Step-by-Step Plan for Product Return Prediction using XGBoost

1. Define Target Variable

- Use column: Returned
  - Binary classification target:
    - 1 or 'Yes' -> Product was returned
    - 0 or 'No' -> Not returned
  - Convert strings to numeric if needed.

2. Select & Engineer Useful Features

Date-based Features:

- From 'Order Date': Order Day, Month, Year, Day of Week
- From 'Ship Date': Shipping Duration = Ship Date - Order Date
- Weekend order flags, Holiday season

Customer Info:

- Use 'Segment'; encode
- Ignore 'Customer Name'

Geographical Info:

- Encode 'Country', 'City'
- Include 'Region', 'Ship Mode'

Product Info:

- Encode 'Category', 'Sub-Category'
- May include 'Product ID' or 'Retail Sales People'

Sales Info:

- Include 'Sales', 'Quantity', 'Discount', 'Profit'
- Derive 'Profit Margin', 'Price Category'

3. Preprocessing

- Handle missing/null values
- Encode categorical features (Label/One-Hot)
- Convert dates to datetime
- Normalize numeric features if needed

4. Create X (features) and y (target)

- X = All engineered and selected features
- y = Returned (0/1)

5. Split the Dataset

- Use 80/20 or 70/30 split
- Or use StratifiedKFold for class balance

6. Train XGBoost Model

- Use XGBClassifier
- Tune hyperparameters: max_depth, learning_rate, n_estimators

7. Evaluate Model

- Accuracy
- Precision / Recall / F1-Score
- ROC-AUC
- Confusion Matrix

8. Feature Importance

- Use XGBoost's .feature_importances_ or plot_importance
- Identify key features for return risk

9. Predict Probabilities

- Use model.predict_proba(X_test) to get return likelihood (0-1)

10. Actionable Output

- Set threshold: e.g., if P(return) > 0.7, mark as 'High Risk'

- Use predictions to optimize logistics and policies

Optional Enhancements

- Add customer/product return history
- Use SMOTE or scale_pos_weight for imbalance
- Use SHAP values for explainability