File structure and explanations in the test task:

<u>1) EDA.ipynb</u>

I investigated the dataset in the notebook in EDA.ipynb. From EDA I highlighted that the dataset has no missing values, ensuring data quality. I observed that **Retail_Price** and **Units_Sold** follow a relatively normal distribution without any significant outliers. The categorical variable Product_Category shows a balanced distribution across categories, with "Health" as the most frequent category, followed closely by "Clothing," "Electronics," "Sports," and "Home & Garden." This balance in Product_Category is confirmed by the pie chart, indicating no significant imbalance.

Correlation analysis revealed that there are no significant relationships between Product_ID, Retail_Price, and Units_Sold, indicating their independence. Segment analysis provided further insights: the bar plots show that units sold are fairly evenly distributed across product categories, materials, and colors, with slight variations. Seasonal products have slightly higher total units sold compared to non-seasonal products. Additionally, materials such as "Metal" and colors like "Blue" show a marginally higher sales volume. These insights can guide marketing and inventory strategies by focusing on high-performing segments.

2) **<u>Part 1 & Part 2 (Part 1, Part 2.ipynb)</u>**

<u>Part 1:</u>

I loaded the dataset, replaced infinite values with NaNs, and filled NaNs with zeros. Log transformation was applied to 'Units_Sold'. Categorical variables were encoded, and new features were engineered, including 'Revenue' and polynomial features to capture interactions.

I chose a **RandomForestRegressor** for its ability to handle various data types and capture complex interactions. The model was trained on the training set and used to predict 'Retail_Price' on the test set.

I achieved these results: **Mean Absolute Error (MAE)**: 0.0036, **Mean Squared Error (MSE)**: 0.0018, **R² Score**: 0.9999. These evaluation metrics demonstrate exceptionally high prediction accuracy, indicating the model's effectiveness.

To further improve the model, I am going to experiment with the following approaches: **Feature Selection**, **Hyperparameter Tuning** (Grid Search or Random Search), **Additional Features**, **Advanced Models** (Experiment with more sophisticated algorithms like Gradient Boosting Machines (GBM), XGBoost, or Neural Networks for potentially better performance), **Cross-Validation**.

Implementing these improvements should further increase the model's accuracy and robustness, making it more reliable for predicting 'Retail_Price' across different datasets.

<u>Part 2:</u>

A **RandomForestClassifier** was chosen for initial model building. To optimize the model, I performed hyperparameter tuning using RandomizedSearchCV. Recursive Feature Elimination (RFE) was then applied to select the top 20 features. A GradientBoostingClassifier was trained on these selected features to predict 'Product_Category'.

The overall accuracy of the model is **60%**, indicating moderate performance with room for improvement. Precision and recall values vary across categories, with 'Home & Garden' achieving the highest precision (0.81) but lower recall (0.54), suggesting many instances are missed despite high prediction accuracy. 'Clothing' and 'Electronics' show balanced but moderate precision and recall, while 'Health' has slightly better recall (0.64) but similar precision (0.53). The F1-scores for these categories reflect this balance, hovering around 0.58-0.60, indicating moderate overall performance. 'Sports' category shows a better precision (0.68) but still struggles with recall (0.56).

To improve the model, additional feature engineering can be implemented, introducing domain-specific features and interaction terms to capture more complex relationships. Hyperparameter tuning can be refined using Grid Search or Bayesian Optimization for more precise adjustments. Data augmentation techniques, such as generating synthetic samples or balancing the dataset, can help improve model training.

I am going to experiment with advanced algorithms like XGBoost or ensemble methods that can capture more intricate patterns and improve performance.

3) **Part 3 (PART 3.ipynb)**

I started by loading and preprocessing the product sales dataset, focusing on the **Retail_Price, Seasonal**, and **Units_Sold** columns. The Seasonal column was encoded using LabelEncoder, and both Retail_Price and Units_Sold were normalized with MinMaxScaler. I built a more complex generator and discriminator for my GAN, adding additional layers and features like **LeakyReLU, BatchNormalization, and Dropout** to improve learning and prevent overfitting. The models were compiled using the Adam optimizer with adjusted learning rates.

During the initial training, I observed that the discriminator accuracy was around **10%**, indicating poor performance. To address this, I ensured that the discriminator was correctly set to be trainable only when needed, and I adjusted the architecture by adding more layers and regularization techniques. By the end of the improved training process, the discriminator accuracy increased significantly to around **74%**, showing that it could effectively distinguish between real and fake data. I monitored the training progress by printing losses and accuracy periodically. I monitored the training progress by printing losses and accuracy periodically.

After training, I generated synthetic Units_Sold values using the trained generator and denormalized these values to convert them back to their original scale. I combined these synthetic values with the original features to create a complete synthetic dataset. By enhancing model complexity and fine-tuning hyperparameters, I aimed to produce higher-quality synthetic data. Regular updates on training metrics allowed me to track progress and make necessary adjustments. The final synthetic dataset was analyzed and compared to the real data to ensure it closely resembled the original distribution and patterns.

After training, I generated synthetic Units_Sold values using the trained generator and denormalized these values to convert them back to their original scale. I combined these synthetic values with the original features to create a complete synthetic dataset. By enhancing model complexity and fine-tuning hyperparameters, I aimed to produce

higher-quality synthetic data. Regular updates on training metrics allowed me to track progress and make necessary adjustments.