Report

Desenvolvimento APOLLO

Autor: Hyago Vieira Lemes Barbosa Silva

Relatório Final: **Classificação de Embeddings para Identificação de Síndromes Genéticas**

1. Metodologia

1.1 Processamento de Dados

Carregamento e Pré-processamento

Para reprodutibilidade dos resultados foi utilizado seed = 0.

O primeiro passo do pipeline é o carregamento dos dados a partir de um arquivo no formato **pickle**, utilizando a biblioteca pickle do Python. Esse arquivo contém uma estrutura hierárquica de dados, que é processada para garantir um formato adequado para análise. O carregamento dos dados é realizado através da biblioteca pickle, com with open.

Após o carregamento, a estrutura dos dados é **achatada** (*flattening*), convertendo a hierarquia original em uma lista de dicionários. Esse processo ocorre em três níveis de iteração (for loops), percorrendo os elementos e extraindo as informações essenciais, como **ID da síndrome**, **ID do sujeito**, **ID da imagem** e **embedding** correspondente.

Além disso, para garantir a integridade dos dados, um segundo conjunto de listas é criado para armazenar **inconsistências**. Durante esse processamento, são realizadas verificações para identificar possíveis problemas, incluindo:

- **Campos vazios**: Todos os campos devem conter informações válidas.
- **Dimensionalidade dos embeddings**: Cada vetor de embedding deve ter **320 dimensões**; caso contrário, é considerado inválido.

Os dados válidos são armazenados em um **DataFrame com biblioteca pandas**, garantindo que a manipulação posterior seja mais eficiente. Já os dados inconsistentes são registrados em um log (issues), possibilitando auditoria e ajustes posteriores.

Criação dos DataFrames Processados

O processamento resulta em três DataFrames distintos, cada um representando uma versão dos embeddings para análise posterior:

- df_original: Contém os embeddings originais, sem transformações.
- df_normalizado: Aplicada Normalização L2, onde cada vetor é ajustado para ter norma unitária.
- df_padronizado: Aplicada Padronização Z-score, onde os valores são transformados para média 0 e desvio padrão 1.

A escolha dessas transformações foi motivada pela necessidade de lidar com a grande variação nos valores dos embeddings, o que pode impactar o desempenho dos modelos de classificação, dando mais importância a específicas imagens, do que outras, dependendo dos valores dos embeddings, assim precisa ter uma padronização ou normalização dos dados, dependendo de cada caso.

No gráfico gerado pelo t-SNE dos dados originais, observou-se que os embeddings apresentam distribuições não padronizadas, e não normalizadas com valores distribuídos de forma heterogênea.

Motivação para a Normalização e Padronização

As duas técnicas foram testadas devido à ausência de uma faixa de valores padronizada nos embeddings, que podem conter valores amplos e variados. Como os dados são embeddings de imagens, a normalização é uma abordagem natural para manter a consistência da magnitude dos vetores. Abaixo, explicamos cada uma delas em detalhes:

1. **Normalização L2 (Norma Unitária)**

A normalização L2 garante que cada vetor de embedding tenha uma norma unitária. Isso é útil especialmente para algoritmos baseados em similaridade de cosseno, pois evita que a magnitude dos vetores interfira na comparação angular. A transformação é aplicada da seguinte forma:

$$x' = \frac{x}{\left|\left|x\right|\right|_2}$$

Onde $\left|\left|x\right|\right|_2$ representa a norma euclidiana do vetor x.

Essa abordagem é a mais adequada para classificadores baseados na distância de cosseno, pois padroniza as relações angulares entre os vetores sem alterar a direção.

2. Padronização (Z-score)

A padronização transforma os dados para uma distribuição com média zero e desvio padrão um, garantindo que todas as dimensões tenham escala comparável. A transformação segue a equação:

$$x' = \frac{x - \mu}{\sigma}$$

onde:

- μ é a média do conjunto de dados;
- σ é o desvio padrão.

Essa abordagem pode ser útil para modelos que se baseiam na distância euclidiana, pois garante que todas as dimensões tenham a mesma influência no cálculo da distância.

IMPORTANTE:

Dado que os embeddings possuem 320 dimensões, a normalização L2 é a escolha mais apropriada, pois mantém a coerência angular entre os vetores, facilitando a distinção entre as síndromes analisadas. E os dados provavelmente não seguem uma distribuição uniforma de desvio padrão 1 e média 0.

Configuração do classificador KNN

O objetivo é avaliar o desempenho do modelo utilizando diferentes valores de **k** e duas métricas de distância: **Euclidiana** e **Cosseno**. Para isso, aplicamos **validação cruzada estratificada** e métricas de avaliação para determinar a melhor configuração do modelo.

A validação cruzada estratificada com 10 folds (n_folds=10) para garantir que todas as classes sejam representadas em cada iteração de treino e teste. O processo de validação cruzada envolve:

- Dividir o conjunto de dados em 10 partes;
- Treinar o modelo em 9 partes e testar na parte restante;
- Argumento adicional foi embaralhar (shuffle=True) para melhorar no treinamento e na comparação entre os folds.

Repetir o processo para cada parte, garantindo que todas as amostras sejam utilizadas para treino e teste.

Para o **K-Nearest Neighbors (KNN)**, um algoritmo baseado na proximidade entre amostras. O modelo é configurado para explorar os valores de **k** no intervalo **1 a 15**, utilizando duas métricas de distância:

- **Distância Euclidiana**: mede a distância direta entre os pontos;
- **Distância Cosseno**: mede a similaridade entre vetores baseando-se no ângulo entre eles. (Melhor para este caso, pois é um vetor (lista) embeddings.

O modelo é treinado para cada combinação de **métrica de distância** e **valor de k**, permitindo avaliar a melhor configuração para o problema.

O Classificador foi usado com os seguintes atributos:

Assim, a configuração recomendada após testes, com weights ‘uniform’, algoritmos ‘ball_tree’ ‘kd_tree’, utiliza:

- Dados originais (ou balanceados) normalizados ou padronizados.
- Métrica de distância: cosseno ou euclidiana,
- weights='distance' para ponderação dos vizinhos,
- algorithm='brute' ou ‘auto’, para alta dimensionalidade, ou para verificar cada caso, depende dos dados que foram enviados.
- E um valor de k variando de 1 a 15.

Após isso, O desempenho do modelo é avaliado por meio das seguintes métricas:

- Área sob a curva ROC (AUC): mede a capacidade do modelo de separar corretamente as classes;
- F1-score: equilíbrio entre precisão e recall;
- Acurácia: percentual de classificações corretas;
- Precisão (macro): média da precisão entre todas as classes;
- Recall (macro): média do recall entre todas as classes;
- Top-K Accuracy: taxa de acertos considerando as K predições mais prováveis;

Para encontrar o melhor valor de K, foi utilizado o índice composto, por 3 variáveis, ou seja. auc * f1 * accuracy * precision * recall * top_k

$$Best_{score_k} = \sqrt[3]{auc * f1 * top_k}$$

1.2 Análise Exploratória e Visualização

Sobre os dados após a conversão (Flatten) para um Dataframe, são vistos que possuem **10 síndromes**, estas são listadas em código.

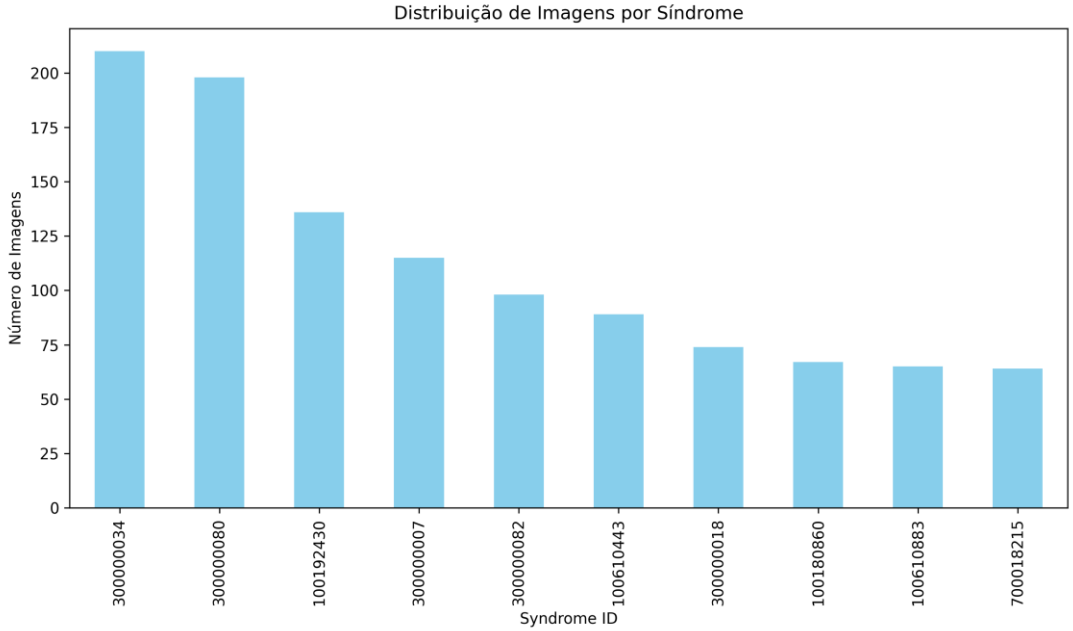
300000034 300000080 100192430 300000007 300000082 300000018 100610883 100610443 100180860 700018215

Destas 10 síndromes para os dados originais, existem, as respectivas quantidades de imagens por cada síndrome.

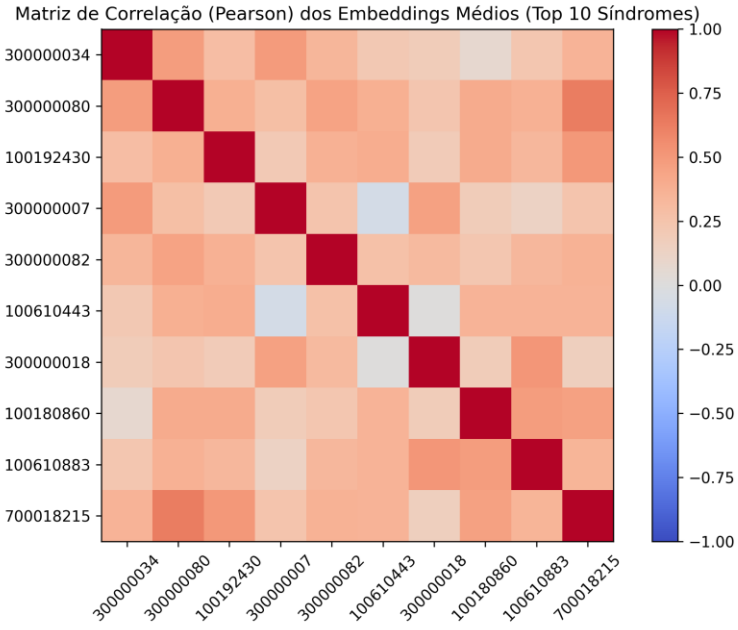
syndrome_id	Quantidade de imagens por síndrome
300000034	210
300000080	198
100192430	136
300000007	115
300000082	98

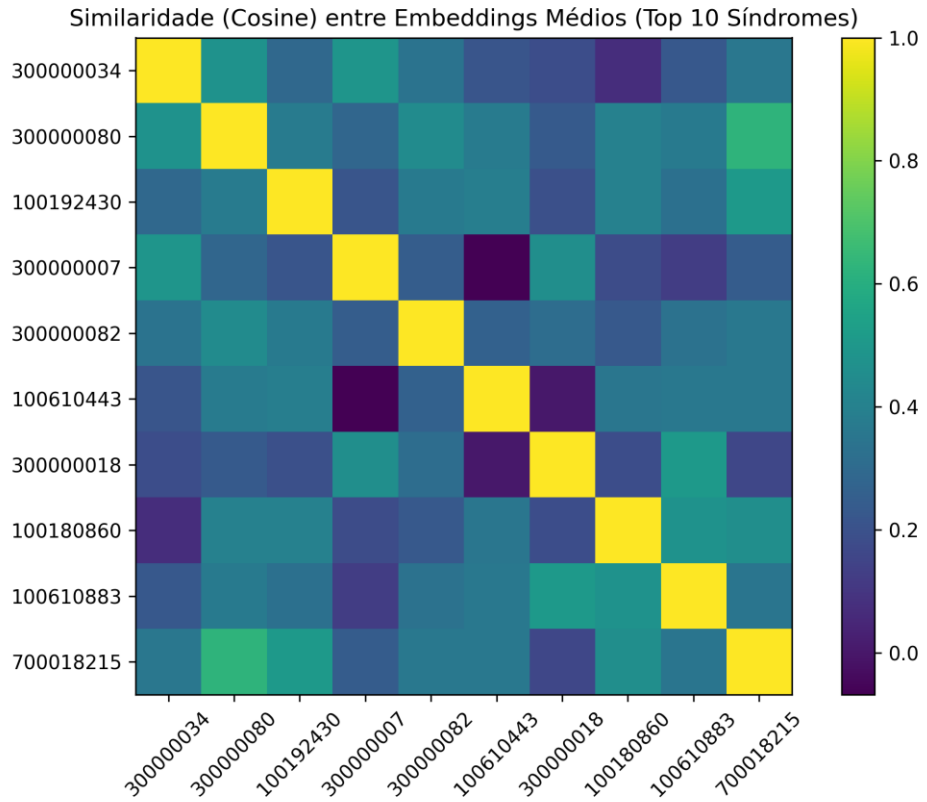
100610443	89
300000018	74
100180860	67
100610883	65
700018215	64

Veja o histograma abaixo da distribuição das imagens. Os dados seguem desbalanceados para cada classe, o que interfere crucialmente no desempenho de classificação das síndromes.



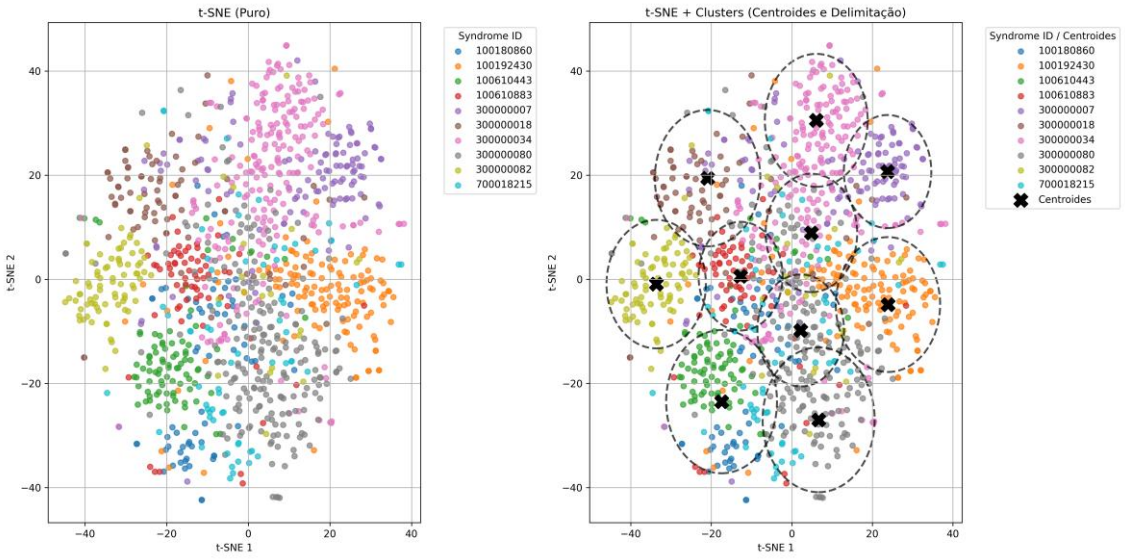
Para algumas análises iniciais, foi utilizada a correlação entre os embeddings de cada síndrome, entre as classes, basicamente foram analisados para ver o que existia de correlação entre os dados. Correlação de Pearson e Similaridade Cosseno.



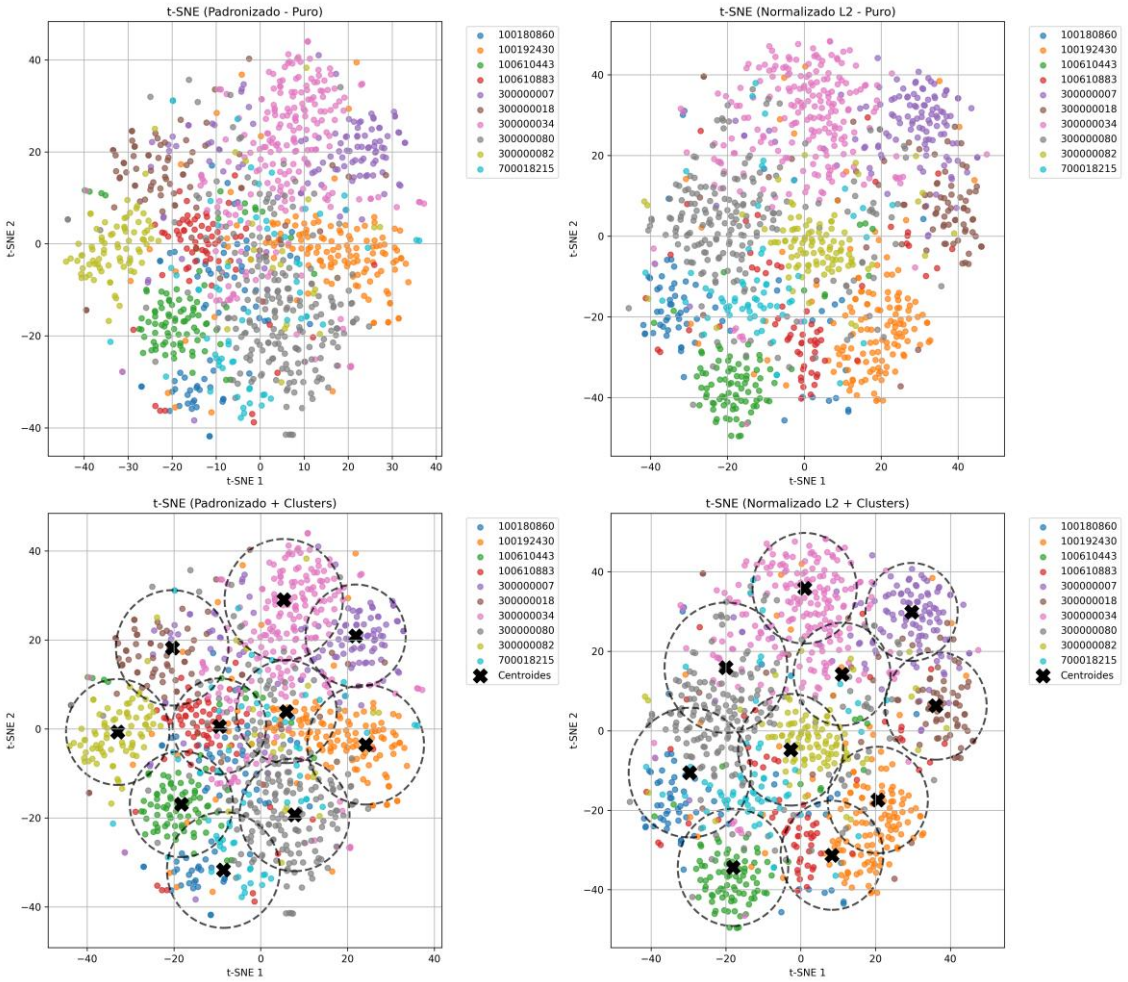


Portanto é visto que entre as classes, a correlação de Person e a similaridade Cosseno, demonstra que a maioria das classes ou síndromes possuem 50% de correlação/similaridade. Algumas especificamente são mais distintas, ou seja, diferentes entre si. Demonstradas nos gráficos acima.

Agora é elaborado o t-SNE para reduzir a dimensionalidade dos embeddings para 2D, permitindo a visualização das relações entre as classes. Também foi realizada o clustering K-Means para identificar padrões nos dados e os centroides assim como os clusters em torno de cada classe. Colorindo cada classe com cores diferentes para melhor visualização.



Agora veja os dados com normalização e padronização aplicada a seguir, e sua comparação.



A padronização não surgiu nenhum efeito praticamente nos dados, pois não é recomendada, pois os dados não seguem uma distribuição normal.

A normalização L2, já consegue ter uma maior expressividade na separação dos dados, dos clusters, mesmo ainda não conseguir separar completamente.

1.3 Balanceamento de Dados

Para corrigir desbalanceamentos no conjunto de dados foi aplicado DownSampling. O processo é dividido em três etapas principais:

Remoção de Outliers (Passo 1)

- Para cada classe, calcula-se o centróide dos embeddings (a média dos vetores da classe).
- As distâncias dos embeddings ao centróide são medidas.

- Um limite superior (definido por outlier_quantile=0.7) é usado para remover amostras que estejam muito afastadas do centróide.
- Isso ajuda a filtrar valores extremos que podem ser ruído no conjunto de dados. Este valor pode ser configurado dentro do código em data processing (python).

Manutenção de Classes Pequenas (Passo 2)

- Se a classe já possui um número pequeno de amostras (≤ min_count), todas as amostras são mantidas.
- Isso evita que classes minoritárias percam ainda mais amostras.

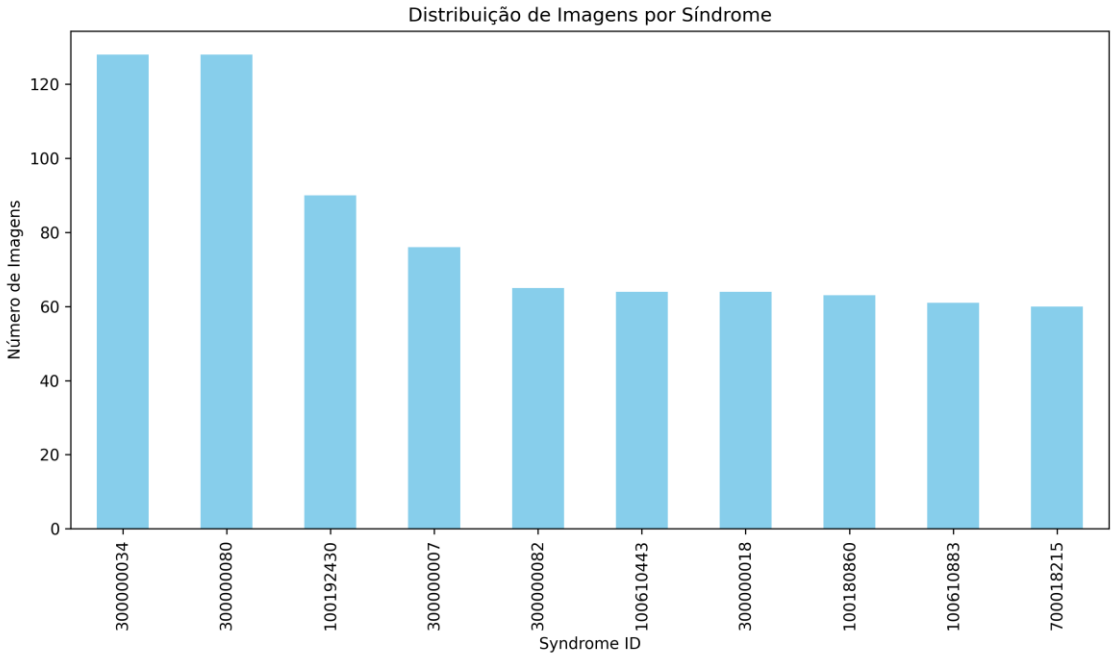
Seleção Estratégica (Passo 3)

- Após a remoção de outliers, as amostras remanescentes são classificadas pela distância ao centróide.
- Apenas as amostras mais próximas do centróide são mantidas, garantindo que representem melhor a classe.
- O número de amostras mantidas varia entre um mínimo (min_count) e um máximo (max_count = max_factor * min_count). Max Count foi basicamente o dobro da classe menor, caso exista essa quantidade de imagens.
- Se houver muitas amostras, as mais distantes são descartadas para manter o equilíbrio.

Após o balanceamento dos dados. Cada syndrome Id ficou com esta quantidade de imagens

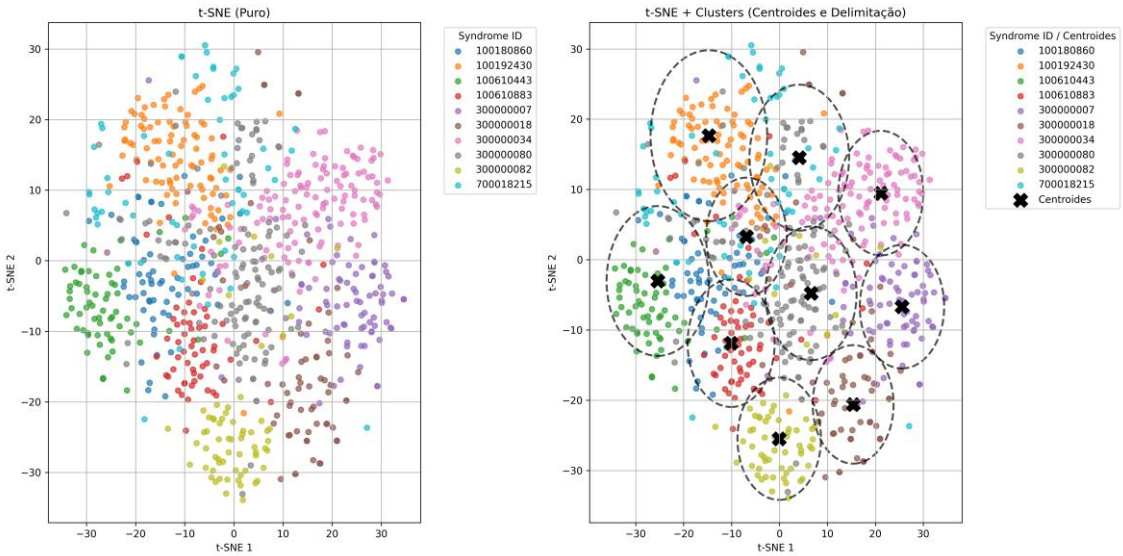
syndrome_id	Quantidade de imagens por síndrome
100180860	63
100192430	90
100610443	64
100610883	61
300000007	76
300000018	64
300000034	128
300000080	128
300000082	65
700018215	60

O histograma desses dados, ficaram.

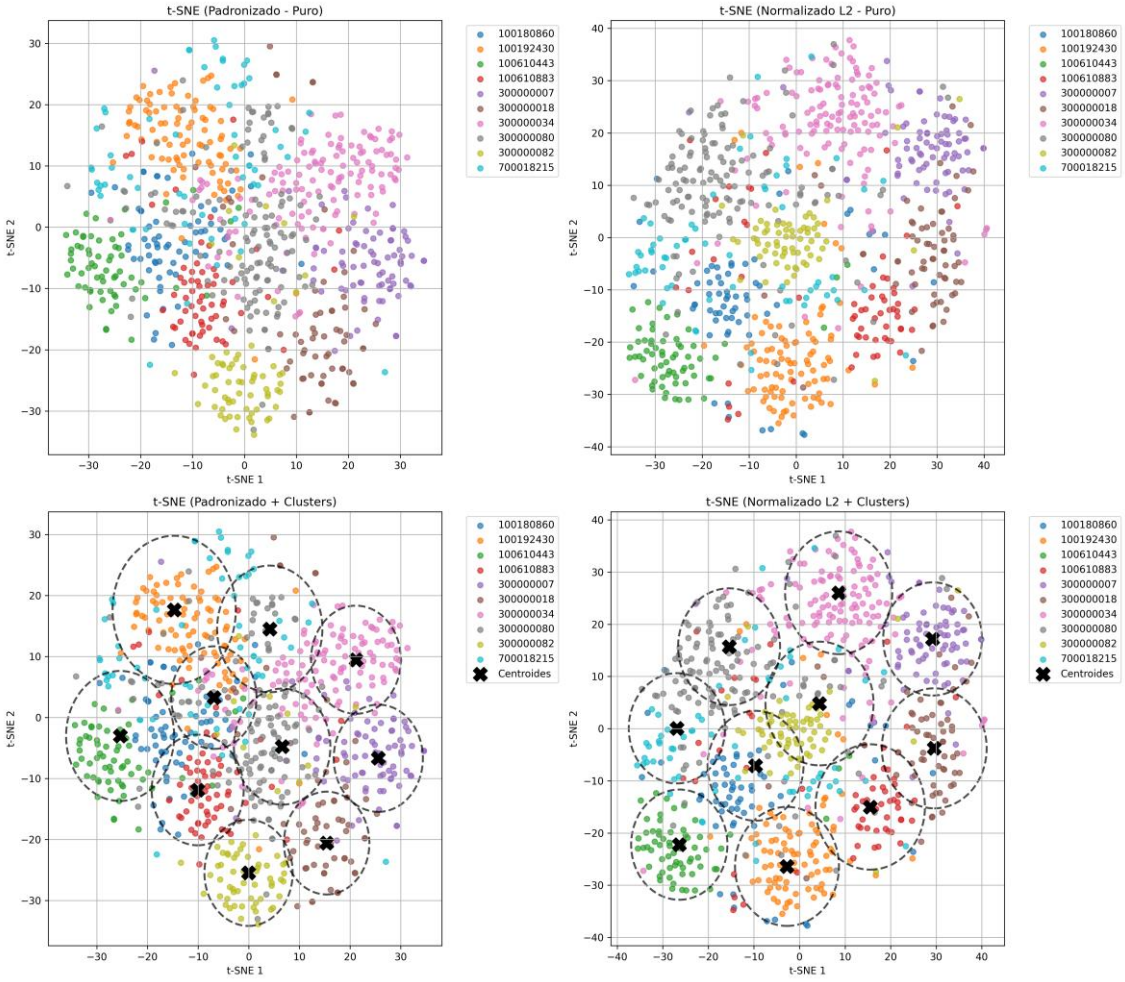


Veja que a uma representatividade alta de algumas classes e outras nem tanto, 50 % aproximadamente das outras, basicamente não possui um balanceamento dos dados.

Veja os dados agora balanceados, ainda sim possuem uma variabilidade muito alta, devido as dimensões analisadas de importância, ainda não é possível retirar essa variabilidade dos dados. Porém já estão mais separadas, a classe minoritária possui o maior problema, que azul claro, 70000...



Agora veja os dados com TSNE padronizado e Normalizado com L2



A padronização ainda não teve performasse sobre os dados, não sei exatamente o que houve, porém ainda não teve nenhuma alteração nos dados, agora a normalização podemos ver que houve uma grande distinção dos dados, e a concentração dos dados por classe, estão mais dentro de cada centroide. O que auxilia na classificação mais correta dos dados.

Nessas técnicas, ainda foram utilizadas outras, ao invés de fazer DownSampling, foi feito UpSampling, o aumento de dados, a forma correta de se fazer isso, seria através de GAN's, redes generativas, Pathology GAN, talvez para Síndromes, dependendo do caso, pix2pix, com redes como U-Net para gerar imagens artificiais, diferentes das reais, porém ainda possuem as características de interesse das imagens, por exemplo em um rosto humano, olhos, nariz, boca, por ai vai. Em imagens de síndromes seria necessário a maior avaliação para reprodução de mais imagens artificiais para compreensão.

Como não possuímos as imagens que deram origem aos embeddings, então foi feito algumas estratégias como:

- **SMOTE**: Gera amostras sintéticas para classes minoritárias. Cria novos pontos sintéticos no espaço de embeddings das classes minoritárias. Ela funciona (de forma resumida) ao interpolar vetores de embeddings vizinhos para gerar um ponto intermediário. É um problema, pois, temos

dados que estão ainda com outlier, e isto poderia causar dados ainda mais fora do escopo do cluster de classificação. No Python, foi usado a biblioteca imbalanced-learn (ou imblearn), que disponibiliza o SMOTE.

- **Noise Augmentation**: Introduz ruído gaussiano para melhorar a robustez do modelo. Adicionar pequenas perturbações aleatórias aos embeddings existentes, criando variações. Mas podem degradar a informação do embedding se o ruído for muito grande, e não necessariamente reflete variações reais das imagens originais

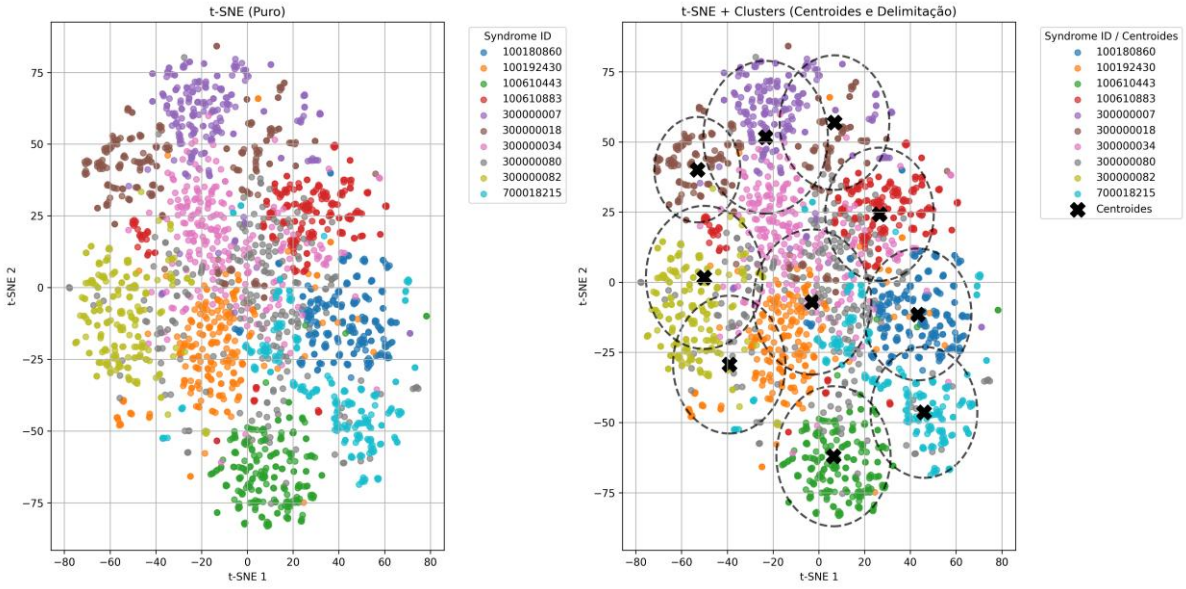
- **Mixup**: Combina embeddings de diferentes imagens para criar novas representações. Uma variação conceitualmente próxima do SMOTE é o Mixup (ou Manifold Mixup). Você pode interpolar embeddings de qualquer par de amostras dentro da mesma classe (ou mesmo entre classes, dependendo do método). A ideia é gerar:

$$embedding_{new} = concat([embeddig_{old}, \alpha * embedding_i + (1 - \alpha) * embedding_j])$$

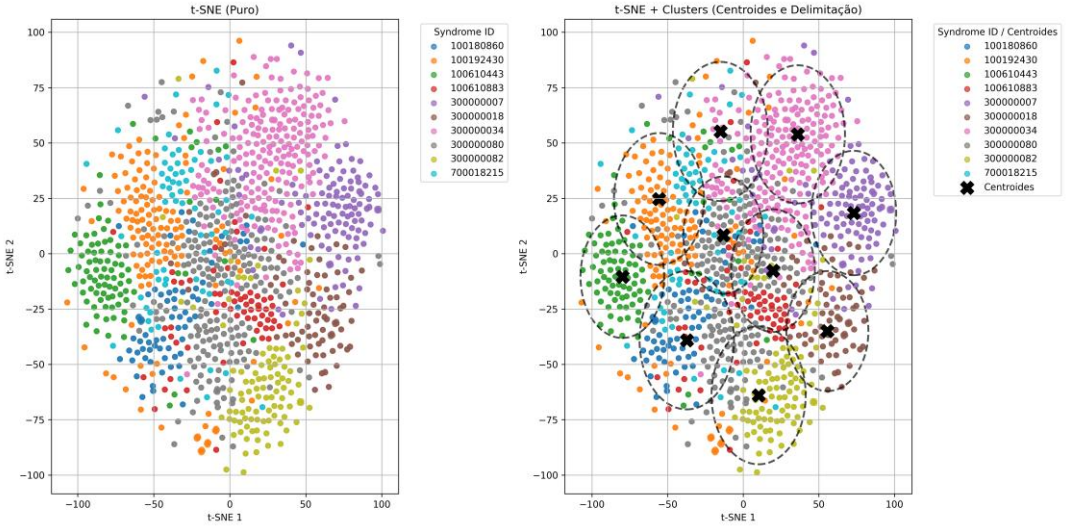
onde α é um escalar no intervalo, de [0,1]. Para classes minoritárias, você pode forçar mais combinações e gerar mais exemplos.

Veja o resultado de cada aplicação desta, para os dados originais.

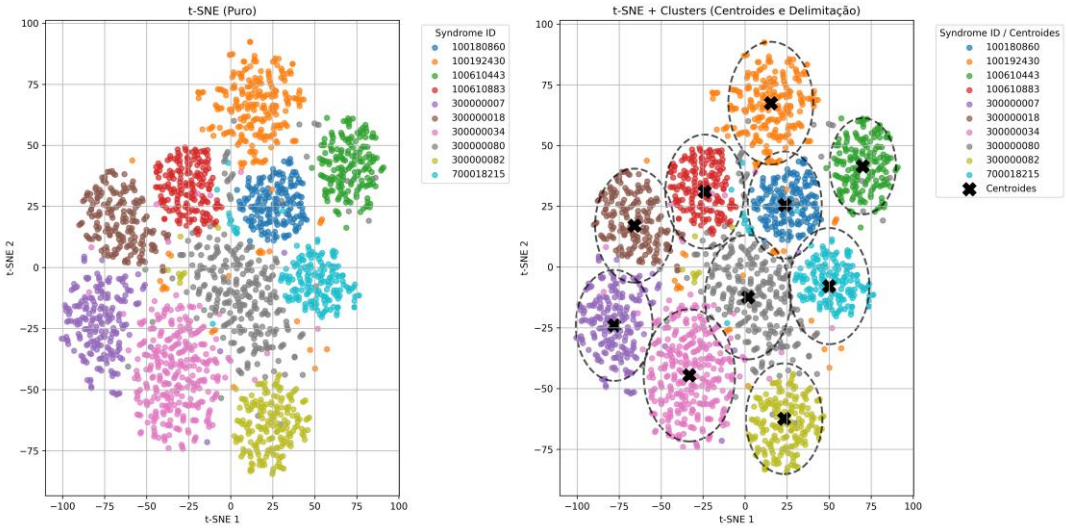
SMOTE



Noise Augmentation



Mixup



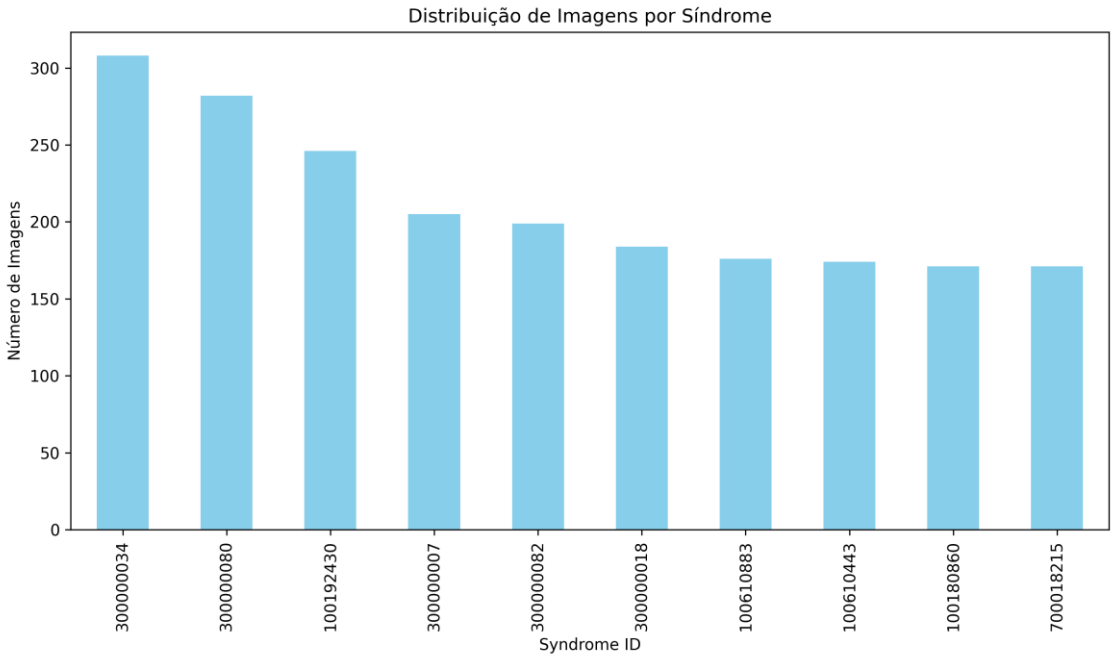
É nítido o resultado que obteve melhor separação dos dados, MIXUP, com isso veja agora os dados normalizados e padronizados, para o resultado com MIXUP, relacionados com os dados balanceados com DownSampling.

Esta escolha foi devido ao DownSampling ter auxiliado na melhoria dos dados, e evitando outliers e outras variantes no processo de classificação, sobre balanceamento dos dados. Após aplicar o Mixup os dados para cada classe ficaram distribuídos desta forma.

Após o mixup dos dados, eles ficaram assim.

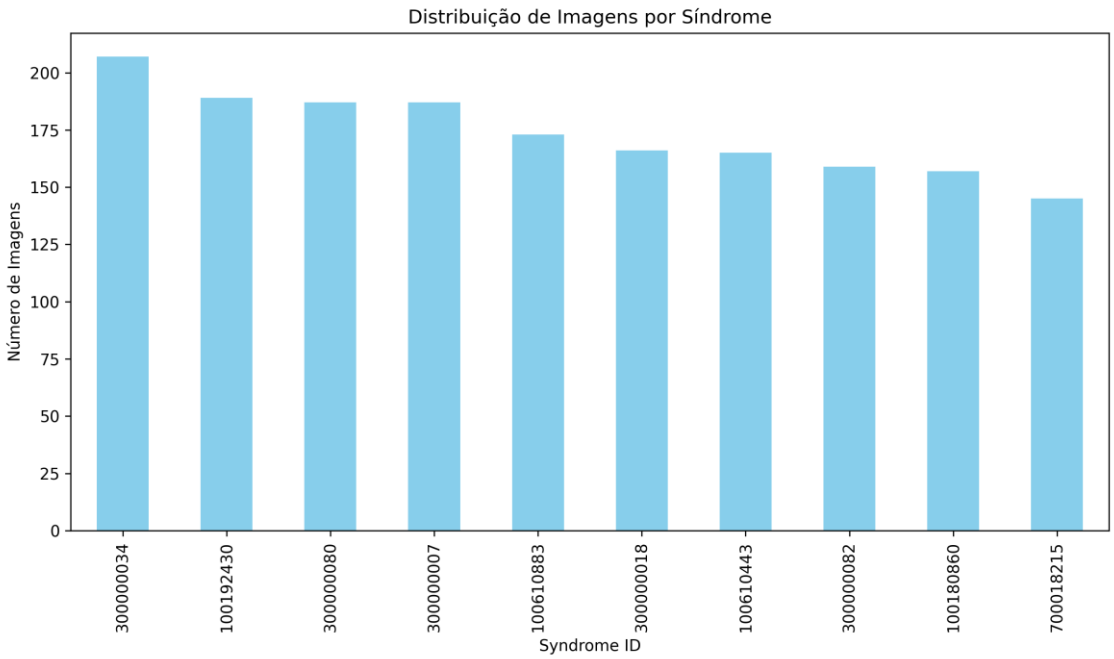
syndrome_id	Quantidade de imagens por síndrome
300000034	308
300000080	282
100192430	246
300000007	205
300000082	199

300000018	184
100610883	176
100610443	174
100180860	171
700018215	171



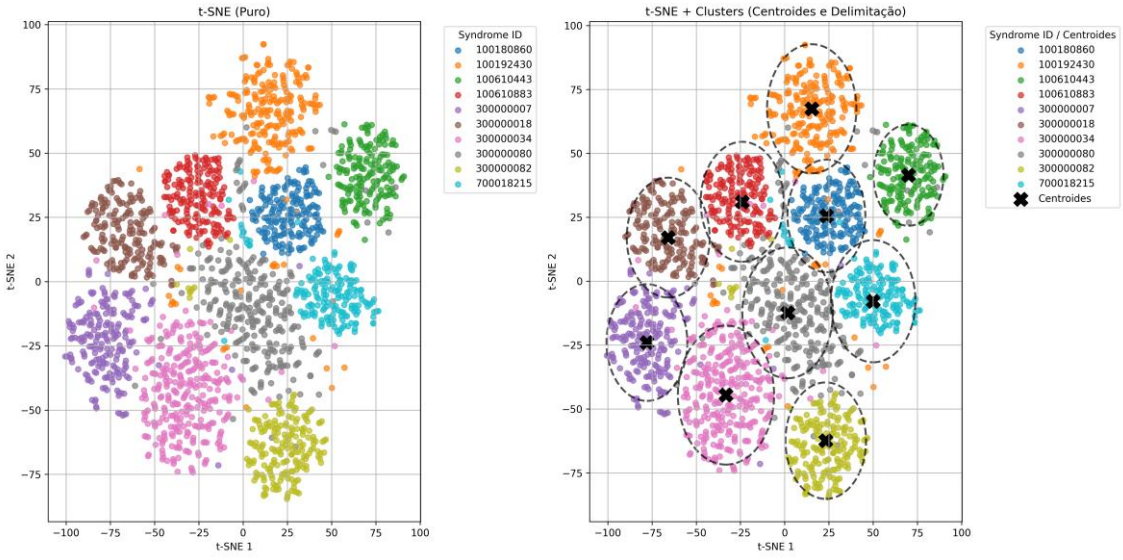
Com balanceamento ficou

syndrome_id	**Quantidade de imagens por síndrome**
300000080	242
300000034	240
100192430	179
300000007	177
100610443	174
300000018	171
100180860	166
700018215	158
100610883	149
300000082	143

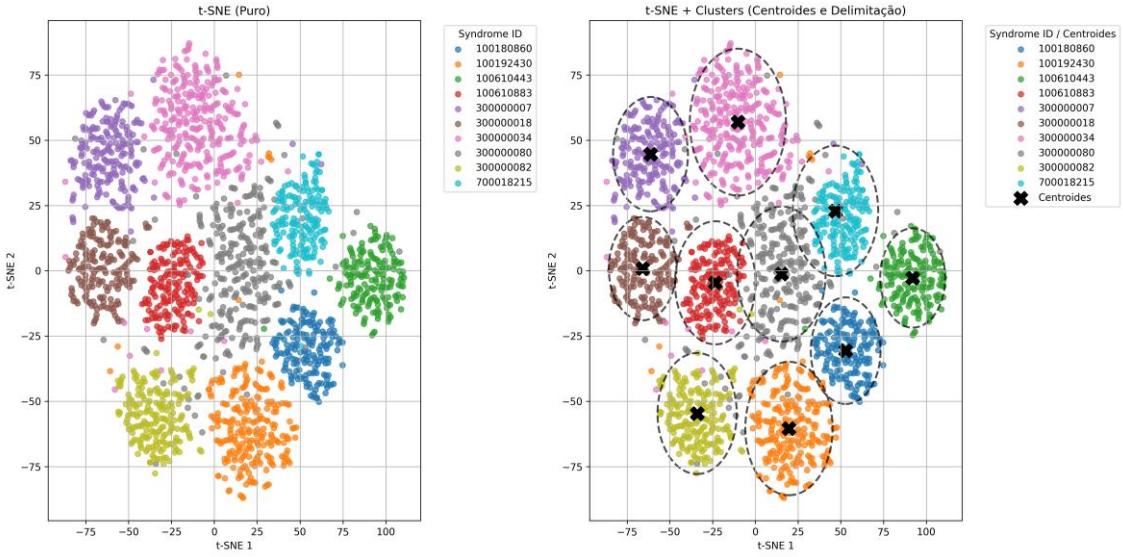


Agora veja o resultado, do TSNE, com e sem DownSampling, aplicando normalização e padronização dos dados. MIXUP dados originais padronizados e normalizados.

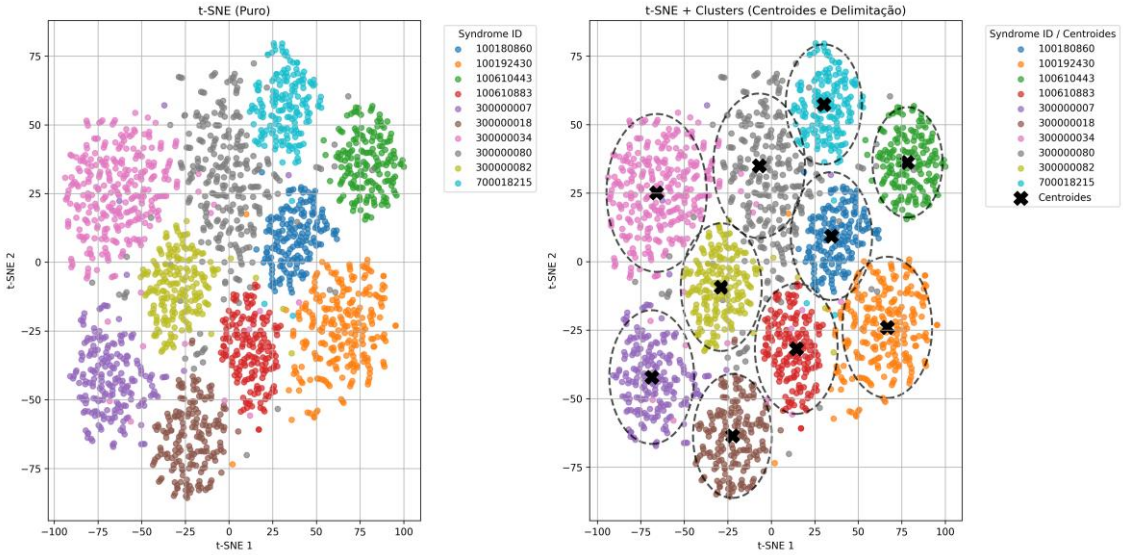
Dados originais Mixup aumento de dados



Normalizado



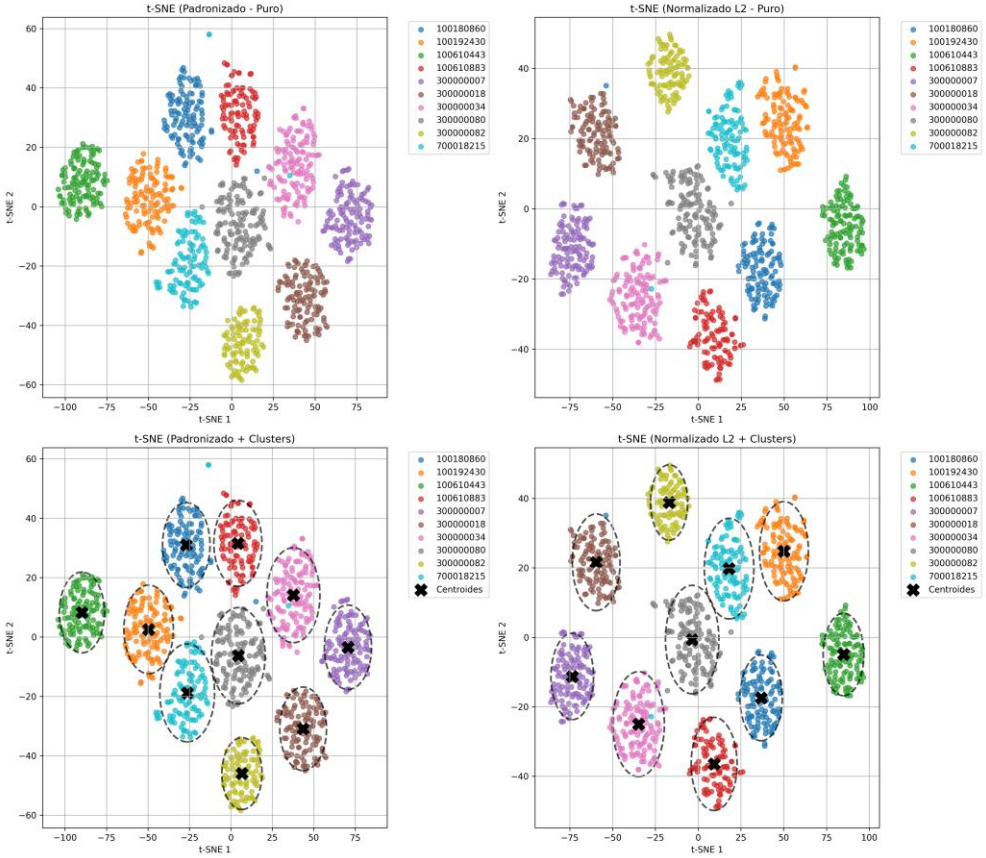
Padronizado



Este resultado do TSNE dos dados aplicando padronização ou normalização, apenas nos dados originais, sem o balanceamento das classes, ou seja, ainda existem os outliers e mantem a estrutura dos dados, antigos, ou seja os dados originais, junto dos dados artificiais.

Agora veja o resultado aplicando DownSampling, dos dados, juntamente com MIXUP com dados padronizados e normalizados.

Dados Mixup com aumento de dados – Balanceados.



A separação é nítida, sobre os dados, mas não fará o modelo sofrer “overffiting”, pelo fato de ainda manter os dados originais, porém com essa abordagem possui uma maior quantidade de dados artificial, que representam a classe, com o balanceamento conseguimos retirar dados que não são específicos da classe.

Este foi o melhor resultado obtido com pré-processamento dos dados e análise estatística e visualização dos resultados.

1.4 Classificação com KNN

Foram gerados gráficos das curvas de cada resultado, por classe e macro (média), além de salvar tabelas em “csv”, para os resultados macro e por classe. Algumas métricas foram avaliadas separadamente de outras, dando enfoque em cada caso.

Armazenamento de Resultados:

Os resultados são agregados (médias dos folds) e, opcionalmente, as predições podem ser armazenadas para análises futuras. Os dados finais são salvos em arquivos CSV.

A proposta é, portanto, oferecer uma avaliação completa e comparativa do KNN sobre os embeddings, explorando diferentes configurações de k e métricas de

Para encontrar o melhor valor de K, foi utilizado o índice composto, por 3 variáveis, ou seja. auc * f1 * accuracy * precision * recall * top_k

$$Best_{score_k} = \sqrt[3]{auc * f1 * top_k}$$

Ou seja, penalizando o score caso tenha um resultado inferior, assim foi escolhido a métrica para avaliar o melhor resultado de K, impresso no terminal.

Como o melhor valor de K resultante da combinação das três métricas acima.

2. Resultados

Os resultados se iniciam, com a análise do melhor processamento feito para o treinamento do classificador KNN.

Para isso foram feitas comparações. Sendo estas relacionadas abaixo:

- Resultado 1
  - Dados originais relacionando a padronização, normalização, ou puros, analisando desempenho das métricas de avaliação do modelo de classificação.
- Resultado 2
  - Dados Balanceados, relacionando a padronização, normalização, ou puros, analisando desempenho das métricas.
- Resultado 3
  - Dados originais, utilizando MIXUP para aumento dos dados, relacionando a padronização, normalização ou puros, analisando desempenho da classificação.
- Resultado 4
  - Dados Balanceados, utilizando MIXUP aumentando os dados, relacionando padronização, normalização e os dados puros. Vendo desempenho da classificação.

Todas as tabelas são grandes então não será possível colocar os valores aqui, mas estão em código.

Observação: Para treinamento foi desconsiderado, pois resultado estava como 100% sempre. Em todos casos.

Resultado 1 – (variando com validação cruzada o valor de k 1 a 15, avaliando o Melhor resultado de acordo com a métrica composta)

- (Dados originais):

distance_metric	k	test_auc	test_f1	test_accuracy	test_precision_macro	test_recall_macro	test_top_k
euclidean	13	0.951	0.743	0.771	0.812	0.730	0.995
cosine	15	0.964	0.770	0.797	0.818	0.756	0.996

Cosine foi melhor no desempenho das métricas originais.

- (Dados Originais Normalizados):

distance_metric	k	test_auc	test_f1	test_accuracy	test_precision_macro	test_recall_macro	test_top_k
euclidean	10	0.959	0.775	0.801	0.811	0.765	0.992
cosine	15	0.964	0.770	0.797	0.818	0.756	0.996

Cosine foi melhor no desempenho das métricas originais. Porém é visto que a distância euclidiana também resultou em melhora.

- (Dados Originais Padronizados)

distance_metric	k	test_auc	test_f1	test_accuracy	test_precision_macro	test_recall_macro	test_top_k
euclidean	13	0.950	0.733	0.758	0.795	0.721	0.995
cosine	14	0.965	0.783	0.807	0.810	0.780	0.995

Cosine foi melhor no desempenho das métricas originais. E a distância euclidiana perdeu performace.

Resultado 2 - Melhor resultado de acordo com a métrica composta

- (Dados balanceados)

distance_metric	k	test_auc	test_f1	test_accuracy	test_precision_macro	test_recall_macro	test_top_k
euclidean	15	0.957	0.762	0.775	0.826	0.751	0.997
cosine	15	0.977	0.807	0.817	0.841	0.798	1.000

Cosine continua sendo melhor no desempenho das métricas originais. E a distância euclidiana melhorou com balanceamento dos dados.

- (Dados Balanceados Normalizados)

distance_metric	k	test_auc	test_f1	test_accuracy	test_precision_macro	test_recall_macro	test_top_k
euclidean	7	0.982	0.884	0.895	0.908	0.879	0.996
cosine	10	0.986	0.882	0.899	0.913	0.878	0.997

Cosine possui uma diferença muito pequena entre a euclidiana, aqui vemos que a normalização para ambos casos auxiliam. Porém em geral ainda Cosine continua como melhor resultado em perdendo para duas métricas apenas recall, e f1-score.

- (Dados Balanceados Padronizados)

distance_metric	k	test_auc	test_f1	test_accuracy	test_precision_macro	test_recall_macro	test_top_k
euclidean	13	0.950	0.733	0.758	0.795	0.721	0.995
cosine	14	0.965	0.783	0.807	0.810	0.780	0.995

Cosine possui uma diferença grande entre a euclidiana, aqui vemos que a padronização para ambos casos auxiliam são diferentes, algumas métricas se mantem porém outras são danificadas.

Resultado 3 - Melhor resultado de acordo com a métrica composta

- (Dados Originais Mixup)

distance_metric	k	test_auc	test_f1	test_accuracy	test_precision_macro	test_recall_macro	test_top_k
euclidean	13	0.951	0.743	0.771	0.812	0.730	0.995
cosine	15	0.964	0.770	0.797	0.818	0.756	0.996

Cosine possui uma diferença pequena entre a euclidiana, mas ainda continua superior, porém o Mixup não foi tão benéfico quanto apenas o balanceamento. Ele puramente introduziu um pouco de overfitting, caso não padronizado ou normalizado.

- (Dados Originais Mixup Normalizados)

distance_metric	k	test_auc	test_f1	test_accuracy	test_precision_macro	test_recall_macro	test_top_k
euclidean	10	0.959	0.775	0.801	0.811	0.765	0.992
cosine	15	0.964	0.770	0.797	0.818	0.756	0.996

Aqui, a distância euclidiana obteve um resultado melhor do que cosine, em média. Inverteu aqui o melhor resultado, porém algumas métricas ainda cosine possui vantagem.

- (Dados Originais Mixup Padronizados)

distance_metric	k	test_auc	test_f1	test_accuracy	test_precision_macro	test_recall_macro	test_top_k
euclidean	13	0.950	0.733	0.758	0.795	0.721	0.995
cosine	14	0.965	0.783	0.807	0.810	0.780	0.995

Aqui consine toma vantagem novamente, sobre todos resultados empatando apenas na métrica de top-K accuracy.

Resultado - 4 Melhor resultado de acordo com a métrica composta

- (Dados Balanceados Mixup)

distance_metric	k	test_auc	test_f1	test_accuracy	test_precision_macro	test_recall_macro	test_top_k
euclidean	15	1.000	0.984	0.983	0.985	0.984	1.000
cosine	15	1.000	0.986	0.986	0.987	0.986	1.000

Aqui você percebe o melhor resultado, desde original normalizado ou padronizado abaixo, os dados seguem delimitações no espaço multidimensional de 2D, ainda mantendo a distruição original, o resultado melhor foi cosine.

- (Dados Originais Mixup Normalizados)

distance_metric	k	test_auc	test_f1	test_accuracy	test_precision_macro	test_recall_macro	test_top_k
euclidean	7	0.954	0.766	0.789	0.795	0.758	0.992
cosine	10	0.959	0.776	0.802	0.812	0.766	0.992

Aqui o desempenho cai, na maioria das métricas, devido ao mantimento dos dados originais, acima, pois após a normalização como criamos dados sintéticos os dados quando normalizados podem sofrer alterações drásticas, com isso o desempenho caiu.

- (Dados Originais Mixup Padronizados)

distance_metric	k	test_auc	test_f1	test_accuracy	test_precision_macro	test_recall_macro	test_top_k
euclidean	13	0.950	0.733	0.758	0.795	0.721	0.995
cosine	14	0.965	0.783	0.807	0.810	0.780	0.995

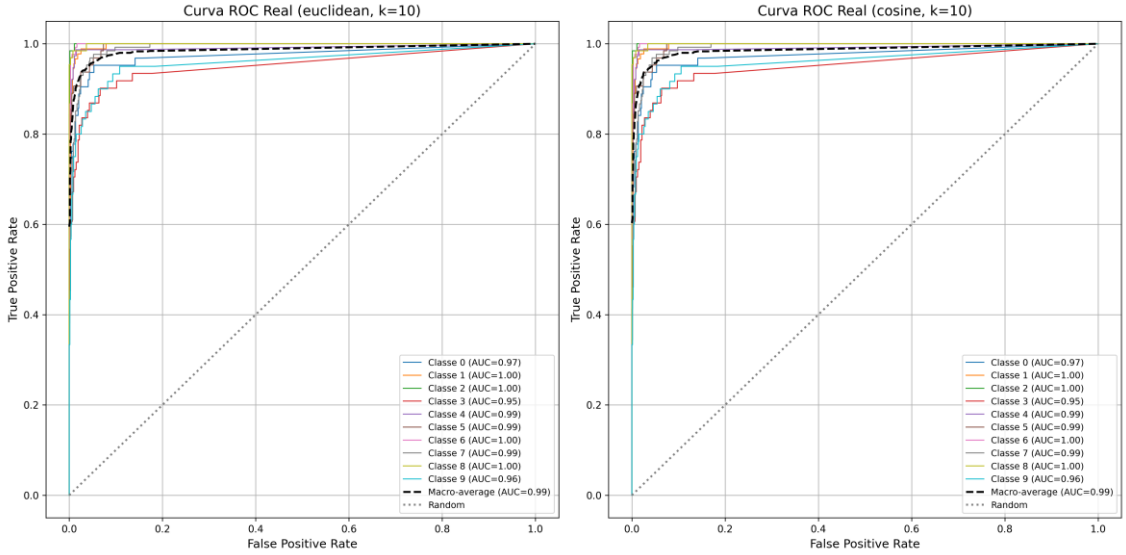
Os resultados são próximos, e piores que a normalização.

2.1 Resultados gráficos – Curva ROC

Com a validação cruzada foi utilizada k=10, para a avaliação, posteriormente colocarei valor de K=1, para avaliação mais comparativa nas conclusões finais.

Resultado 1 - Melhor resultado – Dados Normalizados e balanceados e MIXUP

(Dados Normalizados e balanceados) – K = 10

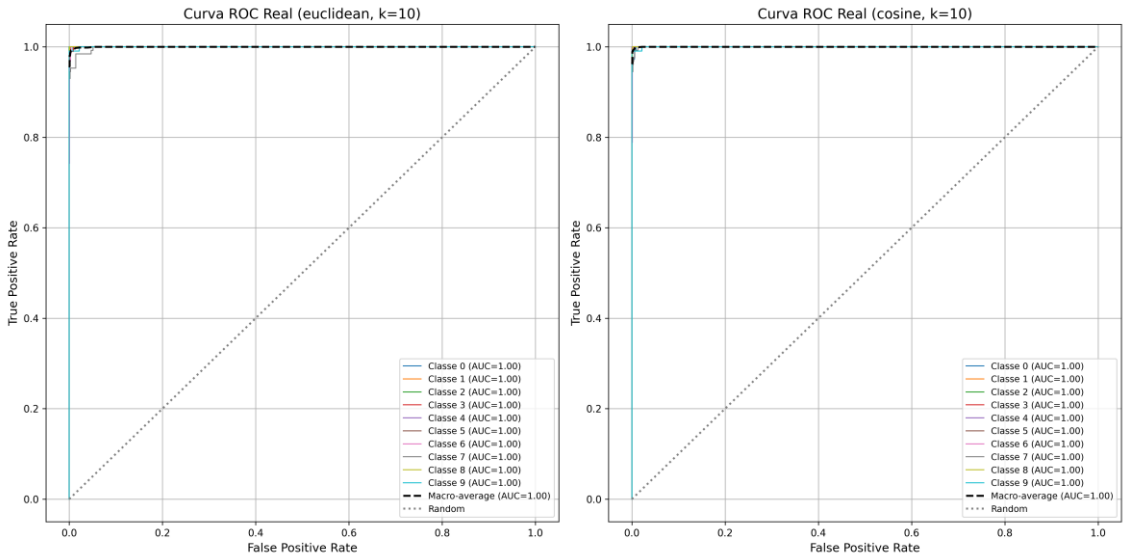


- (Dados Padronizados e balanceados)

Cosine continua levando vantagem, porém a diferença é mínima, para normalização dos dados.

Resultado 3

- Dados Mixup aumentados – K = 10



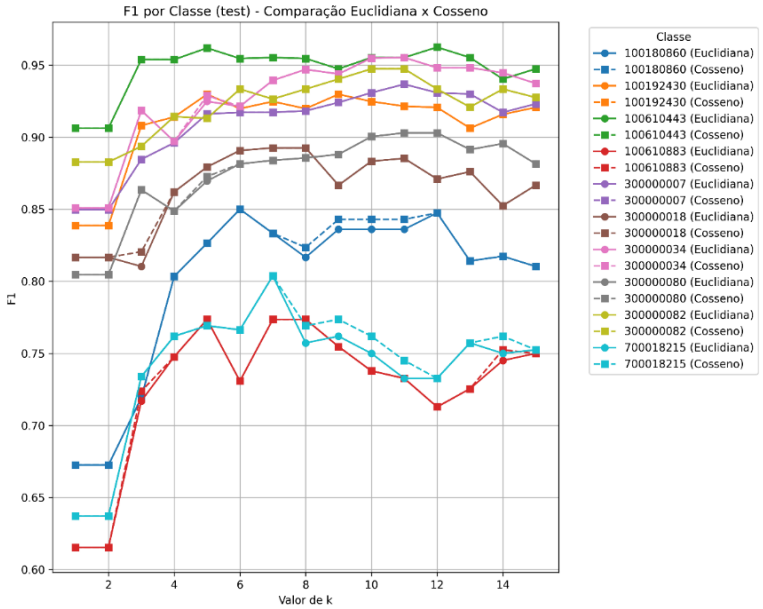
Aqui praticamente não há diferença os dados foram completamente 100% preditos corretamente. Isso é um problema do Mixup de dados, pois influencia em um dado que não se sabe sobre origem (imagem) artificial, os embeddings artificiais podem tomar proporções diferentes.

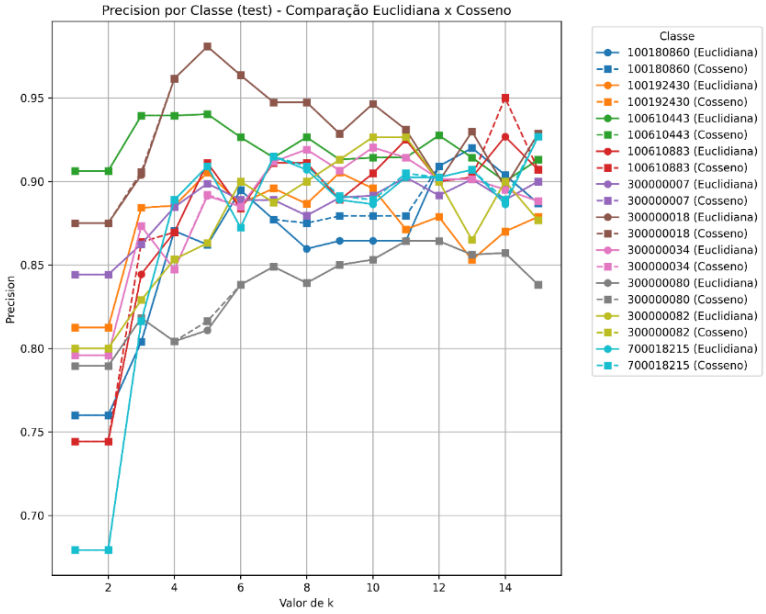
Ao usar normalização ou padronização dos dados, podem manter ainda uma performasse, e manter a distribuição dos dados originais.

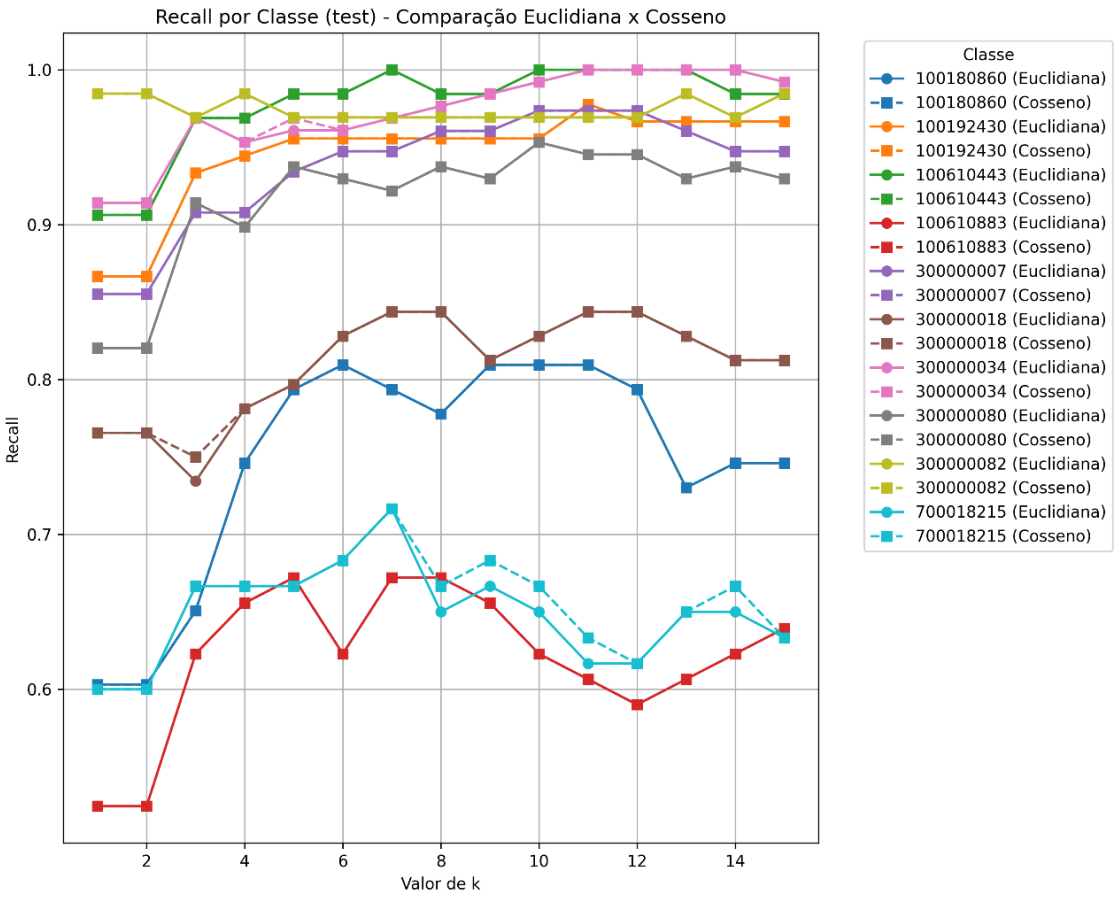
Resultados adicionais – Precision, recall, f1 (Melhores Resultados)

Resultado 1 – (Dados balanceados e normalizados & Mixup original)

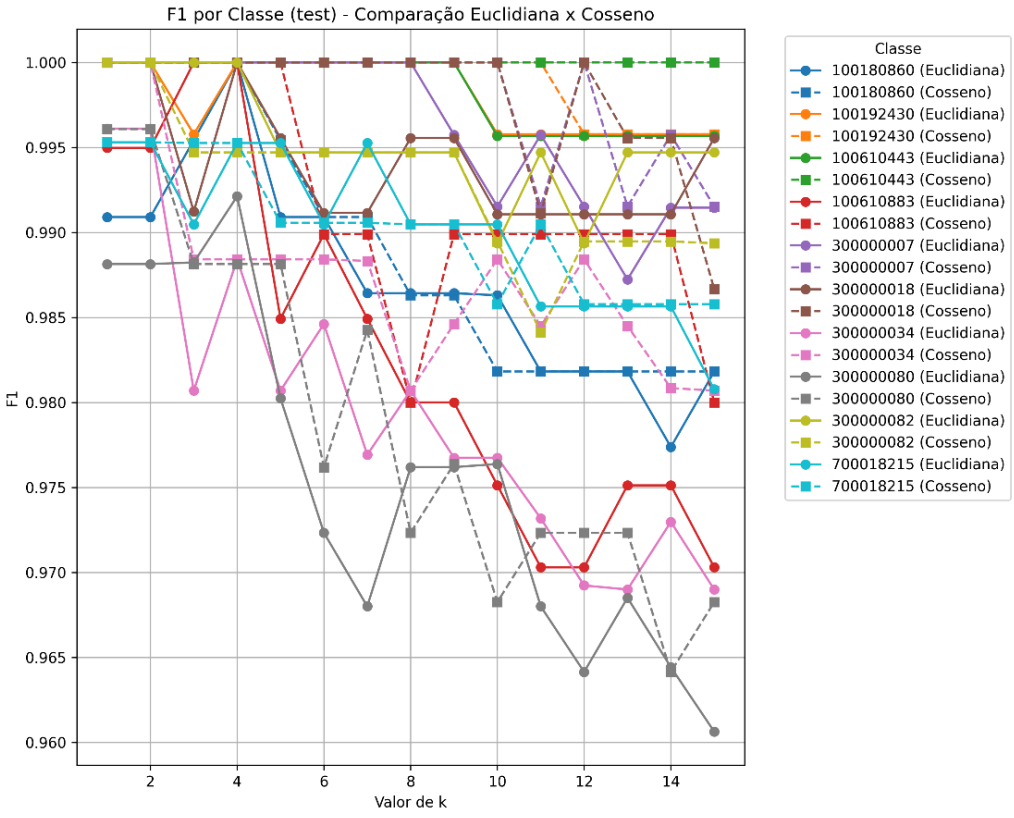
- Dados balanceados normalizados

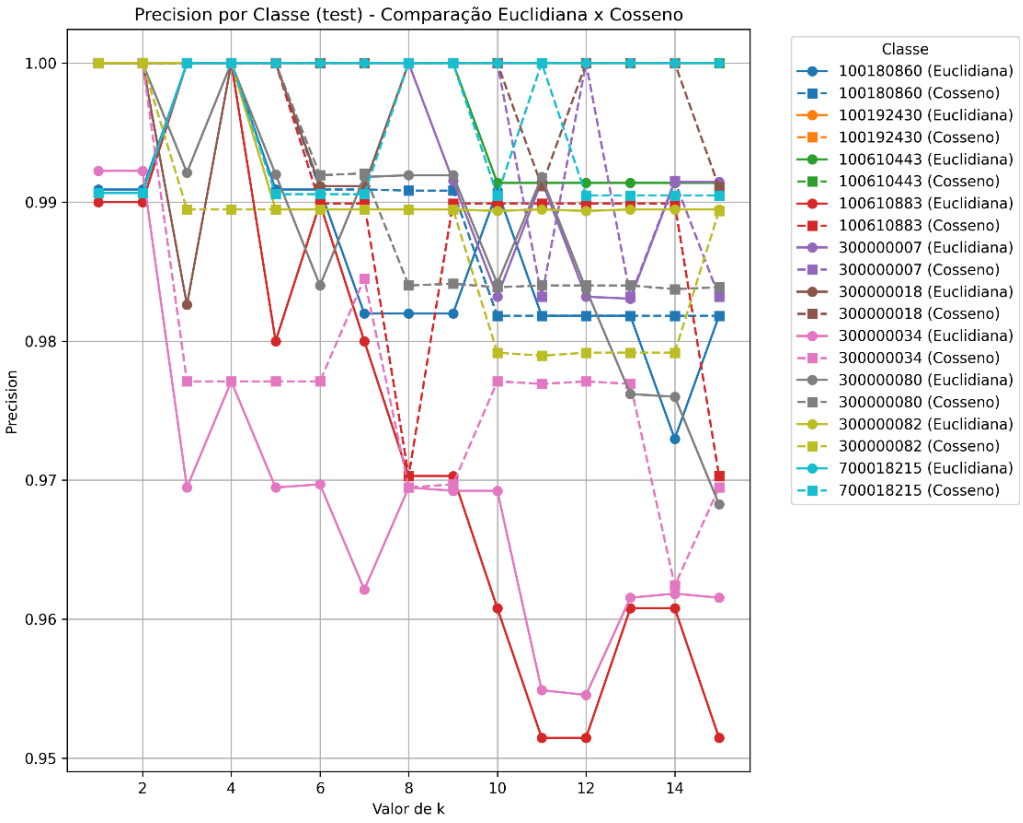


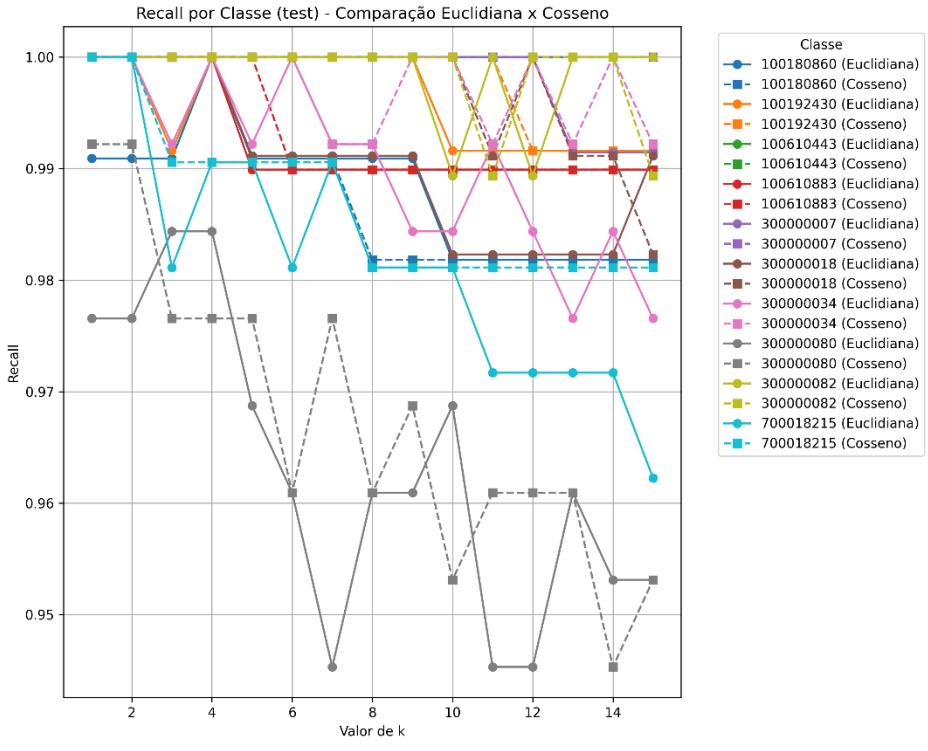




- Dados aumentados MIXUP







Concluindo sobre os resultados, mesmo que não tenha acertado comparando com MIXUP, ou corrigido completamente os clusters com dados originais balanceados e normalizados com L2, este foi o melhor

resultado, porque os demais resultados, com MIXUP mesmo que altos, indicam uma falta e perca da distribuição original dos dados.

Com o balanceamento das bases originais e normalização L2, utilizando Cosine como distância e métrica para o KNN classificador, se conclui que o melhor resultado foi utilizando K=10, com esta configuração, de pre processamento de dados, dados balanceados, normalizados com L2, com K=10.

O mixup, ainda que concatene os dados originais, com os dados artificiais, pode ter problema na questão das imagens, se realmente o embedding_new, for um vetor de características de uma imagem próxima da real. Isto é algo que é complexo ser analisado, e por se tratar de generalização, ambos casos chegaram perto. Então se conclui de forma mais natural, a utilização da base de dados balanceada e normalizada.

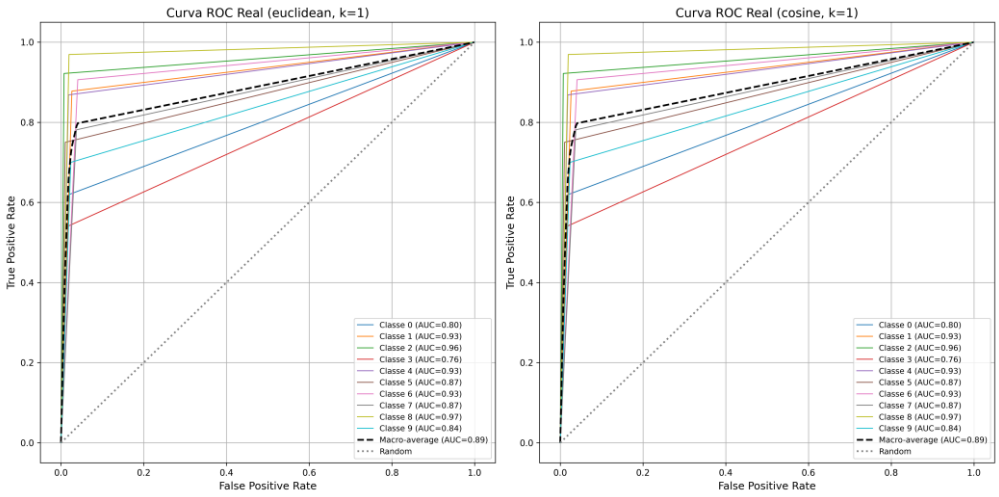
Aplicar MIXUP puramente sem normalizar ou padronizar os dados, não garante que estaremos seguindo a distribuição das características da imagem, portanto é preciso aplicar, isso que foi visto acima não pode ser justificado como melhor escolha, mesmo que os resultados sejam melhores que todos possíveis.

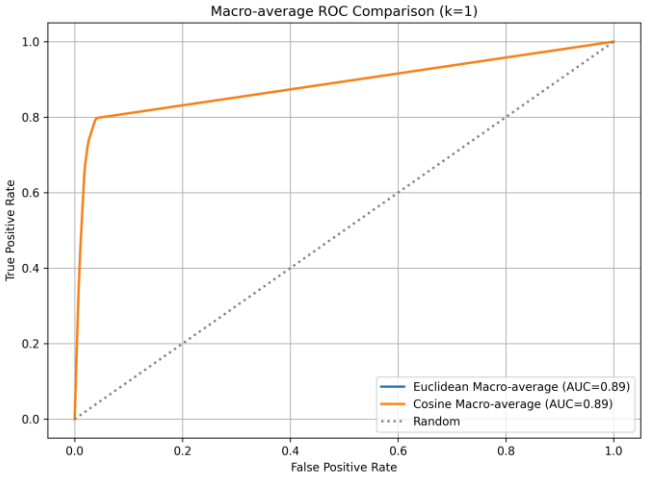
Exemplo do problema de gerar um dado artificial através do embedding. Imagine que eu estou classificando banana e maçã. Pego os embeddings, e ao gerar as características, eu gero os dados artificiais. Pode ser que eu gere uma banana vermelha, uma maçã amarela.

Outro exemplo, eu tenho rostos e quero gerar novos rostos, as características são menores, olhos, nariz, boca, enfim, se eu gero imagens artificiais através dos embeddings sem analisar criteriosamente o que são esses dados, pode ser que eu estou gerando um rosto com olho no lugar da boca, boca no lugar da orelha e assim continua sendo rosto, mas errado!

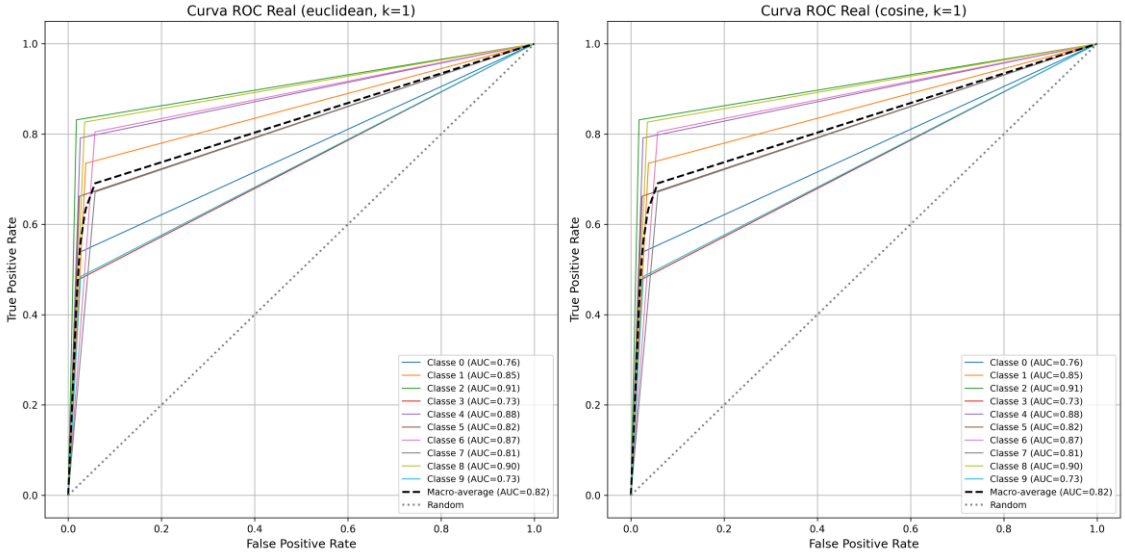
Veja o resultado com K=1. Para melhor comparação.

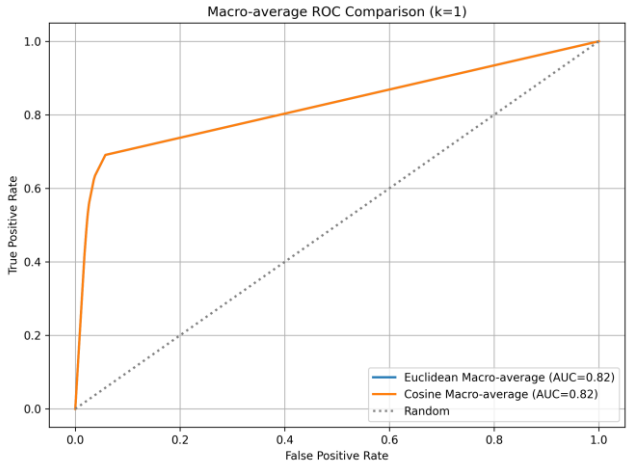
Dados balanceados, com normalização L2.





Dados MIXUP aumentados e (Concatenados + artificiais) normalizados





- Melhores resultados através da análise composta, nos códigos e dos resultados obtidos e análise crítica:

Rank e configuração	Descrição	Composite Score Médio	Métrica de Distância	Melhor k	Composite Score
1 - mixup_balance	Dados Aumentados Mixup - Balanceados (Puro)	1	euclidean	1	1
			cosine	2	1
2 - mixup	Dados Aumentados Mixup (Puro)	0.986	euclidean	2	0.985
			cosine	5	0.987
3 - balance	Dados Balanceados (Puro)	0.909	euclidean	15	0.897
			cosine	13	0.92
4 - norm	Dados Originais - Normalizados (L2)	0.904	euclidean	10	0.903
			cosine	15	0.904
5 - mixup_norm	Dados Aumentados Mixup - Normalizados (L2)	0.904	euclidean	10	0.903
			cosine	15	0.904

É importante notar que a normalização aplicada aqui é aos embeddings extraídos, uma representação de alta-nível das características das imagens, e não à normalização dos pixels. Enquanto a normalização dos pixels é comum para ajustar a intensidade dos valores em imagens, a normalização dos embeddings (L2-normalization) é voltada para garantir que a similaridade seja avaliada com base na direção dos vetores. Essa distinção é crucial porque os embeddings já capturam informações semânticas e a normalização L2 melhora a comparação entre esses vetores, permitindo que a métrica de cosseno tenha melhor desempenho.

O melhor resultado de acordo com todas métricas além desta tabela acima, e conhecimentos sobre o conceito de embeddings, características das imagens de síndromes, que são muito detalhistas, o melhor resultado, configura como os dados balanceados, com DownSampling, com K=15, usando a métrica cosine, com weights='distance' para o KNN. Mas também é considerado importante os dados balanceados e normalizados com L2, K=10, obteve um resultado promissor, mas basta analisar sobre a normalização não afetar as características da imagem, assim como MIXUP, vale verificar também se o aumento com os dados artificiais não afetou as características espaciais, e de canal das imagens reais.

Porque cosine é melhor do que euclidian ?

Foco na Direção, Não na Magnitude. A similaridade de cosseno mede o ângulo entre os vetores, sobre sua magnitude. Se os embeddings forem normalizados (por exemplo, L2-normalizados), o que importa é a direção, ou seja, como os padrões se alinham entre si. Isso pode capturar melhor a similaridade sem ser afetado por diferenças de escala.

Robustez em Espaços de Alta Dimensão. Em espaços de alta dimensionalidade, a distância Euclidiana pode se tornar menos discriminativa, pois as distâncias entre os pontos tendem a se tornar homogêneas. Já o cosseno, focando na orientação, consegue diferenciar melhor os padrões presentes nos embeddings.

Evidências Práticas. Nos resultados que estamos analisando, a configuração utilizando a métrica de cosseno apresentou scores (como AUC, F1, top-k accuracy, etc.) superiores, indicando que ela oferece uma melhor separação das classes para os dados que temos.

Embora a técnica de Mixup tenha potencial para enriquecer o conjunto de dados, ela gera embeddings artificiais através da interpolação de vetores. Isso pode resultar em alguns exemplos anômalos ou menos representativos. Na prática, a diferença entre os dados aumentados via Mixup e os dados originais balanceados não é tão grande; os dados originais balanceados parecem oferecer uma representação mais generalista e robusta das classes.

Portanto, é fundamental realizar uma avaliação qualitativa das imagens que originaram esses embeddings artificiais para confirmar se os novos exemplos gerados pelo Mixup realmente capturam a variabilidade e as características das classes de forma consistente. Essa avaliação ajuda a garantir que a técnica de augmentação não introduza ruído ou distorções que possam prejudicar o desempenho do classificador.

Dados Balanceados e originais e Normalizados (L2): Essa combinação apresentou um excelente desempenho, demonstrando que, ao balancear as classes e normalizar os embeddings, ou utilizar os embeddings originais, o classificador consegue capturar de forma precisa as diferenças semânticas entre as síndromes.

Métrica de Distância: A métrica de cosseno se mostrou superior, pois foca na orientação dos vetores, crucial quando os dados estão L2-normalizados.

5 Desafios e Soluções

Eu encontrei o maior desfio, no preprocessamento dos dados, pois são imagens de síndromes com características e detalhes sucintos, o que pode qualquer alteração acarretar em um grande problema nos dados.

Portanto investiguei, com grok, chatgpt, deepseek, copilot, google, artigos acadêmicos, entre outras fontes, para poder entender soluções que não fazem tanta modificação nos dados, mesmo assim, elas me retornaram algumas coisas alucinadas, não houve uma interpretação tão correta, portanto precisei testar uma a uma, das soluções, que fui analisando, como pre processamento dos dados, tive uma ideia que foi o balanceamento dos dados de acordo com centroid dos dados, e não apenas retirar as amostras que estão nas classes com maior quandidade de dados, assim eu retiro as amostras que estão com outilier para ter uma maior precisão na tarefa de classificação.

Outro insight era saber qual melhor valor de K não sabendo sobre a síndrome e o que cada classe representa em si, para saber sobre VP, FP, VN, FN, então analisei de acordo com métrica composta de dimensões, com 3 dimensões, fazendo uma raiz cúbica dos dados, e avaliando o melhor resultado.

Outro problema é a necessidade de testar empiracamente tudo, agora é necessário aplicar boas práticas no código main, para usar argumentos para as funções, e caso queira aquele resultado você coloca o argumento "—argumento valor", porque, está complexo, para fazer isso nesse pouco tempo.

Então resolvi fazer a execução completa de tudo que foi utilizado de resultado aqui, na execução do código main.

4 Ideais "Insights", conclusões e recomendações.

A distância do cosseno apresentou melhor desempenho para embeddings normalizados.

A técnica de Mixup ajudou a melhorar a generalização do modelo, porém ocorreu de causar danos na distribuição dos dados, ou seja, não mantem a distribuição original dos dados, acrescentando amostras artificiais, elas tomam conta da classificação o que perde a generalização dos dados, e fica estritamente complicado atribuir.

Mixup é uma técnica muito boa, porém se pudesse ter os dados originais, os dados das imagens, pois com apenas os dados sintéticos, podem ocasionar problemas na classificação. Isto é dito caso tenha um pipe line de dados de imagens que se tornam embeddings e posteriormente são classificados.

No processo de isto ocorrer vão ter amostras muito fora do cluster. E este é o problema da generalização com Mixup, mesmo que seja bom, não estamos lidando com as imagens reais e usando data agumentation padrão de imagens ou GANS, para ter imagens artificiais, são dados 320 Dimensões. Que podem sim ter um resultado bom, isto é válido.

Porém se torna estritamente necessário ter as imagens originais e gerar os embeddings para teste e colocar no pipeline de teste para validar se realmente é melhor do que dados balanceados e normalizados com L2 apenas.

O balanceamento dos dados teve impacto positivo nas métricas de precisão e F1-Score.

Este estudo demonstrou a eficácia da aplicação de técnicas de balanceamento e diferentes métricas de distância na classificação de embeddings. Como próximos passos, sugere-se:

Testar arquiteturas mais complexas, como redes neurais **siamesas**. Até mesmo testar auto ML (autogluon, pycaret, autokeras).

Obter as imagens reais, e balancear os dados originais. Obter as imagens originais e aumentar o dataset, com imagens artificiais (GANS, pix2pix, pathology gan) e data agumentation.

Explorar técnicas de embedding supervisionadas para melhorar a separação das classes.

Relatório elaborado por Hyago Vieira Lemes Barbosa Silva