UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO
CURSO DE CIÊNCIAS DA COMPUTAÇÃO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA

Aluno:

Giovane Batista de Lima e Cirne

Tarefa de Machine Learn

Inteligência Artificial

Professora: Rosa Costa

Rio de Janeiro

Descrição do problema

O objetivo central deste projeto é classificar dígitos manuscritos (de 0 a 9) a partir de imagens digitalizadas. O problema é formalmente definido como uma tarefa de classificação multi-classe supervisionada dentro do campo de Aprendizado de Máquina (Machine Learning).

As entradas para os modelos são vetores de características (features) de imagens de dígitos, onde cada imagem 8×8 pixels é transformada em um vetor de 64 dimensões, com cada valor representando a intensidade de um pixel.

As saídas esperadas são os dígitos corretos (0, 1, 2, ..., 9).

O objetivo é treinar modelos para que, dada uma nova imagem de dígito manuscrito, eles possam prever corretamente a qual classe (qual dígito) a imagem pertence.

Justificativa da Ferramenta

Foram escolhidas duas abordagens de classificação supervisionada disponíveis na biblioteca scikit-learn: K-Vizinhos Mais Próximos (KNeighborsClassifier) e Floresta Aleatória (RandomForestClassifier).

K-Vizinhos Mais Próximos (KNeighborsClassifier)

- O KNN classifica um novo ponto de dados com base na maioria dos rótulos de seus k vizinhos mais próximos no espaço de características. Neste projeto, o modelo foi inicializado com os parâmetros padrão (k = 5 vizinhos e métrica de distância Minkowski).

- É um algoritmo simples e não-paramétrico. Sua eficácia é baseada na suposição de que amostras semelhantes existem em estreita proximidade umas das outras. É uma boa escolha inicial para problemas de classificação, especialmente em datasets onde as classes são bem separáveis em um espaço de características.

Floresta Aleatória (RandomForestClassifier)

- A Floresta Aleatória constrói múltiplas árvores de decisão durante o treinamento. A previsão final do modelo é obtida combinando (agregando) as previsões de árvores individuais (por exemplo, por votação majoritária para classificação). Neste projeto, o modelo foi utilizado com 100 árvores (n_estimators=100) e critérios padrão.

- É um método de ensemble baseado em árvores de decisão que geralmente oferece alta precisão, é menos propenso a overfitting do que uma única árvore de decisão e lida bem com a alta dimensionalidade dos dados de imagem (64 dimensões). É robusto e amplamente utilizado para classificação.

Outras ferramentas

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns


digits = load_digits()
```

Análise linha a linha:

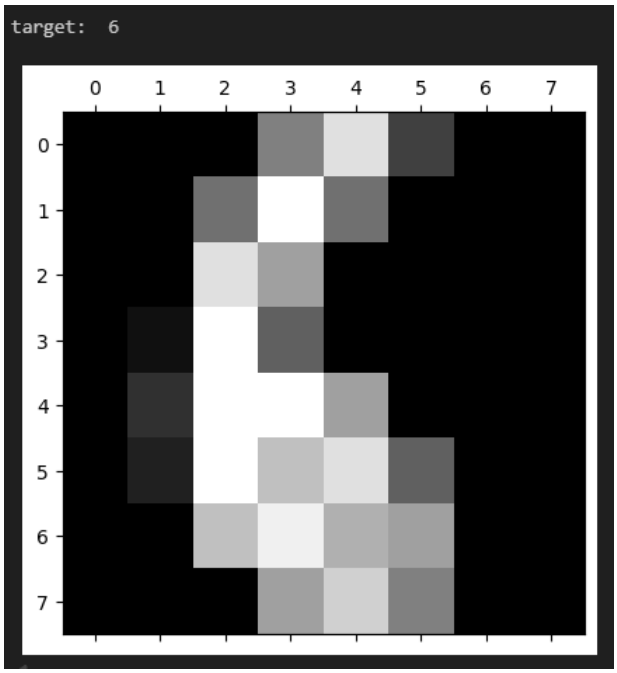
- Importação do dataset a partir da biblioteca sklearn;
- Importação da função de divisão treino/teste a partir da biblioteca sklearn;
- Importação do KNN a partir da biblioteca sklearn;
- Importação da RandomForest a partir da biblioteca sklearn;
- Importação da métrica de matriz de confusão a partir da biblioteca sklearn;
- Importação da biblioteca matplotlib para visualização dos dados;
- Importação da biblioteca numpy para manipulação de dados numéricos;
- Importação da biblioteca seaborn para visualização dos dados;
- Inicialização do dataset;

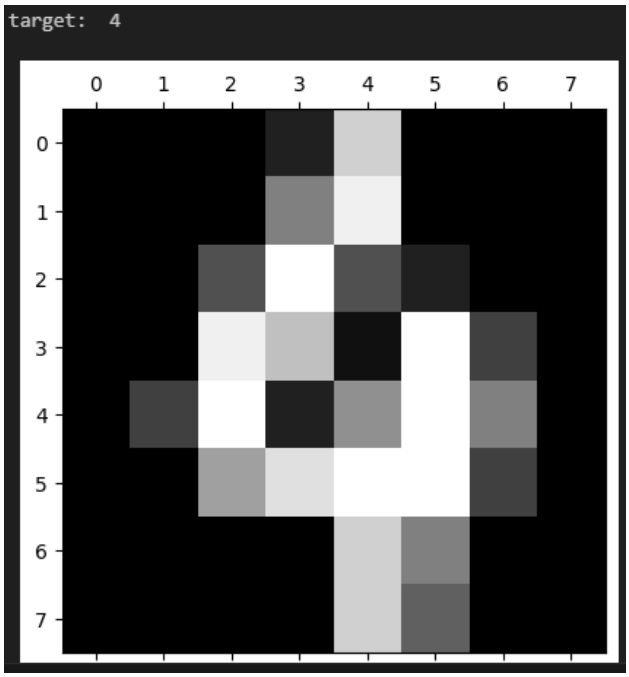
Descrição do dataset

O dataset utilizado no projeto é o "Digits" (Dígitos), que é um conjunto de dados de classificação de acesso rápido (built-in) no scikit-learn. O dataset é carregado diretamente através da função *load_digits()* da biblioteca sklearn.datasets do Python.

O código abaixo nos apresenta uma visualização dos tais dígitos e qual a saída esperada (target) do modelo:

```
for i in range(100,120):
    plt.matshow(digits.images[i], cmap="gray")
    print('target: ', digits.target[i])
    plt.show()
```





Apresentação dos Resultados

Os modelos KNN e Random Forest foram treinados com 80% dos dados (x_train, y_train) e avaliados com 20% dos dados (x_test, y_test).

Acurácia dos Modelos

A acurácia, calculada com *model.score(x_test, y_test)*, representa a proporção de previsões corretas no conjunto de teste.

KNN (model1):

```
model1.score(x_test,y_test)
✓ 0.0s
0.9916666666666667
```

RandomForest (model2):

```
model2.score(x_test,y_test)
✓ 0.0s
0.9777777777777777
```

Análise com Matriz de Confusão

As matrizes de confusão fornecem uma visão detalhada de onde os modelos erraram, comparando os rótulos verdadeiros com os rótulos previstos.

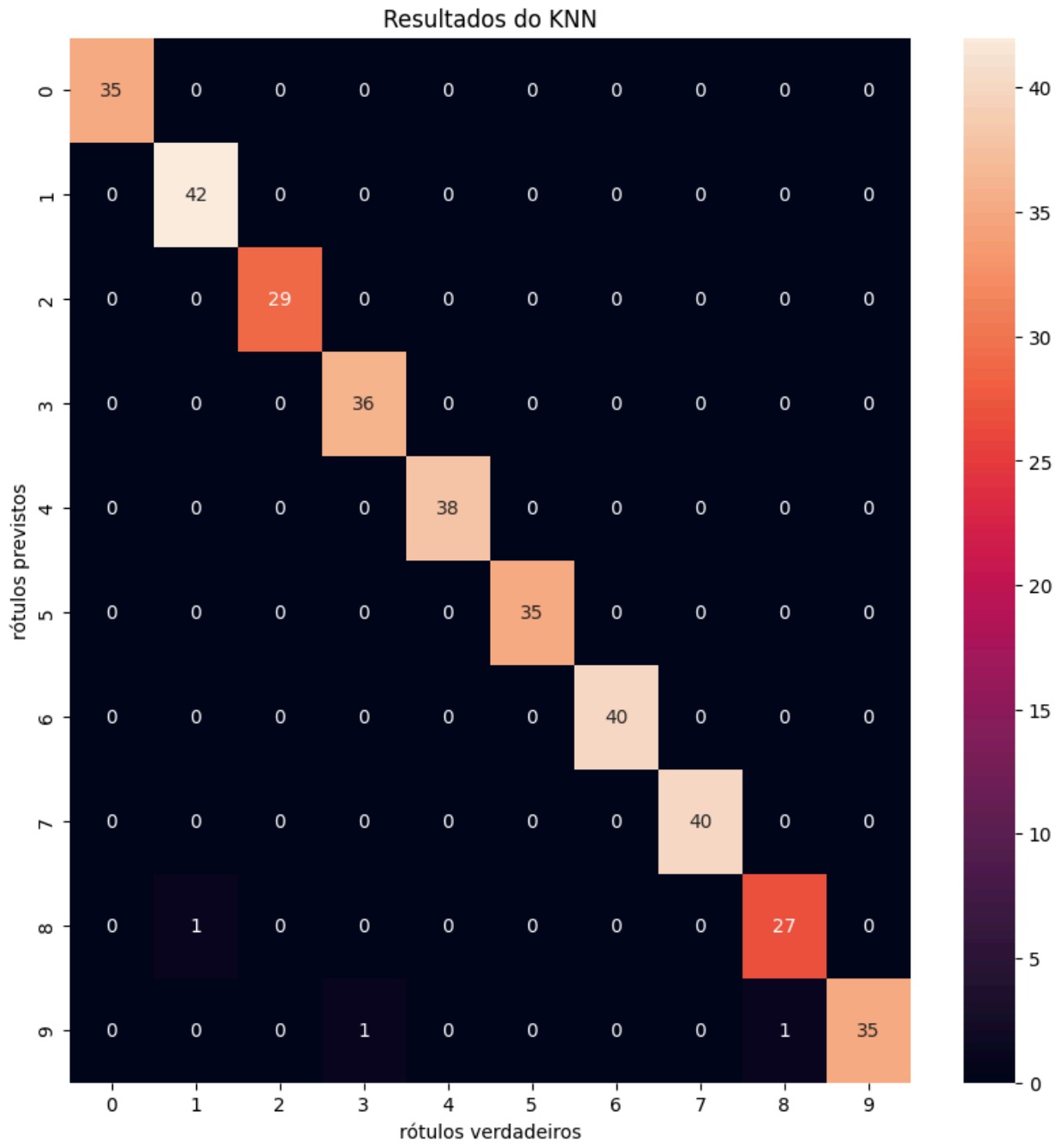
```
y_preds1 = model1.predict(x_test)
y_preds2 = model2.predict(x_test)
```

- y_test é o vetor de resultados verdadeiros.
- y_preds1 é o vetor de predição do KNN (model1)
- y_preds2 é o vetor de predição do RandomForest (model2)

Resultados do KNN (KNeighborsClassifier)

Acurácia: Altíssima (99.16%)

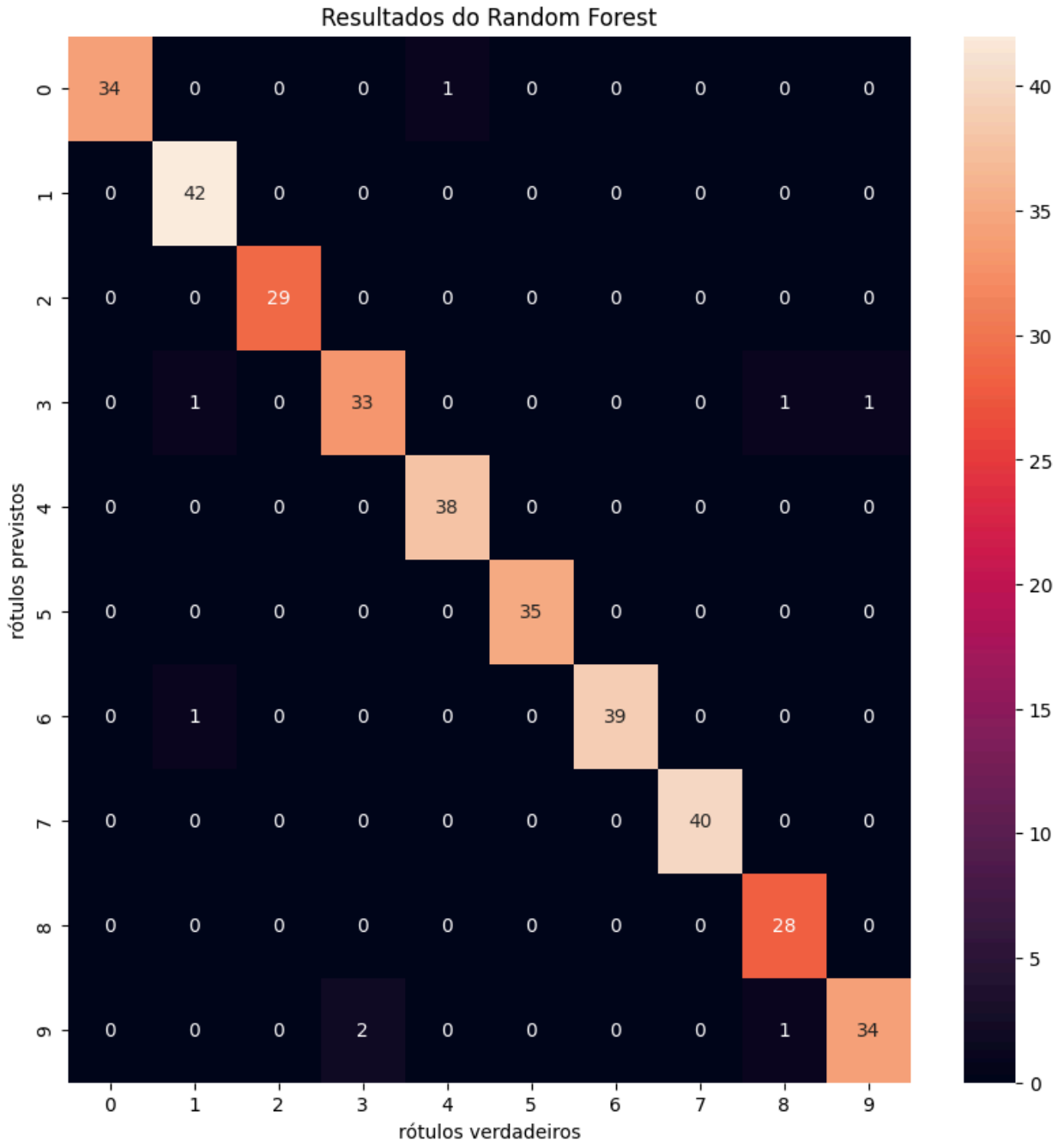
```
fig1, ax1 = plt.subplots(figsize=(10,10))
ax1 = sns.heatmap(confusion_matrix(y_test,y_preds1),annot=True,cbar=True)
plt.title('Resultados do KNN')
plt.xlabel('rótulos verdadeiros')
plt.ylabel('rótulos previstos')
```



Resultados do Random Forest (RandomForestClassifier)

Acurácia: Altíssima (97.77%)

```
fig2, ax2 = plt.subplots(figsize=(10,10))
ax2 = sns.heatmap(confusion_matrix(y_test,y_preds2),annot=True,cbar=True)
plt.title('Resultados do Random Forest')
plt.xlabel('rótulos verdadeiros')
plt.ylabel('rótulos previstos')
```



Discussão dos resultados

Comparação entre Modelos

O KNN superou o Random Forest em termos de acurácia neste dataset. Isso pode ser atribuído à natureza do dataset Digits: as imagens são muito pequenas e as características (pixels) são bem discriminatórias. O KNN, que é um algoritmo baseado em distância, pode ter se beneficiado da proximidade clara entre os vetores de imagens dos mesmos dígitos.

O Random Forest, embora seja um classificador robusto, pode ter tido dificuldades com a alta correlação entre as características (pixels adjacentes) ou pode ter se beneficiado de uma otimização de hiperparâmetros mais profunda (como max_depth ou min_samples_split).

Análise de Erros (Matriz de Confusão)

Ambos os modelos mostram uma tendência a confundir visualmente dígitos semelhantes. Os erros mais frequentes envolvem pares como 7 e 9, e 8 com outros dígitos (1, 5, 9). Isso é esperado, um '7' com um traço cortado pode se assemelhar a um '9', ou um '8' mal desenhado pode parecer um '1' ou '5'.

Bibliografia

https://scikit-learn.org/stable/index.html
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html