WEEK 3 LAB ACTIVITIES COM624

Lab Guide: Data Cleaning and Exploratory Data Analysis with Python

Introduction

Welcome! In this lab, you will learn how to clean messy datasets and perform exploratory data analysis (EDA) using Python. These skills are essential for software engineers working with real-world data. You will work with a messy dataset and learn how to clean it, visualise it, and extract meaningful insights.

Learning Objectives

By the end of this lab, you will be able to:

- Load and inspect messy datasets
- Identify and handle missing values and duplicates
- Calculate mean, median, and mode
- Visualise data before and after cleaning
- Perform full EDA on clean datasets
- Generate insights using charts and graphs

Dataset Description

We will use a simulated messy dataset called retail_sales_final.csv. It contains:

Dataset Here

- Missing values (NaN)
- Duplicates
- Inconsistent formatting
- Numerical and categorical data

You can create your own or download a sample from this GitHub link and modify it to include missing values and duplicates.

Step 1: Setting Up Your Python Environment

```
# This code sets up your Python environment by importing the necessary libraries.

import pandas as pd            # For data manipulation
import numpy as np             # For numerical operations
import matplotlib.pyplot as plt  # For plotting graphs
import seaborn as sns          # For advanced visualisations

# Set a clean visual style for plots
sns.set(style="whitegrid")
```

Step 2: Load and Inspect the Messy Dataset

```
# Load the dataset into a DataFrame
df = pd.read_csv('retail_sales_messy.csv')

# Display the first few rows to understand the structure
print(df.head())

# Check data types and missing values
print(df.info())

# Get summary statistics for all columns
print(df.describe(include='all'))
```

Step 3: Understanding Visual Tools to Identify Missing Data

```
# Bar chart to show count of missing values per column
missing_counts = df.isnull().sum()
missing_counts.plot(kind='bar', color='orange')
plt.title("Missing Values per Column")
plt.ylabel("Count")
plt.show()

# Histogram to show distribution of a numerical column
df['Sales'].plot(kind='hist', bins=20, color='skyblue')
plt.title("Sales Distribution (Messy)")
plt.xlabel("Sales")
plt.show()

# Pie chart to show proportion of missing vs non-missing
missing_total = df.isnull().sum().sum()
non_missing_total = df.size - missing_total
plt.pie([missing_total, non_missing_total], labels=['Missing', 'Non-Missing'],
autopct='%1.1f%%', colors=['red', 'green'])
plt.title("Overall Missing Data Proportion")
plt.show()
```

Step 4: Visualising Messy Data with Box Plots and Heatmaps (Optional)

```
# Heatmap to show missing values
sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.title("Missing Values Heatmap")
plt.show()

# Boxplot to detect outliers in 'Sales'
sns.boxplot(x=df['Sales'])
plt.title("Sales Boxplot (Messy)")
plt.show()
```

Step 5: Cleaning the Dataset: Remove Duplicates

```
# Check for duplicate rows
print("Number of duplicates:", df.duplicated().sum())

# Drop duplicate rows
df = df.drop_duplicates()
```

 # Handle Missing Values

```
# Drop rows with any missing values
df_cleaned = df.dropna()

# Alternatively, fill missing values with the mean
# df['Sales'] = df['Sales'].fillna(df['Sales'].mean())
```

#Standardise Column Names

```
# Clean column names for consistency
df_cleaned.columns = df_cleaned.columns.str.strip().str.lower().str.replace(' ', '_')
```

Step 6: Measures of Central Tendency

```
# Calculate mean, median, and mode for 'sales'
print("Mean Sales:", df_cleaned['sales'].mean())
print("Median Sales:", df_cleaned['sales'].median())
print("Mode Sales:", df_cleaned['sales'].mode()[0])
```

Step 7: Full EDA on Clean Dataset

Univariate Analysis

```
# Histogram of 'sales'
df_cleaned['sales'].plot(kind='hist', bins=20, color='green')
plt.title("Sales Distribution (Clean)")
plt.xlabel("Sales")
plt.show()

# Boxplot of 'sales'
sns.boxplot(x=df_cleaned['sales'])
plt.title("Sales Boxplot (Clean)")
plt.show()
```

Multivariate Analysis

```
# Correlation heatmap
plt.figure(figsize=(8,6))
sns.heatmap(df_cleaned.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```

```
# Scatter plot of 'sales' vs 'profit'
sns.scatterplot(x='sales', y='profit', data=df_cleaned)
plt.title("Sales vs Profit")
plt.show()
```

Step 8: Export the Cleaned Dataset

```
# Save the cleaned dataset to a new CSV file
df_cleaned.to_csv('retail_sales_clean.csv', index=False)
```

Step 9: Write a Short Report

In your own words, write a short report summarising:

- Patterns observed (e.g., correlation between sales and profit)
- Anomalies removed
- Summary statistics
- Visual insights

Step 10: Filling Missing Data (Extra only and it is optional – to learn how to fill unknow in a dataset

In this section, you will learn how to **fill in missing values** in your dataset using Python. Instead of removing rows with missing data, you can intelligently replace them using strategies like the mean, zero, or a placeholder like "Unknown". This helps preserve your dataset and improves your analysis.

Why Fill Missing Data?

- Keeps more rows in your dataset
- Prevents errors in calculations

- Makes your dataset more consistent

Example 1: Fill Missing Sales with the Mean

```
# First, calculate the mean of the 'Sales' column
mean_sales = df['Sales'].mean()

# Then, fill missing values in 'Sales' with the mean
df['Sales'] = df['Sales'].fillna(mean_sales)

# This helps ensure that missing sales data doesn't distort your analysis
```

Example 2: Fill Missing Profit with Zero

```
# Replace missing values in 'Profit' with 0
df['Profit'] = df['Profit'].fillna(0)

# This assumes that missing profit means no profit was made
```

Example 3: Fill Missing Country with “Unknown”

```
# Replace missing values in 'Country' with the string 'Unknown'
df['Country'] = df['Country'].fillna('Unknown')

# This helps maintain consistency in categorical data
```

Example 4: Forward Fill (use previous value)

```
# Fill missing values using the previous row's value
df['Sales'] = df['Sales'].fillna(method='ffill')

# Useful for time-series or sequential data
```

Example 5: Backward Fill (use next value)

```
# Fill missing values using the next row's value
df['Sales'] = df['Sales'].fillna(method='bfill')
```

Finally Check Before and After Filling

```
# Check how many missing values exist before filling
print("Missing before:", df.isnull().sum())

# Fill missing values (example: fill 'Profit' with 0)
df['Profit'] = df['Profit'].fillna(0)

# Check how many missing values remain after filling
print("Missing after:", df.isnull().sum())
```

WHAT YOU NEED TO KNOW

Try different strategies and compare the results. For example:

- Does filling with the mean change your average sales?
- Does using "Unknown" affect how you group countries?

Summary Table

Step	Activity	Outcome
1	Setup	Python environment ready
2	Load Data	Understand messy dataset

Step	Activity	Outcome
3	Visualise	Identify missing data
4	Boxplot & Heatmap	Spot outliers and gaps
5	Clean	Remove errors
6	Stats	Apply central tendency
7	EDA	Explore clean data
8	Export	Save clean version
9	Report	Document insights