What I done:

Step 1: Importing Libraries:

The necessary Python libraries are imported:

- **Data Manipulation**: pandas, numpy
- **Visualization**: seaborn, matplotlib.pyplot
- **Machine Learning**: sklearn modules (e.g., LogisticRegression, RandomForestClassifier, KMeans)
- **Imbalanced Data Handling**: SMOTE from imblearn
- **Streamlit**: Used to create the web application interface
- **Others**: time for measuring performance

Step 2: Generating Synthetic Loan Data:

The generate_loan_data function creates a **synthetic dataset** of loan-related data.

- Features include age, income, credit_score, loan_amount, and repayment_status (target variable).
- Random values are generated for these fields to simulate real-world data.
- The target variable, repayment_status, is imbalanced (80% "No Default" and 20% "Default").

python

CopyEdit

loan_data = generate_loan_data()

This generates a dataset of 1,000 customers with the above fields.

Step 3: Preprocessing Data:

The preprocess_data function prepares the dataset for machine learning:

1. **Feature Selection**: Removes customer_id (not useful for prediction) and separates repayment_status (target variable).
2. **Encoding Categorical Features**: Encodes any categorical variables using one-hot encoding (though the synthetic dataset doesn't have any).
3. **Scaling**: Scales features using StandardScaler to ensure numerical stability for algorithms like Logistic Regression.
4.

Step 4: Exploratory Data Analysis (EDA):

The perform_eda function provides visual and statistical insights into the data:

1. **Summary Statistics**: Displays basic statistics (mean, std, etc.).
2. **Target Variable Distribution**: Shows the proportion of loan defaults vs. non-defaults.
3. **Feature Distributions**: Plots histograms for numeric features (age, income, etc.).
4. **Correlation Heatmap**: Displays relationships between numeric features.
5. **Boxplots**: Highlights how features vary based on default status.

These plots are displayed in the Streamlit interface using st.pyplot.

Step 5: Loan Default Prediction:

The loan_default_prediction function trains machine learning models to predict loan defaults.

1. **Train-Test Split**: Splits the preprocessed data into training (70%) and testing (30%) sets.
2. **Models Used**:
    - Logistic Regression
    - Random Forest
    - Gradient Boosting
3. **Hyperparameter Tuning**:
    - Uses RandomizedSearchCV for optimizing hyperparameters.
4. **SMOTE**:
    - Balances the dataset by oversampling the minority class (default cases).
    - Models are trained **with** and **without** SMOTE to compare performance.
5. **Evaluation Metrics**:
    - Accuracy, Precision, Recall, F1-score, and ROC-AUC.

The best-performing model is recommended based on ROC-AUC.

Step 6: Model Evaluation Helper Functions

Two helper functions support model evaluation:

1. **evaluate_model**: Computes evaluation metrics for a model's predictions.
2. **run_model**:
    - Handles optional hyperparameter tuning (RandomizedSearchCV).
    - Trains the model with or without SMOTE.

- Measures training time and evaluates performance.

Step 7: Customer Segmentation:

The customer_segmentation function uses **KMeans clustering** to group customers into segments.

1. **Feature Selection**: Uses numeric features (age, income, etc.) for clustering.
2. **Scaling**: Normalizes the features for clustering stability.
3. **KMeans Clustering**:
    - Groups customers into 3 clusters (n_clusters=3).
    - Adds the cluster label to the dataset.
4. **PCA Visualization**: (Principal Component Analysis)
    - Reduces feature dimensions to 2 using PCA for visualization.
    - Scatterplot shows clusters in the Streamlit interface.

Step 8: Product Recommendations:

The product_recommendations function demonstrates a **simple recommendation system**:

1. **Dummy Interaction Data**: Creates a synthetic customer-product interaction dataset.
2. **Collaborative Filtering**: Placeholder functionality for building recommendations based on user interactions.

Step 9: Streamlit User Interface:

The main function ties everything together in a Streamlit app:

1. **Sidebar Menu**:
    - Users can select one of four functionalities: **EDA**, **Loan Default Prediction**, **Customer Segmentation**, or **Product Recommendations**.
2. **Conditional Execution**:
    - Based on the user's choice, the corresponding function is called to display results and visualizations.

Step 10: Running the App:

The main() function is executed when the script runs:

python

CopyEdit

```
if __name__ == "__main__":
    main()
```

This launches the Streamlit app in a web browser, allowing users to interact with the analytics features.

Summary:

This code combines:

1. **Data Visualization** for insights (EDA).
2. **Supervised Learning** for loan default prediction with imbalanced data handling (SMOTE).
3. **Unsupervised Learning** for customer segmentation (KMeans).
4. **Recommendation Systems** for suggesting products.