17-11-2024

Training Day – 41

Topic: Visualizing Cleaned Data

- Created histograms and scatter plots for cleaned datasets.
- Example: Visualized the distribution of sales data.

Visualizing cleaned data is an essential step in the data analysis process. Once you've processed and cleaned your data (by removing outliers, handling missing values, normalizing, etc.), visualizations help uncover patterns, trends, and insights. Here are key visualization techniques to consider for cleaned data:

1. Histograms

- **Use:** To visualize the distribution of numerical variables.
- **Why:** Helps identify skewness, normality, and outliers in the data.
- **Tools:** Matplotlib, Seaborn, Plotly.

2. Box Plots

- **Use:** To summarize the distribution of a variable and show outliers.
- **Why:** Provides a five-number summary (minimum, Q1, median, Q3, maximum) and identifies anomalies.
- **Tools:** Matplotlib, Seaborn.

3. Bar Charts

- **Use:** To compare categorical variables.
- **Why:** Visualizes the frequency or proportion of categories.
- **Tools:** Matplotlib, Seaborn, Plotly.

4. Scatter Plots

- **Use:** To visualize relationships between two continuous variables.
- **Why:** Helps identify correlations or trends.
- **Tools:** Matplotlib, Seaborn, Plotly.

5. Pair Plots

- **Use:** To visualize relationships between multiple continuous variables.
- **Why:** Helps identify patterns or trends between variables in a multi-dimensional dataset.

6. Line Charts

- **Use:** To show trends over time.
- **Why:** Ideal for time-series data to observe changes over time.
- **Tools:** Matplotlib, Plotly.

7. Pie Charts

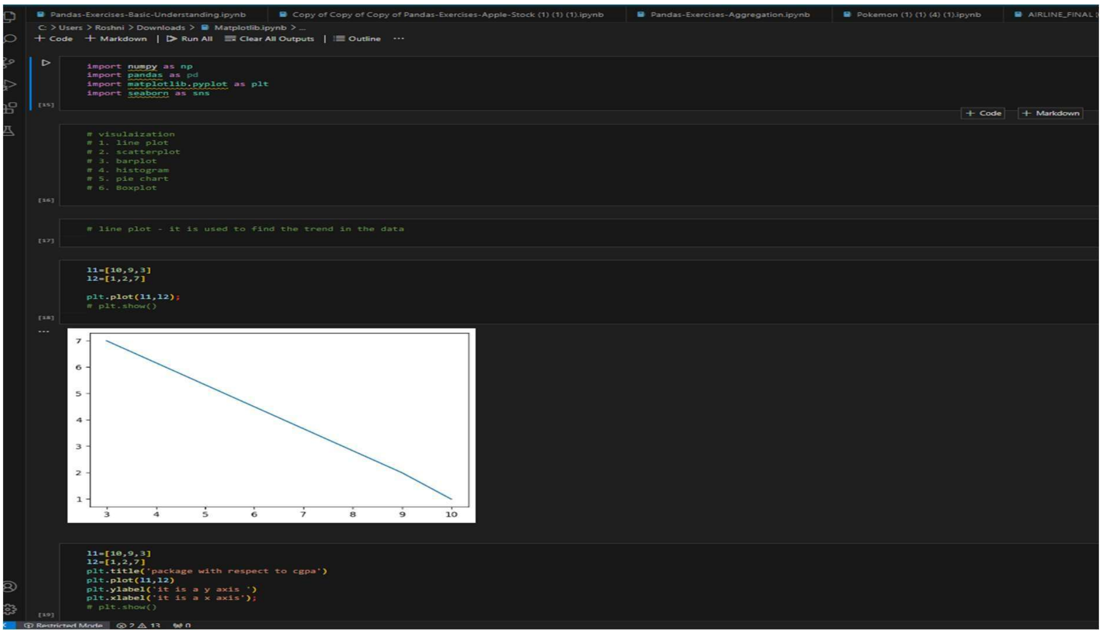
- **Use:** To represent proportions of categorical variables.
- **Why:** Helps quickly understand the composition of a variable.
- **Tools:** Matplotlib.

8. Violin Plots

- **Use:** To show the distribution of a variable across different categories.
- **Why:** Combines box plot and density plot to provide a deeper understanding of the data.
- **Tools:** Seaborn.

Best Practices for Visualizing Cleaned Data:

- **Clarity:** Ensure that visuals are easy to understand, avoiding clutter.
- **Consistency:** Use consistent scales, colors, and labels to ensure comparisons are meaningful.
- **Appropriate charts:** Choose the chart that best represents the data, and avoid using charts that distort the story.



Two or more plots in figure for comparions

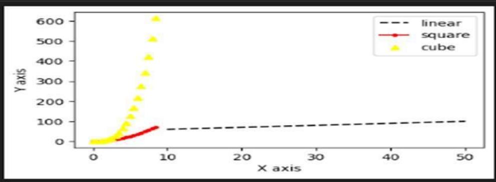
```
x=[10,20,30,40,50]
y=[60,70,80,90,100]
plt.figure(figsize=(5,3))
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.plot(x,y,'--',color='black',label='linear');
x2=np.arange(0,9,0.5)


plt.plot(x2,x2**2,'.-',color='red',label='square');

plt.plot(x2,x2**3,'^',color='yellow',label='cube');
plt.legend()

plt.show()
```

[55]




```
a=np.array([1,2,3,4,5,6])
b=np.array([8,2,4,6,10,12])

plt.subplot(1,3,1)
plt.title('linear graph')
plt.plot(a,b)

plt.subplot(1,3,2)
plt.title('exponential')
plt.plot(a,b**2)


plt.subplot(1,3,3)

plt.plot(a,b)


plt.show()
```

[69]

```
plt.boxplot(new_df_cap['tip']);
```

```
df=sns.load_dataset('tips')
df
```

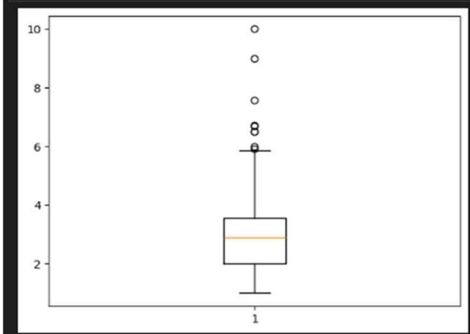
```
df['tip'].mean()
```

2.99827868852459

```
df['tip'].median()
```

2.9

```
plt.boxplot(df['tip']);
```



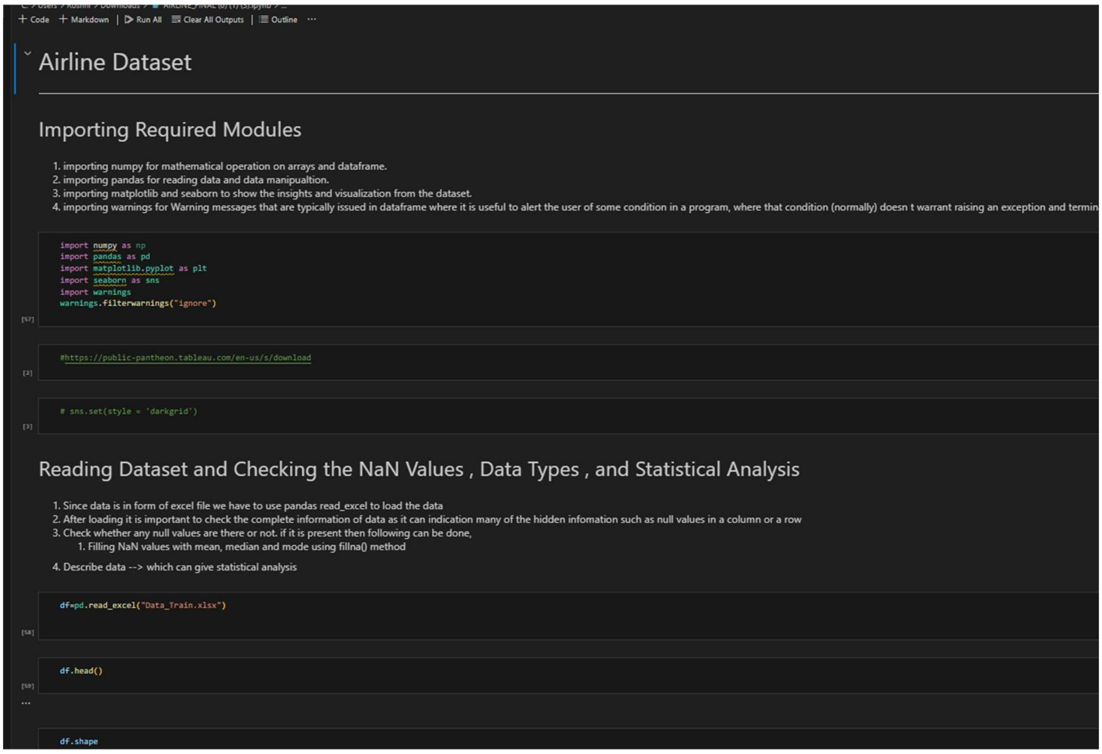

```
# upper_limit=q3+1.5*(IQR)
# lower_limit=q1-1.5*(IQR)
```

18-11-2024

Training Day – 42

Topic: Exporting Processed Data

- Saved cleaned and processed datasets to CSV and Excel formats.
- Example: Exported a cleaned DataFrame to cleaned_data.xlsx.



Airline Dataset

Importing Required Modules

1. importing numpy for mathematical operation on arrays and dataframe.
2. importing pandas for reading data and data manipualtion.
3. importing matplotlib and seaborn to show the insights and visualization from the dataset.
4. importing warnings for Warning messages that are typically issued in dataframe where it is useful to alert the user of some condition in a program, where that condition (normally) doesn t warrant raising an exception and termin

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
#https://public-pantheon.tableau.com/en-us/s/download
```

```
# sns.set(style = 'darkgrid')
```

Reading Dataset and Checking the NaN Values , Data Types , and Statistical Analysis

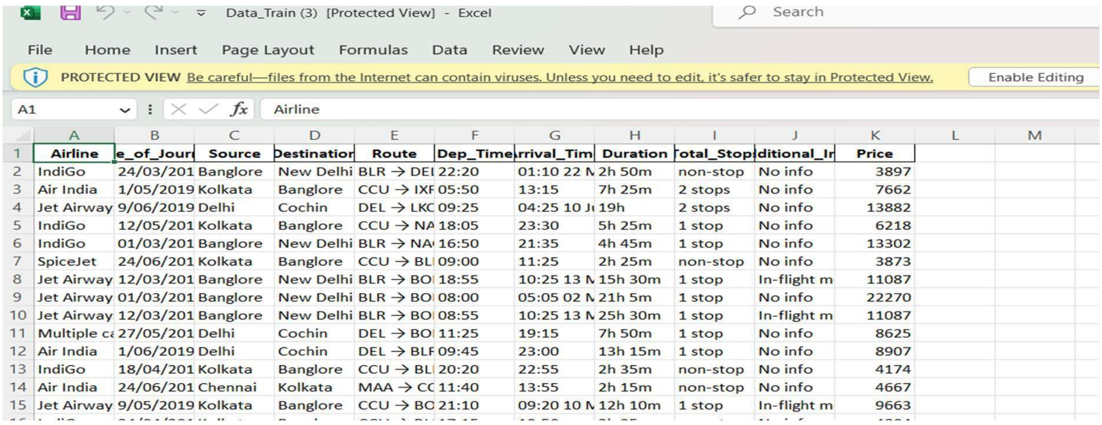
1. Since data is in form of excel file we have to use pandas read_excel to load the data
2. After loading it is important to check the complete information of data as it can indication many of the hidden infomation such as null values in a column or a row
3. Check whether any null values are there or not. if it is present then following can be done,
    1. Filling NaN values with mean, median and mode using fillna() method
4. Describe data --> which can give statistical analysis

```
df=pd.read_excel("Data_Train.xlsx")
```

```
df.head()
```

```
df.shape
```

DATA IN XLSX



	A	B	C	D	E	F	G	H	I	J	K
1	Airline	e_of_Jour	Source	Destination	Route	Dep_Time	rrival_Tim	Duration	otal_Stop	ditional_Ir	Price
2	IndiGo	24/03/201	Banglore	New Delhi	BLR → DEI	22:20	01:10 22 M	2h 50m	non-stop	No info	3897
3	Air India	1/05/2019	Kolkata	Banglore	CCU → IXF	05:50	13:15	7h 25m	2 stops	No info	7662
4	Jet Airway	9/06/2019	Delhi	Cochin	DEL → LKC	09:25	04:25 10 Ju	19h	2 stops	No info	13882
5	IndiGo	12/05/201	Kolkata	Banglore	CCU → NA	18:05	23:30	5h 25m	1 stop	No info	6218
6	IndiGo	01/03/201	Banglore	New Delhi	BLR → NA	16:50	21:35	4h 45m	1 stop	No info	13302
7	SpiceJet	24/06/201	Kolkata	Banglore	CCU → BLI	09:00	11:25	2h 25m	non-stop	No info	3873
8	Jet Airway	12/03/201	Banglore	New Delhi	BLR → BOI	18:55	10:25 13 M	15h 30m	1 stop	In-flight m	11087
9	Jet Airway	01/03/201	Banglore	New Delhi	BLR → BOI	08:00	05:05 02 M	21h 5m	1 stop	No info	22270
10	Jet Airway	12/03/201	Banglore	New Delhi	BLR → BOI	08:55	10:25 13 M	25h 30m	1 stop	In-flight m	11087
11	Multiple ca	27/05/201	Delhi	Cochin	DEL → BOI	11:25	19:15	7h 50m	1 stop	No info	8625
12	Air India	1/06/2019	Delhi	Cochin	DEL → BLF	09:45	23:00	13h 15m	1 stop	No info	8907
13	IndiGo	18/04/201	Kolkata	Banglore	CCU → BLI	20:20	22:55	2h 35m	non-stop	No info	4174
14	Air India	24/06/201	Chennai	Kolkata	MAA → CC	11:40	13:55	2h 15m	non-stop	No info	4667
15	Jet Airway	9/05/2019	Kolkata	Banglore	CCU → BO	21:10	09:20 10 M	12h 10m	1 stop	In-flight m	9663

19-11-2024 Training Day – 43

Topic: Revisiting Data Cleaning Techniques

- Practiced handling outliers and formatting columns.
- Example: Removed outliers using the interquartile range (IQR).

displaying % value for each slice

```
plt.pie(l,labels = df1["species"].unique(),autopct ='%0.0f',startangle=90);
## format string for displaying % : .2f here 2 values after decimal
```

Box Plot

Boxplots can be used to:

1: Identify outliers or anomalous data points

2: To determine if our data is skewed

3: To understand the spread/range of the dataused to detect the outliers

A Box Plot is also known as Whisker plot is created to display the summary of the set of data values having properties like minimum, first quartile, median, third quartile and maximum.

(it is same as describe() in pandas)

In the box plot, a box is created from the first quartile to the third quartile, a vertical line is also there which goes through the box at the median.

Here x-axis denotes the data to be plotted while the y-axis shows the frequency distribution.

Basically : box plot used to display the distribution of data based on five key numbers:

The "minimum",

1st Quartile (25th percentile),

median (2nd Quartile./ 50th Percentile),

the 3rd Quartile (75th percentile),

and the "maximum".

The minimum and maximum values are defined as Q1–1.5 * IQR and Q3 + 1.5 * IQR respectively. Any points that fall outside of these limits are referred to as outliers.

where IQR(Inter quartile range)=Q3-Q1

```
### Creating boxplot for tip column in tips data
```

```
df=sns.load_dataset('tips')
df
```

...

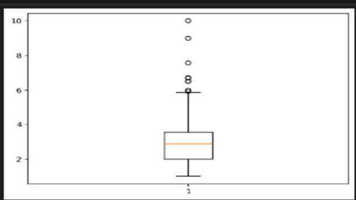
```
df['tip'].mean()
```

```
2.99827868852459
```

```
df['tip'].median()
```

```
2.9
```

```
plt.boxplot(df['tip']);
```




```
# upper_limit=q3+1.5*(IQR)
# lower_limit=q1-1.5*(IQR)

# IQR=q3-q1
```

```
df['tip'].describe()
```

```
count    244.000000
mean       2.998279
std        1.383638
min        1.000000
25%        2.000000
50%        2.900000
75%        3.562500
max       10.000000
Name: tip, dtype: float64
```

```
q1=np.percentile(df['tip'],25)
q1
```

20-11-2024 # Training Day – 44

Topic: Combining Multiple Datasets

- Consolidated datasets into a single clean dataset.
- Example: Used a combination of concat() and merge() for integration.

Combining datasets from different sources or files is a common task in data cleaning and analysis. By integrating datasets into one consolidated clean dataset, you can work with a complete set of information for further analysis or modeling. Two commonly used methods for combining datasets are concat() and merge() functions in Python, particularly with the **Pandas** library.

1. Concatenating Datasets with **concat()**

The concat() function is used to combine datasets along a particular axis (rows or columns). It's useful when datasets have the same structure (e.g., same columns) but come from different sources or time periods.

Example: Concatenating DataFrames by Rows

Suppose you have two DataFrames with identical columns but different rows (e.g., two sets of data collected over different months).

```
import pandas as pd

# Sample DataFrames
df1 = pd.DataFrame({
    'ID': [1, 2, 3],
    'Value': [10, 20, 30]
})

df2 = pd.DataFrame({
    'ID': [4, 5, 6],
    'Value': [40, 50, 60]
})

# Concatenate by rows (axis=0)
df_combined = pd.concat([df1, df2], axis=0, ignore_index=True)
print(df_combined)
```

Example: Concatenating DataFrames by Columns

If your datasets contain different features (columns), you can concatenate them side by side.

```
# Concatenate by columns (axis=1)
df_combined_columns = pd.concat([df1, df2], axis=1)
print(df_combined_columns)
```

Output:

```
Copy code
  ID  Value  ID  Value
0  1     10  4     40
1  2     20  5     50
2  3     30  6     60
```

2. Merging Datasets with **merge()**The merge() function is used when datasets share common columns, and you want to combine them based on matching values. It’s similar to a SQL join (inner, outer, left, or right join).

- **Example: Merging DataFrames on Common Columns**

If you have two DataFrames with a common column (e.g., "ID"), you can merge them to consolidate their information.

```
python
Copy code
df1 = pd.DataFrame({
    'ID': [1, 2, 3],
    'Name': ['Alice', 'Bob', 'Charlie']
})

df2 = pd.DataFrame({
    'ID': [1, 2, 4],
    'Value': [100, 200, 300]
})

# Merge on 'ID'
df_merged = pd.merge(df1, df2, on='ID', how='inner')
print(df_merged)
```

Output:

```
Copy code
  ID    Name  Value
0  1   Alice    100
1  2     Bob    200
```

23-11-2024 **Training Day – 45**

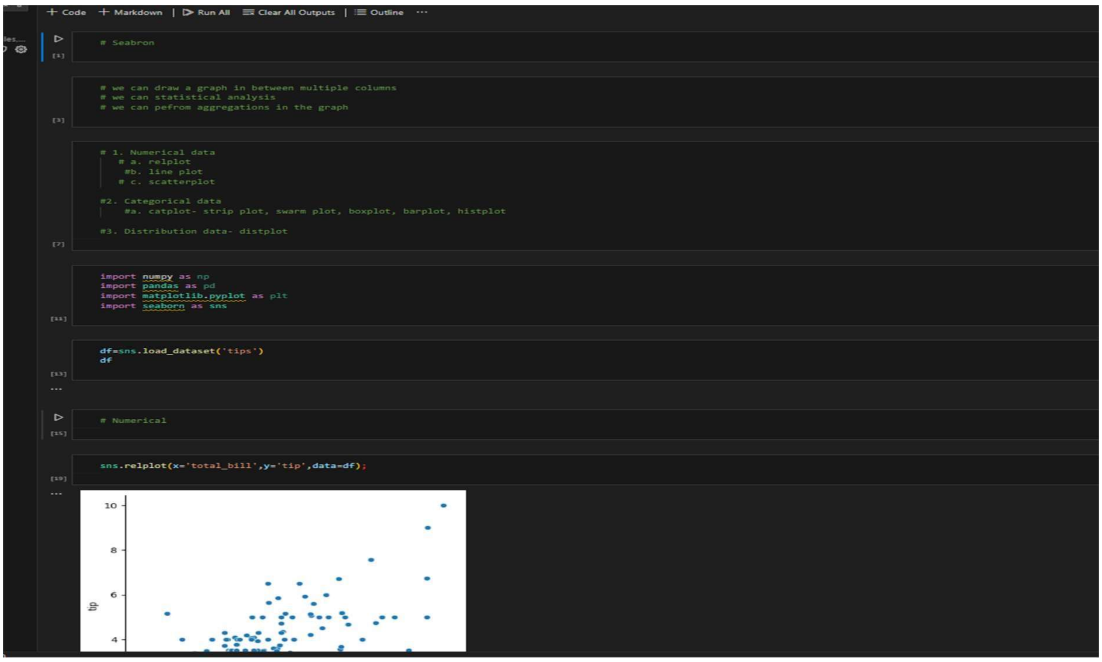
Topic: Advanced Visualizations

- Created multi-line plots to compare trends.
- Example: Compared monthly sales for different products.

Advanced visualizations go beyond simple charts to provide deeper insights into complex datasets. These visualizations can help reveal patterns, trends, and relationships that are not immediately apparent with basic plots. Below are some advanced techniques and types of visualizations that are commonly used in data analysis and data science.

1. Heatmaps

- **Use:** To represent data in a matrix format, where individual values are displayed with color gradients. It's often used to visualize correlation matrices, missing data, or any other form of numerical relationships.
- **Why:** It quickly shows the relationships and magnitude of values across a two-dimensional space.
- **Tools:** Seaborn, Matplotlib, Plotly.
- **Example: Correlation Matrix Heatmap**



+ Code + Markdown | Run All Clear All Outputs | Outline ···

```
import seaborn as sns
df=sns.load_dataset('tips')
```

```
sns.relplot(x='total_bill',y='tip',hue='day',size='sex',style='size',\
            col='time',row='smoker',data=df);
```

[50]