UDACITY

Back to Machine Learning Engineer Nanodegree

Building a Student Intervention System

REVISÃO

REVISÃO DE CÓDIGO

HISTORY

Meets Specifications

Corrigi mais de 100 projetos desse tipo e o seu foi claramente o melhor até agora! Parabéns, desempenho louvável. Continue sempre assim! 👏🏽

Classificação versus regressão

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

Correto!

É um problema de classificação, mais especificamente, uma classificação binária, uma vez que vamos classificar alunos em duas classes: "precisa de uma intervenção" e "não precisa de intervenção", o que é feito treinando-se o algoritmo com um target binário, que representa duas classes. Então vamos utilizar dados de entrada, características dos alunos, e vamos retornar uma etiqueta discreta e que nos dirá se o aluno precisa ou não de intervenção.

No caso de uma regressão, o nosso target seria contínuo, como seria em um problema de previsão do preço de um imóvel.

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**

Preparando os dados

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

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

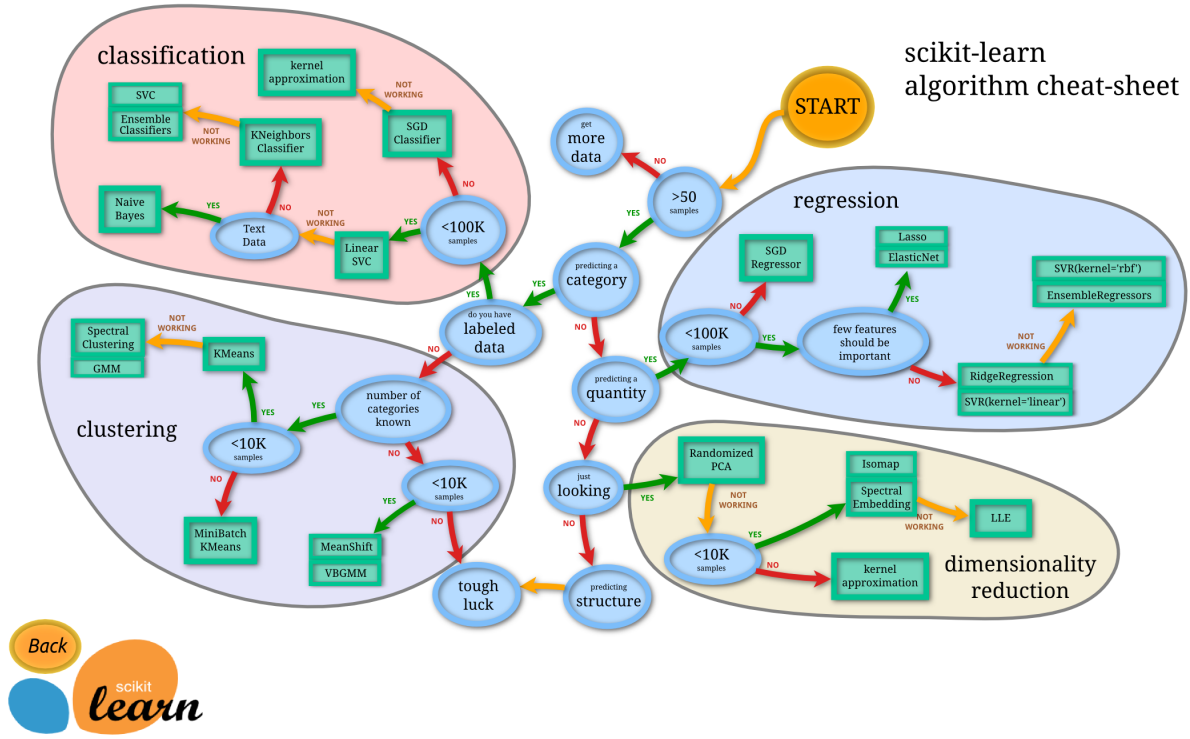
Por que o argumento `random_state` é importante? Esse argumento provê uma seed para o split e faz com que ele seja o mesmo sempre que o mesmo random_state é usado. Para que serve isso? Para reprodutibilidade, ou seja, para quando quisermos rodar novamente e obter os mesmos resultados de antes, assim conseguimos iterar nosso modelo e aplicar melhorias e saber que estamos melhorando e não que os resultados mudaram devido a aleatoriedade da divisão dos dados.

Repare também que o argumento train_size aceita tanto um número inteiro (quantidade exata de pontos que queremos para o conjunto de treino), como uma fração, por exemplo, 0.7, o que significaria deixar 70% dos dados para treino.

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.

Cada conjunto de dados configura um problema diferente, mas existem algumas regrinhas que ajudam a escolher ao menos as primeiras opções para resolver um problema. O scikit-learn oferece um fluxograma para isso:



Para a referência completa, clique aqui

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.

Preencheu a tabela e os resultados para os modelos escolhidos estão dentro do esperado.

Se você se interessa pela questão de desempenho, na página do scikit-learn há um guia sobre otimização e profiling.

Lembre-se que alguns modelos do pacote oferecem multithread, como a `RandomForest`, ou mesmo algumas classes, como `GridSearchCV`. Sempre que houver o argumento `n_jobs` é possível definí-lo como `-1` e ele utilizará todos os cores da máquina em paralelo.

Escolhendo o melhor modelo

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

Justificou a escolha do algoritmo usado as características levantadas, como tempo de treino, teste e a pontuação F1.

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.

Muito bem, sempre fazendo questão de explicar os termos técnicos e fez muito bem em usar uma imagem para ilustrar o conceito. 👏

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.

Aqui usamos a F1 para comparar modelos de classificação, mas existem várias métricas para esse tipo de problema. Sugiro que você dê uma olhada em todas as métricas disponíveis no scikit-learn para esse tipo de problema e tente utilizar outra aqui no GridSearchCV como experimento.

Você pode ver todas aqui.

Qualidade do código

O código reflete a descrição na documentação.

BAIXAR PROJETO

RETORNAR

Avalie esta revisão