UFBA - Universidade Federal da Bahia
Disciplina: IC0007 - Tópicos em Banco de Dados
Alunos: Pablo Henrique Rego dos Santos Cabral, Rodrigo Almeida Bezerra
Data: 10/07/2023

Classificador de Galáxias

1. Dados

Os dados utilizados neste trabalho provêm do cruzamento de informações do Sloan Digital Sky Survey (SDSS) e do Galaxy Zoo. O SDSS é uma colaboração internacional de cientistas que coletam dados de dois telescópios nas Américas do Norte e do Sul para construir imagens tridimensionais detalhadas do universo. O SDSS produziu imagens multicoloridas profundas de um terço do céu e criou espectros de mais de três milhões de objetos astronômicos. O Galaxy Zoo é um projeto online de ciência cidadã no qual voluntários podem ajudar a classificar imagens de galáxias.

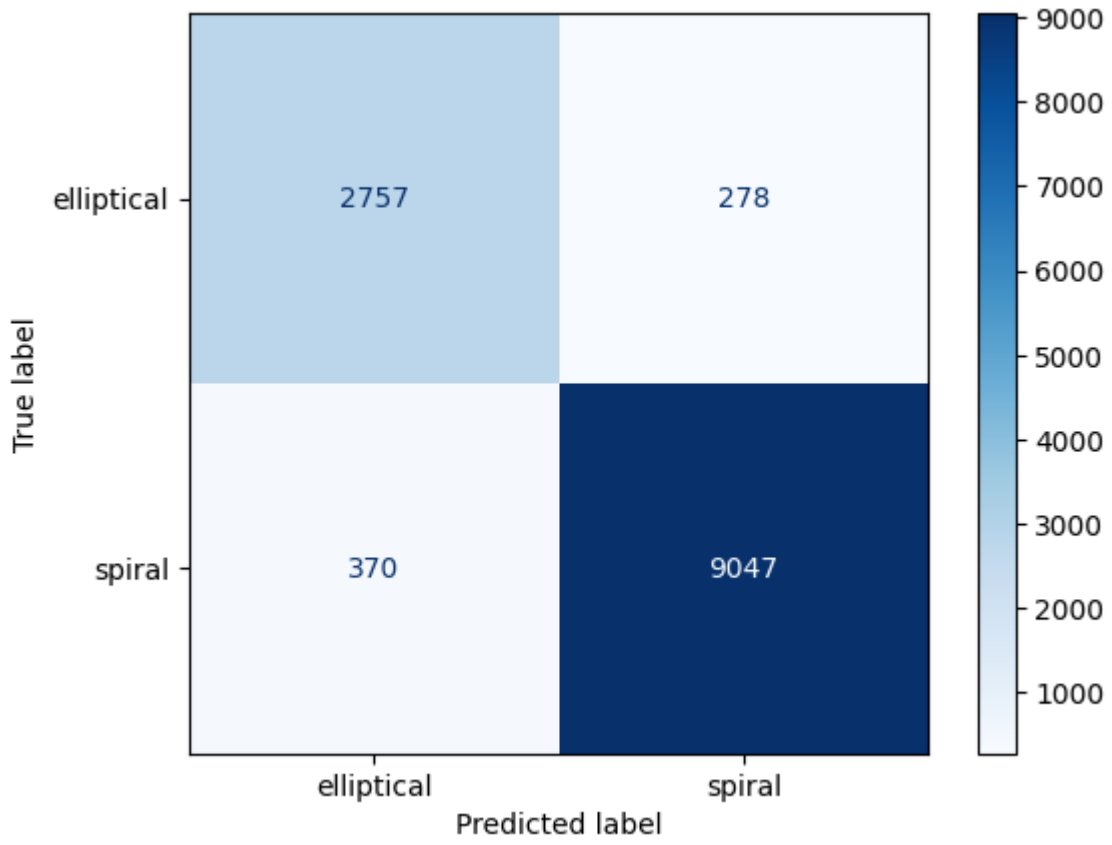
O Galaxy Zoo classificou mais de 1 milhão de galáxias ao longo de suas edições. Para esse trabalho especificamente, utilizamos os dados da tabela 2 da edição 1 do Galaxy Zoo, que fornece a classificação de aproximadamente 251 mil imagens que possuem dados espectroscópicos no *data release* 7 do SDSS. Dessa tabela é possível extrair a informação imparcial de se o objeto observado se trata de uma galáxia elíptica ou espiral. O acesso aos dados foi feito utilizando o astroquery, um pacote de funções e classes da linguagem Python para realizar consultas em bases de dados astronômicas.

2. Modelo

Para realizar a tarefa de classificação, selecionamos alguns dos atributos de saída dos pipelines de fotometria do SDSS que descrevem características morfológicas das galáxias, como as cores astronômicas, o formato (tamanho, orientação), excentricidade e perfil de luminosidade.

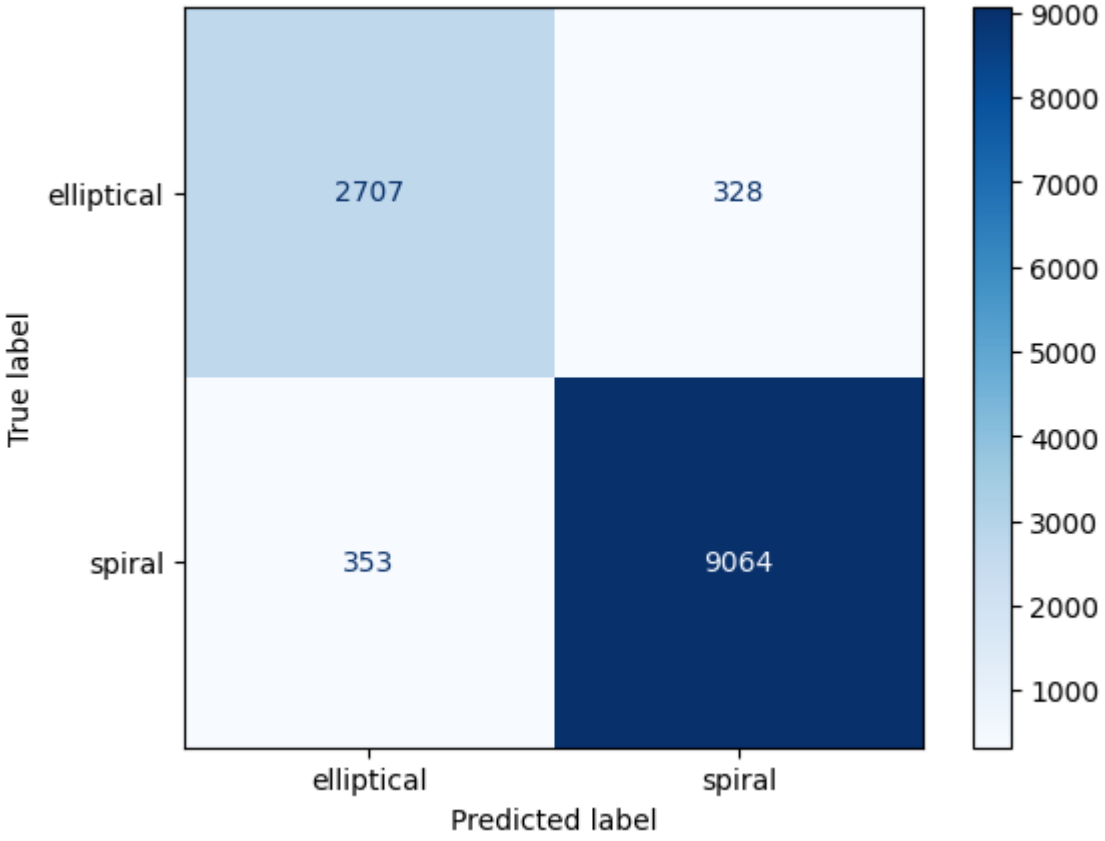
Removemos qualquer instância que, para qualquer um dos atributos, possua algum valor que indique erro de medição ou a ausência do dado. Dividimos o dataset em subconjuntos de treinamento, validação e teste, cada um correspondendo respectivamente a 80%, 10% e 10% do número de instâncias do conjunto original. Realizamos experimentos com os algoritmos KNN (K-Nearest Neighbors), Decision Tree e Random Forest. Utilizamos o RandomizedSearchCV para efetuar uma série de experimentos com o dataset de validação medindo o desempenho desses algoritmos com diferentes hiperparâmetros escolhidos aleatoriamente. Durante esses experimentos foi empregada a técnica 10-fold de validação cruzada para avaliar a capacidade de generalização do modelo. Depois de encontrar os melhores valores para os hiperparâmetros, utilizamos o dataset de treinamento para treinar os modelos. Abaixo estão as matrizes de confusão e a avaliação de desempenho dos modelos. Como as classes estão desbalanceadas, podemos considerar métricas como precisão, revocação ou medida F1 ao invés da acurácia. Como os três modelos obtiveram um desempenho muito próximo, escolhemos o modelo do algoritmo de Decision Tree. Os experimentos foram realizados no Google Colab.

2.1. K-Nearest Neighbors



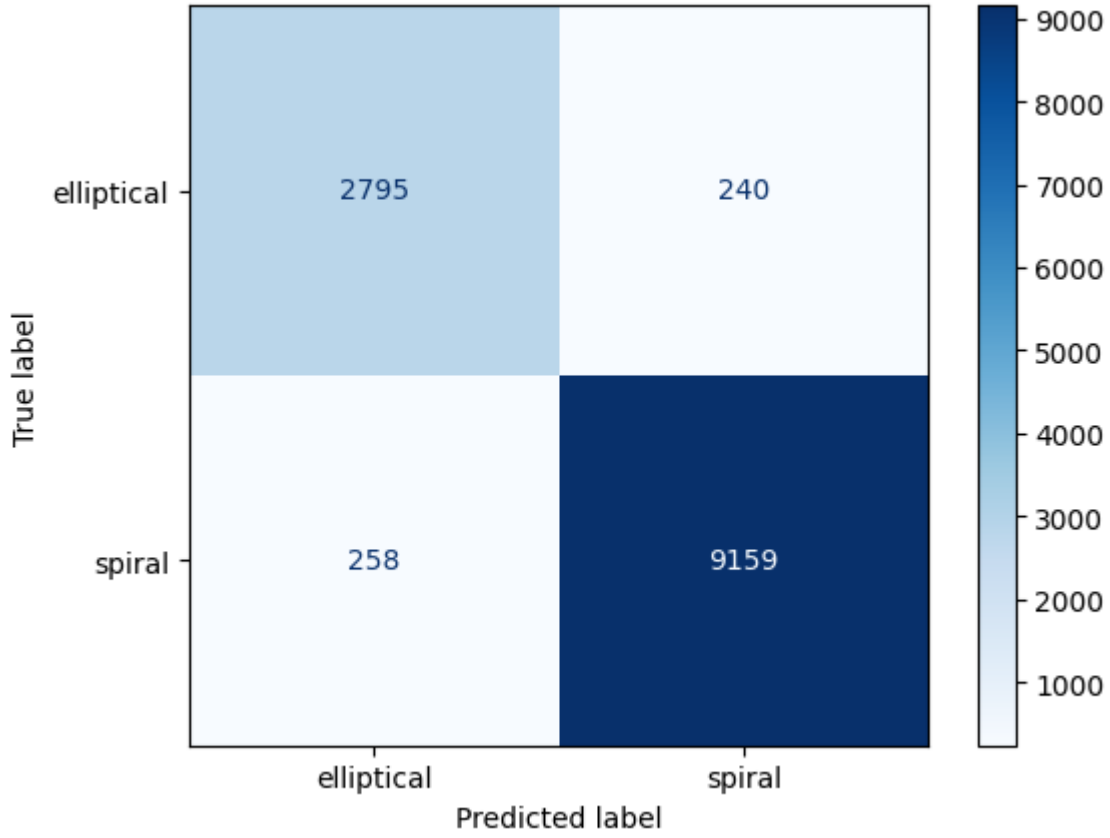
	precision	recall	f1-score
elliptical	0.88	0.91	0.89
spiral	0.97	0.96	0.97
accuracy			0.95
macro avg	0.93	0.93	0.93
weighted avg	0.95	0.95	0.95

2.2. Decision Tree



	precision	recall	f1-score
elliptical	0.88	0.89	0.89
spiral	0.97	0.96	0.96
accuracy			0.95
macro avg	0.92	0.93	0.93
weighted avg	0.95	0.95	0.95

2.3. Random Forest



	precision	recall	f1-score
elliptical	0.92	0.92	0.92
spiral	0.97	0.97	0.97
accuracy			0.96
macro avg	0.94	0.95	0.95
weighted avg	0.96	0.96	0.96

3. Produtização

A API foi feita na linguagem Python, utilizando FastAPI e os mesmos requisitos do modelo com o objetivo de fornecer um endpoint para operação de */POST*. FastAPI é um framework livre para construção de APIs a partir do Python 3.7, com este framework conseguimos realizar a captura dos dados, carregar o modelo treinado e realizar a classificação. Todos os requisitos utilizados nesta API estão no repositório do projeto, o requirements.txt. Ela funciona em conjunto com o uvicorn, um servidor ASGI utilizado para execução da API.

A API é acessível pelo endereço *localhost:8000/classify* e a entrada de dados é em formato JSON, Arquivo - modelo de requisição. Após a construção e teste da API iniciamos a construção do Docker utilizando o mesmo arquivo requirements.txt. Foi salvo todos os arquivos na mesma pasta como forma de facilitar a construção da imagem, rodamos o comando *docker build classificador-galaxias-docker* pra gerar a imagem. Assim realizamos alguns testes o que gerou o arquivo de modelo-requisições.txt que está disponível no GitHub após isso salvamos nossa imagem com o comando *docker save classificador-galaxias-docker > classificador-galaxias-docker.tar* e como forma de teste realizamos o *load* da imagem executando o comando *docker load -i classificador-galaxias-docker.tar.* Com a imagem carregada fizemos mais testes e obtivemos sucesso na produtização do nosso modelo.

No GitHub do projeto existe um readme detalhando como executar, um passo a passo e a imagem está disponível neste link foi necessário realizar o upload no drive devido ao tamanho da imagem de 2,1 GB consideramos que poderia ficar pesado o download e futuras modificações no modelo assim como o retreino e monitoramento.

4. Retreino e Monitoramento

Inicialmente consideramos realizar o retreino do nosso modelo de forma automática e a partir de um determinado número de novas requisições, por exemplo supondo que tivesse em nosso banco de dados 1000 novos dados de galáxias para classificação o próprio código da api iria executar o modelo novamente e retornar ao início. Porém, por questões de ética, segurança e até mesmo alguns tópicos do LGPD sobre captura e armazenamento de dados do usuário isso foi descartado.

Apesar desta parte não ter sido implementada como trabalhos futuros a ideia seria utilizar F1-score, Matriz de confusão, ROC/AUC e recall para monitoramento, isto seria realizado na mesma granularidade que o modelo é executado. Assim seria feito de maneira a tornar o acompanhamento disso simples, através da geração de um relatório exibindo as métricas que foram escolhidas e definir um range para que quando essas métricas saírem deste limite seja emitido um alerta.

Com base neste relatório gerado iríamos de forma manual, realizar o retreino do modelo com novos dados e assim atualizar nosso modelo. A frequência levaria em consideração dois fatores: quando o range das métricas forem ultrapassados ou quando se passar mais de um ano sem retreino e sem emitir alerta. Este tempo de retreino é devido a própria SDSS não ter uma periodicidade de dados novos em menos de um ano e não ter tantas modificações no conjunto de dados disponibilizados por eles.