Class Notes

Date: 17/01/2025

Step 1: Start with a Plan

Before visualizing, identify:

1. **Feature Types**:
   - Numerical (continuous/discrete).
   - Categorical.
   - Date/Time.
2. **EDA Goals**:
   - Understand distributions of individual features.
   - Identify relationships between features (correlations, trends, etc.).
   - Spot anomalies, missing values, or outliers.
   - Answer domain-specific questions (e.g., trends, patterns).

Step 2: Visualizations for Individual Features

1. Numerical Features:

- **Histogram**: To view the distribution (e.g., normal, skewed, etc.).

- **Boxplot**: To detect outliers and variability.
- **KDE Plot**: To smooth distributions and identify data peaks.

2. Categorical Features:

- **Bar Chart**: To analyze frequencies of categories.
- **Pie Chart**: For proportions (use sparingly).

3. Datetime Features:

- **Line Plot**: To observe trends over time.
- **Heatmaps**: For time-series data to identify seasonal patterns.

Step 3: Visualizations for Relationships

1. Numerical-Numerical:

- **Scatter Plot**: To detect patterns or correlations.
- **Heatmap of Correlations**: To quantify relationships between variables.
- **Pairplot**: To visualize pairwise relationships across all numerical features.

2. Numerical-Categorical:

- **Boxplot**: To compare distributions across categories.
- **Violin Plot**: Combines KDE and boxplot for richer insights.

3. Categorical-Categorical:

- **Stacked Bar Chart**: To observe proportions across categories.
- **Clustered Bar Chart**: To compare category combinations.

4. Multivariate:

- **FacetGrid/Small Multiples**: To slice data by one variable and plot another.
- **3D Scatter Plot**: To visualize relationships across three numerical features.
- **Bubble Plot**: To add a fourth variable using bubble size.

Step 4: Advanced Techniques

1. Dimensionality Reduction:

- **PCA (Principal Component Analysis)**: For visualizing high-dimensional data in 2D/3D.
- **t-SNE/UMAP**: For clustering and pattern recognition.

2. Feature Engineering Insights:

- **Interaction Effects**: Use color or size to encode additional variables.
- **Cluster Analysis**: Combine clustering techniques with visualizations to find natural groupings.

3. Missing Data:

- **Heatmap**: To show missing value patterns.
- **Bar Chart**: To view the count of missing values per feature.

Step 5: Tools and Libraries

1. **Python Libraries**:
    - **Matplotlib**: Low-level and highly customizable.
    - **Seaborn**: High-level API for statistical plots.
    - **Plotly**: Interactive plots.
    - **Altair**: Declarative visualization for insightful patterns.
    - **Pandas Visualization**: Quick and simple.
2. **Built-In Techniques**:
    - Use `pandas-profiling` or `sweetviz` for automated EDA reports.
    - `Yellowbrick` for visual diagnostics of machine learning models.

Step 6: Workflow to Gain Insights

1. **Iterate**: Start with simple plots, then refine with advanced visualizations.
2. **Ask Questions**: Frame hypotheses about your data and validate them visually.
3. **Automate Patterns**: Use automation for large datasets but manually inspect anomalies.
4. **Summarize Findings**: Note key observations for each visualization.

Here's the complete Python workflow for EDA visualization, which you can add to your editing area:

```python

# 1. Setup Environment

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

# 2. Load Dataset

# Load the data

df = pd.read_csv('your_data.csv')

# Display the first few rows to understand the data structure

df.head()

# 3. Visualizing Individual Features

## 3.1 Numerical Features

# Histogram: To visualize the distribution of a numerical feature

plt.figure(figsize=(8,6))
```

```
sns.histplot(df['numerical_column'], kde=True, bins=30)

plt.title('Distribution of Numerical Feature')

plt.show()

# Boxplot: To detect outliers and understand spread

plt.figure(figsize=(8,6))

sns.boxplot(x=df['numerical_column'])

plt.title('Boxplot of Numerical Feature')

plt.show()

## 3.2 Categorical Features

# Bar Chart: To visualize the frequency of categories

plt.figure(figsize=(8,6))

sns.countplot(x=df['categorical_column'])

plt.title('Frequency of Categories')

plt.show()

## 3.3 Datetime Features

# Line Plot: To observe trends over time

df['date_column'] = pd.to_datetime(df['date_column'])

plt.figure(figsize=(10,6))

sns.lineplot(x=df['date_column'], y=df['numerical_column'])

plt.title('Trends Over Time')

plt.show()
```

```
# 4. Visualizing Relationships Between Features

## 4.1 Numerical-Numerical Relationships

# Scatter Plot: To identify linear or non-linear relationships

plt.figure(figsize=(8,6))

sns.scatterplot(x=df['numerical_column_1'], y=df['numerical_column_2'])

plt.title('Scatter Plot Between Two Numerical Features')

plt.show()

# Correlation Heatmap: To identify correlations

plt.figure(figsize=(10,8))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)

plt.title('Correlation Heatmap')

plt.show()

## 4.2 Numerical-Categorical Relationships

# Boxplot: To visualize the distribution of a numerical feature across categories

plt.figure(figsize=(8,6))

sns.boxplot(x=df['categorical_column'], y=df['numerical_column'])

plt.title('Boxplot: Numerical Feature vs Categorical Feature')

plt.show()

# Violin Plot: For more detailed distributions

plt.figure(figsize=(8,6))

sns.violinplot(x=df['categorical_column'], y=df['numerical_column'])
```

```
plt.title('Violin Plot: Numerical Feature vs Categorical Feature')

plt.show()

## 4.3 Categorical-Categorical Relationships

# Stacked Bar Chart: To see the relationship between two categorical features

ct = pd.crosstab(df['categorical_column_1'], df['categorical_column_2'])

ct.plot(kind='bar', stacked=True, figsize=(10,6))

plt.title('Stacked Bar Chart: Categorical Feature 1 vs Categorical Feature 2')

plt.show()

# 5. Visualizing Multivariate Relationships

## 5.1 Pairplot: To visualize pairwise relationships across all numerical features

sns.pairplot(df[['numerical_column_1', 'numerical_column_2', 'numerical_column_3']])

plt.title('Pairplot of Numerical Features')

plt.show()

## 5.2 FacetGrid: To visualize relationships between features split by categories

g = sns.FacetGrid(df, col="categorical_column", height=5, aspect=1.5)

g.map(sns.scatterplot, 'numerical_column_1', 'numerical_column_2')

g.set_axis_labels('Numerical Feature 1', 'Numerical Feature 2')

g.set_titles('Category: {col_name}')

plt.show()

# 6. Visualizing Missing Data

## Missing Data Heatmap: To visualize missing values in the dataset
```

```
plt.figure(figsize=(10,6))

sns.heatmap(df.isnull(), cbar=False, cmap='viridis')

plt.title('Missing Data Heatmap')

plt.show()

# 7. Interactive Visualizations (Optional)

## Plotly: Interactive Scatter Plot using Plotly

fig = px.scatter(df, x='numerical_column_1', y='numerical_column_2',
color='categorical_column')

fig.update_layout(title='Interactive Scatter Plot')

fig.show()
```