esperado nos últimos anos. A escola 'X' 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 disciplinas 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, se um aluno terá sucesso ou não na disciplina em questão. Para isso, você vai utilizar uma técnica de classificação binária.

Vamos utilizar a base de dados "*student-performance*" disponível no repositório *UCI Machine Learning*. Essa base de dados contém informações sobre alunos 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.

1) Entendendo o problema

A partir do enunciado você sabe que precisa prever se o aluno vai ter sucesso na disciplina de matemática ou português a partir de notas parciais e de outros dados.

- Você sabe que é um problema de classificação binária. Você precisa classificar o aluno como sucesso (geralmente representado por "1") ou insucesso (geralmente representado por "0 (zero)").
- Observe que não temos uma variável categórica de sucesso e que você vai ter que criá-la a partir da variável G3. Considere que o aluno teve sucesso se a média em G3 for maior ou igual a 12 (as notas em Portugal vão de 0 a 20).

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 agora fazer algo que não vimos ainda nessa disciplina: 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 `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.

3.2) Você pode também usar `print(df.dtypes)` para saber os tipos dos dados e `print(df.shape)` para ver o formato do dataframe (número de instâncias X número de variáveis). 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.

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:

```
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 pela linha `count`. Todas as colunas têm 395 células, assim não temos dados faltantes.

3.4) Você pode ainda verificar a distribuição das classes. Em problemas de classificação, conjuntos de dados desbalanceados (mais dados de uma classe de saída do que de outras) podem precisar de cuidados especiais. Assim, é importante verificar o balanceamento das classes. Você pode verificar isso com o comando abaixo, onde `'class'` deve ser o nome da variável (coluna) output. Mas cuidado que em alguns problemas, como desse exercício, a classe output precisa ser criada primeiramente.

```
class_counts = df.groupby('class').size()
print(class_counts)
```

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.

4.2) Você pode ainda inserir o código para remover as instâncias duplicadas, caso existam.

4.3) 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 conjunto de treinamento não pode ser contaminado indiretamente pelo processamento que você vai realizar (sim, a gente não tomou esse cuidado na videoaula 2.3! Mas a ideia é ir vendo os conteúdos aos poucos). 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 qualquer pré-processamento. Você pode usar o comando `train_test_split` visto na videoaula 2.3.

4.4) Separe em dois dataframes diferentes as variáveis de entrada e saída do algoritmo de treinamento. Observe que não temos uma variável categórica de saída sucesso e que você vai ter que criá-la a partir da variável G3. Considere que o aluno teve sucesso se a média for maior ou igual a 12.

4.5) Agora você pode padronizar os dados de treinamento de variáveis com distribuição normal. A padronização consiste em escalar os dados de tal forma que a média seja 0 e o desvio padrão 1. Uma maneira de verificar se os dados possuem distribuição normal é visualizar o histograma e/ou rodar o teste Shapiro (ver itens 4.1 e 4.2). 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.

- Para saber mais sobre sse tema, sugiro: https://machinelearningmastery.com/data-preparation-without-data-leakage/

4.6) Mesmo os dados que foram padronizados, podem ser normalizados. Isso é interessante porque embora a padronização ajuste os dados para uma faixa menor com média 0 e desvio padrão 1, ela não vai colocar na mesma escala das outras variáveis normalizadas (entre 0 e 1, ou seja com valo rmínimo de 0 e valor máximo de 1). Assim, algo que você pode fazer agora é normalizar todos os dados numéricos usando o objeto `MinMaxScaler()` e seus métodos.

```
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

4.7) Segundo vários autores nem sempre há uma regra que define quando se deve normalizar ou padronizar os dados. Assim, talvez você queira criar o código necessário para testar as várias possibilidades: (1) sem re-escalar os dados, (2) apenas padronizando, e (3) apenas normalizando. A maneira mais simples é criar uma célula diferente para cada cenário e comentar os cenários que não quer testar. Mas fique livre para pensar em outras formas.

4.8) Não esqueça de verificar se você tem variáveis categóricas e realizar o processo de *one-hot-encoding*.

5) Treinando o modelo

Agora você está pronto para treinar o seu modelo.

6) Avaliando o seu modelo

Agora você pode avaliar o seu modelo. Para avaliar você tem que fazer predições com o conjunto de teste e usar as métricas para comparar o valor predito com o valor real.

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. Na aula 9, aprenderemos a trabalhar com cross-validação e falaremos também de conjunto hold-outo e outras técnicas para melhorar nossas avaliações de modelos criados.