**Exploratory Data Analysis (EDA)**

Exploratory Data Analysis (EDA) is a crucial initial step in data science projects. It involves analyzing and visualizing data to understand its key characteristics, uncover patterns, and identify relationships between variables refers to the method of studying and exploring record sets to apprehend their predominant traits, discover patterns, locate outliers, and identify relationships between variables. EDA is normally carried out as a preliminary step before undertaking extra formal statistical analyses or modeling.

Key aspects of EDA include:

* Distribution of Data: Examining the distribution of data points to understand their range, central tendencies (mean, median), and dispersion (variance, standard deviation).
* Graphical Representations: Utilizing charts such as histograms, box plots, scatter plots, and bar charts to visualize relationships within the data and distributions of variables.
* Outlier Detection: Identifying unusual values that deviate from other data points. Outliers can influence statistical analyses and might indicate data entry errors or unique cases.
* Correlation Analysis: Checking the relationships between variables to understand how they might affect each other. This includes computing correlation coefficients and creating correlation matrices.
* Handling Missing Values: Detecting and deciding how to address missing data points, whether by imputation or removal, depending on their impact and the amount of missing data.
* Summary Statistics: Calculating key statistics that provide insight into data trends and nuances.
* Testing Assumptions: Many statistical tests and models assume the data meet certain conditions (like normality or homoscedasticity). EDA helps verify these assumptions.

**Visualize Data Using Python: Learn Visualization Using Pandas, Matplotlib, and Seaborn**

Data driven Graphs :

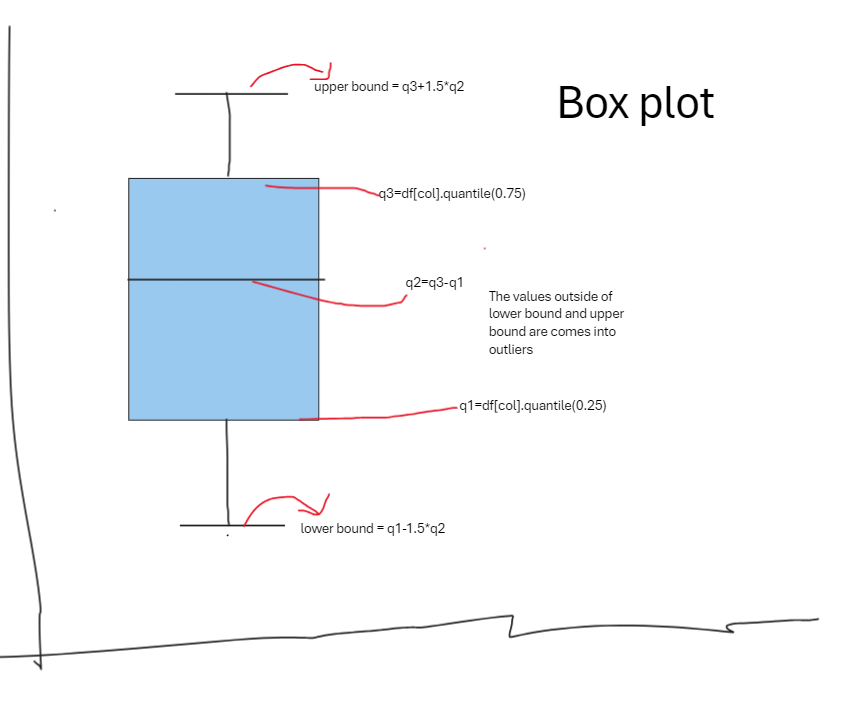
Data is combination of numerical variables and categorical variables.

* Distribution charts - histograms, boxplots, KDS plots
* Trend charts - line charts, area charts
* Comparison charts
* Contribution charts - tree maps, frequency bar charts, frequency pie charts , stacked bar charts

Python libraries :

Pandas, Matplotlib, seaborn , Altair , Ggplot , Plotly

**Box plot**

****

**Line chart :**

For analysing continues values with respect to date and time

**Bar charts:**

For comparing categorical data with numerical data

vertical bar plots, horizontal bar plots, and clustered bar plots.

Bar can be applied using pandas dataframe

gender\_counts.plot(kind='bar', color=['red', 'blue'])

here gender\_count is dataframe

Bar also implemented using matplotlib.pyplot

1)plt.bar(categories,values) – vertical bar

2) plt.barh(categories,value) – horizontal bar

3)stacked bar charts can be drawing using loops

for i in range(len(subcategories)):

plt.bar(categories, values[:, i],bottom=np.sum(values[:, :i],axis=1),label=subcategories[i])

**Histograms:**

Used to visualize the “distribution of numerical data”

plt.hist(scores, bins=6, edgecolor='black')

Bins defines the range(interval) of data , y-axis represents the frequency of values fall under those respective range.

As increasing bins values results in better visualization.

All bins have equal sizes .

**Scatter plot:**

These are used to visualize the relationship between two continuous variables.

The data point (x,y) represents the values of two variables

plt.scatter(hours\_studied, exam\_scores, color='blue')

multiple plots :

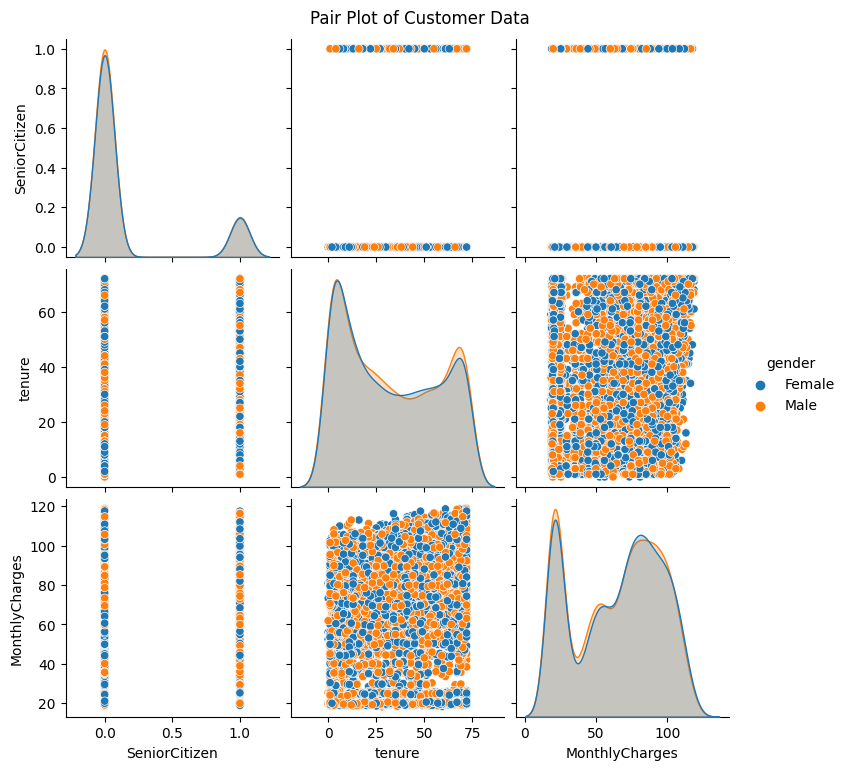
comparing more than two variables at a time

sns.pairplot(df2, diag\_kind='kde',hue='gender')

plt.suptitle('Pair Plot of Customer Data', y=1.02)

plt.show()

hue: differentiates the plots by some category like one colour for one category and other for different colour



**Treemaps:**

Used to display hierarchical data using nested rectangles

It uses another package “**squarify”**

import squarify

squarify.plot(sizes=gender\_counts.values, label=gender\_counts.index, alpha=.8)

**Heat maps:**

To estimate correlation between variables.

Sns.heatmap()

**countPlot/stacked bar plot:**

Used to analyse two different categorical variables.

# Stacked bar plot of Churn vs. Contract Type

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

sns.countplot(x='Contract Type', hue='Churn', data=df, palette='pastel')

plt.title('Churn vs. Contract Type')

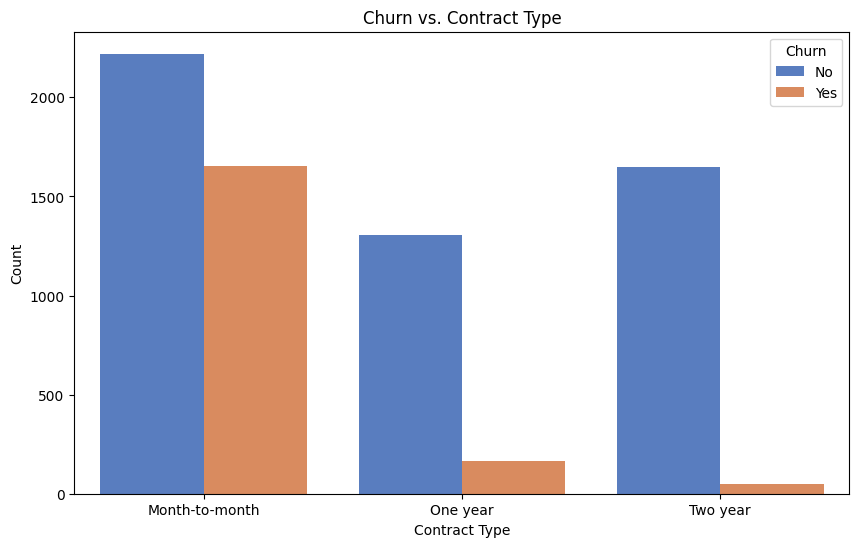
plt.xlabel('Contract Type')

plt.ylabel('Count')

plt.legend(title='Churn', loc='upper right')

plt.show()

palettes = ['deep', 'muted', 'pastel', 'bright', 'dark', 'colorblind']



**Summary:**

Matplotlib

Functions: plot(), scatter(), bar(), hist(), boxplot(), pie(), imshow(), etc.

Seaborn

Functions: countplot(), barplot(), boxplot(), violinplot(), heatmap(), pairplot(), etc.

Pandas

Functions: .plot(kind='line'), .plot(kind='bar'), .plot(kind='hist'), .plot(kind='box'), .plot(kind='scatter'), .plot(kind='pie'), etc.

Plotly

Functions: plotly.graph\_objects for creating figures, plotly.express

Altair

Functions: .mark\_line(), .mark\_bar(), .mark\_point(), .mark\_circle(), .mark\_rect(), .mark\_area(), etc.

Bokeh

Functions: figure(), line(), scatter(), bar(), histogram(), heatmap(), etc.