Data Harmonization and Insights Extraction

BY:PRIYANSHU KUMAR

DATE:- JAN 2025

Agenda

- Project overview
- Data wrangling and cleaning steps
- Key challenges and how they were addressed
- Insights and findings from the dataset

Project Overview

This project focuses on cleaning and unifying multiple datasets by resolving issues such as missing values, duplicate entries etc. Using advanced data-wrangling techniques, the goal is to standardize and integrate data, ensuring accuracy and consistency. The resulting clean dataset will serve as a reliable foundation for business analysis and predictive modeling.

Data wrangling and cleaning steps

Importing libraries and loading dataset 1

[1]:
```
import pandas as pd     #Used for Data manipulation and analysis And Frequently used for reading/writing data (CSV, Excel, SQL, etc.).
import matplotlib.pyplot as plt  #  Used for Data visualization
                                 #It is Highly customizable for plotting graphs like line plots, bar charts, scatter plots, etc.
import numpy as np     #Numerical computing, Includes mathematical functions to operate on these arrays efficiently.
import seaborn as sns    #Statistical data visualization, Supports visualizations like heatmaps, violin plots, and pair plots.
from sklearn.preprocessing import LabelEncoder  #Converts categorical labels into numeric labels.
```

[2]:
```
df1=pd.read_csv("dataset_1 - dataset_1.csv")
```

[3]:
```
df1
```

[3]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	weathersit	temp
0	1	01-01-2011	1	0	1	0	False	6	1	0.24
1	2	01-01-2011	1	0	1	1	False	6	1	0.22
2	3	01-01-2011	1	0	1	2	False	6	1	0.22
3	4	01-01-2011	1	0	1	3	False	6	1	0.24
4	5	01-01-2011	1	0	1	4	False	6	1	0.24
...	...	...	...	...	...	...	...	...	...	...
605	606	28-01-2011	1	0	1	11	False	5	3	0.18
606	607	28-01-2011	1	0	1	12	False	5	3	0.18
607	608	28-01-2011	1	0	1	13	False	5	3	0.18
608	609	28-01-2011	1	0	1	14	False	5	3	0.22
609	610	28-01-2011	1	0	1	15	False	5	2	0.20

610 rows × 10 columns

Data wrangling and cleaning steps

Loading dataset 2 and performing left join with 2nd dataset

```
[436]: df2=pd.read_csv("dataset_2.xlsx - dataset_2.csv")

[437]: #df2

[438]: #pd.merge Combines two DataFrames by aligning rows based on one or more shared columns or indices.
       #by default it use left join
       combined = pd.merge(df1, df2, on='instant')

[439]: combined
```

[439]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	weathersit	temp	Unnamed: 0	atemp	hum	windspeed	casual	registered	cnt
0	1	01-01-2011	1	0	1	0	False	6	1	0.24	0	0.2879	0.81	0.0000	3	13	16
1	2	01-01-2011	1	0	1	1	False	6	1	0.22	1	0.2727	0.80	0.0000	8	32	40
2	3	01-01-2011	1	0	1	2	False	6	1	0.22	2	0.2727	0.80	0.0000	5	27	32
3	4	01-01-2011	1	0	1	3	False	6	1	0.24	3	0.2879	0.75	0.0000	3	10	13
4	5	01-01-2011	1	0	1	4	False	6	1	0.24	4	0.2879	0.75	0.0000	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
605	606	28-01-2011	1	0	1	11	False	5	3	0.18	605	0.2121	0.93	0.1045	0	30	30
606	607	28-01-2011	1	0	1	12	False	5	3	0.18	606	0.2121	0.93	0.1045	1	28	29
607	608	28-01-2011	1	0	1	13	False	5	3	0.18	607	0.2121	0.93	0.1045	0	31	31
608	609	28-01-2011	1	0	1	14	False	5	3	0.22	608	0.2727	0.80	0.0000	2	36	38
609	610	28-01-2011	1	0	1	15	False	5	2	0.20	609	0.2576	0.86	0.0000	1	40	41

610 rows × 17 columns

Data wrangling and cleaning steps

Loading dataset 3 and appending it with the merged dataset

```
[441]: df3=pd.read_csv("dataset_3 - dataset_3.csv")

[442]: #df3

[443]: # here fd = final data set
       # in combined(df1+df2)sets and in the 3rd dataset both have same name of header to join one above the other we use conacat function.
       df=pd.concat([combined,df3],ignore_index=True)

[444]: df
```

[444]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	weathersit	temp	Unnamed: 0	atemp	hum	windspeed	casual	registered	cnt
0	1	01-01-2011	1	0	1	0	False	6	1	0.24	0.0	0.2879	0.81	0.0000	3	13	16
1	2	01-01-2011	1	0	1	1	False	6	1	0.22	1.0	0.2727	0.80	0.0000	8	32	40
2	3	01-01-2011	1	0	1	2	False	6	1	0.22	2.0	0.2727	0.80	0.0000	5	27	32
3	4	01-01-2011	1	0	1	3	False	6	1	0.24	3.0	0.2879	0.75	0.0000	3	10	13
4	5	01-01-2011	1	0	1	4	False	6	1	0.24	4.0	0.2879	0.75	0.0000	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	615	28-01-2011	1	0	1	20	False	5	2	0.24	NaN	0.2273	0.70	0.1940	1	61	62
996	616	28-01-2011	1	0	1	21	False	5	2	0.22	NaN	0.2273	0.75	0.1343	1	57	58
997	617	28-01-2011	1	0	1	22	False	5	1	0.24	NaN	0.2121	0.65	0.3582	0	26	26
998	618	28-01-2011	1	0	1	23	False	5	1	0.24	NaN	0.2273	0.60	0.2239	1	22	23
999	619	29-01-2011	1	0	1	0	False	6	1	0.22	NaN	0.1970	0.64	0.3582	2	26	28

1000 rows × 17 columns

Data wrangling and cleaning steps

Missing value imputation

```
[446]: # It used to detect and summarize missing values in a DataFrame.
       df.isnull().sum()
```

```
[446]: instant         0
       dteday          0
       season          0
       yr              0
       mnth            0
       hr              0
       holiday         0
       weekday         0
       weathersit      0
       temp            0
       Unnamed: 0    390
       atemp          11
       hum             0
       windspeed       0
       casual          0
       registered      0
       cnt             0
       dtype: int64
```

```
[447]: # fill.na is used to fill the missing values.
       df["atemp"]=df["atemp"].fillna(df["atemp"].mean())
```

```
[448]: df.isnull().sum()
```

```
[448]: instant         0
       dteday          0
       season          0
       yr              0
       mnth            0
       hr              0
       holiday         0
```

Data wrangling and cleaning steps

Dropping redundant columns

[449]:
```
# drop command is used to remove the particular column from the dataset
df.drop("Unnamed: 0",axis=1, inplace=True)
```

[450]:
```
df
```

[450]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	01-01-2011	1	0	1	0	False	6	1	0.24	0.2879	0.81	0.0000	3	13	16
1	2	01-01-2011	1	0	1	1	False	6	1	0.22	0.2727	0.80	0.0000	8	32	40
2	3	01-01-2011	1	0	1	2	False	6	1	0.22	0.2727	0.80	0.0000	5	27	32
3	4	01-01-2011	1	0	1	3	False	6	1	0.24	0.2879	0.75	0.0000	3	10	13
4	5	01-01-2011	1	0	1	4	False	6	1	0.24	0.2879	0.75	0.0000	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	615	28-01-2011	1	0	1	20	False	5	2	0.24	0.2273	0.70	0.1940	1	61	62
996	616	28-01-2011	1	0	1	21	False	5	2	0.22	0.2273	0.75	0.1343	1	57	58
997	617	28-01-2011	1	0	1	22	False	5	1	0.24	0.2121	0.65	0.3582	0	26	26
998	618	28-01-2011	1	0	1	23	False	5	1	0.24	0.2273	0.60	0.2239	1	22	23
999	619	29-01-2011	1	0	1	0	False	6	1	0.22	0.1970	0.64	0.3582	2	26	28

1000 rows × 16 columns

Data wrangling and cleaning steps

Data cleaning

```
*[451]: # It is used to split the date and time and saperate them
        parts = df["dteday"].str.split(" ", n=2, expand=True)
        df["date"] = parts[0]
        df["time"] = parts[0].str[:2].astype('int')
        df.head()
```

[451]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt	date	time
0	1	01-01-2011	1	0	1	0	False	6	1	0.24	0.2879	0.81	0.0	3	13	16	01-01-2011	1
1	2	01-01-2011	1	0	1	1	False	6	1	0.22	0.2727	0.80	0.0	8	32	40	01-01-2011	1
2	3	01-01-2011	1	0	1	2	False	6	1	0.22	0.2727	0.80	0.0	5	27	32	01-01-2011	1
3	4	01-01-2011	1	0	1	3	False	6	1	0.24	0.2879	0.75	0.0	3	10	13	01-01-2011	1
4	5	01-01-2011	1	0	1	4	False	6	1	0.24	0.2879	0.75	0.0	0	1	1	01-01-2011	1

```
*[452]: # unique command is used to show the unique values of column of the data frame
        df["holiday"].unique()
```

```
[452]: array([False,  True])
```

```
[453]: df["windspeed"].unique()
```

```
[453]: array([0.    , 0.0896, 0.2537, 0.2836, 0.2985, 0.194 , 0.2239, 0.1343,
               0.1642, 0.3284, 0.4478, 0.3582, 0.4179, 0.3881, 0.1045, 0.4925,
               0.5522, 0.4627, 0.5224, 0.5821])
```

```
*[454]: # here i drop the dteday column from the final data frame.
        df.drop("dteday",axis=1, inplace=True)
```

```
[455]: df
```

[455]:

instant	season	yr	mnth	hr	holiday	weekday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt	date	time

Key challenges and how they were addressed

1. combined = pd.merge(df1, df2, on='instant')

To add extra columns, we performed a left join on the first two datasets.
The left join was implemented using the default behavior of the merge function in pandas.

2. df=pd.concat([combined,df3],ignore_index=True)

To append the new data to the previously joined data, we used the concat function in pandas.

3. df.isnull().sum()

Finding the total number of nulls in all the columns of the data frame

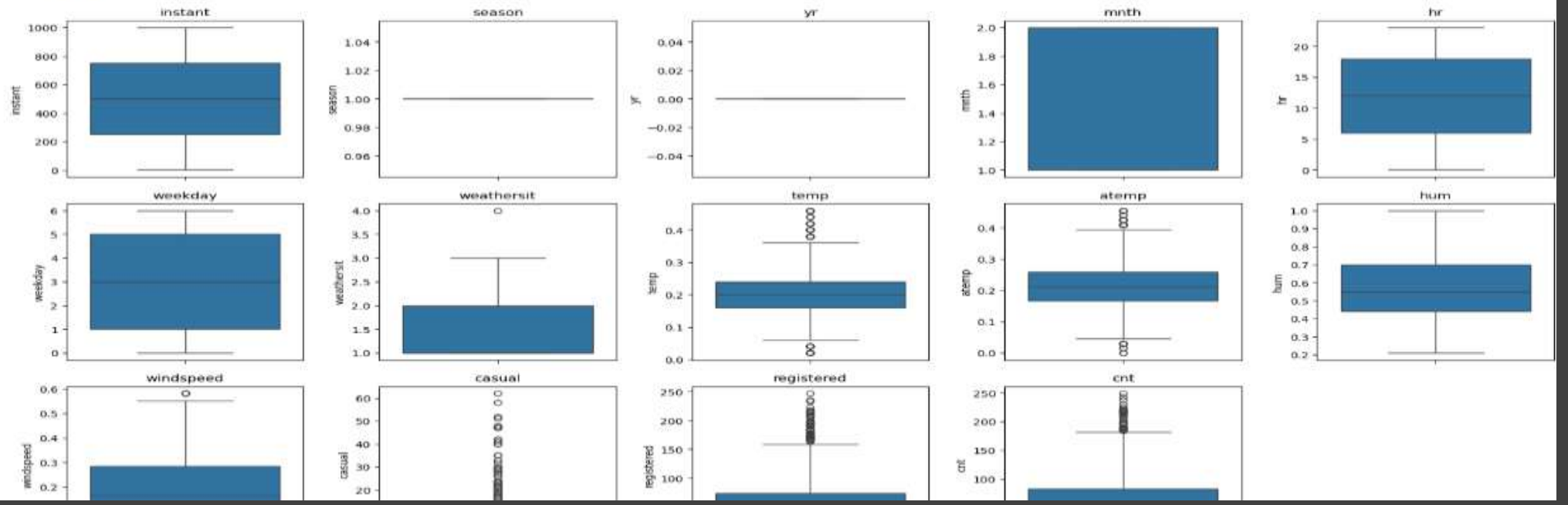
4. df["atemp"]=df["atemp"].fillna(df["atemp"].mean())

As there was only one column with missing values, and it was of data type int, filling the missing values with the mean helps preserve the column's overall distribution and central tendency, ensuring it remains representative of the dataset.

Insights and findings from the dataset

Finding the distribution of data in different columns

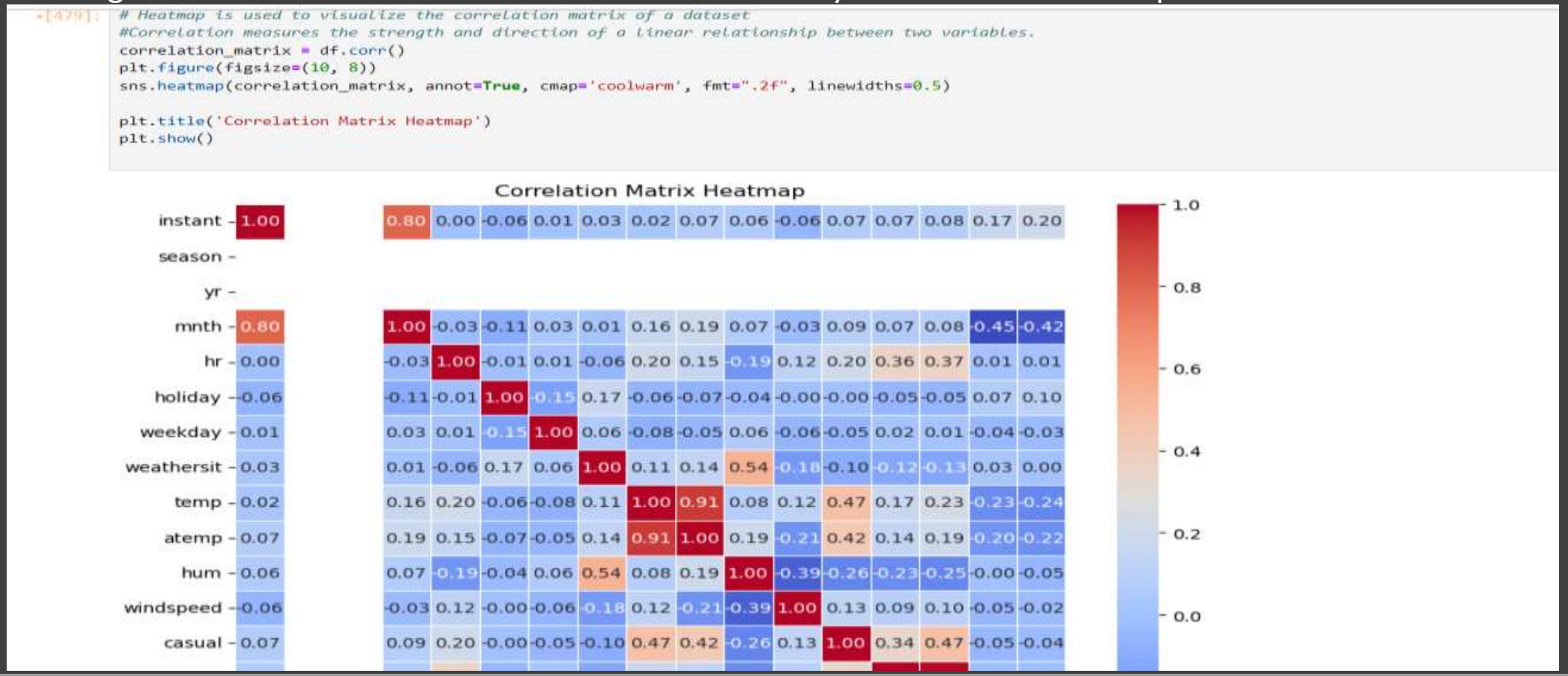
```
[270]: #Boxplots are great for visualizing the distribution, central tendency, and potential outliers in the data.
plt.figure(figsize=(18,15))  # It shows the width and height of the figure in inches.
for i,col in enumerate(numerical,1):
    plt.subplot(5, 5, i)  #Creates a subplot in a grid with 5 rows and 5 columns, i specifies the position of the current plot in this grid.
    sns.boxplot(y=df[col])  #Creates a boxplot for the current column in the DataFrame.
    plt.title(col)

plt.tight_layout()
plt.show()
```



Insights and findings from the dataset

Finding correlation between different columns to identify the linear relationship between two variables

```
[479]: # Heatmap is used to visualize the correlation matrix of a dataset
#Correlation measures the strength and direction of a linear relationship between two variables.
correlation_matrix = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)

plt.title('Correlation Matrix Heatmap')
plt.show()
```



Insights and findings from the dataset

Identifying different quartiles

[482]:
```
#A quartile is a statistical term used to describe the division of a dataset into four equal parts.
#Each quartile represents a specific percentage of the data distribution, helping to understand the spread and central tendency of the data.
q1=df["windspeed"].quantile(0.25)
q2=df["windspeed"].quantile(0.50)
q3=df["windspeed"].quantile(0.75)
print("The first quantile is",q1)
print("The second quantile is",q2)
print("The third quantile is",q3)
```

```
The first quantile is 0.1045
The second quantile is 0.1642
The third quantile is 0.2836
```

[483]:
```
#IQR (Interquartile Range) is a measure of statistical dispersion, which is the spread of data values.
#It describes the range within which the middle 50% of the data lies
#focusing on the central portion of the dataset.
#IQR=Q3(Third Quartile)-Q1(First Quartile)

IQR=(q3-q1)
print('The inner quantile is',IQR)
```

```
The inner quantile is 0.17910000000000004
```

[484]:
```
lower_limit=q1-1.5*IQR
upper_limit=q3+1.5*IQR
print('The lower limit is',lower_limit)
print('The upper limit is',upper_limit)
```

```
The lower limit is -0.16415000000000007
The upper limit is 0.5522500000000001
```

[485]:
```
df[(df["windspeed"]<lower_limit)|(df["windspeed"]>upper_limit)]
```

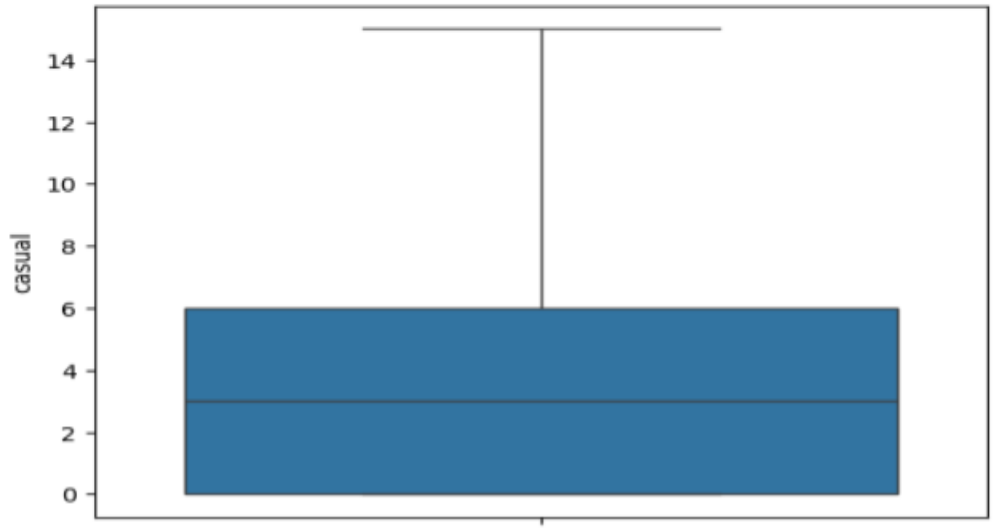
[485]:

	instant	season	yr	mnth	hr	holiday	weekday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt	time	date_code
265	266	1	0	1	12	False	3	1	0.20	0.1515	0.47	0.5821	3	52	55	12	22
467	468	1	0	1	12	False	5	1	0.22	0.1818	0.27	0.5821	11	67	78	21	34
468	469	1	0	1	13	False	5	1	0.20	0.1515	0.21	0.5821	8	65	73	21	34

Insights and findings from the dataset

Removing outliers

```
[512]: # Removing The Outliers

[513]: # np.where() is a conditional function in NumPy that works like an if-else statement but is applied element-wise to an entire array or column of data.
       #If the value is less than 163, the condition evaluates to True; otherwise, it evaluates to False.
       #If the condition is False the value in the "registered" column is replaced with 162.5.

       df["casual"] = np.where(df["casual"] < 16, df["casual"], 15)

[514]: #df

[515]: sns.boxplot(df["casual"])

[515]: <Axes: ylabel='casual'>
```



Thankyou