EXPLOATORY DATA ANALYSIS

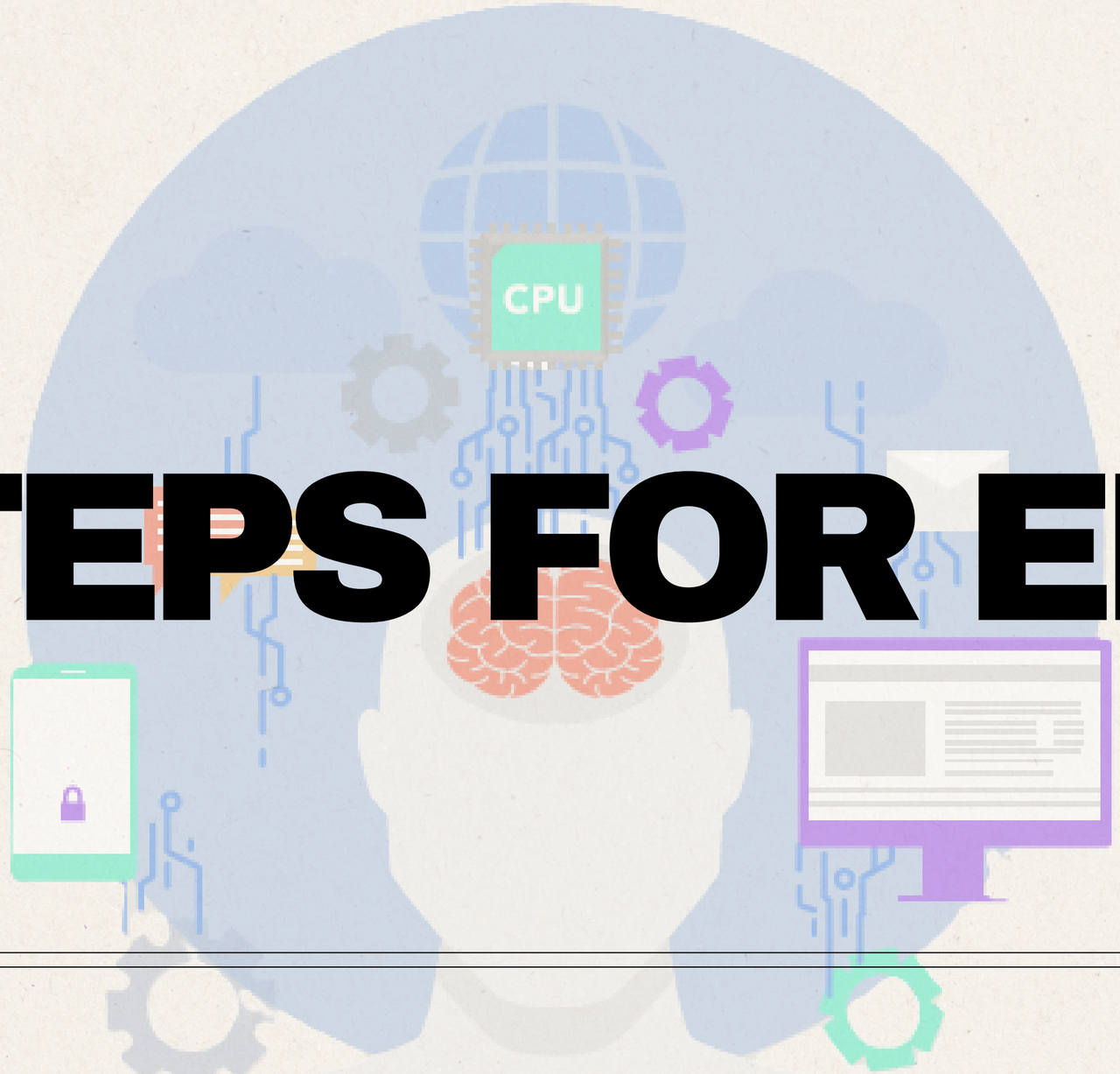
Through

PYTHON LIBRARIES

WHAT IS EDA?

- Exploratory Data Analysis (EDA) is a method of analyzing datasets to understand their main characteristics. It involves summarizing data features, detecting patterns, and uncovering relationships through visual and statistical techniques. EDA helps in gaining insights and formulating hypotheses for further analysis.

STEPS FOR EDA



STEP 1

- Reading the dataset:

```
#Read the CSV file
df = pd.read_csv("/content/80085.csv")
#Print the first few rows
print(df.head())
```

- Analysing the dataset:

```
#Number of rows and coloums
df.shape
#Shows Type of data
#number of entries per coloum
#whether any values are missing
df.info()
#Statistical summary of dataframe
df.describe()
```

STEP 2

- Checking missing Values:

```
df.isnull().sum()
```

This checks for missing values in each column and returns the total number of null values per column helping us to identify any gaps in our data.

OUTPUT:

	0
fixed acidity	0
volatile acidity	0
alcohol	0
quality	0
Id	0

dtype: int64

STEP 3

- Checking for duplicate Values:

```
df.nunique()
```

This function tells us how many unique values exist in each column which provides insight into the variety of data in each feature.

OUTPUT:

	0
fixed acidity	91
volatile acidity	135
alcohol	61
quality	6
Id	1143

dtype: int64

STEP 4

- UNIVARIATE ANALYSIS

In Univariate analysis plotting the right charts can help us to better understand the data making the data visualization so important.We will analyse individual variables to understand their distribution and characteristics. We will be mostly using matplot to create the plots mostly. Seaborn is also a python library built on top of Matplotlib that uses short lines of code to create and style statistical plots from Pandas and Numpy. Categorical variables can be visualized using a Count plot, Bar Chart, Pie Plot, etc. Numerical Variables can be visualized using Histogram, Box Plot, Density Plot, etc.

Quick Note : Numerical variables can be counted or measured while categorical variables are categories or groups.

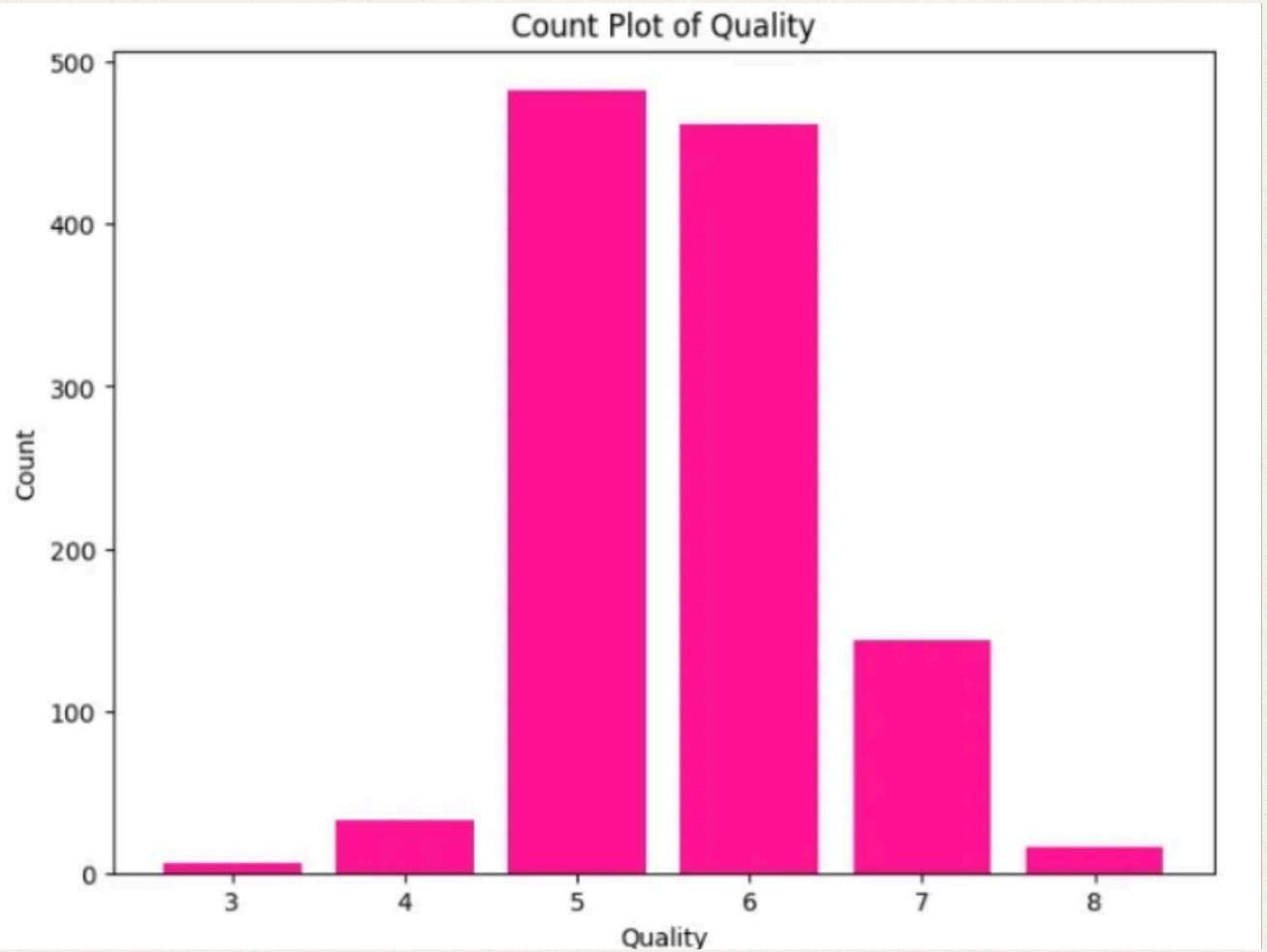
CONTINUED.....

- Example Bar Plot for evaluating the count of the wine with its quality rate with code:

```
# Count the occurrences of each unique value in the 'quality' column
quality_counts = df['quality'].value_counts()
# Create a new figure with specified size
plt.figure(figsize=(8, 6))
# Plot a bar chart using quality levels (index) and their counts
plt.bar(quality_counts.index, quality_counts, color='deeppink')
# Set the title of the plot
plt.title('Count Plot of Quality')
# Label the x-axis as 'Quality'
plt.xlabel('Quality')
# Label the y-axis as 'Count'
plt.ylabel('Count')
# Display the plot
plt.show()
```

CONTINUED.....

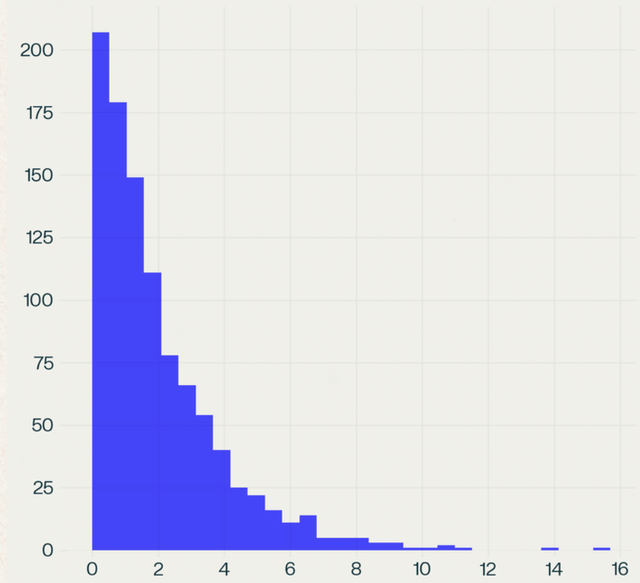
- Example Bar Plot for evaluating the count of the wine with its quality rate with code:



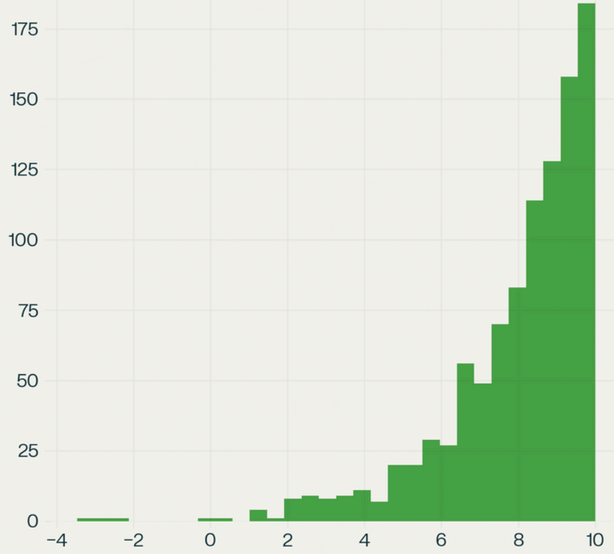
CONTINUED.....

- Kernel Density Plots for Understanding variance in datasets:

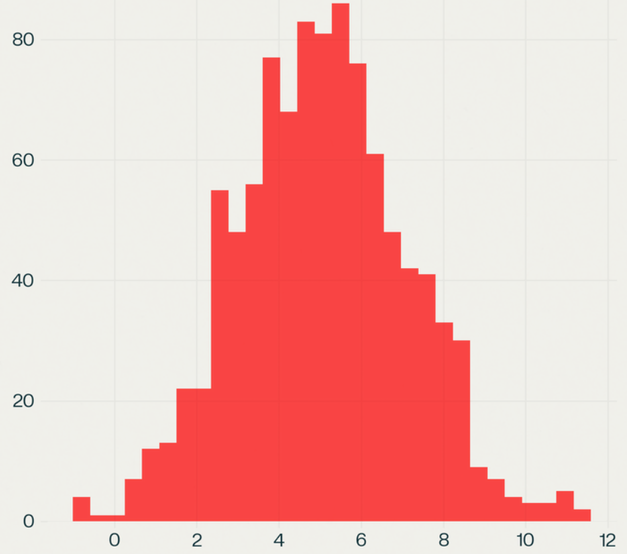
  A kernel density plot (or kernel density estimate, KDE) is a way to visualize the distribution of a continuous variable. It's a smoothed version of a histogram and provides an estimate of the probability density function (PDF) of the variable.
  Skewness is a measure of the asymmetry of a distribution. The features in the dataset with a skewness of 0 shows a symmetrical distribution. If the skewness is 1 or above it suggests a positively skewed (right-skewed) distribution. In a right-skewed distribution the tail extends more to the right which shows the presence of extremely high values.



Right Skewed



Left Skewed



Symmetric

CONTINUED.....

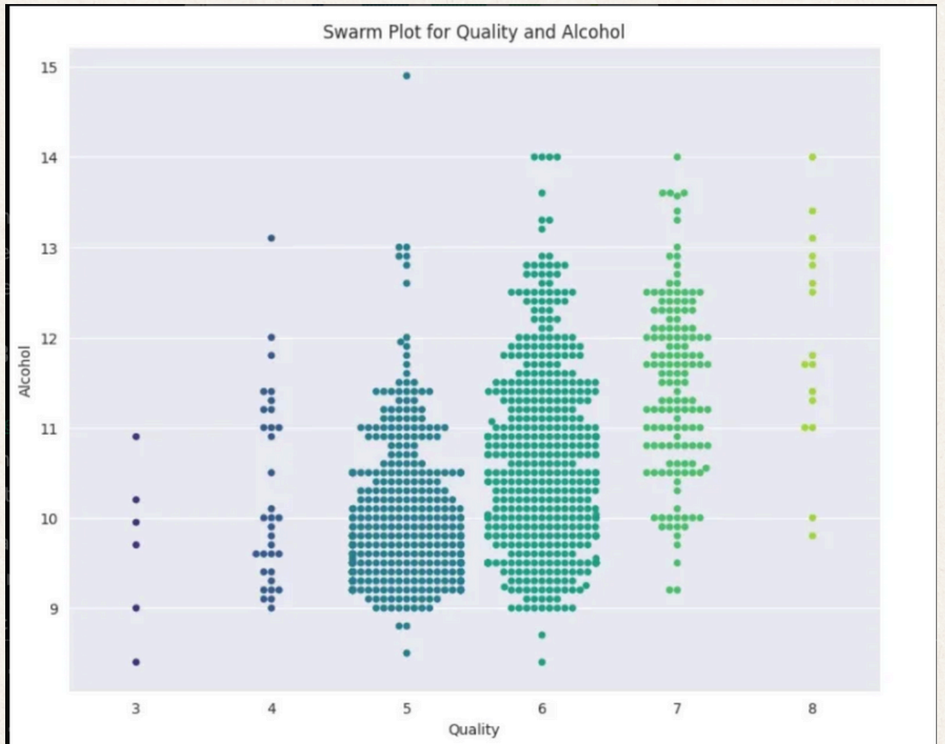
- CODE

```
# Set Seaborn style
sns.set(style="whitegrid")
# Generate datasets
np.random.seed(0)
data_symmetric = np.random.normal(loc=0, scale=1, size=1000)  # Symmetric (Normal)
data_pos_skew = skewnorm.rvs(a=10, loc=0, scale=1, size=1000)  # Positive skew
data_neg_skew = skewnorm.rvs(a=-10, loc=0, scale=1, size=1000) # Negative skew
# Plot KDEs
plt.figure(figsize=(10, 6))
sns.kdeplot(data_symmetric, label=f"Symmetric (skew={np.round(np.mean(data_symmetric), 2)})", color="blue")
sns.kdeplot(data_pos_skew, label=f"Positively Skewed (skew={np.round(np.mean(data_pos_skew), 2)})", color="green")
sns.kdeplot(data_neg_skew, label=f"Negatively Skewed (skew={np.round(np.mean(data_neg_skew), 2)})", color="red")
# Labels and title
plt.title("KDE Plots of Distributions with Different Skewness")
plt.xlabel("Value")
plt.ylabel("Density")
plt.legend()
plt.tight_layout()
plt.show()
```

CONTINUED.....

- Swarm Plots :

A swarm plot is a type of scatter plot that displays individual data points, but in a way that avoids overlapping them. It's often used to visualize the distribution of a categorical variable (like 'Quality') against a continuous variable (like 'Alcohol'). The higher point density in certain areas shows where most of the data points are concentrated. Points that are isolated and far from these clusters represent outliers highlighting uneven values in the dataset.



CONTINUED.....

- CODE:

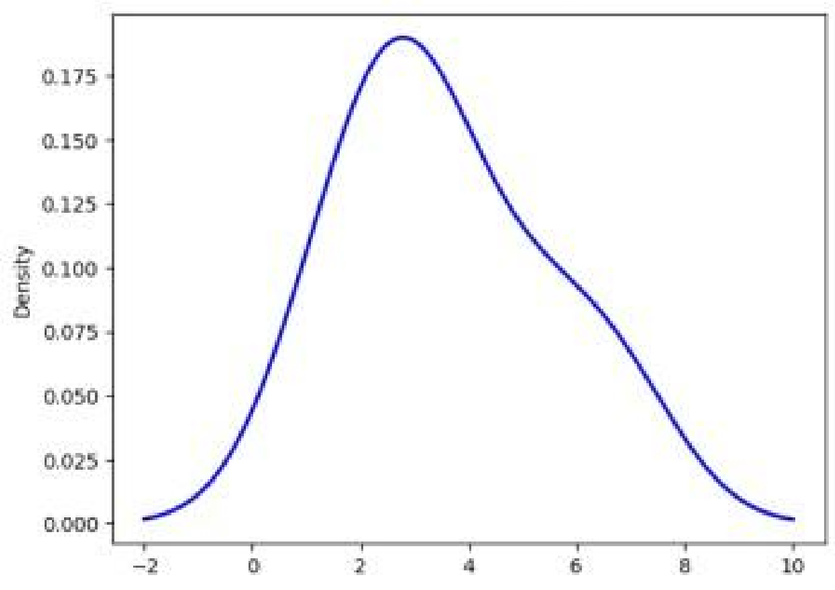
```
plt.figure(figsize=(10, 8))

sns.swarmplot(x="quality", y="alcohol", data=df, palette='viridis')

plt.title('Swarm Plot for Quality and Alcohol')
plt.xlabel('Quality')
plt.ylabel('Alcohol')
plt.show()
```

CONTINUED.....

- Kernel Density Plots for Understanding variance in datasets:

A Density Plot (also known as a Kernel Density Plot) is a smooth curve that shows the distribution of data points across a range, similar to a histogram but without bars. Higher curves indicate more concentration of data in that area. It provides a clearer view of data distribution, useful for comparing datasets. In Pandas, you can create a density plot using the plot() function with Seaborn or Matplotlib. You can create a density plot using either of the following functions:

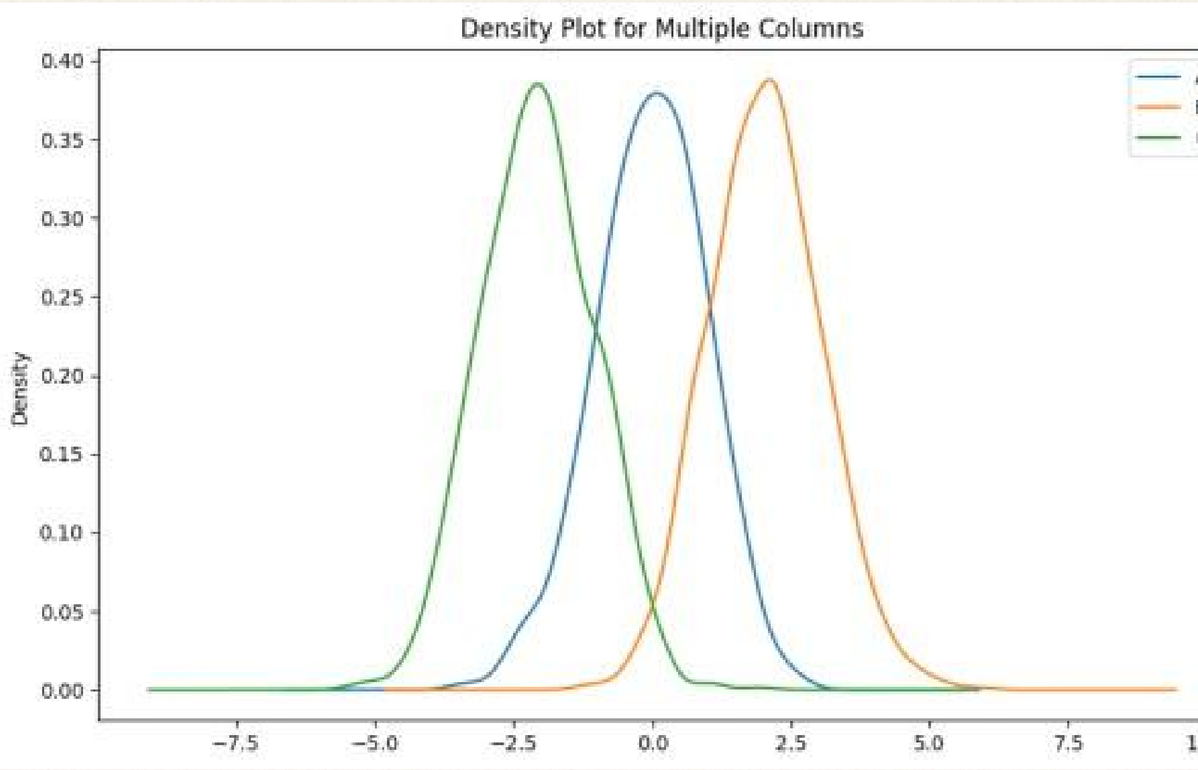
- pandas.DataFrame.plot.density()
- pandas.DataFrame.plot.kde()



CONTINUED.....

- CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.DataFrame({
    'A': np.random.randn(1000),
    'B': np.random.randn(1000) + 2,
    'C': np.random.randn(1000) - 2
})

# Plot density for all columns
df.plot.density(bw_method='silverman', figsize=(10, 6))

plt.title('Density Plot for Multiple Columns')
plt.show()
```



STEP 5

- BIVARIATE ANALYSIS

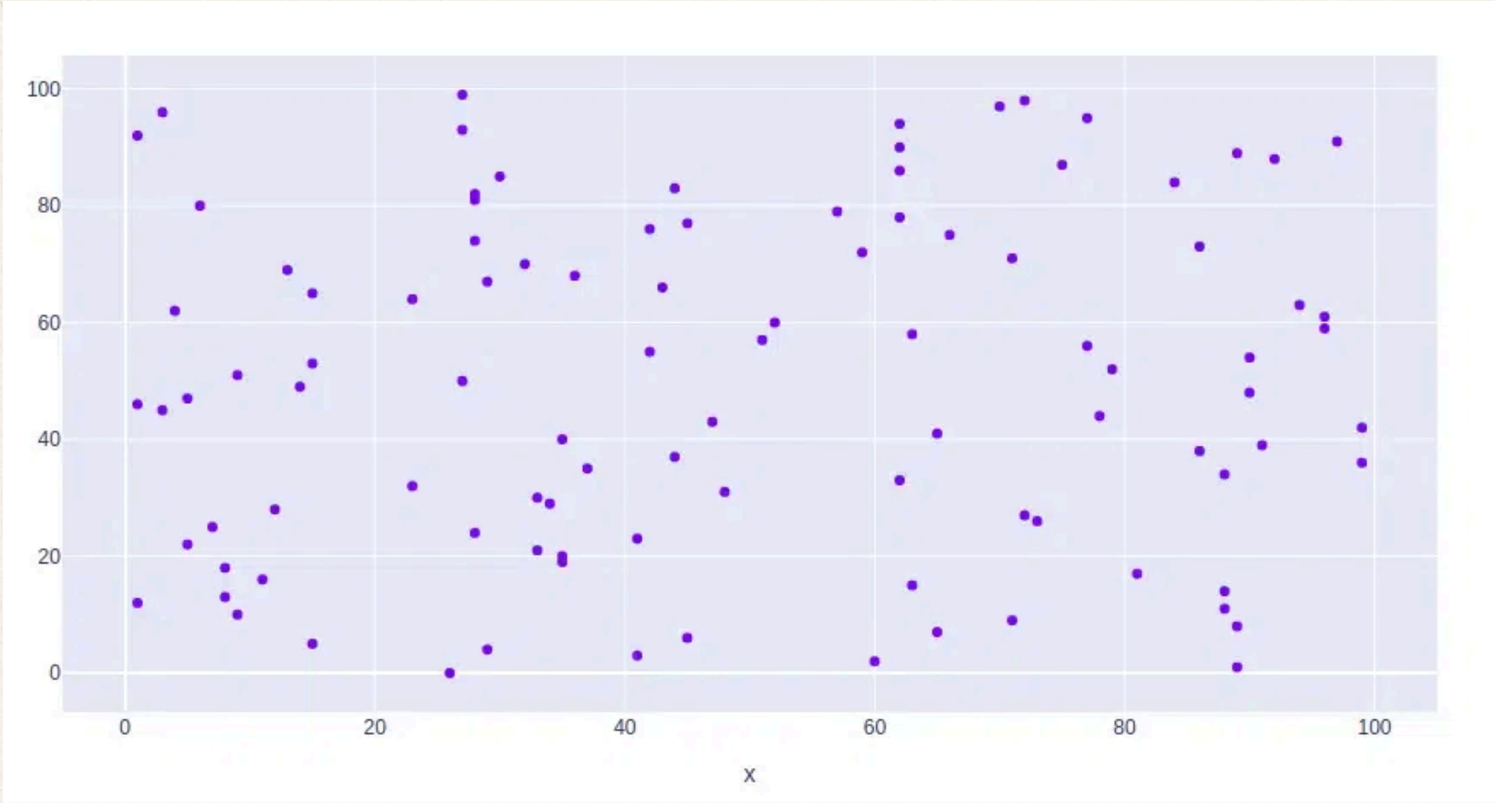
Bivariate analysis is a statistical method used to investigate the relationship between two variables. It is often used in quality-of-life research. It's a straightforward form of quantitative analysis which examines two variables denoted as X and Y. For instance, consider a study examining the relationship between exercise duration (X) and heart rate (Y) during physical activity. By analyzing this bivariate data, researchers can determine if there's a correlation between the duration of exercise and heart rate.

In bivariate analysis, researchers aim to understand to what extent one variable (possibly independent, like exercise duration) predicts the value of another (possibly dependent, like heart rate). It's a basic yet valuable tool for testing hypotheses of association. Bivariate analysis contrasts with univariate analysis, which examines only one variable at a time.

CONTINUED.....

- Example Scatter Plot for sepal width with sepal length:

```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length")
fig.show()
```



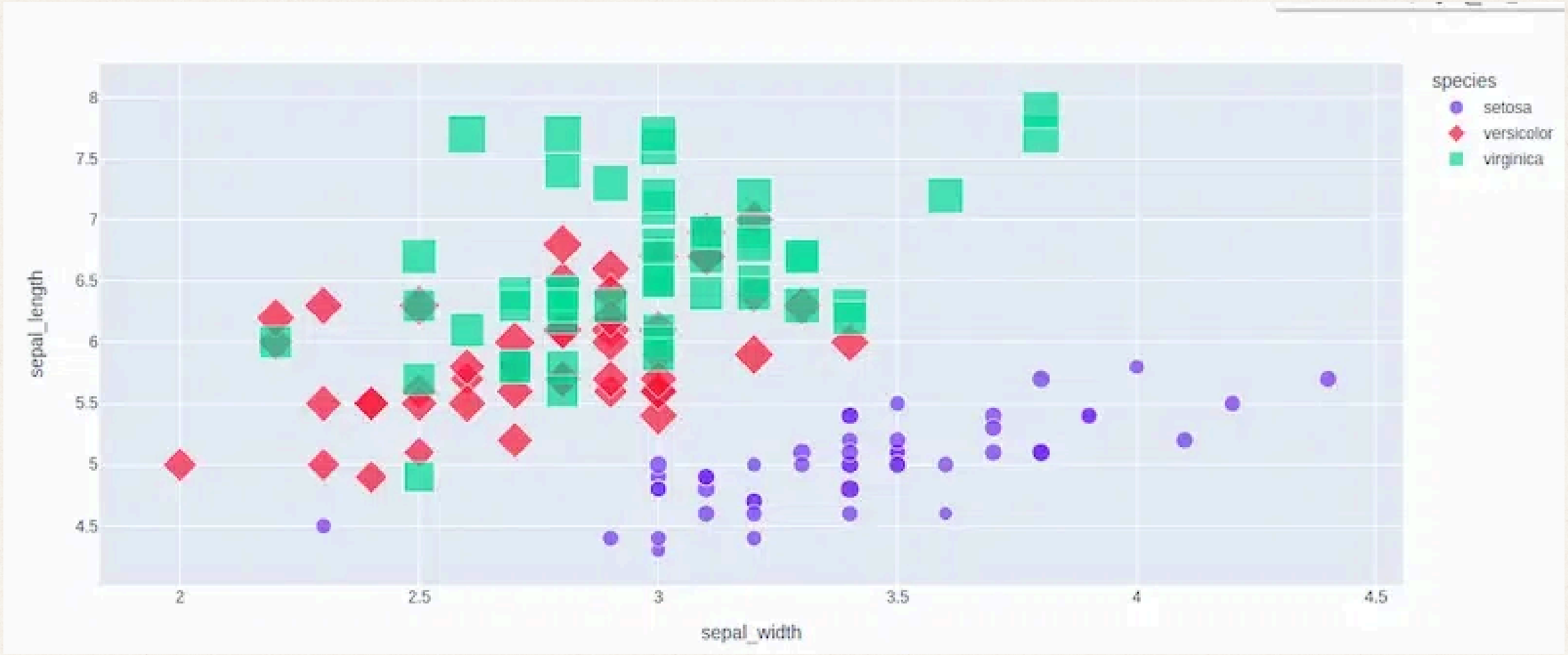
CONTINUED.....

- Example Scatter Plot for sepal width with sepal length:

```
import plotly.express as px
df = px.data.iris()

# Create a scatter plot with different symbols for each species
fig = px.scatter(df, x="sepal_width", y="sepal_length",
                 color="species", size='petal_length',
                 symbol="species",  # This parameter changes the symbols based on species
                 hover_data=['petal_width'])
fig.show()
```

CONTINUED.....

- Example Scatter Plot for sepal width with sepal length:



CONTINUED.....

- Correlation Analysis

The Pearson correlation coefficient is the most often used metric of correlation. It expresses the linear relationship between two variables in numerical terms. The Pearson correlation coefficient, written as "r," is as follows:

$$r_{xy} = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma(x_i - \bar{x})^2 \Sigma(y_i - \bar{y})^2}}$$

r_{xy} = correlation coefficient between x and y

x_i = the values of x within a sample

y_i = the values of y within a sample

$\bar{x}$ = the average of the values of x within a sample

$\bar{y}$ = the average of the values of y within a sample

CONTINUED.....

- Example code for correlation coefficient between 2 sample data:

```
import pandas as pd

# Create a DataFrame with sample data
data = pd.DataFrame({'X': [1, 2, 3, 4, 5], 'Y': [5, 7, 3, 9, 1]})

# Calculate correlation coefficient
correlation_coefficient = data['X'].corr(data['Y'])

print(&quot;Correlation Coefficient:&quot;, correlation_coefficient)
```

```
Correlation Coefficient: -0.3
```

CONTINUED.....

- Regression Analysis

Regression Analysis is a supervised learning analysis where supervised learning is the analyzing or predicting the data based on the previously available data or past data. For supervised learning, we have both train data and test data. Regression analysis is one of the statistical methods for the analysis and prediction of the data. Regression analysis is used for predictive data or quantitative or numerical data.
Types of Regression Analysis:

- Simple Linear Regression
- Multiple Linear Regression
- Exponential Regression
- Logistic regression

We'll study more about these in the next lecture!!!

THANK YOU!

CONTACT DETAILS

Parnika Mittal	+91 81263 34477	mparnika23@iitk.ac.in
Aayushman Kumar	+91 81088 04449	aayushmank23@iitk.ac.in
Ikrima Badr Shamim Ahmed	+91 80978 44011	ikrimab23@iitk.ac.in