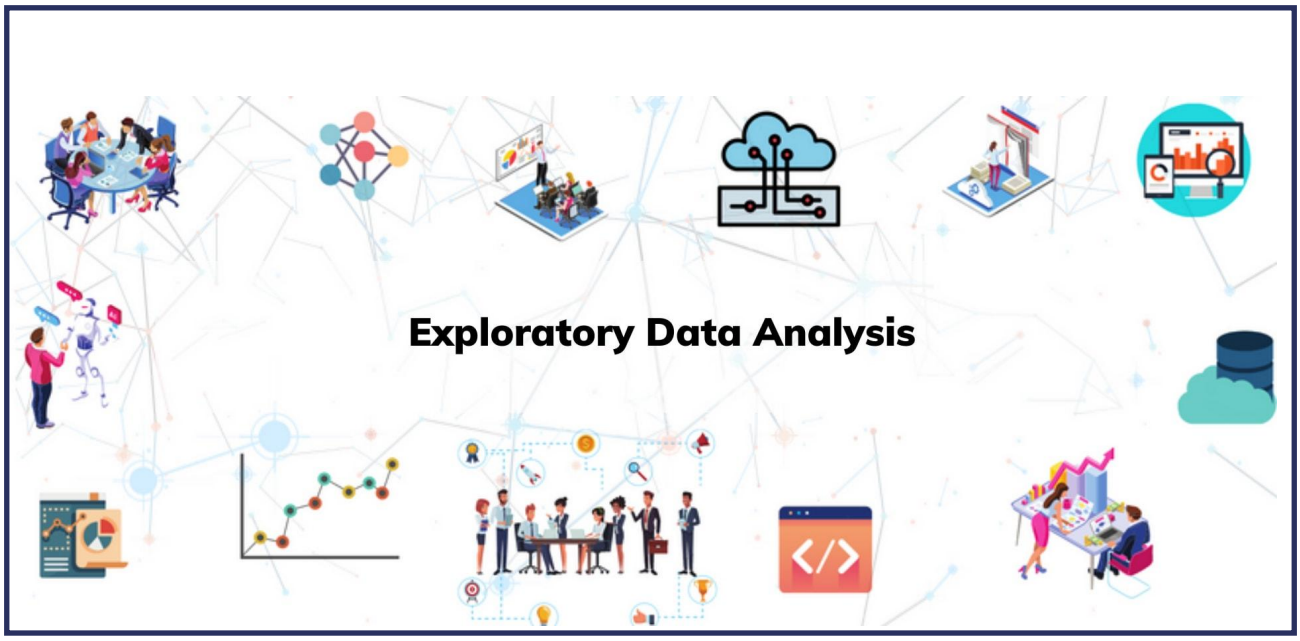
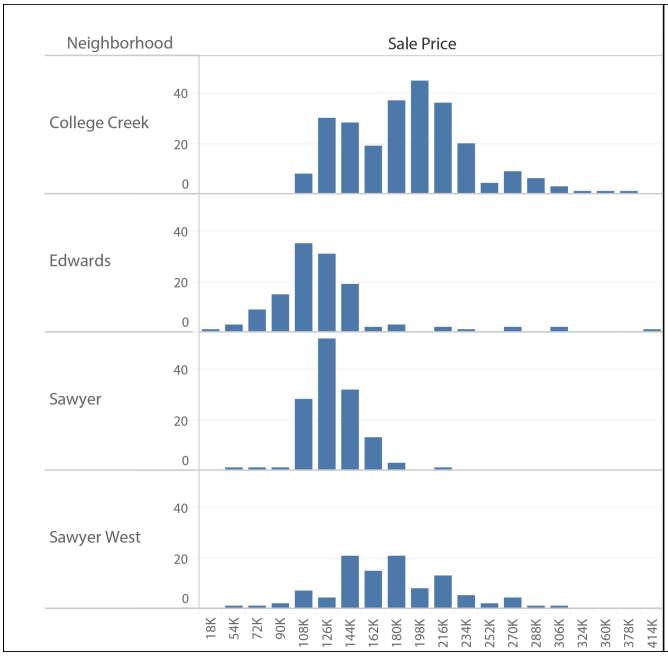
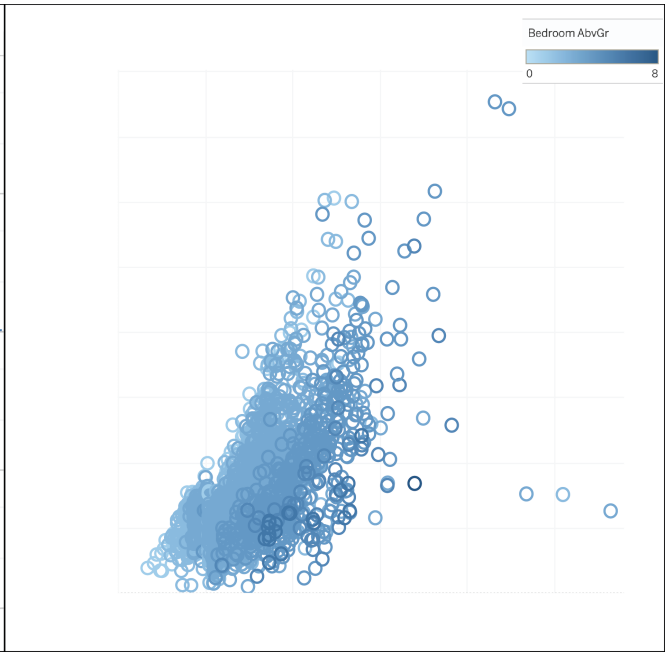



# What is Exploratory Data Analysis ?




Follow: siddhesh Kumbhar @ www.linkedin.com/in/siddhesh-kumbhar-517627b4

## What is Exploratory Data Analysis ?

**Exploratory Data Analysis (EDA)** is an approach to analyze the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.

## Dataset Used

For the simplicity of the article, we will use a single dataset. 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 Used:** Employees.csv

Let's read the dataset using the Pandas module and print the 1st five rows. To print the first five rows we will use the **head()** function.

**Example:**

Python3

```
import pandas as pd
import numpy as np

df = pd.read_csv('employees.csv')
df.head()
```

**Output:**

| | 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 see the shape of the data using the shape.

**Example:**

Python3

```
df.shape
```

**Output:**

```
(1000, 8)
```

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

Let's get a quick summary of the dataset using the **describe()** method. 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. describe() function gives a good picture of the distribution of data.

**Example:**

Python3

```
df.describe()
```

**Output:**

| | 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 |

Now, let's also the columns and their data types. For this, we will use the info() method.

Python3

```
df.info()
```

**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)
memory usage: 62.6+ KB
```

Till now we have got an idea about the dataset used. Now Let's see if our dataset contains any missing value or not.

## Handling Missing Values

You all must be wondering why a dataset will contain any missing value. 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, some users may choose not to share the address in this way many datasets went missing. Missing Data is a very big problem in real-life scenarios. Missing Data can also refer to as NA(Not Available) values in pandas. There are several useful functions for detecting, removing, and replacing null values in Pandas DataFrame :

- isnull()
- notnull()
- dropna()
- fillna()
- replace()
- interpolate()

Now let's check if there are any missing values in our dataset or not.

**Example:**

Python3

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

**Output:**

```
First Name           67
Gender              145
Start Date            0
Last Login Time       0
Salary                0
Bonus %               0
Senior Management    67
Team                 43
dtype: int64
```

We can see that every column has a different amount of missing values. Like Gender as 145 missing values and salary has 0. Now for handling these missing values there can be several cases like dropping the rows containing NaN or replacing NaN with either mean, median, mode, or some other value.

Now, let's try to fill the missing values of gender with the string "No Gender".

**Example:**

Python3

```
df["Gender"].fillna("No Gender", inplace = True)

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

**Output:**

```
First Name           67
Gender                0
Start Date            0
Last Login Time       0
Salary                0
Bonus %               0
Senior Management    67
Team                 43
dtype: int64
```

We can see that now there is no null value for the gender column. Now, Let's fill the senior management with the mode value.

**Example:**

Python3

```
mode = df['Senior Management'].mode().values[0]
df['Senior Management']= df['Senior Management'].replace(np.nan, mode)

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

**Output:**

```
First Name           67
Gender                0
Start Date            0
Last Login Time       0
Salary                0
Bonus %               0
Senior Management     0
Team                 43
dtype: int64
```

Now for the first name and team, we cannot fill the missing values with arbitrary data, so, let's drop all the rows containing these missing values.

**Example:**

Python3

```
df = df.dropna(axis = 0, how ='any')

print(df.isnull().sum())
df.shape
```

**Output:**

```
First Name           0
Gender               0
Start Date           0
Last Login Time      0
Salary               0
Bonus %              0
Senior Management    0
Team                 0
dtype: int64

(899, 8)
```

We can see that our dataset is now free of all the missing values and after dropping the data the number of also reduced from 1000 to 899.

**Note:** For more information, refer Working with Missing Data in Pandas.

After removing the missing data let's visualize our data.

# Data visualization

Data Visualization is the process of analyzing data in the form of graphs or maps, making it a lot easier to understand the trends or patterns in the data. There are various types of visualizations -

- **Univariate analysis:** This type of data consists of only one variable. The analysis of univariate data is thus the simplest form of analysis since the information deals with only one quantity that changes. It does not deal with causes or relationships and the main purpose of the analysis is to describe the data and find patterns that exist within it.
- **Bi-Variate analysis:** This type of data involves two different variables. The analysis of this type of data deals with causes and relationships and the analysis is done to find out the relationship among the two variables.
- **Multi-Variate analysis:** When the data involves three or more variables, it is categorized under multivariate.

Let's see some commonly used graphs -

> ***Note:*** *We will use Matplotlib and Seaborn library for the data visualization. If you want to know about these modules refer to the articles -*
>
> - *Matplotlib Tutorial*
> - *Python Seaborn Tutorial*

## Histogram

It can be used for both uni and bivariate analysis.

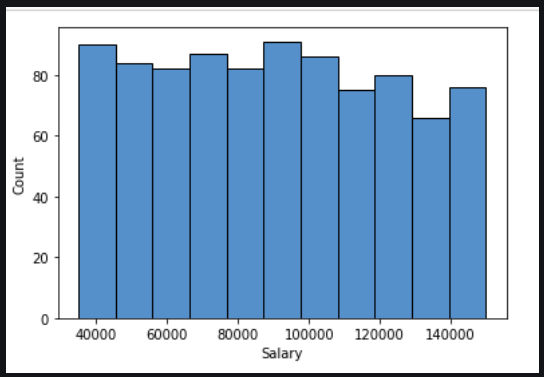
**Example:**

Python3

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


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

**Output:**




## Boxplot

It can also be used for univariate and bivariate analyses.

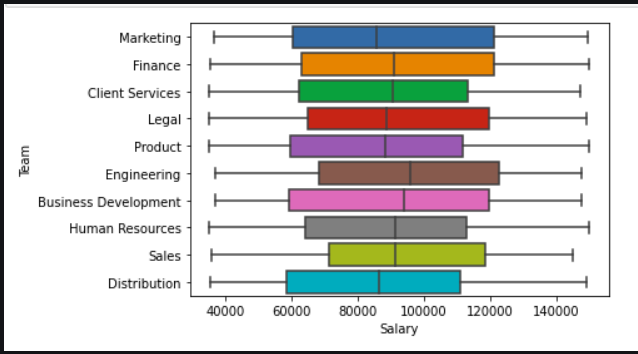
**Example:**

Python3

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


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

**Output:**




### Scatter Plot

It can be used for bivariate analyses.

**Example:**

Python3

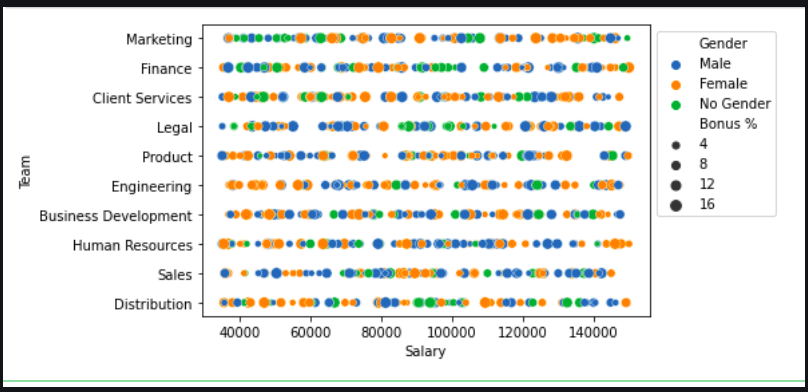
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt


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:**




For multivariate analysis, we can the pairplot()method of seaborn module. We can also use it for the multiple pairwise bivariate distributions in a dataset.

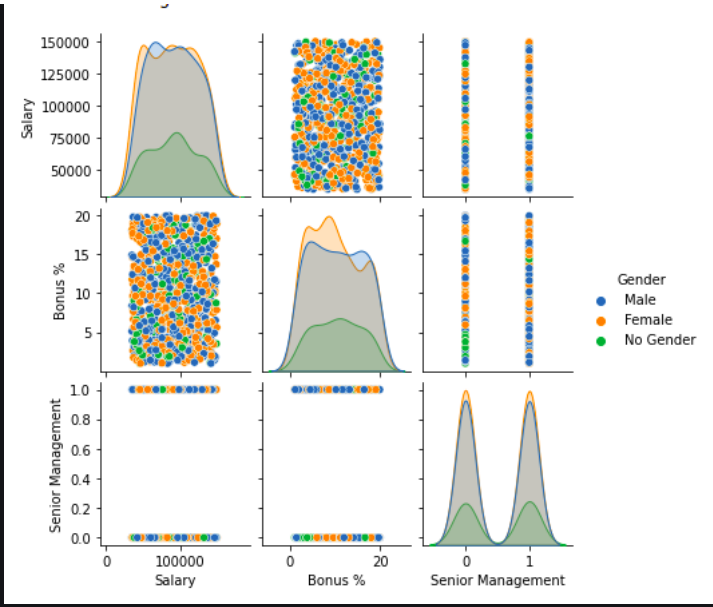
**Example:**

Python3

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


sns.pairplot(df, hue='Gender', height=2)
```

**Output:**




## Handling Outliers

An Outlier is a data-item/object that deviates significantly from the rest of the (so-called normal)objects. They can be caused by measurement or execution errors. The analysis for outlier detection is referred to as outlier mining. There are many ways to detect the outliers, and the removal process is the data frame same as removing a data item from the panda's dataframe.

Let's consider the iris dataset and let's plot the boxplot for the SepalWidthCm column.

**Example:**

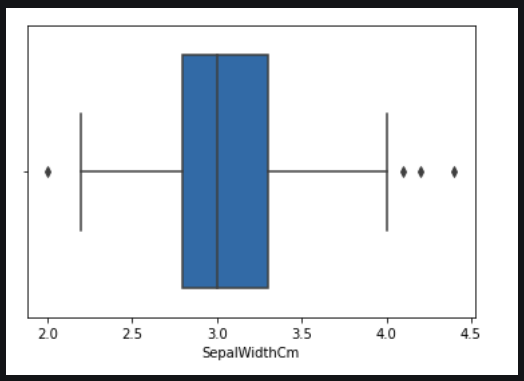
Python3

```
# importing packages
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:**




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

### Removing Outliers

For removing the outlier, one must follow the same process of removing an entry from the dataset using its exact position in the dataset because in all the above methods of detecting the outliers end result is the list of all those data items that satisfy the outlier definition according to the method used.

**Example:** We will detect the outliers using IQR and then we will remove them. We will also draw the boxplot to see if the outliers are removed or not.

Python3

```
# Importing
import sklearn
```

```
from sklearn.datasets import load_boston
import pandas as pd
import seaborn as sns

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

# IQR
Q1 = np.percentile(df['SepalWidthCm'], 25,
                   interpolation = 'midpoint')

Q3 = np.percentile(df['SepalWidthCm'], 75,
                   interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", df.shape)

# Upper bound
upper = np.where(df['SepalWidthCm'] >= (Q3+1.5*IQR))

# Lower bound
lower = np.where(df['SepalWidthCm'] <= (Q1-1.5*IQR))

# Removing the Outliers
df.drop(upper[0], inplace = True)
df.drop(lower[0], inplace = True)

print("New Shape: ", df.shape)

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

**Output:**

```
Old Shape:  (150, 6)
New Shape:  (146, 6)

<AxesSubplot:xlabel='SepalWidthCm'>
```