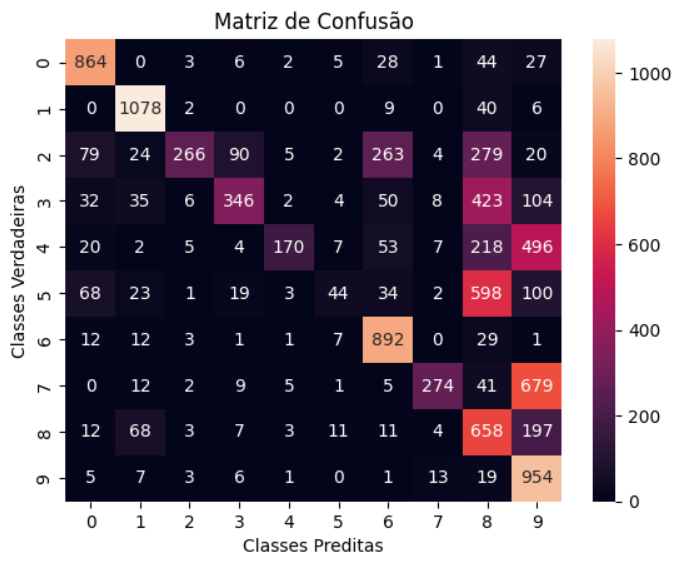
TRABALHO 2

INTELIGÊNCIA ARTIFICIAL

Larissa Magalhães Pereira

Nº USP: 13747904



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Taxa de Erro | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Total |
| Teste | 0.118 | 0.050 | 0.742 | 0.657 | 0.827 | 0.951 | 0.069 | 0.733 | 0.324 | 0.055 | 0.5546 |

1. Com os resultados obtidos, podemos concluir que o uso do Naive Bayes Gaussiano não é muito eficiente, já que foi alcançado apenas 55% de acurácia. Caso utilizássemos o Naive Bayes Multinomial, conseguiríamos obter melhores resultados (84% de acurácia). Além disso, pela matriz de confusão e a taxa de erro, observamos que os números 2, 3, 4, 5 e 7 foram os com maiores erros de classificação, com taxas de erro maiores que 50%.
2. Para obter os resultados, eu utilizei a linguagem de programação Python e as bibliotecas pandas e sklearn.

Primeiramente, foi feito a leitura dos dados dos arquivos CSV. Nos arquivos, alterei/adicionei apenas a primeira linha para colocar o nome dos atributos, como ‘label’, ‘pixel 1’, ‘pixel 2’...

Depois, realizei a normalização dos dados e a separação dos conjuntos treinamento e teste, além da separação de entradas (pixels) e saídas (label) em cada um desses conjuntos.

Utilizando o método GaussianNB, criei o classificador e realizei o treinamento com o conjunto treinamento. Posteriormente, foi feito o teste.

Finalizado o treinamento, além da sklearn, utilizei as bibliotecas seaborn, matplotlib e numpy para mostrar as métricas (como a acurácia), a matriz de confusão dos dados de teste e a taxa de erro por classe e geral do classificador, além da visualização das médias por classe em um formato de imagem 28x28.

Código: <https://colab.research.google.com/drive/1MB6QceoeFdbOwTSLmfW3yi96zm6aU0Fw?usp=sharing>

\*Código e arquivos CSV anexados