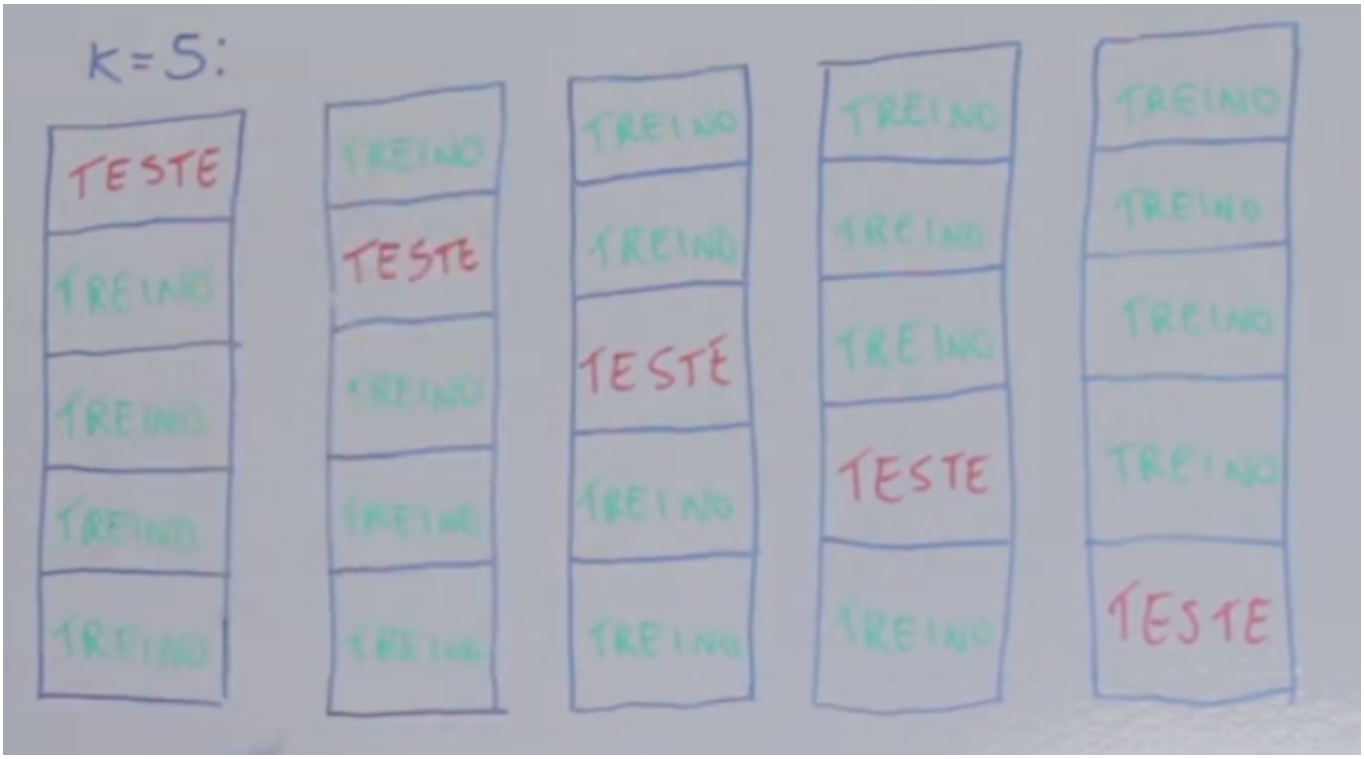
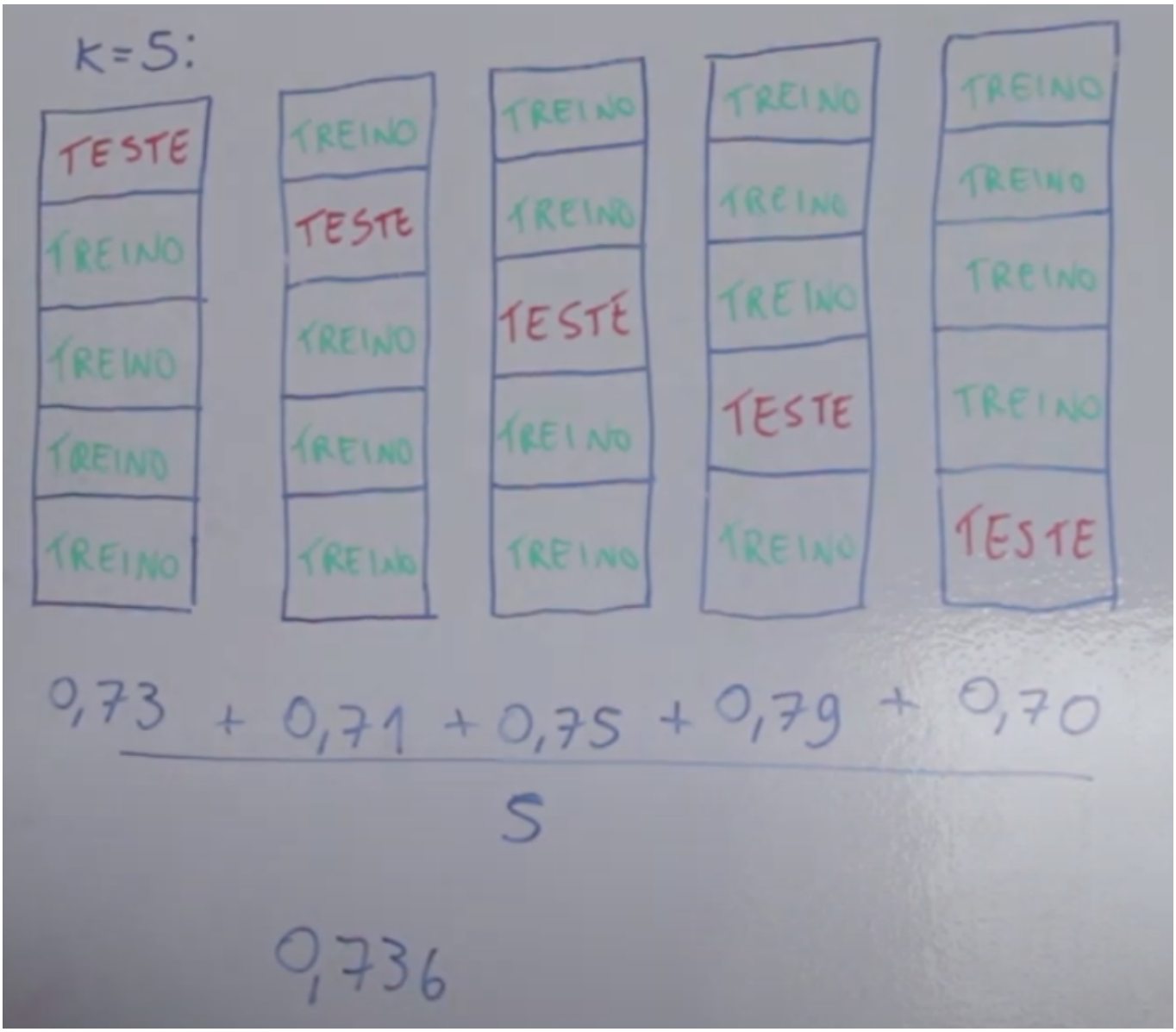
Validação cruzada Kfold

Nessa técnica a base de dados é subdividida em um tamanho "k", por exemplo 5, onde teremos 25% da base em cada subdivisão. A partir disso podemos selecionar uma dessas subdivisões para teste e o restante para treino, gerando assim 5 treinos com amostras de treino e teste distintas.



Após treinar cada bloco de treino, com uma amostra distinta para teste, podemos utilizar uma métrica como resultado de cada bloco de treino como por exemplo o R2, e a partir disso somar todos os resultados e gerar uma média como demonstrado abaixo:



A vantagem disso é que, podemos utilizar um pedaço da base sempre como teste, e ao tirar uma média dos resultados, temos uma métrica mais generalizada de como o modelo está se comportando, ou seja, ao invés de utilizar aleatoriamente um percentual como teste e outro como treino como o train_test_split nativamente faz, obteremos um resultado final mais fidedigno de como o modelo performa.

A desvantagem está no custo computacional, pois faremos diversas baterias de teste dependendo de "k", uma vez que quanto maior esse número, mais custoso e mais treinos serão feitos. Como por exemplo usando k = 5, o custo computacional é 5 vezes maior do que utilizar apenas uma base de treino e teste.

Recomendado utilizar K entre 5 e 10

OBS:

Um detalhe importante que esqueci de comentar nesse vídeo é que a função kfold vem com o parâmetro shuffle=False como default, então é importante estar atento a isso (a melhor prática é informar shuffle=True). Quando esse parâmetro está marcado como True, o algoritmo irá primeiro embaralhar randomicamente as amostras antes de iniciar os cálculos. Isso é importante pois se as amostras não estão randomicamente distribuídas, a divisão dos dados entre treino e teste pode ficar desbalanceada. A função train_test_split já faz isso por default (o parâmetro shuffle vem marcado como True por default), enquanto nas funções kfold, o shuffle vem marcado como False por default.