UNIVERSIDADE FEDERAL DE MINAS GERAIS
Nome: João Marcos Ribeiro Tolentino

Organização Geral: O código está bem organizado conforme o esperado com as funções bem separadas e o código, identado, bem limpo e comentado.
Comentários: Os comentários são sucintos, mas fornecem uma breve explicação sobre o que o código está fazendo. Nesse sentido, o objetivo foi explicar mais as partes que não são auto explicativas. Do mesmo modo, não foi muito detalhado outras para não se criar um código sujo desnecessariamente.
Uso de Bibliotecas: numpy, pandas, dateutil, pytz e six são as bibliotecas permitidas como descrito na atividade, porém só foi utilizadas as seguintes: numpy e pandas.
Clareza: O código é relativamente claro em sua funcionalidade, mas é necessário um certo entendimento em relação aos loops mais aninhados, mas não é nada demais para um desenvolvedor em geral mesmo que iniciante. Ou seja, é um código de fácil compreensão caso o leitor tenha pelo menos um entendimento básico de desenvolvimento de software.
Variáveis Globalmente Definidas: Variáveis como: “user_factors”, “item_factors”, e outras são definidas globalmente. É recomendável passar essas variáveis como argumentos para funções em vez de tê-las globalmente, para melhorar a modularidade e evitar efeitos colaterais inesperados.
Comentários de Código: Foi incluído alguns comentários ao longo do código, principalmente para dar nomes às variáveis. Isso é útil para entender o que cada variável representa e o que foi feito.
Comentários de Função: A função main() não tem um comentário de função que explique sua finalidade e o que ela faz. Foi adicionado também um breve comentário de função para fornecer uma visão geral do objetivo e cada função na main.
Fluxo do Programa:

-Carregamento de Dados: Carrega os dados de classificação a partir da base de dados “rating” e para fazer previsão são carregados o arquivo “targets”.

- Pré-processamento de Dados: É calculado a média das classificações no conjunto de dados de classificação, extraídas as colunas relevantes e os dados são armazenados em listas separadas.

Modularidade: O código foi dividido em várias funções, cada uma com uma responsabilidade específica, como carregar os dados, pré-processar os dados, treinar o modelo, fazer previsões e salvar resultados. Isso torna o código mais fácil de entender e manter. Observe como o código foi separado em 5 funções:

```python
def main():
    ratings_df, targets_df = load_data()
    ratings_matrix, user_index, item_index, mean_rating = preprocess_data(ratings_df)
    user_factors, item_factors = train_model(ratings_df, ratings_matrix, user_index, item_index, mean_rating
    predictions_df = make_predictions(user_factors, item_factors, user_index, item_index, targets_df)
    save_predictions(predictions_df)
```

Uso de Funções: As ações específicas são encapsuladas em funções, o que facilita a reutilização e o teste de cada parte do código separadamente.
Uso de Parâmetros: Hiperparâmetros do modelo, como o número de fatores, taxa de aprendizado e regularização, são definidos de forma explícita e podem ser facilmente modificados como argumentos das funções. Nesse sentido, os valores foram testados várias vezes para: num_factors, learning_rate, regularization e num_epochs para uma melhor filtragem colaborativa baseado em fatores latentes e uma otimização do modelo, além de evitar o overfitting.
Tratamento de Viés: Foi adicionada uma correção de viés ao cálculo das previsões, usando a média dos ratings, o que pode melhorar o desempenho do modelo (Isso melhorou bastante a minha pontuação no Kaggle).
Estruturação Geral: O código está estruturado de forma a seguir um fluxo mais lógico, desde o carregamento dos dados até a geração das previsões e o salvamento dos resultados.

Em geral, o código executa a tarefa de treinamento de um modelo de recomendação e geração de previsões com base nos dados de entrada. No entanto, para melhorar a qualidade e manutenção do código foi feita essa documentação para tornar o código mais acessível para outros leitores e desenvolvedores.

ESTRUTURA DE DADOS E COMPLEXIDADE COMPUTACIONAL

Concluindo a complexidade computacional do código é dominada pelo treinamento do modelo, que envolve loops aninhados sobre as classificações e as épocas. Portanto, a complexidade global é principalmente quadrática em relação ao número de classificações e linear em relação ao número de épocas. Assim, para leitura de dados, pré-processamento e criação de previsões: têm complexidade linear.

Em relação aos resultados alcançados aos se criar o código inicialmente tive uma pontuação em torno de: **1.33.** Dessa forma, foram testadas várias alterações nos hiperparâmetros definidos que acabaram dando um resultados piores. Só que o que melhorou a pontuação para **1.26** foi calcular a previsão com a correção de viés ou seja tirando a diferença entre a predição e a média da base de dados.