Perguntas

1. Qual comando principal para iniciar o treinamento de um modelo?

Existem duas alternativas: utilizar o `fit` que vai realizar o treinamento considerando as features X e o objetivo y. Contudo, o `scikit-learn` fornece uma alternativa chamada `fit_transform` disponível para alguns modelos que, além de realizar o treinamento, aplica as transformações aos dados em um único passo.

1. Qual comando utilizado para classificação das amostras?

Para classificação devemos utilizar a função `predict` onde podemos passar as features da(s) amostra(s) e termos a lista de classificação.

1. Para que servem os pipelines?

As pipelines são estruturas que nos permites "aninhar" objetos que serão chamados em sequência. No contexto de pré-processamento dos dados e treinamento, ao invés de criar 2 objetos separados e chamar um a um, podemos criar um pipeline que irá chamar o objeto de pré-processamento e resultado será passado para o modelo para treinamento.

Em resumo o pipeline serve para encadear tarefas umas as outras, onde a saída de uma tarefa é a entrada de outra.

1. Como podemos usar uma árvore de decisão para realizar uma predição numérica?

Utilizar uma árvore de decisão para regressão numérica é semelhante a tarefa de classificação. Com a diferença de que, para a regressão utilizamos a classe `DecisionTreeRegressor` e as targets y deve ser números a serem predizidos ao invés de logits das classes.

1. O que são as fronteiras de decisão?

As fronteiras de decisão são estruturas que contém as combinações de limite para as regras aprendidas dos dados de

treinamento. Ou seja, as fronteiras de decisão definem as possíveis escolham de uma árvore de decisão.

1. Qual comando utilizado para particionamento de dados entre treino e teste? Quais seus principais argumentos?

O `scikit-learn` disponibilidade uma função chamada `train_test_split`. Essa função recebe o dataset completo para uma determinada tarefa e separa o dataset entre dois novos conjuntos: dados de treinamento e dados de teste. Os dados de treinamento são utilizados para realizar o treinamento do modelo e os dados de teste são utilizados para testar o modelo, isto é, validar o modelo em dados diferentes dos dados de treinamento.

Alguns dos principais argumentos são:

- `train_size`: a proporção do dataset original que deve ser utilizado para treinamento
- `test_size`: a propoção do dataset que deve ser utilizado para teste
- `stratify`: esse parâmetro indica que o dataset deve ser dividido de forma que a mesma proporção de classes estejam presentes em cada dataset.

1. Considerando um classificador do tipo vizinhos mais próximos, como especificar quantos vizinhos devem ser considerados no processo de classificação?

A classe `KNeighborsClassifier`, no momento de sua inicialização, recebe o parâmetro `n_neighbors` que define a quantidade de vizinhos que devem ser considerados na classificação.

1. Considerando um classificador do tipo vizinhos mais próximos, qual o impacto em se utilizar poderação tipo uniforme ou baseada em distância?

Com base na ponderação utilizada, podemos configurar os pesos dos vizinhos com base distância. Ao utilizar a ponderação **uniforme** todos os vizinhos têm o mesmo impacto na decisão, enquanto a ponderação baseada em distância, ou seja, vizinhos

mais próximos têm maior influência do que os vizinhos mais distantes.

1. Considerando o classificar do tipo Naive Bayes, como especificar as probabilidades a priori das classes diretamente, ou seja, sem computar a partir dos dados?

Durante a inicialização das classes de classificação **Naive Bayes** existem parâmetros que podemos definir para configurar a probabilidade a priori das amostras.

- `GaussianNB` dispõe do parâmetro `priors` que define a probabilidade inicial das amostras
- As classes `MultinomialNB` e `ComplementNB` fornecem o parâmetro `class_prior`

1. Considerando os datasets “moons”, “circles” e “linearly separable”. Qual classificador apresenta a melhor eficácia para cada um e qual apresenta a pior eficácia? Justifique.

- `Moons`: para esse dataset, o melhor classificador foi o `Nearest Neighbors` que obteve um **score** de 0.97 e o pior foi o `QDA` que obteve **score** de 0.85. Além do `Nearest Neighbors` outros classificadores obtiveram score igual ou próximo, contudo, esse classificador foi capaz de aproximar bem aos dados sem overfitting, enquanto outros classificadores como o `RBF SVM` ficou extremamente ligado aos dados, isto é, com alta variância e portanto, não generaliza para novos dados.
- `Circles`: para esse dataset, o `Nearest Neighbors` também foi o melhor classificador em termos de **score**. Enquanto o pior modelo foi o `Linear SVM`.
- `Linear Separable`: Esse dataset é o mais complexo de decidir o melhor modelo visto que 7 alternativas obtiveram **score** igual a 0.95 e os piores modelos obtiveram 0.93 de **score**. Não existe um modelo ideal claro, outros testes de validação precisariam ser realizados. Mas, dado que o dataset contém dados linearmente separáveis, muitos modelos são boas escolhas para esse problema.

1. Considerando classificadores do tipo árvore de decisão, o que é chamado de “caminho de decisão”?

O **Caminho de Decisão** são os nós que os determinada amostra passou até a classificação final. Ele nos permite visualizar as regras validadas e o caminho escolhido por uma determinada amostra.

Com isso, podemos visualizar claramente as decisões tomadas por um caso de teste.