Comprehensive Case Studies: Data Generation, Preprocessing, and Visualization

Numan Shafi

October 7, 2024

Contents

1 Case Study 1: Customer Purchasing Behavior Analysis

1.1 Scenario

You are tasked with analyzing customer purchasing behavior for an online retail store. The store wants insights into customer demographics, purchasing patterns, and potential factors affecting purchases. You will generate a dummy dataset, preprocess it, and visualize trends, correlations, and relationships for further model training.

1.2 Step 1: Generate Dummy Data

Task: Create a dataset with 1000 rows and the following columns:

- **CustomerID**: A unique identifier for each customer.
- **Age**: Random values between 18 and 70.
- **Annual Income**: Random values between 20,000 and 120,000.
- **Gender**: Randomly chosen from 'Male', 'Female'.
- **Purchased**: Binary values indicating if a customer purchased a product (0 = No, 1 = Yes).

Hint: Use `NumPy` to generate random numbers for numerical columns and random choices for categorical data. Organize the data into a Pandas DataFrame.

1.3 Step 2: Explore and Inspect the Data

Task: Display the first 10 rows of the dataset to understand its structure.
Hint: Use a Pandas function to preview the DataFrame.
Task: Check for missing values across the dataset.
Hint: Use Pandas to detect any missing data in the columns.

1.4 Step 3: Handling Missing Data

Scenario: After inspection, you notice some missing values in the `Annual Income` column.
Task: Fill in the missing `Annual Income` values using the median income.
Hint: Use a method to replace missing values with the median of the column.

1.5 Step 4: Encoding Categorical Data

Task: Convert the `Gender` and `Purchased` columns into numerical values for machine learning.
Hint: Explore Pandas or Scikit-learn's encoding methods to convert the categorical columns.

1.6 Step 5: Feature Scaling

Task: Scale the `Age` and `Annual Income` columns using min-max scaling.
Hint: Use normalization to scale the data, ensuring values are between 0 and 1.

1.7 Step 6: Data Visualization

Task: Create a histogram to show the distribution of `Age`.
Hint: Use `Matplotlib` to create a histogram that highlights the most common age groups.

Task: Generate a scatter plot to visualize the relationship between `Age` and `Annual Income`.
Hint: Use `Matplotlib` to create a scatter plot showing the spread of income by age.

1.8 Step 7: Correlation Analysis

Task: Calculate the correlation between `Age`, `Annual Income`, and `Purchased` to check if age or income impacts purchasing behavior.
Hint: Use a correlation matrix to see the relationships between these variables.

1.9 Step 8: Feature Engineering

Task: Create a new feature called `Income_per_Age` by dividing `Annual Income` by `Age`.
Hint: Use basic Pandas operations to generate this feature and add it to the DataFrame.

1.10 Step 9: Prepare Data for Modeling

Task: Drop the `CustomerID` column as it is irrelevant for prediction.
Hint: Use Pandas to remove columns that don't contribute to your analysis.

Task: Split the data into training and testing sets.
Hint: Use `train_test_split` from Scikit-learn to divide the dataset into 80% training and 20% testing sets.

2 Case Study 2: Employee Performance Prediction

2.1 Scenario

A company wants to predict employee performance based on their demographic data and work characteristics. You are tasked with generating a dummy dataset, cleaning it, and visualizing relationships between variables to help build a prediction model.

2.2 Step 1: Generate Dummy Data

Task: Create a dataset with 1500 rows and the following columns:

- **EmployeeID**: A unique identifier for each employee.
- **Age**: Random values between 22 and 60.
- **Years of Experience**: Random values between 1 and 40.
- **Gender**: Randomly chosen from 'Male', 'Female'.
- **Performance Rating**: Random values between 1 and 5 (1 = Low, 5 = Excellent).

Hint: Use `NumPy` to generate random numbers for the numerical data and random selections for the categorical data. Organize the data into a Pandas DataFrame.

2.3 Step 2: Explore and Inspect the Data

Task: Display the first 15 rows of the dataset to understand its structure.
Hint: Use a Pandas method to display the head of the DataFrame.

Task: Check for any missing values in the dataset.
Hint: Use Pandas to check for null or missing values.

2.4 Step 3: Handling Missing Data

Scenario: You notice missing values in the `Years of Experience` column.
Task: Fill the missing `Years of Experience` values with the mean experience.
Hint: Use Pandas to fill missing data using the mean.

2.5 Step 4: Encoding Categorical Data

Task: Convert the `Gender` column into numerical values for machine learning.
Hint: Use Pandas' encoding functions to convert the categorical column to numeric values.

2.6 Step 5: Outlier Detection

Task: Identify and handle outliers in the `Years of Experience` column (e.g., values over 40).
Hint: Use statistical methods or visualizations like box plots to detect outliers and decide how to handle them (e.g., remove or cap outliers).

2.7 Step 6: Feature Scaling

Task: Scale the `Age` and `Years of Experience` columns using standardization (z-score normalization).
Hint: Use a standard scaling method to normalize these columns.

2.8 Step 7: Data Visualization

Task: Create a box plot for the `Performance Rating` to understand the distribution of ratings.
Hint: Use `Matplotlib` to create a box plot that shows the spread and central tendencies of the performance ratings.

Task: Create a scatter plot to visualize the relationship between `Years of Experience` and `Performance Rating`.
Hint: Use `Matplotlib` to generate a scatter plot that shows if experience correlates with better performance.

2.9 Step 8: Correlation Analysis

Task: Calculate the correlation between `Age`, `Years of Experience`, and `Performance Rating`.
Hint: Use Pandas' correlation method to create a matrix of correlations between these variables.

2.10 Step 9: Feature Engineering

Task: Create a new feature called `Experience_per_Age` by dividing `Years of Experience` by `Age`.
Hint: Use Pandas to generate this feature and add it to the DataFrame.

2.11 Step 10: Prepare Data for Modeling

Task: Drop irrelevant columns, like `EmployeeID`, from the dataset.
Hint: Use Pandas to remove columns that do not contribute to the prediction task.

Task: Split the data into training and testing sets.
Hint: Use `train_test_split` from Scikit-learn to create an 80% training and 20% testing split.