11

Implementando mais de um algoritmo no código

Dentro do arquivo `classifica_buscas.py`, além de importar apenas o `MultinomialNB`, importe o `AdaBoostClassifier` por meio da biblioteca `sklearn.ensemble`. Então crie um modelo para cada um dos algoritmos, por exemplo, `modeloMultinomail` e `modeloAdaBoost`. Agora reutilize todo o código que treina e testa os algoritmos de classificação, ou seja, crie a função `fit_and_predict`.

Dentro da função `fit_and_predict` cole todo o código que realiza o treino e teste dentro da função, segue abaixo o código:

```
modelo.fit(treino_dados, treino_marcacoes)

resultado = modelo.predict(teste_dados)

acertos = resultado == teste_marcacoes

total_de_acertos = sum(acertos)
total_de_elementos = len(teste_dados)

taxa_de_acerto = 100.0 * total_de_acertos / total_de_elementos
print("Taxa de acerto do algoritmo: %f" % taxa_de_acerto)
```

Perceba que as variáveis `treino_dados`, `treino_marcacoes`, `teste_dados`, `teste_marcacoes` não fazem parte do escopo dessa função, em outras palavras, elas precisam ser enviadas por parâmetro da função `fit_and_predict`. Agora que iremos rodar mais de um algoritmo ao mesmo tempo, precisamos identificar cada algoritmo e seu resultado, porém, perceba que a mensagem atual é muito genérica, ou seja, precisaremos modificá-la, altere a mensagem para:

```
msg = "Taxa de acerto do algoritmo {0}: {1}".format(nome, taxa_de_acerto)
print(msg)
```

Porém perceba que adicionamos a variável `nome`, portanto, precisamos recebê-la também por parâmetro na função `fit_and_predict`. Por fim, faça a chamada do `fit_and_predict` para cada um dos modelos criados e execute o arquivo tanto para o `buscas.csv` quanto para o `buscas2.csv`. Por fim, cole o código e o resultado de ambos os arquivos de dados:

Responda

INSERIR CÓDIGO		FORMATAÇÃO