Documentação

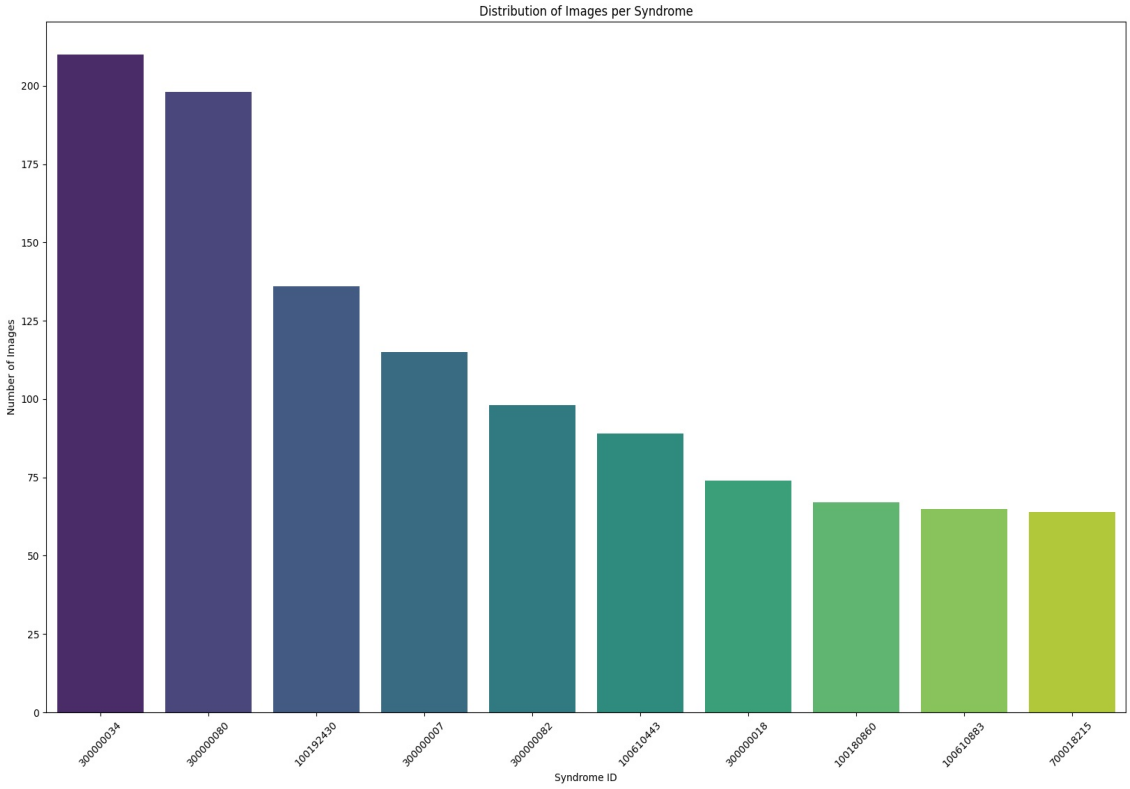
Esse é um arquivo feito especialmente para documentar meus pensamentos sobre o código e análises dos exercícios numerados 1 ao 5.

1. Processamento de Dados

Análise e Discussão do Gráfico de Imagem por Síndrome

De acordo com o gráfico gerado em imagens por síndrome, é possível notar que de fato existem duas síndromes que possuem muito mais imagens que o resto das outras do dataset.

É claro que os dados poderiam estar melhor balanceados, mas não acredito que isso cause um problema tão grande, visto que os dois são os únicos dentre os 8 que estão mais normalizados.



2. Visualização de Dados

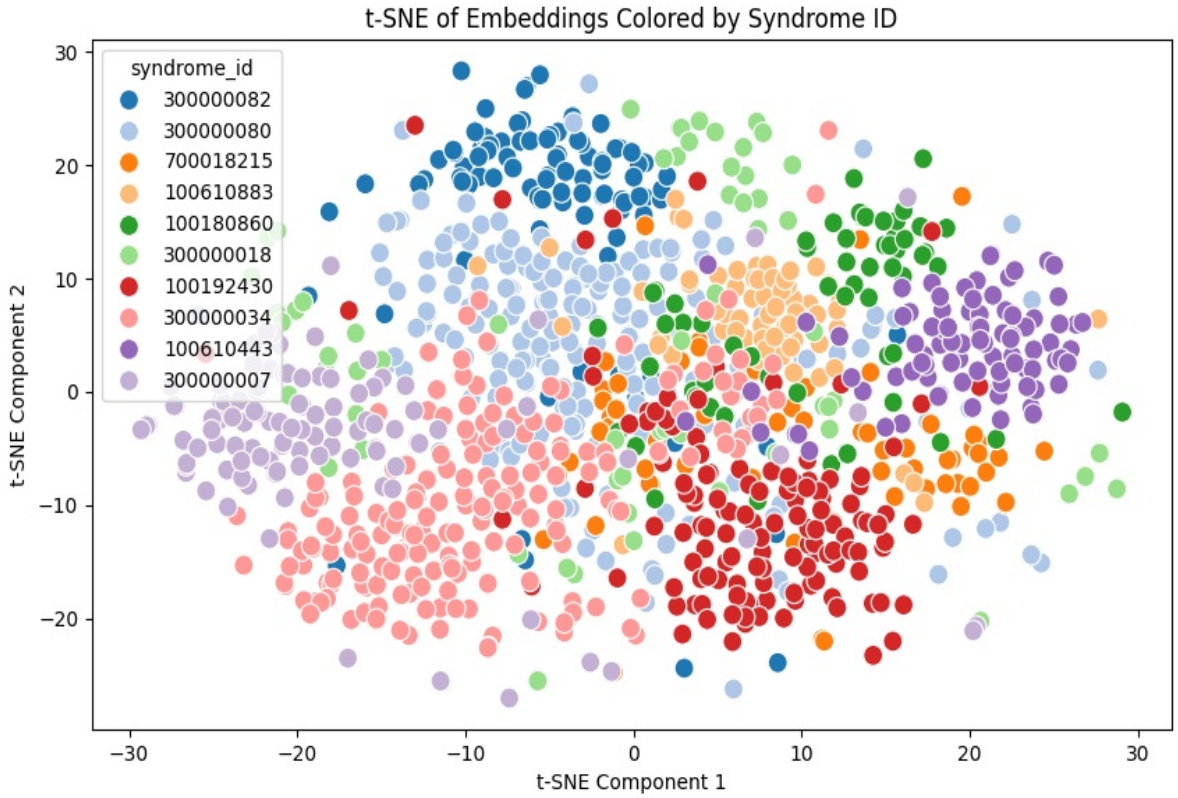
Análise e Discussão do Gráfico de dispersão t-Sne

É possível observar que as imagens da mesma síndrome tendem a ficar agrupadas próximas umas das outras. No entanto, existem outliers em todas as síndromes. Algumas imagens específicas estão posicionadas no lado oposto do gráfico, o que pode sugerir uma possível similaridade entre síndromes. Acredito que síndromes com prefixos semelhantes possam também ser mais próximas em termos de embeddings. Por exemplo, as síndromes 300000007 e 300000018 podem ser variações de uma mesma síndrome.

Uma evidência que poderia apoiar essa teoria é o fato de que os clusters de prefixos similares estão adjacentes entre si. No entanto, essa observação não é conclusiva, pois não há contexto suficiente sobre os dados para garantir essa relação.

O cluster mais caótico é o relacionado à síndrome 700018215, que coincidentemente é aquele que possui uma das menores presenças no dataset.

No geral, o gráfico t-SNE conseguiu realizar a clusterização de forma eficaz, alocando a maioria dos dados corretamente em seus devidos clusters.



3. Classificação

Comparando os Modelos (Métricas)

De acordo com as métricas geradas após o treinamento dos classificadores, é possível notar que, independentemente da métrica avaliada, o KNN utilizando a métrica cosseno é melhor que a euclidiana. Afinal, o AUC score de ambos é efetivamente equivalente, ambos possuem Top-k Accuracy de 1.0, e a única métrica avaliada onde ambos divergem é no f1-score, onde o cosseno se mostra superior por pouco, mas ainda se destaca.

Metric	AUC	F1-Score	Top-K Accuracy
Euclidean	0.9452	0.7099	1.0
Cosine	0.9447	0.7584	1.0

Discussão

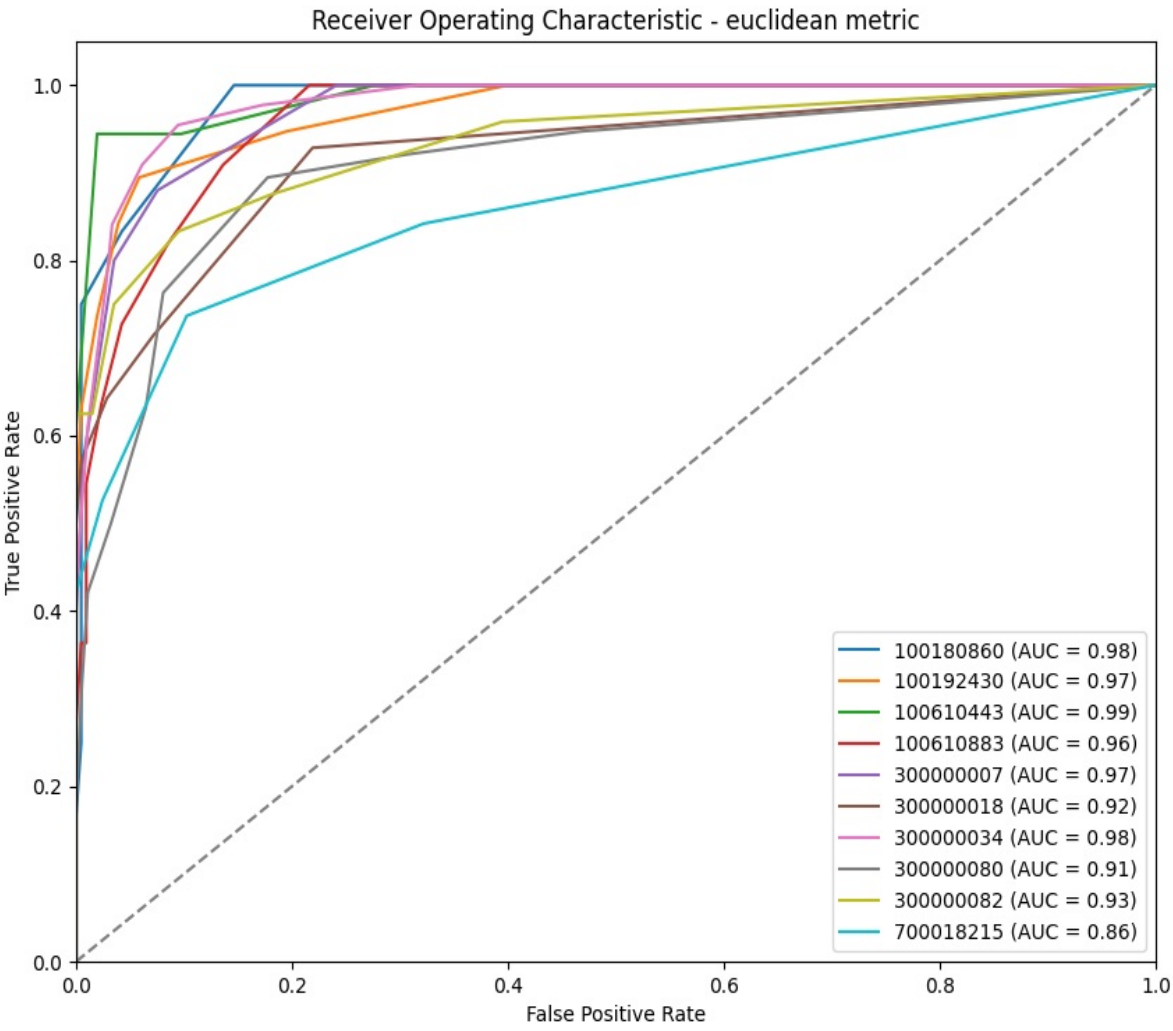
Pelo que aprendi ao realizar o exercício, a explicação para o KNN cosseno ser melhor que o euclidiano é por conta do dataset ser mais disperso, e ele realiza aproximação utilizando o ângulo em vez da distância absoluta. Acredito que isso permitiu que ele possivelmente classificasse melhor os outliers que foram observados nos gráficos de t-SNE.

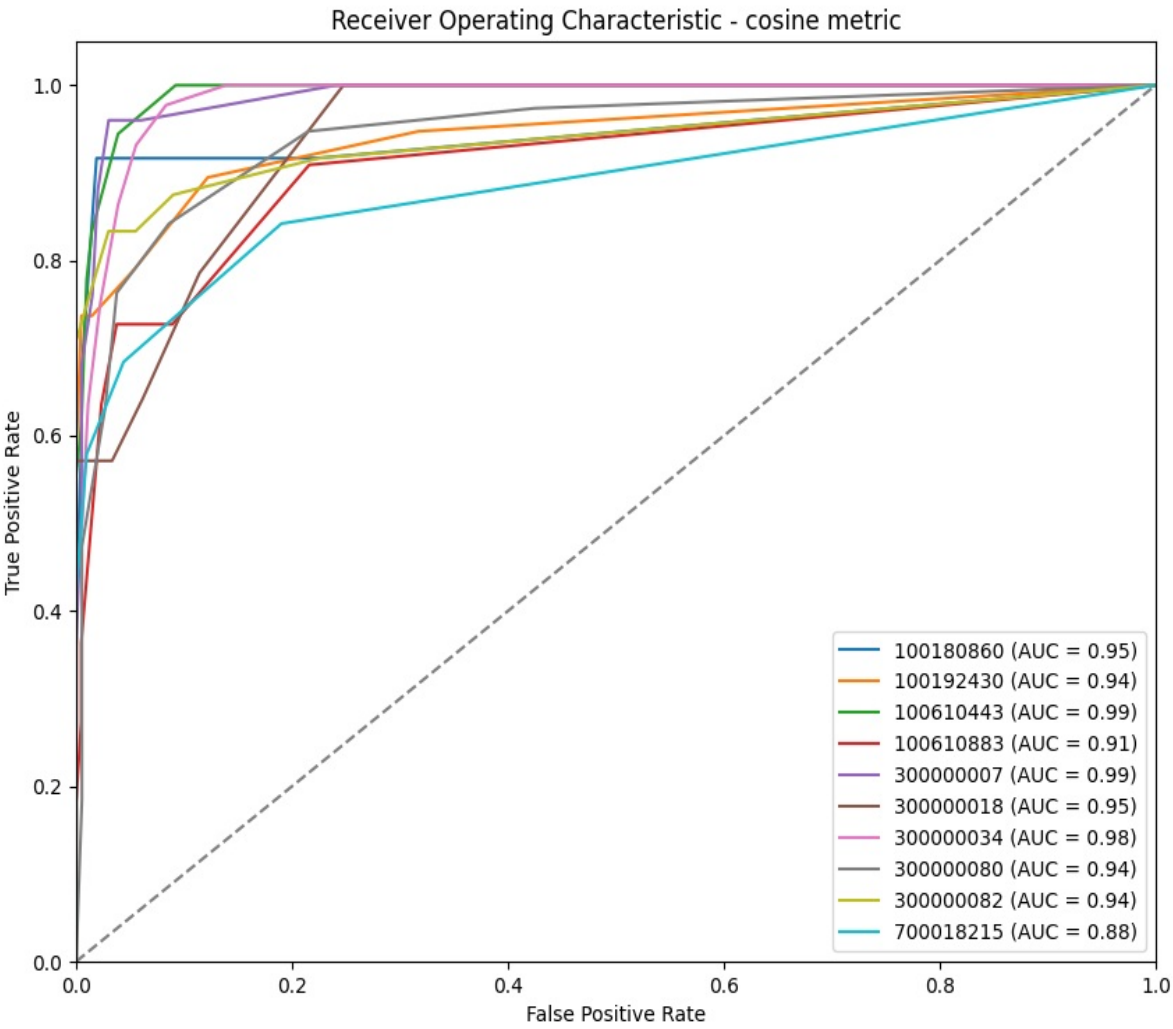
4. Métricas

Gráficos

Ao analisar ambos os gráficos, é possível reforçar a análise da terceira seção. É possível identificar que, apesar do modelo baseado em cosseno ser melhor, ele tem um desempenho ligeiramente inferior em algumas classes individuais. No entanto, essa diferença é insignificante, pois o algoritmo euclidiano continua sendo levemente inferior nas outras classes.

Dito isso, a escolha do melhor modelo deve ser baseada na métrica mais importante, considerando o objetivo da tarefa. Se o foco for classificar corretamente as síndromes, sem dúvidas o cosseno é a melhor escolha, mesmo que por uma margem pequena. Mas se a prioridade for classificar corretamente uma classe específica, então, dependendo da classe, o euclidiano pode ser a melhor opção.


Receiver Operating Characteristic - euclidean metric
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
100180860 (AUC = 0.98)
100192430 (AUC = 0.97)
100610443 (AUC = 0.99)
100610883 (AUC = 0.96)
300000007 (AUC = 0.97)
300000018 (AUC = 0.92)
300000034 (AUC = 0.98)
300000080 (AUC = 0.91)
300000082 (AUC = 0.93)
700018215 (AUC = 0.86)



5. Relatório

Metodologia

Passos

Durante todo o projeto, meu objetivo foi não apenas completar o exercício, mas também aprender. Assim, nos primeiros quatro dias, me dediquei a aprender e relembrar conceitos e práticas por meio de videoaulas no YouTube. Afinal, fazia algum tempo que eu não trabalhava com código envolvendo Machine Learning. Como meu método preferido de aprendizado é o Project-Based Learning, passei esses dias criando um projeto para relembrar os conceitos e depois o adaptei para seguir as diretrizes do teste.

A maneira que utilizei para verificar se tudo estava funcionando corretamente foi escrever o código aos poucos em um Jupyter Notebook e, em seguida, adaptá-lo para um script Python comum.

Caso encontrasse erros, consultava a documentação da biblioteca, procurava vídeos específicos no YouTube, buscava por soluções no Stack Overflow e, como última alternativa, recorria à IA generativa conforme permitido nas diretrizes do teste.

Repetindo esse padrão, eventualmente consegui completar o teste.

Preprocessamento de dados e algoritmos

A escolha dos algoritmos para o pré-processamento de dados foi inicialmente motivada por um vídeo que assisti sobre t-SNE. O primeiro passo foi a aplicação de um imputer, para lidar com valores ausentes sem recorrer à simples remoção dos dados. Em seguida, utilizei o StandardScaler, pois, após pesquisas, verifiquei que a normalização dos dados poderia facilitar os cálculos do t-SNE e melhorar a análise.

Escolha de parâmetros

Para a otimização dos hiperparâmetros, utilizei a função GridSearch, focando na busca pelo melhor valor de K para cada métrica do algoritmo KNN. Optei por ignorar outros parâmetros do KNN, pois considerei que não seriam essenciais para o exercício. Além disso, devido ao feriado de

Carnaval e à incerteza sobre uma possível resposta por e-mail, decidi prosseguir sem essa confirmação.

Resultados e Análise

Os resultados detalhados estão descritos ao longo do texto, mas, em resumo, a métrica do cosseno apresentou o melhor desempenho geral, embora por uma margem pequena. Já a distância euclidiana mostrou-se mais eficaz para classificar uma síndrome específica com base nos gráficos. Assim, se a prioridade for essa síndrome em particular, a euclidiana se torna a melhor opção.

Desafios e Soluções

Já descrevi o processo adotado, então aqui focarei nos principais desafios que enfrentei, sem uma ordem específica.

1 - Achatamento de Dados

Inicialmente, considerei remover o subject_id, pois o achei irrelevante para a análise. A ideia era simplificar a estrutura dos dados, algo como:

```
{
  "syndrome_id": [
    "image_id": ["320dim Array"]
  ]
}
```

No entanto, logo percebi que essa abordagem não mudaria nada. Após ler um artigo específico e assistir a um vídeo sobre o assunto, compreendi melhor o que deveria ser feito e ajustei minha estratégia de forma mais eficiente.

2 - T-sne

Antes desse trabalho, eu nunca tinha ouvido falar do algoritmo t-SNE. Ele me lembrou um pouco do K-means, mas minha dificuldade estava justamente na falta de conhecimento sobre o t-SNE. Felizmente, após assistir a um vídeo teórico seguido de um tutorial prático, fui capaz de entender e aplicar o algoritmo corretamente.

3 - ROC Curve

A implementação de uma Curva ROC foi uma novidade para mim. Encontrei dificuldades com o conjunto de dados por conta disso, principalmente ao tentar adaptar para o tratamento de múltiplas classes. Durante o desenvolvimento, descobri que era necessário realizar a binarização das classes para aplicar a curva corretamente. Foi um processo de tentativa e erro até conseguir implementar a solução de forma adequada, mas os artigos e vídeos que consultei foram essenciais para superar essa dificuldade.

Recomendações

1 - Tipagem

Acredito que a principal dificuldade que encontrei ao realizar a tarefa foi a ausência de tipagem em certos pontos. Sou um grande defensor da tipagem forte, pois ela ajuda a identificar e corrigir erros antes que se tornem problemas maiores. Dedicar um tempo para melhorar o projeto nesse aspecto pode ser altamente benéfico para a manutenibilidade a longo prazo.

2 - Arquitetura

Do ponto de vista arquitetural, acredito que a pipeline foi bem estruturada, com as responsabilidades dos componentes devidamente separadas. No entanto, vejo espaço para melhorias, especialmente ao adotar melhores práticas de padrões de projeto, como o SOLID. A Inversão de Dependências, por exemplo, poderia tornar o projeto mais robusto, permitindo uma maior flexibilidade caso alguma biblioteca precise ser alterada no futuro. Além disso, a modularidade do projeto seria significativamente aprimorada. E vale lembrar que isso é apenas um dos princípios do SOLID.

Outra recomendação seria a adição de um arquivo .env ou config.yaml para gerenciar os file_paths e outras configs que são utilizados nos arquivos, o que traria mais flexibilidade e organização para o projeto.

3 - Continuação da Pipeline

O projeto pode ser aprimorado ao completar a pipeline com etapas voltadas para o deploy. Por exemplo, um arquivo de configuração Terraform poderia ser utilizado para integrar o modelo com um serviço na AWS. Outra sugestão seria a implementação de uma API simples usando Serverless ou EC2, ou até mesmo o Firebase. Com a dockerização do ambiente, todo esse processo se tornaria muito mais ágil e eficiente.

6. Recursos

Esta seção é dedicada a mostrar os recursos que utilizei para a realização do teste.

De maneira geral, os sites que mais utilizei foram:

- GeeksforGeeks
- Scikit-Learn
- YouTube
- StackOverflow

- Para dúvidas específicas ChatGPT

Um exemplo de dúvida específica que consultei foi: “Por que a Curva ROC necessita de tratamento para o multiclasse?” Esse tipo de questão não encontrei facilmente em minhas pesquisas. Minhas dúvidas eram focadas principalmente em garantir o meu entendimento do que estava produzindo, e não em resolver problemas de código diretamente.

A IA generativa também foi utilizada para identar alguns arquivos, já que a extensão do VSCode não estava funcionando por algum motivo. Foi também utilizada para corrigir erros ortográficos em logs.

Abaixo, segue uma lista de alguns recursos específicos de cada site que utilizei:

- Machine Learning Pipeline In Python | How to run pipeline in python machine learning
- Playlist - End To End Data Science Playlist - Get Prepared With Industry Ready Projects
- Article - Flatten a List of Lists in Python
- Video - Data Cleaning in Pandas | Python Pandas Tutorials
- Video - Creating Visualizations using Pandas Library | Python Pandas Tutorials
- Video - Exploratory Data Analysis in Pandas | Python Pandas Tutorials
- Video - StatQuest: t-SNE, Clearly Explained
- Docs - KNeighborsClassifier
- Video - Matplotlib Crash Course
- Video - Seaborn Crash Course
- Video - Machine Learning Tutorial Python - 18: K nearest neighbors classification with python code
- Video - Cosine Distance vs Euclidean Distance in Machine Learning and NLP with Word2Vec or Glove Vectors
- Article - How to choose the right distance metric in KNN?
- Article - Cosine Similarity
- Article - Euclidean Distance
- Video - ROC AUC | Machine Learning with Scikit-Learn Python
- Article - AUC ROC Curve in Machine Learning

7. Considerações Finais

Gostaria de expressar minha sincera gratidão ao avaliador por dedicar seu tempo à leitura, análise e avaliação deste trabalho. Reconheço que, apesar de ser parte de suas responsabilidades, a avaliação de um projeto exige atenção e esforço, e por isso valorizo muito ao menos a oportunidade de ser avaliado. Ao longo da última semana, dediquei-me intensamente para concluir este projeto, buscando não apenas cumprir as exigências, mas também aprofundar meu entendimento sobre os conceitos abordados.

Independentemente do retorno que receberei, estou convencido de que a jornada de desenvolvimento deste trabalho foi valiosa. Se o feedback for positivo, ficará a sensação de que meu esforço foi recompensado. Caso contrário, ficarei grato pela oportunidade de aprender com os pontos que possam ser melhorados. Achei o projeto desafiador, no bom sentido.

Atenciosamente, Guilherme Pereira Schneidt