**EDA Practical Code Explanation**

### *1. Importing Libraries*

**import numpy as np**

**import pandas as pd**

**from scipy import stats**

* numpy (np): Used for numerical operations like mean, variance, std deviation.
* pandas (pd): Used for data manipulation and analysis.
* scipy.stats: Provides statistical functions like z-score.

### *2. Mean, Variance, and Standard Deviation*

**data = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]**

**mean\_value = np.mean(data)**

**variance = np.var(data, ddof=1)**

**std\_dev = np.std(data, ddof=1)**

* np.mean(data): Calculates the average.
* np.var(data, ddof=1): Computes sample variance. ddof=1 applies Bessel’s correction.
* np.std(data, ddof=1): Computes sample standard deviation.

**ddof** stands for **"Delta Degrees of Freedom"**.

* It's a parameter used in **variance** and **standard deviation** calculations in NumPy and Pandas

### *3.* *Correlation and Heatmap*

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**data = {**

**'Math': [...],**

**'Science': [...],**

**'English': [...] }**

**df = pd.DataFrame(data)**

**correlation\_matrix = df.corr()**

**sns.heatmap(correlation\_matrix, annot=True, cmap="coolwarm")**

* **df.corr():** Computes correlation between numerical columns.
* **sns.heatmap(...):** Visualizes correlation matrix with colors.
* **annot=True**: Shows numbers inside boxes.
* **cmap="coolwarm"**: Sets color theme.

### *4. Outlier Detection – IQR Method*

**data = [10, 12, 14, 15, 17, 20, 30, 100]**

**df = pd.DataFrame(data, columns=['values'])**

**Q1 = df['values'].quantile(0.25)**

**Q3 = df['values'].quantile(0.75)**

**IQR = Q3 - Q1**

**lower\_bound = Q1 - 1.5 \* IQR**

**upper\_bound = Q3 + 1.5 \* IQR**

**outliers = df[(df['values'] < lower\_bound) | (df['values'] > upper\_bound)]**

* **quantile(0.25/0.75)**: Returns Q1 and Q3.
* **IQR**: Middle 50% spread.
* **Outliers** are those beyond 1.5\*IQR from Q1 or Q3.

### *5. Outlier Detection – Z-Score Method*

**data\_array = np.array(data)**

**z\_scores = np.abs(stats.zscore(data\_array))**

**outliers = data\_array[z\_scores > 3]**

**stats.zscore()**: Standardizes data.

**np.abs(...)**: Takes absolute values.

**z\_scores > 3**: Detects points more than 3 std deviations from mean.

### *6. Scatter Plot for Outliers*

**plt.scatter(range(len(data)), data)**

**plt.scatter([data.index(100)], [100], color='red')**

Highlights outlier using red dot.

**data.index(100)**: Finds index of outlier.

### *7. Categorical Data Analysis*

**data = pd.DataFrame({'Category': ['Apple', 'Banana', ...]})**

**data['Category'].value\_counts()**

**data['Category'].value\_counts(normalize=True) \* 100**

* .value\_counts(): Counts category frequency.
* normalize=True: Converts count to %.

### *8. Categorical Visualization*

**df['Category'].value\_counts().plot(kind='bar')**

**sns.countplot(x=df['Category'])**

* First uses pandas’ plot()
* Second uses Seaborn’s countplot()

### *9. Encoding Categorical Variables*

**from sklearn.preprocessing import LabelEncoder**

**label\_encoder = LabelEncoder()**

**df['Fruit\_Label'] = label\_encoder.fit\_transform(df['Fruit'])**

**pd.get\_dummies(df['Fruit'])**

**LabelEncoder**: Converts categories to integers.

**get\_dummies()**: Creates one-hot encoding.

### *10. Automated EDA with ydata-profiling*

**from ydata\_profiling import ProfileReport**

**df = pd.read\_csv("https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.csv")**

**profile = ProfileReport(df, explorative=True)**

**profile.to\_file("titanic\_profile\_report.html")**

* Generates HTML report with summaries, distributions, correlations, missing values.

### *11. Automated EDA with Sweetviz*

**import sweetviz as sv**

**report = sv.analyze(df)**

**report.show\_html("titanic\_sweetviz.html")**

* Visual and interactive report generator.
* **analyze()** inspects DataFrame.
* **show\_html()** generates an HTML file.

### *12. D-Tale Dashboard*

**import dtale**

**d = dtale.show(df)**

**d.open\_browser()**

* Opens an interactive GUI in the browser for exploring the DataFrame.