WHAT IS EDA IN MACHINE LEARNING??

Exploratory Data Analysis (EDA) is a crucial step in the data analysis process that involves visually and statistically exploring a dataset. The primary objectives of EDA are to understand the structure of the data, discover patterns, identify outliers, and generate hypotheses that can guide further analysis.

key aspects of Exploratory Data Analysis:

- Summary Statistics: Calculate and examine basic statistical measures such as mean, median, standard deviation, etc., to get a sense of the central tendency and dispersion of the data.
- Data Visualization: Use various graphical representations such as histograms, box plots, scatter plots, and more to visually inspect the distribution, relationships, and patterns within the data. Visualization is powerful for gaining insights that might not be apparent in raw data.
- Data Cleaning: Identify and handle missing data, outliers, or anomalies that may affect the quality and accuracy of the analysis. This step is crucial for ensuring the reliability of the results.
- Feature Engineering: Derive new variables or features from existing ones to enhance the dataset and potentially improve model performance.
- Correlation Analysis: Examine the relationships between variables to understand how they influence each other. This often involves calculating correlation coefficients.
- Pattern Recognition: Look for trends, clusters, or other patterns in the data that might be indicative of underlying structures.
- Hypothesis Generation: Use the insights gained from the analysis to form hypotheses that can be tested in subsequent stages of the data analysis process.

Exploratory Data Analysis (EDA) encompasses a variety of techniques and methods to analyse and visualize data. Here are some common types or approaches to EDA:

- Univariate Analysis:

Histograms: A graphical representation of the distribution of a univariate dataset.

- Box Plots: Display the distribution of data based on a five-number summary (minimum, first quartile, median, third quartile, maximum).
- Scatter Plots: Visualize the relationship between two continuous variables.
- Correlation Analysis: Quantify and assess the strength and direction of the linear relationship between two variables.
- Pair Plots: Display scatter plots for all possible pairs of numerical features in a dataset.
- Heatmaps: Visualize the correlation matrix of multiple variables using color gradients.
- Time Series Plots: Display data points in chronological order to observe patterns or trends over time.

Exploratory Data Analysis (EDA) plays a crucial role in the machine learning (ML) pipeline. It helps data scientists and machine learning practitioners to understand the structure and characteristics of the data before building and training ML models. Here's more on the relationship between EDA and machine learning:

- Data Preprocessing:

EDA often reveals missing values, outliers, or inconsistencies in the data. Addressing these issues is part of the data preprocessing phase in machine learning.

Techniques such as imputation, outlier handling, and normalization may be applied based on insights gained during EDA.

- Feature Engineering:

EDA can guide feature selection and engineering by identifying relevant features that contribute to the predictive power of the model.

Understanding relationships between variables during EDA may suggest transformations or combinations of features to improve model performance.

- Model Selection:

EDA insights may influence the choice of machine learning models. For instance, understanding the distribution of the target variable may guide whether a regression or classification model is more appropriate.

EDA can help identify the complexity of the relationships in the data, guiding the selection of models that can capture these patterns.

- Hyperparameter Tuning:

Parameters of machine learning models can be tuned based on insights gained from EDA. For example, the range of hyperparameters for a decision tree model may be adjusted based on the observed variability in the data.

- Data Understanding for Stakeholders:

EDA results are often visualized and communicated to stakeholders, providing a clear understanding of the data's characteristics and challenges.

Clear communication resulting from EDA can help align expectations and goals between data scientists and stakeholders.

- Detecting Data Drift:

Continuous monitoring of data distributions, a part of ongoing EDA, can help detect data drift. This is essential for ensuring model performance over time as the underlying data may change.

- Optimizing Model Performance:

EDA insights can be used to optimize and fine-tune machine learning models. For example, if certain subsets of the data exhibit different characteristics, models may be customized or ensemble methods employed.

Interpreting Model Outputs:

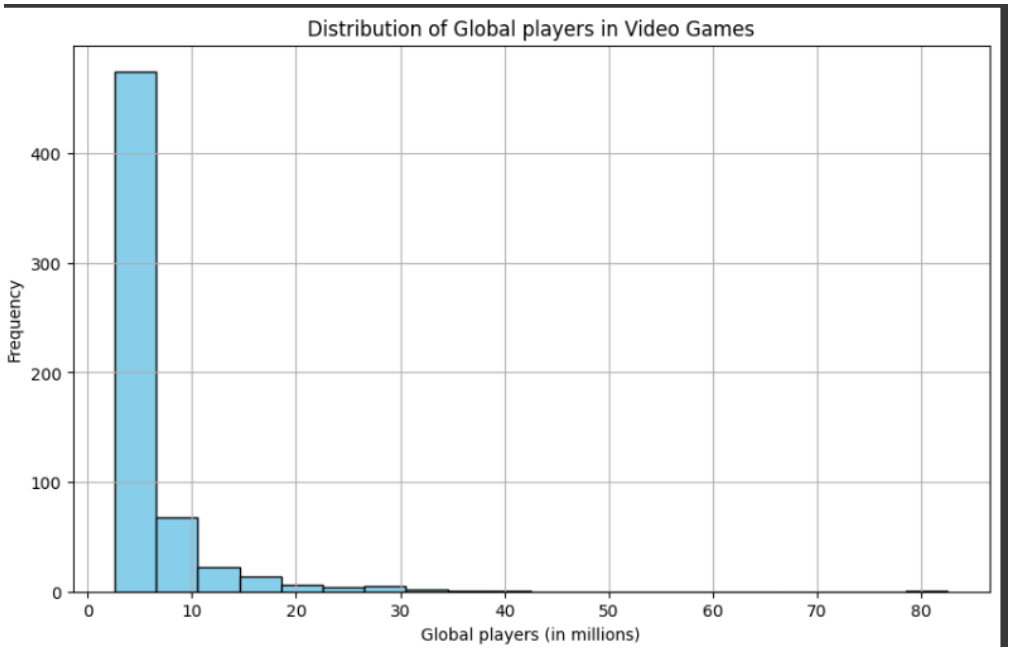
Understanding the characteristics of the input data through EDA aids in the interpretation of model outputs. This is particularly important for models where interpretability is crucial, such as in healthcare or finance.

FUNCTIONS IN EDA WITH CODE SNIPPETS :

1. Summary analysis :
   Description: Get an overview of the dataset's central tendencies and distributions.
   Code:
   `summary_stats = df.describe()`
2. Missing Values :
   Description : Visualize the distribution of a numerical variable.
   Code :
   `missing_values = df.isnull().sum()`
3. Histogram :
   Description : Identify outliers and visualize the distribution of a numerical variable.

```
# Load your video game dataset
# Replace 'your_dataset.csv' with the actual file path or URL of your
dataset
video_games_df = pd.read_csv('video_game.csv')

# Choose a numerical column for the histogram (e.g., 'Global_Sales')
numerical_column = 'Global_players'

# Create a histogram
plt.figure(figsize=(10, 6))
plt.hist(video_games_df[numerical_column], bins=20, color='skyblue',
edgecolor='black')
plt.title('Distribution of Global players in Video Games')
plt.xlabel('Global players (in millions)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

output :

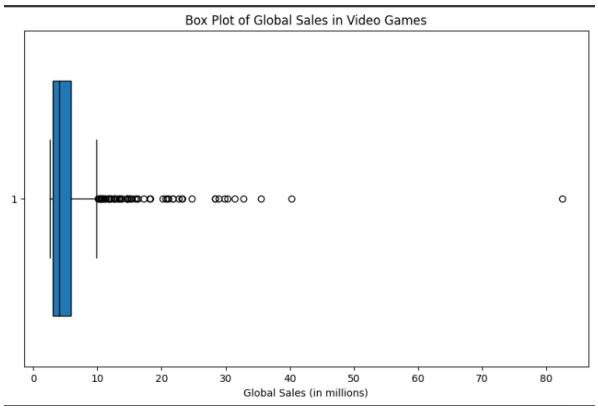


4. Box Plot:

Description: Identify outliers and visualize the distribution of a numerical variable.

```
    # Create a box plot using matplotlib

plt.figure(figsize=(10, 6))
plt.boxplot(video_games_df[numerical_column], vert=False, widths=0.7,
patch_artist=True, medianprops={'color':'black'})
plt.title('Box Plot of Global Sales in Video Games')
plt.xlabel('Global Sales (in millions)')
plt.show()
```

output :



5. Correlation Matrix:

   Description: Explore relationships between numerical variables.

   Code :

```
# Select numerical columns for correlation analysis
numerical_columns =
video_games_df.select_dtypes(include='number').columns

# Create a correlation matrix
correlation_matrix = video_games_df[numerical_columns].corr()

# Display the correlation matrix
print(correlation_matrix)
```

   Output:

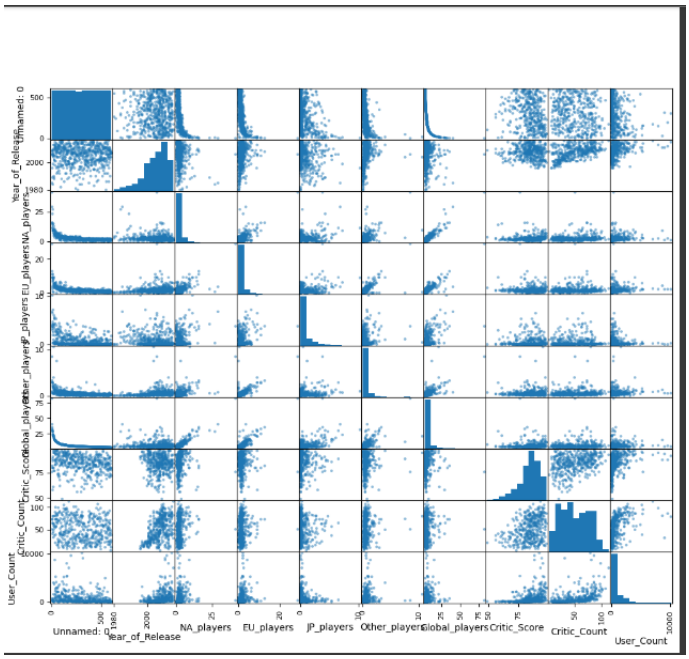
```
                 Unnamed: 0  Year_of_Release  NA_players  EU_players  \
Unnamed: 0         1.000000        -0.003298   -0.535721   -0.531012
Year_of_Release   -0.003298         1.000000   -0.138876    0.154587
NA_players        -0.535721        -0.138876    1.000000    0.652980
EU_players        -0.531012         0.154587    0.652980    1.000000
JP_players        -0.404714        -0.222176    0.366535    0.332400
Other_players     -0.409136         0.203395    0.492003    0.608607
Global_players    -0.617900        -0.045608    0.918441    0.851586
Critic_Score      -0.196164        -0.056531    0.049258   -0.008439
Critic_Count      -0.144198         0.490660    0.137865    0.060190
User_Count        -0.253525         0.364947    0.109091
0.113670
```

6. Pair Plot:
   Description: Visualize pairwise relationships between numerical variables.
   Code:

```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix

# Load your video game dataset
# Replace 'your_dataset.csv' with the actual file path or URL of your
dataset
video_games_df = pd.read_csv('video_game.csv')

# Select numerical columns for pair plot
numerical_columns =
video_games_df.select_dtypes(include='number').columns

# Create a pair plot
scatter_matrix(video_games_df[numerical_columns], figsize=(12, 10))
```

```
plt.suptitle('Pair Plot for Video Game Dataset', y=1.02)
plt.show()
```

Output:



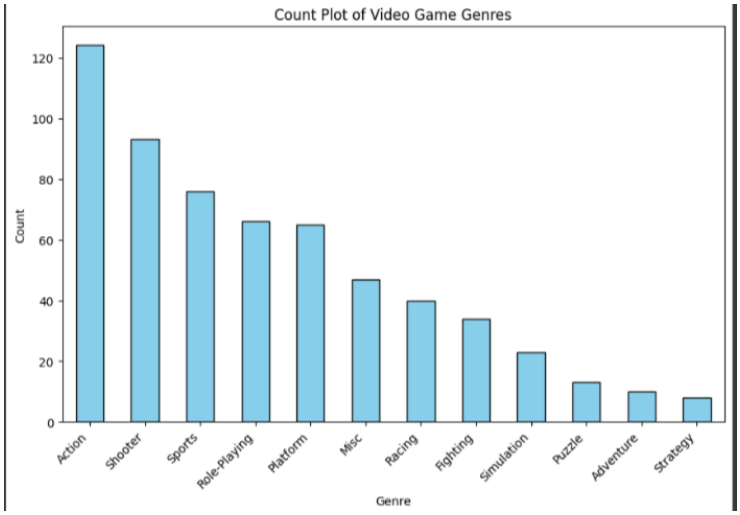
7. Count Plot:

Description: Display the distribution of categorical variables.
Code:

```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix

# Load your video game dataset
# Replace 'your_dataset.csv' with the actual file path or URL of your
dataset
video_games_df = pd.read_csv('video_game.csv')

# Choose a categorical column for the count plot (e.g., 'Genre')
categorical_column = 'Genre'

# Count the occurrences of each category
category_counts = video_games_df[categorical_column].value_counts()

# Create a count plot
plt.figure(figsize=(10, 6))
category_counts.plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('Count Plot of Video Game Genres')
plt.xlabel('Genre')
```

```
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.show()
```

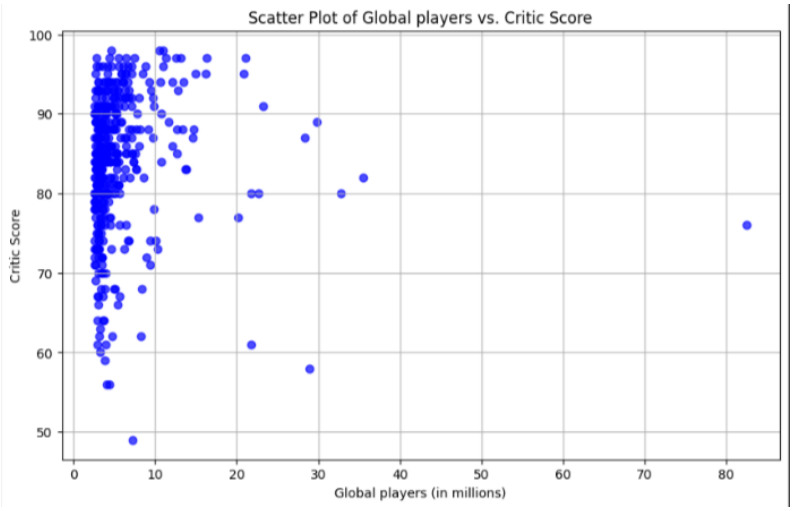
Output:



8. Scatter Plot:
   Description: Explore relationships between two numerical variables.
   Code:

```
# Choose two numerical columns for the scatter plot (e.g.,
'Global_Sales' and 'Critic_Score')
x_column = 'Global_players'
y_column = 'Critic_Score'

# Create a scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(video_games_df[x_column], video_games_df[y_column],
alpha=0.7, color='blue')
plt.title('Scatter Plot of Global players vs. Critic Score')
plt.xlabel('Global players (in millions)')
plt.ylabel('Critic Score')
plt.grid(True)
plt.show()
```

Output:

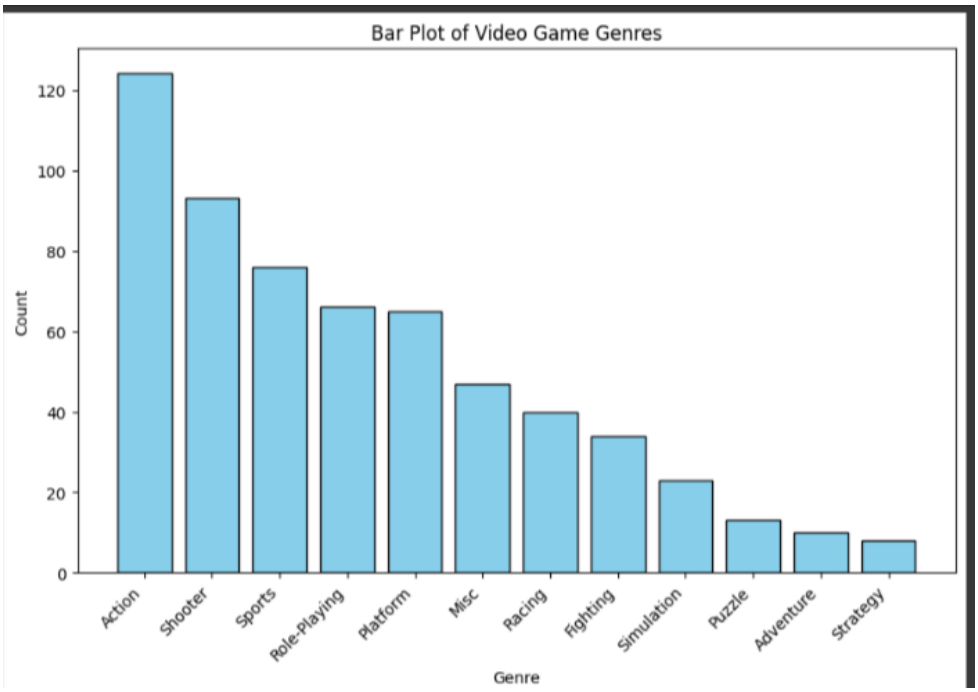


9. Bar Plot:

Description: Visualize the distribution or aggregation of data for categorical variables.
Code:

```
# Choose a categorical column for the bar plot (e.g., 'Genre')
categorical_column = 'Genre'

# Count the occurrences of each category
category_counts = video_games_df[categorical_column].value_counts()

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.bar(category_counts.index, category_counts, color='skyblue',
edgecolor='black')
plt.title('Bar Plot of Video Game Genres')
plt.xlabel('Genre')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.show()
```

Output:


Bar Plot of Video Game Genres
Count
120
100
80
60
40
20
0
Action
Shooter
Sports
Role-Playing
Platform
Misc
Racing
Fighting
Simulation
Puzzle
Adventure
Strategy
Genre