Biotech ML Challange

Minha metodologia usada foi entender o problema como um todo para depois criar uma abordagem e resolver as etapas descritas, as referências externas que eu usei estão todas anexadas em cada etapa, neste documento.

O documento está separado por etapa do desafio, contendo a metodologia usada para resolver a etapa específica e também meus pensamentos e conclusões.

Em relação aos desafios e soluções, tive mais desafios na etapa de classificação, como será explicado mais detalhadamente na parte específica do documento.

Data Processing

Nessa etapa utilizei as bibliotecas Pickle e Pandas para realizar a análise de dados primária do dataset, para obter o resultado basta somente rodar o arquivo data_processing.py

A leitura do arquivo resultou em um DataFrame com as informações do dataset, tal qual as que foram especificadas no desafio: syndrome_id, subject_id, image_id e o embedding da imagem como uma matriz.

As métricas que busquei foram as especificadas no desafio: número de síndromes e imagens por síndrome, porém também registrei outras métricas que julguei ser útil: representação em porcentagem da quantidade de imagem por síndrome, número total de imagens e média de imagens.

```
$ python data_processing.py
Number of uniques syndromes: 10
Total number of images: 1116
Average of images: 111.6
   syndrome_id  images_count  images_count_percent
0    100180860            67              6.003584
1    100192430           136             12.186380
2    100610443            89              7.974910
3    100610883            65              5.824373
4    300000007           115             10.304659
5    300000018            74              6.630824
6    300000034           210             18.817204
7    300000080           198             17.741935
8    300000082            98              8.781362
9    700018215            64              5.734767
```

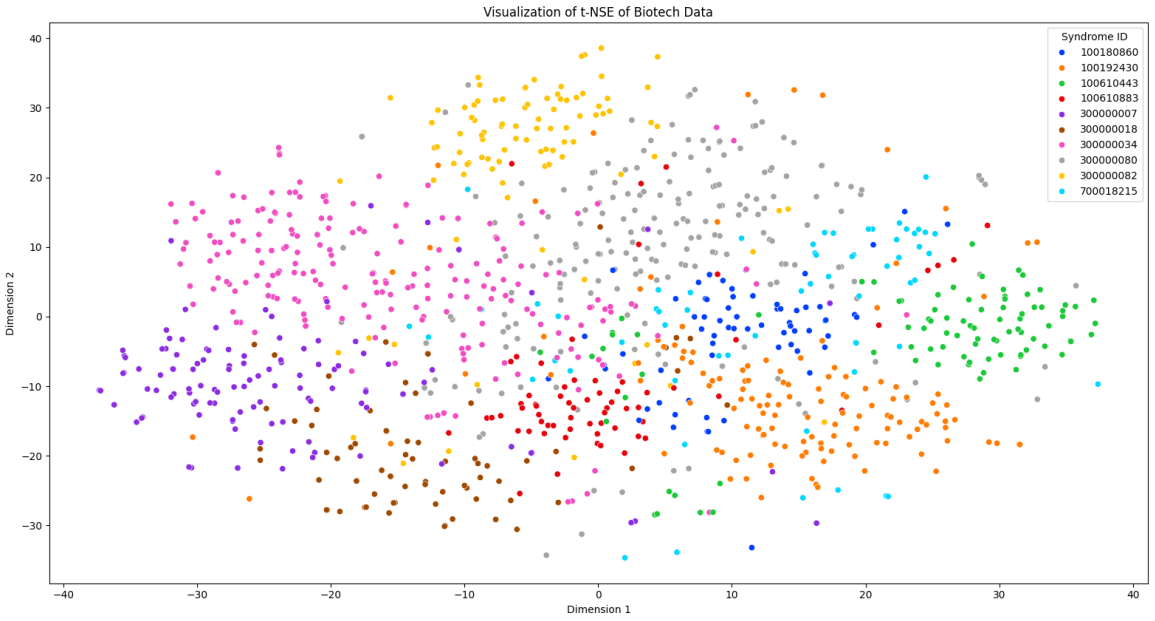
Analisando esses dados eu cheguei a conclusão que há um desbalanceamento no dataset, e refiro-me a quantidade de imagens da síndrome com id 300000034 e 300000080, que possuem uma quantidade de imagens muito elevada, o que pode impactar a avaliação de performance ou o desempenho das classes que tem menos imagens no dataset.

Data Visualization

Referências usadas:

https://scikit-learn.org/1.5/modules/generated/sklearn.manifold.TSNE.html

Para realizar essa etapa utilizei a mesma criação do DataFrame da etapa de **processamento de dados** e após isso apliquei o algoritmo de T-SNE da biblioteca sklearn, os resultados serão discutidos após a visualização, o gráfico pode ser gerado ao rodar o arquivo data_visualization.py



Em relação aos agrupamentos e padrões da visualização, consegui identificar que os 3 maiores agrupamentos, que também são os síndromes id com maior imagens no dataset (300000034, 300000080 e 100192430) são os que apresentam maior dispersão na visualização. Enquanto dentre os 3 menores clusters, o menor de todos (700018215) apresenta uma grande dispersão comparado aos outros 2 (100610883 e 100180860) que apresentam uma dispersão bem menor, ocupando uma área bem concisa na visualização.

Estas categorias de síndrome id que possuem uma grande dispersão pode representar uma subcategoria dentro dessa síndrome principal, porém isso requer uma análise mais minuciosa. Isso também mostra uma alta variabilidade nos dados.

Classification Task

Referências usadas:

matplotlib.pyplot.subplots — Matplotlib 3.10.0 documentation

KNeighborsClassifier — scikit-learn 1.5.2 documentation

3.4. Metrics and scoring: quantifying the quality of predictions — scikit-learn 1.5.2 documentation

Multiclass classification evaluation with ROC Curves and ROC AUC | by Vinícius Trevisan | Towards Data Science

Tive algumas dificuldades para me adaptar a situação, utilizando KNN e Cross Validation com o problema multiclasse. Nesse caso o cálculo das métricas (por exemplo f1-score e acurácia) estavam quebrando o algoritmo e precisei debuggar várias vezes para entender o que estava acontecendo.

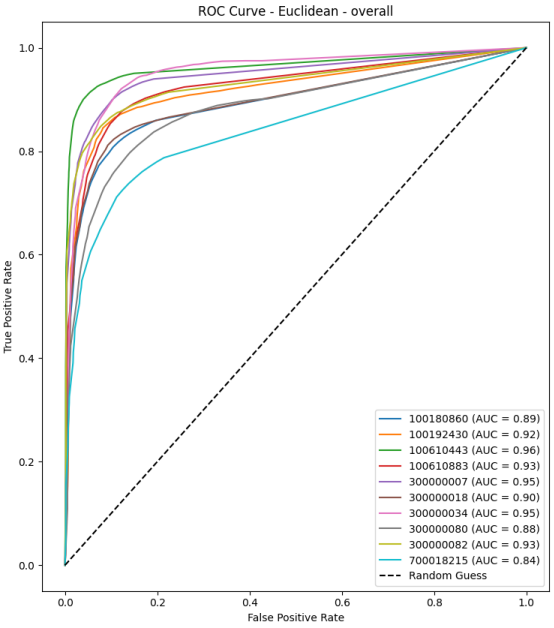
Além disso precisei reescrever o código algumas vezes, debuggando com ajuda de IA para conseguir chegar em uma solução que compreendia todos os requisitos.

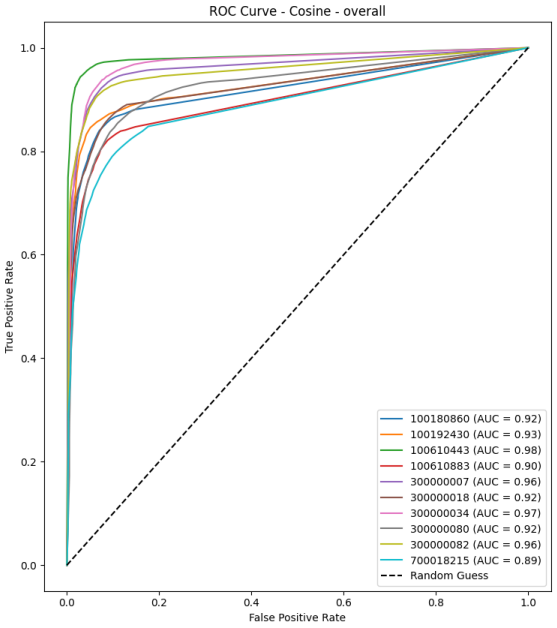
O meu código passou por algumas etapas e versões antes de eu chegar na versão final que irei discutir os resultados:

1. Na primeira versão, eu estava calculando apenas algumas métricas: precisão, recall e f1-score, além da AUC, porém a lógica do cálculo da AUC estava incorreto pois estava utilizando somente os dados da última iteração de K para este calculo, o que gerava uma ROC incorreta.
2. Nessa versão eu adicionei a métrica de acurácia para ser calculada, e além disso corrigi o cálculo da AUC e geração do ROC, porém nessa versão eu estava criando de maneira desnecessária o AUC e ROC da melhor iteração de K, gerando dois gráficos “desnecessários” pois mostravam apenas um recorte do melhor cenário de classificação. O que poderia resultar em interpretações erradas, por isso resolvi remover isso na próxima versão.
3. Na última versão, adicionei a métrica correta para o top-k-accuracy e também removi os cálculos desnecessários para ROC e AUC que comentei anteriormente, também adicionei uma etapa para salvar os DataFrames (Métricas do modelo e AUC para cada classe) em um arquivo .csv para facilitar a criação dos gráficos destas métricas.

Para obter este gráfico basta somente rodar o arquivo classification_task.py

Plot da ROC e AUC por classe





Logo de cara eu identifiquei que a classe com a menor AUC foi a classe 70018215, que é a mesma classe que apresentou a maior dispersão na etapa de visualização com T-SNE mostrando uma correlação entre essas duas métricas.

Também é notório que ao utilizar a métrica com Cosseno para o cálculo de distâncias tive um resultado geral melhor se comparar com a métrica Euclidiana.

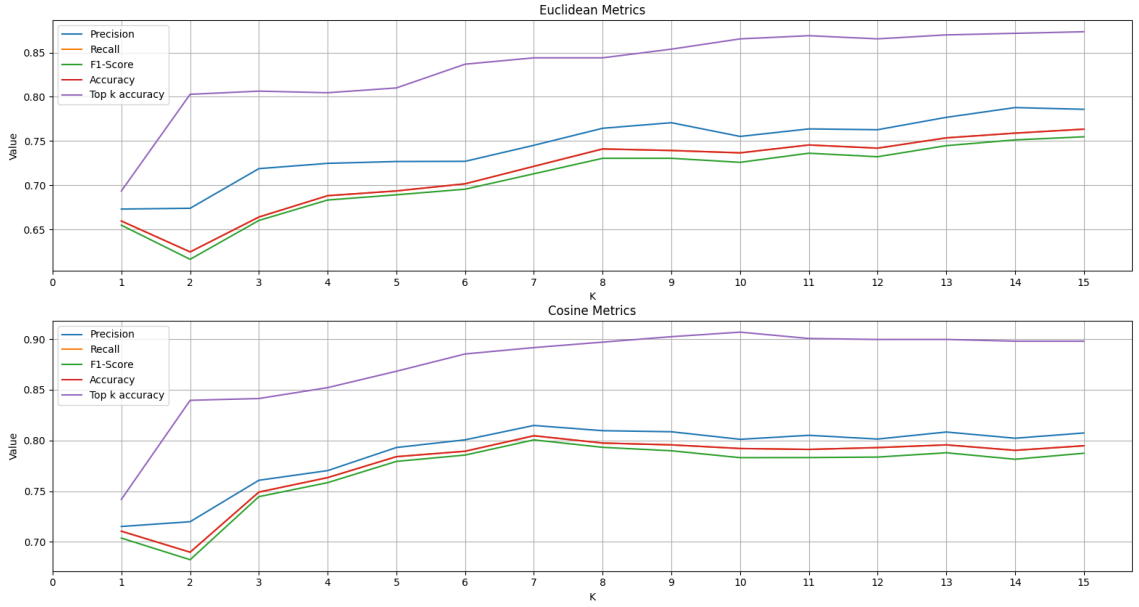
Porém no geral, pude concluir que obtive um bom resultado para a curva de ROC e da Área sob a curva.

Metrics and Evaluation

Aqui tenho a evolução das métricas em cada iteração de K para o Cross Validation.

Nessa etapa eu reescrevi várias vezes o código tentando buscar a melhor visualização que deixasse claro essa evolução, acabei escolhendo por esse gráfico de linhas acoplando todas as métricas.

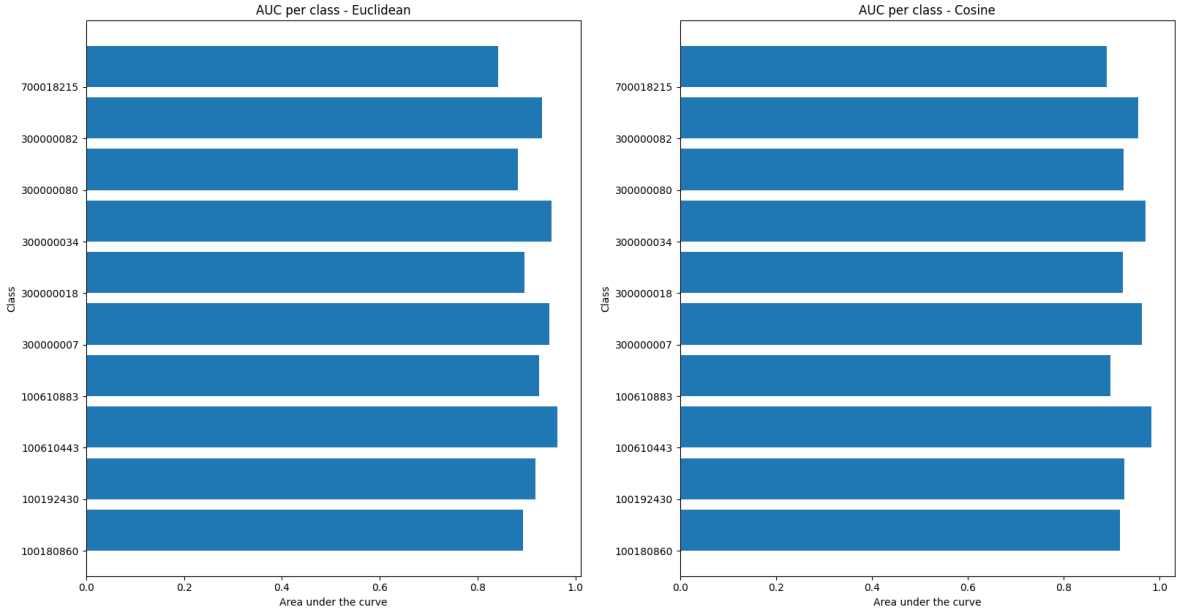
Para gerar tanto o gráfico das métricas (Precisão, Recall, F1-Score...) e o gráfico de AUC por classe é necessário criar primeiramente os arquivos .csv com o código da etapa de classificação dos dados (classification_task.py) e depois rodar código do arquivo metrics.py



Pude perceber que não há diferenças gritantes entre a escolha de cálculo de distâncias com Cosseno ou Euclidiano, mas novamente, o uso do KNN com Cosseno teve um resultado ligeiramente superior nas métricas em geral, em específico o Top K Accuracy.

Com essa visualização também é possível notar que o Recall não aparece no gráfico, isso por que ele é **igual** à acurácia. Isso não é usual em dados desbalanceados como o dataset que temos.

Uma outra visualização para a AUC por classe



Next Steps

Desde a primeira análise superficial é possível identificar que há classes desbalanceadas, para próximos passos eu tentaria aumentar o número de dados para as classes desbalanceadas, tentando minimizando o underfitting dessas classes. Em relação a classe com syndrome id 700018215 eu focaria em entender melhor o motivo dele ser muito disperso na visualização T-SNE e também o motivo de ter a menor AUC, análises mais aprofundadas revelariam se realmente há uma subclasse dentro dessa categoria.

Em relação ao recall médio ser igual à acurácia média na classificação do KNN, eu implementaria uma abordagem diferente e calcularia as estas métricas individualmente por classe, identificando assim se as classes tem desempenho semelhante no KNN.