Exploratory Data Analysis

Introduction

Welcome to the fourth module of our AI course. Today we're going to delve into Exploratory Data Analysis (EDA) - a crucial step in any data science or machine learning project. EDA is the **process of examining and investigating data** to discover patterns, spot anomalies, test hypotheses, and check assumptions.

Think of EDA as detective work - you're investigating your data to understand its **"story"** before applying more complex techniques. By the end of this module, you'll understand how to extract meaningful insights from data before formal modeling.

1. What is Exploratory Data Analysis?

Exploratory Data Analysis (EDA) is an approach to **analyzing datasets** to summarize their main characteristics, often with visual methods. It's about **getting to know your data**—understanding its **structure, variables, and potential issues** before applying more complex analysis techniques.

EDA is like investigating a crime scene: you carefully examine the evidence (data), look for patterns and unusual occurrences, and form initial theories about what might be happening.

EDA helps us to:

- Identify patterns, trends, and anomalies in the data
- Detect errors and missing values
- Understand relationships between variables
- Formulate hypotheses for further investigation
- Guide the selection of appropriate analytical techniques

💡 **Interactive Question #1: When should you perform Exploratory Data Analysis in a data science project?**

A) Only after building your predictive models

B) At the very beginning, before any modeling

C) Only if your initial models don't perform well

D) Only when working with unfamiliar datasets

Consider the data science workflow. Where would understanding your data be most valuable?

2. The EDA Process

2.1 Understanding the Data Structure

The first step in EDA is to **understand what data you're working** with:

- How many observations (rows) and variables (columns) are there?
- What types of variables are present (numerical, categorical, etc.)?
- Are there any missing values or duplicates?

In Python, we can use simple commands to get this information:

Dataset : https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv

No Copy-Paste, Only Typing

```python
import pandas as pd
import numpy as np

# Load the dataset (using the Titanic dataset as an example)
df = pd.read_csv('https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/ti

# View the first few rows
print(df.head())

# Get basic information about the dataset
print(df.info())

# Summary statistics
print(df.describe())

# Check for missing values
print(df.isnull().sum())

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

No Copy-Paste, Only Typing

💡 **Interactive Question #2: If a dataset has 1000 rows and 20 columns, which command would tell you how many missing values exist in each column?**

A) dataset.count()

B) dataset.isnull()

C) dataset.isnull().sum()

D) dataset.describe()

2.2 Univariate Analysis

Univariate analysis examines variables **one at a time**. This helps us understand the distribution and characteristics of individual variables before looking at relationships.

For Numerical Variables:

- Calculating central tendency (mean, median, mode)
- Measuring dispersion (range, variance, standard deviation)
- Understanding the shape of distributions (symmetric, skewed, bimodal)
- Identifying outliers

Below is the sample python code:

No Copy-Paste, Only Typing

```python
# Example code for numerical analysis
# Calculate basic statistics
print(df['Age'].describe())

# Find mode
print(f"Mode of Age: {df['Age'].mode()[0]}")

# Calculate variance and standard deviation
print(f"Variance of Age: {df['Age'].var()}")
print(f"Standard deviation of Age: {df['Age'].std()}")

# Calculate range
print(f"Range of Age: {df['Age'].max() - df['Age'].min()}")

# Calculate quantiles
print(df['Age'].quantile([0.25, 0.5, 0.75]))

# Calculate skewness and kurtosis
print(f"Skewness of Age: {df['Age'].skew()}")
print(f"Kurtosis of Age: {df['Age'].kurt()}")
```

No Copy-Paste, Only Typing

For Categorical Variables:

- Frequency counts and proportions
- Unique values
- Mode (most common category)
- Cardinality (number of unique categories)

Below is the sample code:

No Copy-Paste, Only Typing

```
# Example code for categorical analysis
# Frequency counts
print(df['Pclass'].value_counts())

# Proportions
print(df['Pclass'].value_counts(normalize=True))

# Unique values
print(f"Unique values in Sex: {df['Sex'].unique()}")

# Number of unique categories
print(f"Cardinality of Embarked: {df['Embarked'].nunique()}")

# Mode (most common category)
print(f"Most common passenger class: {df['Pclass'].mode()[0]}")
```

No Copy-Paste, Only Typing

2.3 Bivariate Analysis

Bivariate analysis examines the **relationship between two variables**. This helps us understand how variables interact and influence each other.

Numerical vs. Numerical:

• **Correlation coefficients**: Numbers that measure **how strongly two variables are related and in what direction**

- **Pearson correlation**: Ranges from -1 to +1. +1 means strong positive relationship (as one increases, the other increases), -1 means strong negative relationship (as one increases, the other decreases), and 0 means no linear relationship.
- **Spearman correlation**: Similar to Pearson but works better when relationships aren't straight lines or when data has outliers.

• **Covariance**: Also measures relationship between variables but depends on the scale of the variables, making it harder to interpret than correlation.

Below is the sample code:

No Copy-Paste, Only Typing

```python
# Calculate correlation coefficients
print("Pearson correlation between Age and Fare:")
print(df['Age'].corr(df['Fare'], method='pearson'))

print("\nSpearman correlation between Age and Fare:")
print(df['Age'].corr(df['Fare'], method='spearman'))

# Calculate covariance
print("\nCovariance between Age and Fare:")
print(df['Age'].cov(df['Fare']))

# Testing for independence (if needed)
from scipy.stats import pearsonr
corr, p_value = pearsonr(df['Age'].dropna(), df['Fare'].dropna())
print(f"\nPearson correlation: {corr:.3f}")
print(f"P-value: {p_value:.6f}")
if p_value < 0.05:
    print("The correlation is statistically significant.")
else:
    print("There is no statistically significant correlation.")
```

No Copy-Paste, Only Typing

Difference Between Covariance and Correlation

- Think of **covariance** as saying:

  "Do these two things move in the same direction?"

- Think of **correlation** as saying:

  "How closely and strongly do they move together — and how much can we trust that?"

Categorical vs. Numerical:

When analyzing a numerical variable across different categories:

- **Group statistics**: Calculate summary statistics for each category (like average income by education level)
- **ANOVA (Analysis of Variance)**: Tests whether the averages differ significantly between groups

AMEYA

Below is the sample code:

No Copy-Paste, Only Typing

```python
# Group statistics
print("Mean age by passenger class:")
print(df.groupby('Pclass')['Age'].mean())

print("\nMedian age by passenger class:")
print(df.groupby('Pclass')['Age'].median())

print("\nStandard deviation of age by passenger class:")
print(df.groupby('Pclass')['Age'].std())

# ANOVA test to see if average ages differ between classes
from scipy.stats import f_oneway

class1_age = df[df['Pclass'] == 1]['Age'].dropna()
class2_age = df[df['Pclass'] == 2]['Age'].dropna()
class3_age = df[df['Pclass'] == 3]['Age'].dropna()

f_stat, p_value = f_oneway(class1_age, class2_age, class3_age)
print(f"\nANOVA test for age difference across classes:")
print(f"F-statistic: {f_stat:.3f}")
print(f"P-value: {p_value:.6f}")
if p_value < 0.05:
    print("There is a statistically significant difference in ages between classes.")
else:
    print("There is no statistically significant difference in ages between classes.")
```

No Copy-Paste, Only Typing

Categorical vs. Categorical:

For relationships between two categorical variables (like gender and survival status):

- **Contingency tables**: Tables showing counts for all combinations of categories (like number of survivors by gender)
- **Chi-square test**: Determines if there's a significant relationship between the categories

💡 **Interactive Question #3: When analyzing the relationship between income (numerical) and education level (categorical), which statistical approach would be most appropriate?**

A) Pearson correlation

B) Group comparison of means or medians

C) Chi-square test

D) Simple linear regression

No Copy-Paste, Only Typing

```
# Contingency table
survival_by_class = pd.crosstab(df['Pclass'], df['Survived'],
                                margins=True, margins_name='Total')
print("Contingency table of survival by passenger class:")
print(survival_by_class)

# Chi-square test
from scipy.stats import chi2_contingency
chi2, p_value, dof, expected = chi2_contingency(pd.crosstab(df['Pclass'], df['Survived']))
print(f"\nChi-square test result:")
print(f"P-value: {p_value:.6f}")
if p_value < 0.05:
    print("There is a statistically significant relationship between passenger class and
else:
    print("There is no statistically significant relationship between passenger class and
```

No Copy-Paste, Only Typing

2.4 Multivariate Analysis

Multivariate analysis examines **relationships between three or more variables** simultaneously. This helps us understand complex interactions and patterns that might not be visible in simpler analyses.

Key multivariate techniques include:

- Conditional means and proportions
- Multiple correlation analysis
- Partial correlation coefficients
- Multiple regression (exploratory, not predictive at this stage)
- Principal Component Analysis (conceptual overview)
- Factor Analysis (conceptual overview)
- Cluster Analysis (conceptual overview)

💡 **Interactive Question #4: Which analysis technique would be best to examine relationships among multiple numerical variables simultaneously?**

A) Series of bivariate correlations

B) Multiple regression

C) Multivariate correlation analysis

D) Principal Component Analysis

No Copy-Paste, Only Typing

```python
# Conditional analysis
print("Mean fare by passenger class and survival status:")
print(df.groupby(['Pclass', 'Survived'])['Fare'].mean())

# Proportion analysis
survival_by_class_sex = pd.crosstab(
    [df['Pclass'], df['Sex']],
    df['Survived'],
    normalize='index'
)
print("\nSurvival rate by passenger class and sex:")
print(survival_by_class_sex)

# Partial correlation
from scipy.stats import pearsonr

# Example: Correlation between Age and Fare, controlling for Pclass
# (Simplified approach - in practice, use partial_corr from pingouin or statsmodels)
correlations = []
for pclass in df['Pclass'].unique():
    temp_df = df[df['Pclass'] == pclass].dropna(subset=['Age', 'Fare'])
    if len(temp_df) > 5:  # Ensure enough data points
        corr, _ = pearsonr(temp_df['Age'], temp_df['Fare'])
        correlations.append((pclass, corr))

print("\nPartial correlations between Age and Fare by passenger class:")
for pclass, corr in correlations:
    print(f"Class {pclass}: {corr:.3f}")
```

No Copy-Paste, Only Typing

3. Dealing with Data Issues During EDA

3.1 Missing Values

One of the key insights from EDA is **understanding missing data patterns**. Missing data analysis helps determine whether values are **missing completely at random (MCAR)**, **missing at random (MAR)**, or **missing not at random (MNAR)**.

💡 **Interactive Question #5: What does it indicate when a variable has over 90% missing values?**

A) The data collection process was flawed

B) The variable might not be relevant for most observations

C) The data is unsuitable for any analysis

D) Missing values are likely random

No Copy-Paste, Only Typing

```
# Calculate missing values
missing_values = df.isnull().sum()
missing_percentage = missing_values / len(df) * 100
missing_df = pd.DataFrame({'Missing Values': missing_values,
                           'Percentage': missing_percentage})
print(missing_df[missing_df['Missing Values'] > 0].sort_values('Percentage', ascending=Fa

# Analyze missing value patterns
# Are missing values related to other variables?
# Example: Are missing ages related to passenger class?
age_missing_by_pclass = df.groupby('Pclass')['Age'].apply(lambda x: x.isnull().sum() / le
print(f"Percentage of missing ages by passenger class:\n{age_missing_by_pclass}")

# Check if missing values in one column correlate with missing values in another
missing_age = df['Age'].isnull()
missing_cabin = df['Cabin'].isnull()
missing_correlation = pd.crosstab(missing_age, missing_cabin,
                                  normalize='all',
                                  rownames=['Missing Age'],
                                  colnames=['Missing Cabin'])
print("\nCrosstab of missing values:")
print(missing_correlation)
```

No Copy-Paste, Only Typing

3.2 Outliers

Identifying and analyzing outliers is another crucial aspect of EDA. Outliers can significantly impact statistical analyses and machine learning models.

There are several methods for detecting outliers:

1. Statistical Methods:

- **Z-score method** (parametric)
- IQR method (non-parametric)
- Modified Z-score (for skewed distributions)
- Grubbs test and Dixon's Q test

2. Distance-based Methods:

- Mahalanobis distance (for multivariate outliers)
- Local Outlier Factor (LOF)
- DBSCAN

AMEYA

No Copy-Paste, Only Typing

```python
# Z-score method for outlier detection
from scipy import stats

z_scores = stats.zscore(df['Fare'].dropna())
abs_z_scores = np.abs(z_scores)
outliers_z = (abs_z_scores > 3)
print(f"Number of outliers detected (Z-score method): {np.sum(outliers_z)}")

# IQR method for outlier detection
Q1 = df['Fare'].quantile(0.25)
Q3 = df['Fare'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers_iqr = ((df['Fare'] < lower_bound) | (df['Fare'] > upper_bound))
print(f"Number of outliers detected (IQR method): {outliers_iqr.sum()}")

# Modified Z-score (more robust for skewed data)
median = df['Fare'].median()
MAD = stats.median_abs_deviation(df['Fare'].dropna())
modified_z_scores = 0.6745 * (df['Fare'] - median) / MAD
outliers_mod_z = np.abs(modified_z_scores) > 3.5
print(f"Number of outliers detected (Modified Z-score method): {outliers_mod_z.sum()}")

# Analyzing outliers - are they errors or legitimate extreme values?
outlier_indices = df.index[outliers_iqr]
print("\nSample of potential outliers:")
print(df.loc[outlier_indices.tolist()[:5], ['PassengerId', 'Pclass', 'Fare', 'Sex']])
```

No Copy-Paste, Only Typing

After identifying outliers, we need to decide how to handle them. Options include:

- Keeping them (if they're legitimate and important)
- Removing them (if they're errors or very rare cases)
- Capping them (winsorizing)
- Transforming the data (e.g., log transformation)
- Treating them as a separate category

💡 **Interactive Question #6: Which method defines outliers as data points that fall more than 1.5 times the interquartile range (IQR) below Q1 or above Q3?**

A) Z-score method

B) Standard deviation method

C) Box plot method

D) Percentile method

3.3 Skewed Distributions

Identifying and understanding non-normal distributions is essential in EDA. Many statistical techniques assume normal distribution, so recognizing skewness helps in selecting appropriate methods.

Types of Skewness:

- **Right (Positive) Skew**: When most values are on the left with a long tail extending to the right (mean > median)
- **Left (Negative) Skew**: When most values are on the right with a long tail extending to the left (mean < median)
- **Symmetric**: When values are evenly distributed around the center (mean ≈ median)

Measuring Skewness:

No Copy-Paste, Only Typing

```
# Calculate skewness
from scipy import stats

print(f"Skewness of Fare: {df['Fare'].skew()}")
# Quick interpretation guide:
# > 0.5 (positively skewed), < -0.5 (negatively skewed), -0.5 to 0.5 (approximately symme

# Check mean and median difference (another sign of skewness)
print(f"Mean of Fare: {df['Fare'].mean()}")
print(f"Median of Fare: {df['Fare'].median()}")
```

No Copy-Paste, Only Typing

Common Transformations for Skewed Data:

- **Logarithmic transformation**: Takes the log of values; great for right-skewed data
- **Square root transformation**: Less dramatic than log; good for moderately skewed data
- **Box-Cox transformation**: Finds the optimal power transformation automatically

No Copy-Paste, Only Typing

```
# Log transformation example
df['Fare_log'] = np.log1p(df['Fare'])  # log1p handles zeros
print(f"Skewness after log transformation: {df['Fare_log'].skew()}")

# Square root transformation example
df['Fare_sqrt'] = np.sqrt(df['Fare'])
print(f"Skewness after square root transformation: {df['Fare_sqrt'].skew()}")
```

No Copy-Paste, Only Typing

Why transformations matter:

1. Many statistical tests assume normal distribution
2. Models like linear regression perform better with normally distributed variables
3. Transformations can help reveal patterns hidden by skewed distributions

💡 **Interactive Question #7: Which transformation is commonly used for right-skewed data?**

A) Squaring the values

B) Logarithmic transformation

C) Exponential transformation

D) Standardization (z-score)

4. Feature Engineering Ideas from EDA

EDA often suggests new features that might improve model performance. Feature engineering is the process of **creating new variables from existing data** to improve model accuracy or interpretability.

Common Feature Engineering Techniques:

- Creating interaction terms
- Binning continuous variables
- One-hot encoding categorical variables
- Feature extraction from text or dates
- Creating domain-specific features

💡 **Interactive Question #8: What type of feature engineering is happening when we convert the continuous variable 'Age' into categories like 'Child', 'Teenager', etc.?**

A) Scaling

B) Dimensionality reduction

C) Discretization

D) One-hot encoding

AMEYA

No Copy-Paste, Only Typing

```python
# Creating a family size feature
df['FamilySize'] = df['SibSp'] + df['Parch'] + 1  # +1 for the passenger themselves
print("Distribution of family size:")
print(df['FamilySize'].value_counts().sort_index())

# Creating age groups
df['AgeGroup'] = pd.cut(df['Age'], bins=[0, 12, 18, 65, 100],
                        labels=['Child', 'Teenager', 'Adult', 'Elderly'])
print("\nDistribution of age groups:")
print(df['AgeGroup'].value_counts())

# Creating fare categories
df['FareCategory'] = pd.qcut(df['Fare'], q=4, labels=['Low', 'Medium', 'High', 'Very High
print("\nDistribution of fare categories:")
print(df['FareCategory'].value_counts())

# Extracting titles from names
df['Title'] = df['Name'].str.extract(' ([A-Za-z]+)\.', expand=False)
print("\nFrequency of titles:")
print(df['Title'].value_counts())

# Analyzing the relationship between new features and the target
print("\nSurvival rate by family size:")
survival_by_family = df.groupby('FamilySize')['Survived'].mean().sort_index()
print(survival_by_family)

print("\nSurvival rate by age group:")
survival_by_age = df.groupby('AgeGroup')['Survived'].mean()
print(survival_by_age)
```

No Copy-Paste, Only Typing

Summary

Exploratory Data Analysis (EDA) is a crucial first step in any data science project. It helps us understand the structure and characteristics of our data, identify patterns and relationships, and prepare for modeling. The key components of EDA include:

1. **Understanding data structure**: Examining the shape, types, and basic properties of the dataset
2. **Univariate analysis**: Studying individual variables' distributions and properties
3. **Bivariate analysis**: Exploring relationships between pairs of variables
4. **Multivariate analysis**: Investigating complex interactions among multiple variables
5. **Handling data issues**: Addressing missing values, outliers, and skewed distributions
6. **Feature engineering**: Creating new variables from existing ones to improve model performance

By thoroughly exploring your data before modeling, you can make more informed decisions, avoid potential pitfalls, and ultimately build better predictive models.

Remember that EDA is an iterative and creative process. There's no fixed recipe - the specific techniques you use will depend on your data and questions. The key is to approach your data with curiosity and critical thinking, always asking "What does this tell me?" and "What else should I investigate?"

AMEYA

Glossary

• **Central tendency**: Measures that describe the center of a distribution (mean, median, mode)

• **Correlation**: A measure of the relationship between two variables, indicating how they change together

• **Correlation coefficients**: Numerical values that quantify the strength and direction of relationships between variables:

- **Pearson correlation coefficient (r)**: Measures linear relationships between continuous variables; ranges from -1 (perfect negative) to +1 (perfect positive)
- **Spearman rank correlation (ρ)**: Non-parametric measure of monotonic relationships; less sensitive to outliers than Pearson

• **Covariance**: Measures how two variables change together but is not standardized, making comparisons difficult across different scales

• **Contingency table**: A cross-tabulation showing the frequency distribution of variables across different categories

• **Dimensionality reduction**: Techniques to reduce the number of variables while preserving information; includes PCA, t-SNE, and factor analysis

• **Dispersion**: Measures of how spread out the data is:

- **Variance**: Average of squared deviations from the mean
- **Standard deviation**: Square root of variance, in the same units as the original data
- **IQR (Interquartile Range)**: The difference between the 75th and 25th percentiles; robust to outliers
- **Range**: Difference between maximum and minimum values
- **Mean Absolute Deviation (MAD)**: Average of absolute deviations from the mean

• **Feature engineering**: The process of creating new variables from existing data to improve model performance

• **Multicollinearity**: High correlation between predictor variables that can cause problems in regression models by making coefficient estimates unstable

• **Normality**: The extent to which a distribution follows a normal/Gaussian distribution (bell curve)

• **Outlier**: A data point that differs significantly from other observations; can be identified using methods like the IQR method, Z-score, or visual inspection

• **Skewness**: A measure of the asymmetry of a distribution:

AMEYA

- **Positive skew**: Longer tail on the right, mean > median
- **Negative skew**: Longer tail on the left, mean < median

• **Statistical tests**:

- **Chi-square test**: Tests independence between categorical variables
- **ANOVA (Analysis of Variance)**: Compares means across three or more groups

• **Transformation**: Mathematical operations applied to variables to change their distribution or scale:

- **Log transformation**: Useful for right-skewed data, compresses high values
- **Square root transformation**: Milder than log, good for count data
- **Box-Cox**: Family of power transformations to find optimal normalization
- **Z-score standardization**: (value - mean)/standard deviation; centers data around 0 with SD of 1
- **Min-Max scaling**: Transforms data to range between 0 and 1

Multiple Choice Answers

1. B) At the very beginning, before any modeling
2. C) dataset.isnull().sum()
3. B) Group comparison of means or medians
4. D) Principal Component Analysis
5. B) The variable might not be relevant for most observations
6. C) Box plot method
7. B) Logarithmic transformation
8. C) Discretization

AMEYA