DAY 15 **[11 - JULY - 2025]**

EXPLORATORY DATA ANALYSIS (EDA)

Introduction to EDA

Exploratory Data Analysis (EDA) is a critical step in the machine learning pipeline that involves analyzing and visualizing datasets to understand their structure, patterns, and anomalies before building predictive models. EDA helps data scientists uncover insights, identify data quality issues, and inform feature engineering and model selection. By exploring the data, we can ensure that the machine learning model is built on a solid foundation.

Objectives of EDA

- **Understand Data Structure**: Identify the size, shape, and types of data in the dataset.
- **Detect Anomalies**: Find missing values, outliers, or inconsistencies.
- **Discover Patterns**: Identify relationships, trends, and correlations between variables.
- **Prepare for Modeling**: Guide feature selection, scaling, and transformation decisions.

Basic Operations in EDA

Below are the fundamental operations commonly performed during EDA in a machine learning context, illustrated with Python code using libraries like Pandas, NumPy, Matplotlib, and Seaborn.

1. Loading the Dataset

The first step in EDA is to load the dataset into a suitable format, typically a Pandas DataFrame, for analysis.

```
import pandas as pd
import numpy as np

# Load a sample dataset (e.g., a CSV file)
# Replace 'dataset.csv' with the path to your dataset
df = pd.read_csv('dataset.csv')

# Display the first 5 rows to inspect the data
print("First 5 rows of the dataset:")
print(df.head())
```

2. Understanding Data Structure

To get an overview of the dataset, we check its shape, column names, and data types.

```
# Check the shape of the dataset (rows, columns)
print("Dataset Shape:", df.shape)
```

```
# List column names
print("Column Names:", df.columns.tolist())

# Check data types of each column
print("Data Types:\n", df.dtypes)
```

3. Summary Statistics

Summary statistics provide insights into the central tendency, dispersion, and distribution of numerical features.

```
# Generate summary statistics for numerical columns
print("Summary Statistics:\n", df.describe())
```

For categorical variables, we can check unique values and their counts.

```
# Check unique values and counts for a categorical column
print("Unique Values in Categorical Column:\n", df['categorical_column'].value_counts())
```

4. Handling Missing Values

Missing data can bias machine learning models. EDA involves identifying and addressing missing values.

```
# Check for missing values
print("Missing Values:\n", df.isnull().sum())

# Replace missing values with the mean for numerical columns
df['numerical_column'].fillna(df['numerical_column'].mean(), inplace=True)

# Replace missing values with the mode for categorical columns
df['categorical_column'].fillna(df['categorical_column'].mode()[0], inplace=True)

# Verify missing values are handled
print("Missing Values After Handling:\n", df.isnull().sum())
```

5. Data Visualization

Visualizations help uncover patterns, distributions, and relationships in the data.

Univariate Analysis

Analyze individual variables to understand their distribution.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Histogram for a numerical column
plt.figure(figsize=(8, 6))
```

```
sns.histplot(df['numerical_column'], kde=True)
plt.title('Distribution of Numerical Column')
plt.xlabel('Numerical Column')
plt.ylabel('Frequency')
plt.show()

# Count plot for a categorical column
plt.figure(figsize=(8, 6))
sns.countplot(x='categorical_column', data=df)
plt.title('Count of Categorical Column')
plt.xlabel('Categorical Column')
plt.ylabel('Count')
plt.show()
```

Bivariate Analysis

Explore relationships between two variables.

```
# Scatter plot to visualize relationship between two numerical columns
plt.figure(figsize=(8, 6))
sns.scatterplot(x='numerical_column1', y='numerical_column2', data=df)
plt.title('Scatter Plot of Numerical Columns')
plt.xlabel('Numerical Column 1')
plt.ylabel('Numerical Column 2')
plt.show()

# Box plot to compare numerical column across categories
plt.figure(figsize=(8, 6))
sns.boxplot(x='categorical_column', y='numerical_column', data=df)
plt.title('Box Plot of Numerical Column by Category')
plt.xlabel('Categorical Column')
plt.ylabel('Numerical Column')
plt.show()
```

6. Correlation Analysis

Correlation analysis helps identify relationships between numerical features, which is crucial for feature selection in machine learning.

```
# Compute correlation matrix
correlation_matrix = df.corr()

# Visualize correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```