Exploratory Data Analysis (EDA)

Ankit Rajput

Indian Institute of Technology, Guwahati

r.ankit@iitg.ac.in

11 November 2023

Contents

What is EDA?

- Exploratory Data Analysis is an approach in data analysis that focuses on summarizing, visualizing, and understanding the main characteristics of a dataset.
- Exploring data for patterns, trends, underlying structure, deviations from the trend, anomalies and strange structures.
- EDA allow us to get a better feel of our data and finds useful patterns in it.

The Foremost Goals of EDA

- **Data Cleaning**: EDA involves examining the information for errors, lacking values, and inconsistencies. It includes techniques including records imputation, managing missing statistics, and figuring out and getting rid of outliers.
- **Descriptive Statistics**: EDA utilizes precise records to recognize the important tendency, variability, and distribution of variables. Measures like suggest, median, mode, preferred deviation, range, and percentiles are usually used.
- **Data Visualization**: EDA employs visual techniques to represent the statistics graphically. Visualizations consisting of histograms, box plots, scatter plots, line plots, heatmaps, and bar charts assist in identifying styles, trends, and relationships within the facts.

The Foremost Goals of EDA

- **Feature Engineering**: EDA allows for the exploration of various variables and their adjustments to create new functions or derive meaningful insights. Feature engineering can contain scaling, normalization, binning and encoding variables.
- **Correlation and Relationships**: EDA allows discover relationships and dependencies between variables. Techniques such as correlation analysis, scatter plots, and pass-tabulations offer insights into the power and direction of relationships between variables.
- **Data Segmentation**: EDA can contain dividing the information into significant segments based totally on sure standards or traits. This segmentation allows advantage insights into unique subgroups inside the information and might cause extra focused analysis.

Types of EDA

Depending on the number of columns, There are various types of EDA techniques and methods, including:

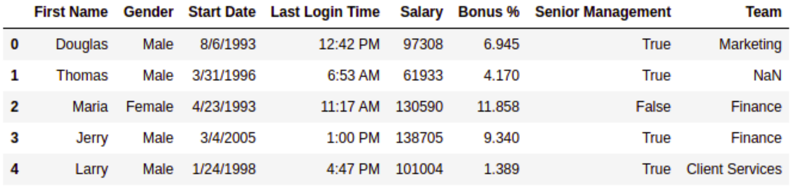
- **Univariate Analysis**: This type of EDA focuses on analyzing individual variables or features in the dataset. It involves techniques like histograms, bar charts, pie charts, summary statistics, and distribution plots to understand the characteristics of each variable.
- **Bivariate Analysis**: Bivariate EDA explores the relationship between two variables. Common techniques include scatter plots, correlation analysis, and two-way frequency tables. It helps in understanding how two variables interact with each other.

- **Multivariate Analysis**: Multivariate EDA extends the analysis to more than two variables simultaneously. It often involves techniques like heatmap correlation matrices, parallel coordinates plots, and 3D scatter plots to explore complex relationships within the dataset.
- **Outlier Detection**: Identifying and analyzing outliers is a critical aspect of EDA. Methods such as box plots, z-scores, and the IQR (Interquartile Range) can help detect and understand the impact of outliers on the dataset.

Exploratory Data Analysis (EDA) Using Python Libraries

We will use the employee data for this. It contains 8 columns namely – First Name, Gender, Start Date, Last Login, Salary, Bonus, Senior Management, and Team.

Dataset:



	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services

Getting Insights About The Dataset

Let's read the dataset using the Pandas.

```
import pandas as pd
import numpy as np
# read datasdet using pandas
df = pd.read_csv('employees.csv')
df.head()
```

shape of the data using the shape.

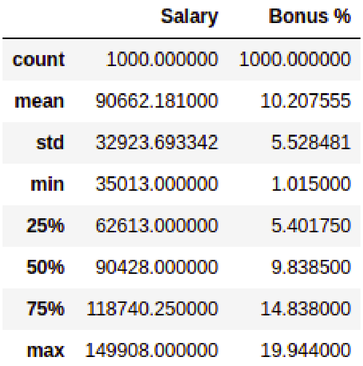
```
df.shape
```

Output: (1000, 8)

This means that this dataset has 1000 rows and 8 columns.

```
df.describe()
```

The describe() function applies basic statistical computations on the dataset like extreme values, count of data points standard deviation, etc. Any missing value or NaN value is automatically skipped.



	Salary	Bonus %
count	1000.000000	1000.000000
mean	90662.181000	10.207555
std	32923.693342	5.528481
min	35013.000000	1.015000
25%	62613.000000	5.401750
50%	90428.000000	9.838500
75%	118740.250000	14.838000
max	149908.000000	19.944000

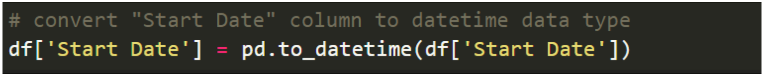
```
# information about the dataset
df.info()
```

To get the columns and their data types. For this, we will use the info() method.

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   First Name         933 non-null    object
 1   Gender             855 non-null    object
 2   Start Date         1000 non-null   object
 3   Last Login Time    1000 non-null   object
 4   Salary             1000 non-null   int64
 5   Bonus %            1000 non-null   float64
 6   Senior Management  933 non-null    object
 7   Team               957 non-null    object
dtypes: float64(1), int64(1), object(6)
```

- Changing Dtype from Object to Datetime
  Start Date is an important column for employees. However, it is not of much use if we can not handle it properly to handle this type of data pandas provide a special function datetime() from which we can change object type to DateTime format.

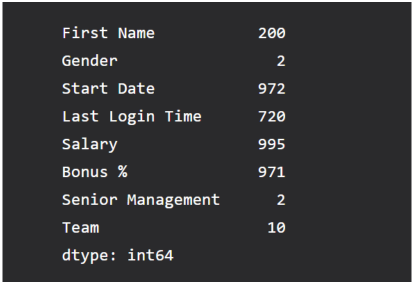


```
# convert "Start Date" column to datetime data type
df['Start Date'] = pd.to_datetime(df['Start Date'])
```

- We can see the number of unique elements in our dataset. This will help us in deciding which type of encoding to choose for converting categorical columns into numerical columns.

```
df.nunique()
```

Output:



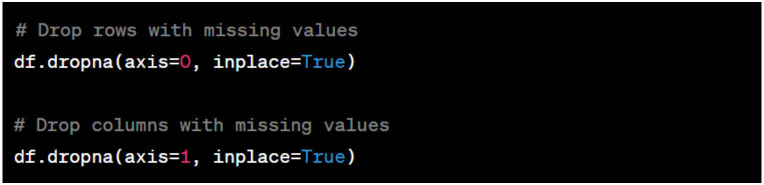
```
First Name           200
Gender                 2
Start Date           972
Last Login Time      720
Salary               995
Bonus %              971
Senior Management      2
Team                  10
dtype: int64
```

Handling Missing Values

- Why there is missing value in dataset: It can occur when no information is provided for one or more items or for a whole unit. For Example, Suppose different users being surveyed may choose not to share their income, and some users may choose not to share their address in this way many datasets went missing.

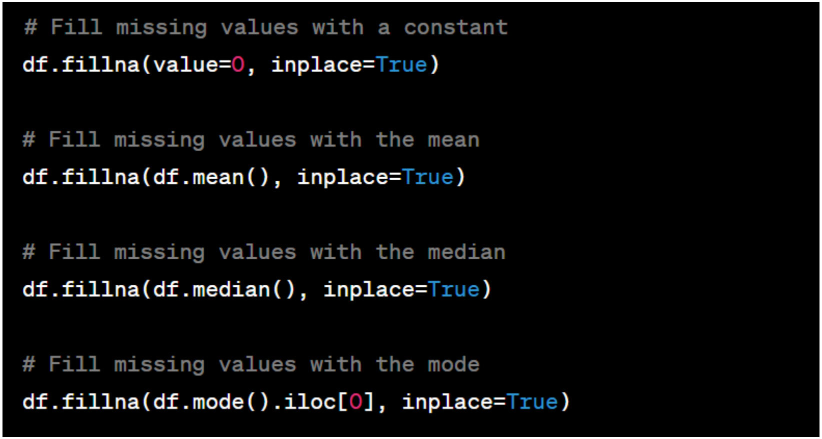
To handle this, there are several ways:

1. Removing Rows or Columns:



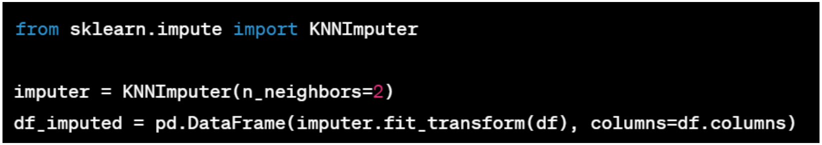
```
# Drop rows with missing values
df.dropna(axis=0, inplace=True)

# Drop columns with missing values
df.dropna(axis=1, inplace=True)
```

2. Imputation:
Fill missing values with a specific constant, mean, median, or mode.



```python
# Fill missing values with a constant
df.fillna(value=0, inplace=True)

# Fill missing values with the mean
df.fillna(df.mean(), inplace=True)

# Fill missing values with the median
df.fillna(df.median(), inplace=True)

# Fill missing values with the mode
df.fillna(df.mode().iloc[0], inplace=True)
```

3. Using Machine Learning Models:

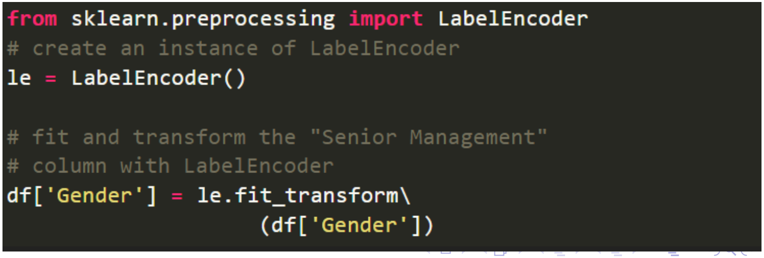


```
from sklearn.impute import KNNImputer

imputer = KNNImputer(n_neighbors=2)
df_imputed = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)
```

There are other techniques also, we have to choose the method that best suits our data and problem context. It's often a good practice to analyze the reasons for missing values before deciding on the imputation strategy.

Data Encoding

- There are some models like Linear Regression which does not work with categorical dataset in that case we should try to encode categorical dataset into the numerical column. we can use different methods for encoding like Label encoding or One-hot encoding.
- pandas and sklearn provide different functions for encoding in our case we will use the LabelEncoding function from sklearn to encode the Gender column.



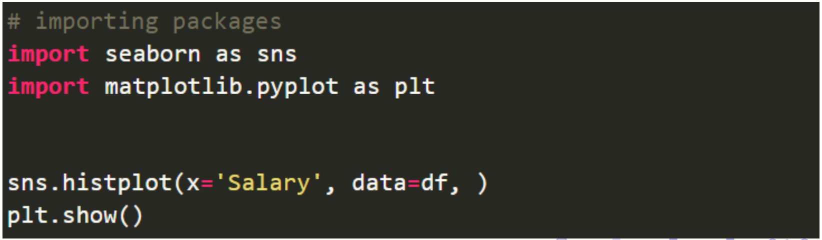
```python
from sklearn.preprocessing import LabelEncoder
# create an instance of LabelEncoder
le = LabelEncoder()

# fit and transform the "Senior Management"
# column with LabelEncoder
df['Gender'] = le.fit_transform\
                   (df['Gender'])
```

Data visualization

Data visualization is a powerful way to explore, analyze, and communicate patterns and insights in your data. In Python, there are several libraries that you can use for creating effective visualizations.
We will use Matplotlib and Seaborn library for the data visualization.

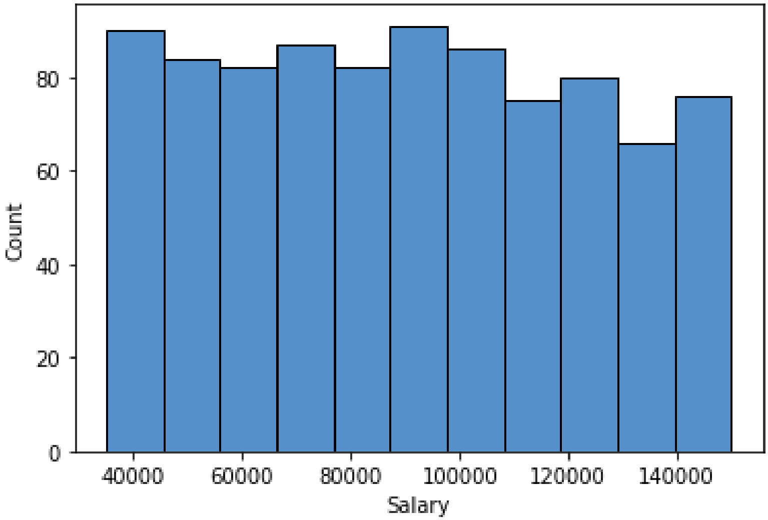
- Histogram

It can be used for both uni and bivariate analysis.

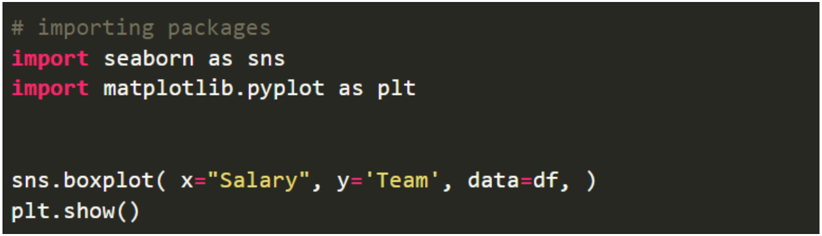


```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt


sns.histplot(x='Salary', data=df, )
plt.show()
```

Output:

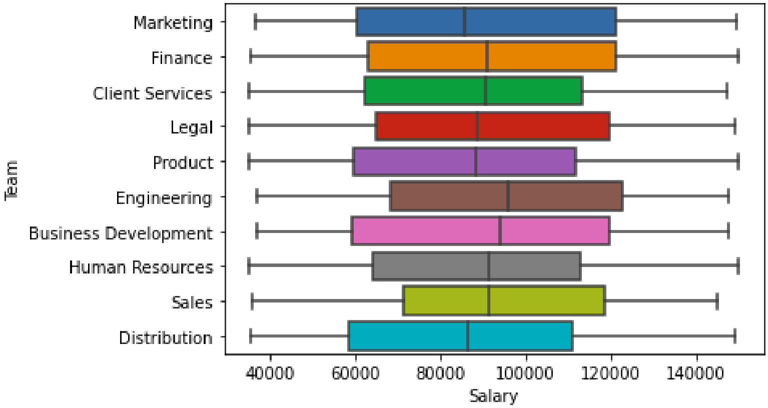

Count
0
20
40
60
80
40000
60000
80000
100000
120000
140000
Salary

- Boxplot
  It can also be used for univariate and bivariate analyses.



```python
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt


sns.boxplot( x="Salary", y='Team', data=df, )
plt.show()
```

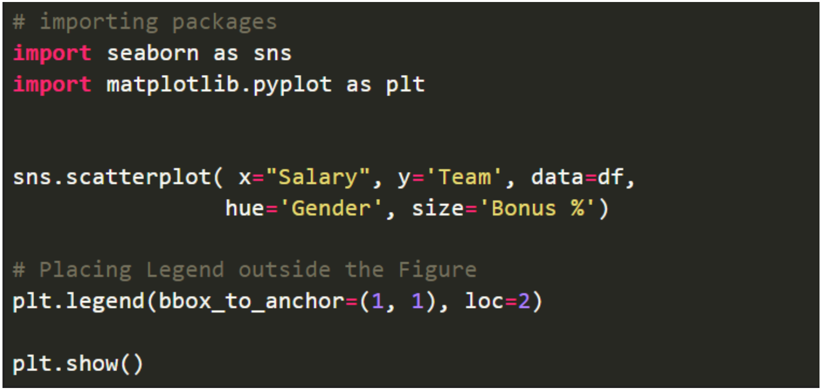
Output:



Marketing
Finance
Client Services
Legal
Product
Engineering
Business Development
Human Resources
Sales
Distribution
Team
40000
60000
80000
100000
120000
140000
Salary

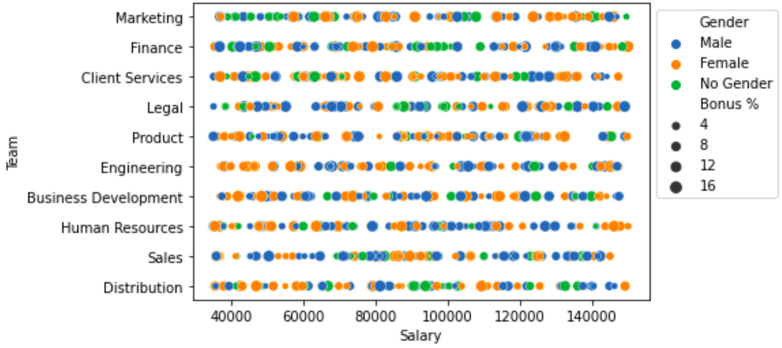
- Scatter plot

It can be used for bivariate analyses.



```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt


sns.scatterplot( x="Salary", y='Team', data=df,
                hue='Gender', size='Bonus %')

# Placing Legend outside the Figure
plt.legend(bbox_to_anchor=(1, 1), loc=2)

plt.show()
```

Output:

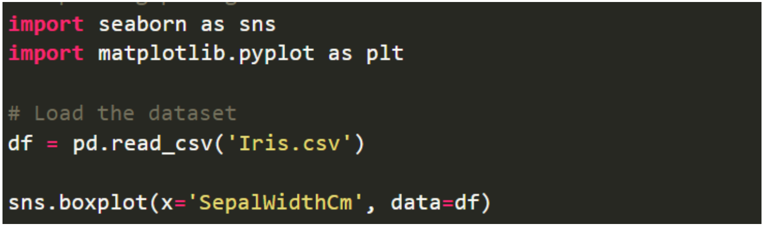

Marketing
Finance
Client Services
Legal
Product
Engineering
Business Development
Human Resources
Sales
Distribution
Team
40000
60000
80000
100000
120000
140000
Salary
Gender
Male
Female
No Gender
Bonus %
4
8
12
16

Handling Outliers

- Outliers are data points that deviate significantly from the rest of the data in a dataset. Handling outliers is important to prevent them from skewing statistical analysis or machine learning models

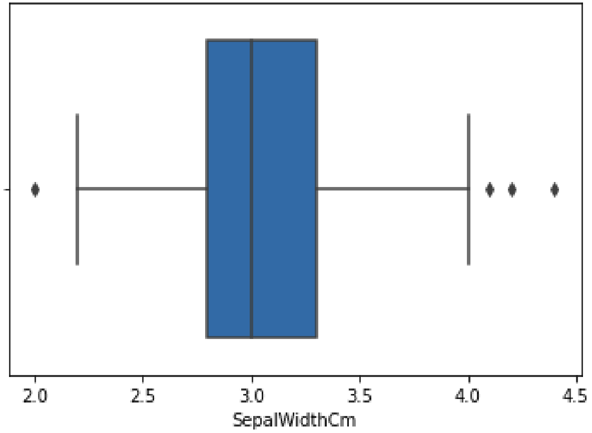
Identifying Outliers

1. Visual Inspection:
   - Use box plots, scatter plots, or histograms to visually inspect the data for any unusual patterns or points that are far from the bulk of the data.



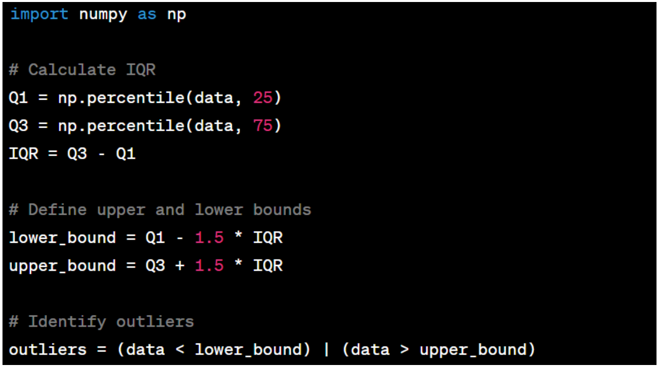
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv('Iris.csv')

sns.boxplot(x='SepalWidthCm', data=df)
```

Output:



the values above 4 and below 2 are acting as outliers.

2. Summary Statistics:
   - Calculate summary statistics like mean, median, standard deviation, and interquartile range (IQR) to identify values that fall outside normal ranges.



```python
import numpy as np

# Calculate IQR
Q1 = np.percentile(data, 25)
Q3 = np.percentile(data, 75)
IQR = Q3 - Q1

# Define upper and lower bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identify outliers
outliers = (data < lower_bound) | (data > upper_bound)
```

Conclusion

- The Exploratory Data Analysis (EDA) revealed valuable insights into the dataset, uncovering key patterns, trends, and relationships. Through visualizations and statistical summaries, we gained a comprehensive understanding of the data's distributions and identified potential outliers. The data cleaning and preprocessing steps addressed missing values and improved the overall quality of the dataset.

References

- https://www.geeksforgeeks.org/what-is-exploratory-data-analysis/
- https://pandas.pydata.org/
- https://matplotlib.org/
- https://seaborn.pydata.org/
- https://numpy.org/

Thank You