UDACITY

PROJETO

Building a Student Intervention System

Uma parte do Machine Learning Engineer Nanodegree Program

REVISÃO DO PROJETO

REVISÃO DE CÓDIGO

COMENTÁRIOS

COMPARTILHE SUA REALIZAÇÃO!

Requires Changes

3 ESPECIFICAÇÕES NECESSITAM DE MUDANÇAS

Parabéns por uma ótima primeira versão deste projeto! Algumas alterações são necessárias, mas espero que os comentários e sugestões abaixo ajudem você a atender a todos os requisitos do projeto da próxima vez. Bom trabalho e mantenha o ritmo!

Classificação versus regressão

O aluno identifica corretamente o tipo de estimativa que o problema requer e a justifica de maneira razoável.

Excelente

Bom trabalho! De fato, o importante para definir se um problema de aprendizado supervisionado é uma classificação ou uma regressão é a **variável-alvo**: um alvo discreto significa que estamos lidando com um problema de classificação, enquanto o problema é de regressão se o alvo for contínuo.

Observando os dados

A resposta do aluno aborda as características mais importantes do conjunto de dados e usa essas características para informar o processo de tomada de decisão. Características importantes incluem:

- **Número de pontos**
- **Número de atributos**
- **Número de alunos aprovados**
- **Número de alunos reprovados**
- **Taxa de graduação**

Alteração necessária

Uma das colunas em `student_data` é a variável-alvo, e não um atributo. Considerando isso, corrija seu código para que `n_features` represente corretamente o número de atributos no conjunto de dados.

Preparando os dados

O código foi executado com a saída apropriada e sem erros.

Sugestão

Antes de salvar a última versão do seu *notebook*, é uma boa ideia reinicializar o *kernel* e rodar todas as células de código. Com isso você pode se assegurar de que o código vai rodar corretamente do início ao fim, sem depender de variáveis ou cálculos realizados anteriormente no *notebook*.

Conjuntos de treinamento e teste foram criados por meio de amostragem aleatória do conjunto completo.

Sugestão

Na função `train_test_split()`, você também pode definir `num_test` (ou `num_train`) como um número inteiro, e nesse caso a função interpretará o parâmetro como o número absoluto de dados de teste (ou treino). No nosso caso, por exemplo, definir `num_train=300` garante que 300 dados serão usados para treino e o restante (95) para teste.

Treinando e avaliando modelos

Três modelos supervisionados são escolhidos e razoavelmente justificados. Prós e contras de cada modelo são fornecidos, além de uma discussão de aplicações gerais de cada.

Excelente

Você foi além do necessário aqui, descrevendo e utilizando cinco modelos em vez de três. Parabéns!

Sugestão

A documentação do scikit-learn inclui um fluxograma feioso para ajudar o usuário a escolher o melhor algoritmo para um problema. De acordo com esse fluxograma, quais seriam os algoritmos recomendados para o nosso problema?

Comentários

Como máquinas de vetores de suporte calculam a **distância** entre observações, é importante normalizar) os atributos, de forma que todos eles variem ao longo da mesma escala. Você vai aprender mais sobre esses procedimentos na lição para o projeto 3. :)

De fato, como você indica em sua resposta, árvores de decisão podem facilmente sobreajustar os dados de treino. A boa notícia, porém, é que é relativamente fácil controlar essa tendência, definindo limitações como a profundidade da árvore (vimos isso no projeto anterior do curso).

Uma nota técnica: a classe GaussianNB do scikit-learn foi desenvolvido para lidar com atributos contínuos, enquanto a MultinomialNB lida com dados categóricos. Quando há atributos dos dois tipos (como é o caso aqui), pode-se transformá-los todos em atributos categóricos (separando-se os atributos variáveis por percentis, por exemplo) ou treinar modelos separados para juntá-los em seguida. Este post do stackoverflow - escrito por Olivier Grisel, um dos desenvolvedores do scikit-learn - apresenta mais detalhes.

Ainda que a premissa de independência entre variáveis não seja verdadeira, o modelo de Naïve Bayes ainda pode alcançar resultados satisfatórios em alguns casos. Neste vídeo, Charles e Michael falam sobre isso (embora mesmo eles parecem não ter muita certeza de como isso dá certo!).

Os tempos e pontuações F1 de cada modelo com cada tamanho de conjunto de treinamento são preenchidos na tabela dada. A métrica de desempenho é razoável em comparação a outros modelos medidos.

Excelente

Ótimo uso de loops para simplificar o código necessário para testar seus classificadores.

Escolhendo o melhor modelo

É justificado qual modelo parece ser o melhor, comparando o custo computacional e a precisão de cada método.

Alteração necessária

Na **Questão 3**, peço para você também considerar o tempo gasto por cada modelo para treinamento e previsão. Sua decisão não precisa mudar! Porém, você precisa considerar também esse fator na hora de tomá-la.

Comentário

Escolher um modelo com alta variância com base nesses resultados iniciais pode não ser uma má ideia, uma vez que o modelo será posteriormente calibrado para reduzir esse problema.

O aluno é capaz de descrever de forma clara e concisa como o modelo ótimo funciona, em termos leigos para uma pessoa que não tem experiência técnica nem familiaridade com machine learning.

Alteração necessária

Excelente descrição de máquinas de vetores de suporte! Peço apenas que você adicione uma menção ao *kernel trick*. Como um modelo SVM pode lidar com dados que não são linearmente separáveis considerando sua dimensionalidade original?

O modelo escolhido é calibrado corretamente, usando busca em matriz de pelo menos um parâmetro com no mínimo três configurações diferentes. Se o modelo não precisa de nenhuma calibração, isso é dito explicitamente e explicado de forma satisfatória.

A pontuação F1 é calculada para o modelo calibrado e tem desempenho igual ou melhor ao do modelo escolhido com as configurações padrão.

Sugestão

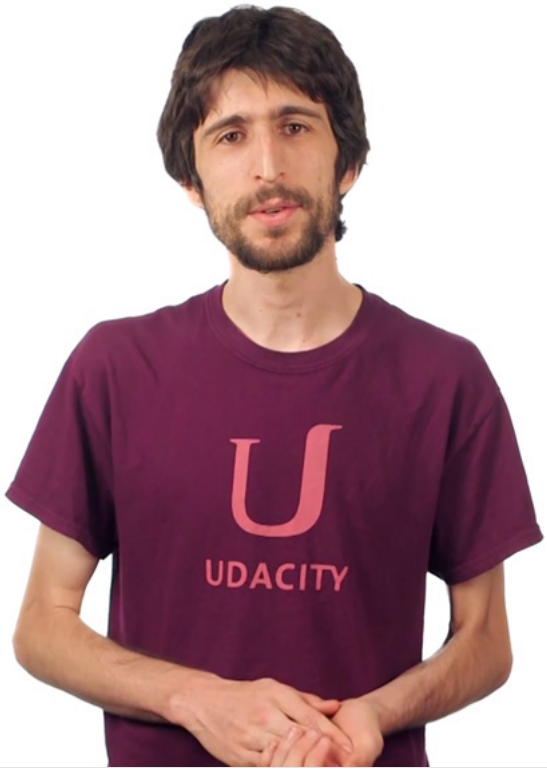
- Além de não termos uma grande quantidade de dados, as classes da variável-alvo são desbalanceadas - ou seja, uma delas é muito mais comum no conjunto de dados do que a outra. Isso pode causar problemas para a validação cruzada, pois a proporção entre classes pode variar bastante para cada conjunto de de validação.
- Uma solução para esse problema é **estratificar** a divisão de dados, para se assegurar de que cada subconjunto tem aproximadamente a mesma proporção entre classes. Caso queira fazer isso, você pode dar uma olhada na classe StratifiedShuffleSplit do scikit-learn.

Qualidade do código

O código reflete a descrição na documentação.

REENVIAR

BAIXAR PROJETO



Melhores práticas para sua resubmissão do projeto

Ben compartilha 5 dicas úteis para a revisão ressubmissão do seu projeto.

Assistir Vídeo (3:01)

RETORNAR

Avalie esta revisão

FAQ do Estudante