# Exploratory Data Analysis (EDA)

## 1. Introduction to EDA

Exploratory Data Analysis (EDA) is a critical first step in data analysis that employs various techniques to:

- Understand data characteristics
- Find patterns and relationships
- Detect anomalies
- Test assumptions
- Support hypothesis development

## 2. Data Understanding

### 2.1 Basic Data Examination

python
Copy

```
# Basic data information
df.info()
df.shape
df.head()
df.tail()
df.describe()
df.columns

# Data types
df.dtypes
df.select_dtypes(include=['number'])
df.select_dtypes(include=['object'])
```

### 2.2 Missing Values Analysis

python
Copy

```
# Check missing values
df.isnull().sum()
df.isnull().mean() * 100  # Percentage of missing values
```

```python
# Visualize missing values
import missingno as msno
msno.matrix(df)
msno.heatmap(df)
```

# 3. Univariate Analysis

## 3.1 Numerical Variables

python
Copy

```python
# Statistical measures
df['column'].describe()
df['column'].mean()
df['column'].median()
df['column'].mode()
df['column'].std()
df['column'].var()
df['column'].skew()
df['column'].kurtosis()

# Visualizations
# Histogram
plt.hist(df['column'])

# Box plot
plt.boxplot(df['column'])

# Kernel Density Plot
sns.kdeplot(df['column'])
```

## 3.2 Categorical Variables

python
Copy

```python
# Frequency counts
df['column'].value_counts()
df['column'].value_counts(normalize=True)  # Proportions

# Visualizations
```

```python
# Bar plot
sns.barplot(x=df['column'].value_counts().index,
            y=df['column'].value_counts().values)

# Pie chart
plt.pie(df['column'].value_counts().values,
        labels=df['column'].value_counts().index)
```

# 4. Bivariate Analysis

## 4.1 Numerical vs Numerical

python
Copy

```python
# Correlation analysis
df.corr()
sns.heatmap(df.corr(), annot=True)

# Scatter plot
plt.scatter(df['column1'], df['column2'])
sns.scatterplot(data=df, x='column1', y='column2')

# Joint plot
sns.jointplot(data=df, x='column1', y='column2')
```

## 4.2 Numerical vs Categorical

python
Copy

```python
# Box plot
sns.boxplot(x='categorical_col', y='numerical_col', data=df)

# Violin plot
sns.violinplot(x='categorical_col', y='numerical_col', data=df)

# Bar plot with confidence intervals
sns.barplot(x='categorical_col', y='numerical_col', data=df)
```

## 4.3 Categorical vs Categorical

python

Copy
```python
# Contingency table
pd.crosstab(df['cat_col1'], df['cat_col2'])

# Stacked bar plot
pd.crosstab(df['cat_col1'], df['cat_col2']).plot(kind='bar', stacked=True)

# Chi-square test
from scipy.stats import chi2_contingency
chi2_contingency(pd.crosstab(df['cat_col1'], df['cat_col2']))
```

# 5. Multivariate Analysis

## 5.1 Multiple Numerical Variables

python
Copy
```python
# Scatter matrix
pd.plotting.scatter_matrix(df[['col1', 'col2', 'col3']])

# Pair plot
sns.pairplot(df[['col1', 'col2', 'col3']])

# Correlation heatmap
sns.heatmap(df[['col1', 'col2', 'col3']].corr(), annot=True)
```

## 5.2 Mixed Variable Types

python
Copy
```python
# Faceted plots
g = sns.FacetGrid(df, col='categorical_col')
g.map(plt.hist, 'numerical_col')

# Multiple box plots
sns.boxplot(x='cat_col1', y='numerical_col', hue='cat_col2', data=df)
```

# 6. Advanced Analysis Techniques

### 6.1 Distribution Analysis

python
Copy

```
from scipy import stats

# Normality tests
stats.normaltest(df['column'])
stats.shapiro(df['column'])

# QQ Plot
import statsmodels.api as sm
sm.qqplot(df['column'], line='45')
```

### 6.2 Outlier Detection

python
Copy

```
# Z-score method
from scipy import stats
z_scores = stats.zscore(df['column'])
outliers = abs(z_scores) > 3

# IQR method
Q1 = df['column'].quantile(0.25)
Q3 = df['column'].quantile(0.75)
IQR = Q3 - Q1
outliers = (df['column'] < (Q1 - 1.5 * IQR)) | (df['column'] >
(Q3 + 1.5 * IQR))
```

# 7. Time Series Analysis

### 7.1 Basic Time Series Plots

python
Copy

```
# Line plot
df.plot(x='date_column', y='value_column')
```

```python
# Seasonal decomposition
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(df['column'], period=12)
decomposition.plot()
```

### 7.2 Time-Based Features

python
Copy

```python
# Extract time components
df['year'] = df['date_column'].dt.year
df['month'] = df['date_column'].dt.month
df['day'] = df['date_column'].dt.day
df['dayofweek'] = df['date_column'].dt.dayofweek
```

## 8. Feature Engineering Insights

### 8.1 Feature Creation

python
Copy

```python
# Binning
df['age_group'] = pd.cut(df['age'], bins=[0, 18, 35, 50, 65,
100])

# Log transformation
df['log_value'] = np.log1p(df['value'])

# Interaction terms
df['interaction'] = df['feature1'] * df['feature2']
```

### 8.2 Feature Selection

python
Copy

```python
# Correlation-based selection
correlation_matrix = df.corr().abs()
upper =
correlation_matrix.where(np.triu(np.ones(correlation_matrix.shap
e), k=1).astype(bool))
```

```python
high_corr_features = [column for column in upper.columns if any(upper[column] > 0.95)]
```

# 9. Statistical Tests

## 9.1 Parametric Tests

python
Copy

```python
# T-test
from scipy import stats
stats.ttest_ind(group1, group2)

# ANOVA
stats.f_oneway(group1, group2, group3)
```

## 9.2 Non-parametric Tests

python
Copy

```python
# Mann-Whitney U test
stats.mannwhitneyu(group1, group2)

# Kruskal-Wallis H-test
stats.kruskal(group1, group2, group3)
```

# 10. Reporting and Visualization Best Practices

## 10.1 Visualization Guidelines

- Use appropriate chart types
- Maintain consistent styling
- Include clear labels and titles
- Add legends where necessary
- Consider color-blind friendly palettes

## 10.2 Reporting Structure

1. Data Overview
2. Data Quality Assessment

3. Univariate Analysis Results
4. Bivariate Analysis Results
5. Multivariate Analysis Results
6. Key Findings and Insights
7. Recommendations for Further Analysis