

Data Collection and Preprocessing Phase

Date	12th October 2024
Team ID	LTVIP2024TMID24968
Project Title	TrafficTelligence Advanced Traffic Volume Estimation with Machine Learning
Maximum Marks	6 Marks

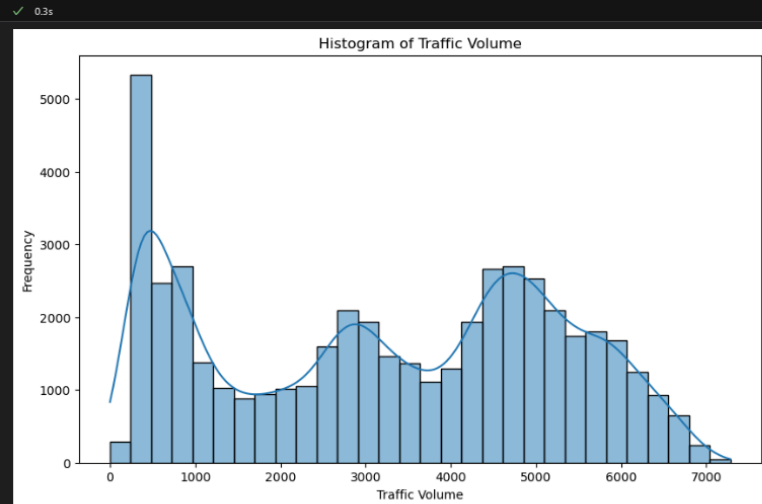
Data Exploration and Preprocessing Template

Dataset variables will undergo statistical analysis to uncover patterns and identify outliers, utilizing Python for preprocessing tasks such as normalization and feature engineering. Data cleaning will address missing values and outliers, ensuring high-quality data for subsequent analysis and modeling, ultimately establishing a solid foundation for insights and accurate predictions.

Section	Description																																													
Data Overview	<div><div><div>> ▾</div><div>[8]</div><div>...</div></div><div><pre>data.describe()</pre><table><tr><th></th><th>temp</th><th>rain</th><th>snow</th><th>traffic_volume</th></tr><tr><td>count</td><td>48151.000000</td><td>48202.000000</td><td>48192.000000</td><td>48204.000000</td></tr><tr><td>mean</td><td>281.205351</td><td>0.334278</td><td>0.000222</td><td>3259.818355</td></tr><tr><td>std</td><td>13.343675</td><td>44.790062</td><td>0.008169</td><td>1986.860670</td></tr><tr><td>min</td><td>0.000000</td><td>0.000000</td><td>0.000000</td><td>0.000000</td></tr><tr><td>25%</td><td>272.160000</td><td>0.000000</td><td>0.000000</td><td>1193.000000</td></tr><tr><td>50%</td><td>282.460000</td><td>0.000000</td><td>0.000000</td><td>3380.000000</td></tr><tr><td>75%</td><td>291.810000</td><td>0.000000</td><td>0.000000</td><td>4933.000000</td></tr><tr><td>max</td><td>310.070000</td><td>9831.300000</td><td>0.510000</td><td>7280.000000</td></tr></table></div></div>		temp	rain	snow	traffic_volume	count	48151.000000	48202.000000	48192.000000	48204.000000	mean	281.205351	0.334278	0.000222	3259.818355	std	13.343675	44.790062	0.008169	1986.860670	min	0.000000	0.000000	0.000000	0.000000	25%	272.160000	0.000000	0.000000	1193.000000	50%	282.460000	0.000000	0.000000	3380.000000	75%	291.810000	0.000000	0.000000	4933.000000	max	310.070000	9831.300000	0.510000	7280.000000
	temp	rain	snow	traffic_volume																																										
count	48151.000000	48202.000000	48192.000000	48204.000000																																										
mean	281.205351	0.334278	0.000222	3259.818355																																										
std	13.343675	44.790062	0.008169	1986.860670																																										
min	0.000000	0.000000	0.000000	0.000000																																										
25%	272.160000	0.000000	0.000000	1193.000000																																										
50%	282.460000	0.000000	0.000000	3380.000000																																										
75%	291.810000	0.000000	0.000000	4933.000000																																										
max	310.070000	9831.300000	0.510000	7280.000000																																										

Univariate Analysis

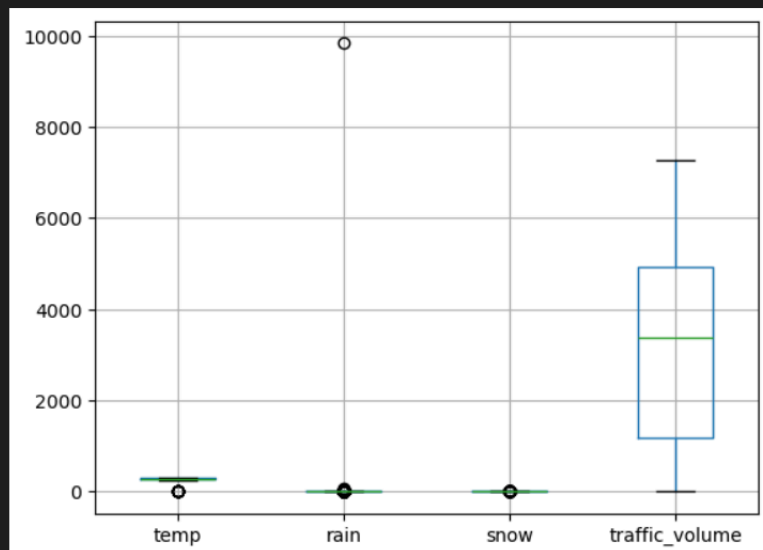
```
plt.figure(figsize=(10, 6))
sns.histplot(data['traffic_volume'], bins=30, kde=True)
plt.title('Histogram of Traffic Volume')
plt.xlabel('Traffic Volume')
plt.ylabel('Frequency')
plt.show()
```



Bivariate Analysis

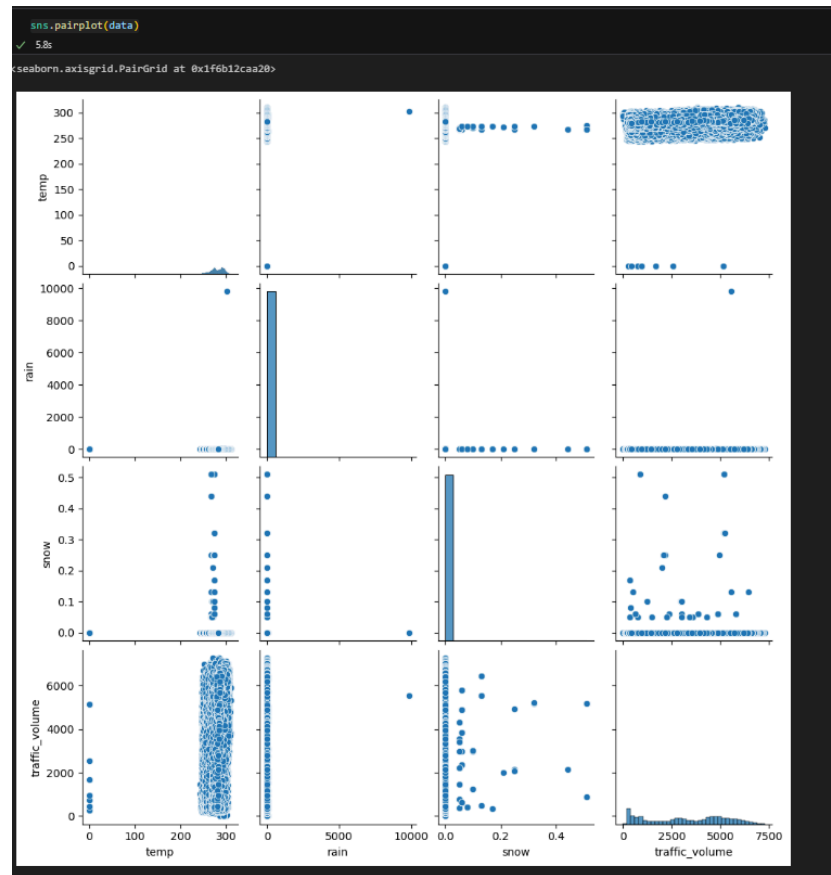
```
data.boxplot()
```

<Axes: >



Multivariate Analysis

Pairplot

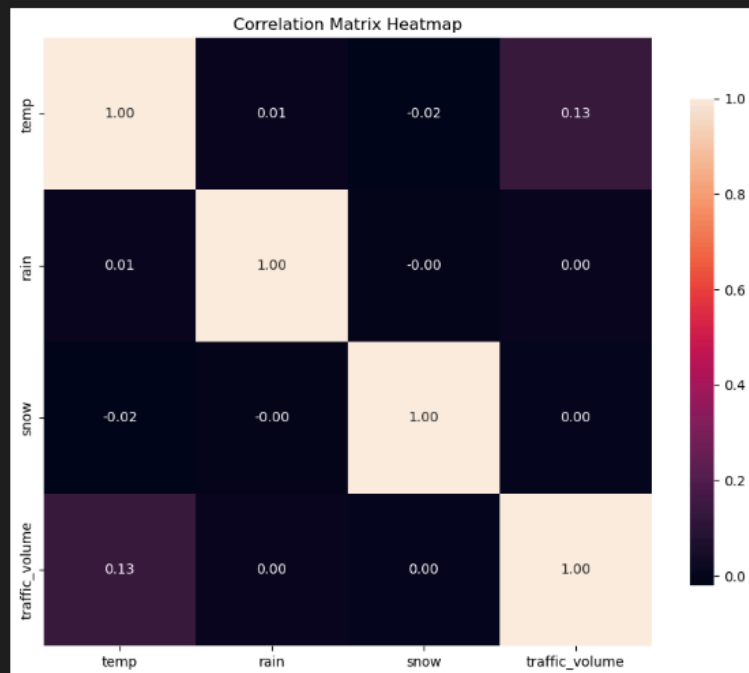


Heat Map

```
numeric_df = data.select_dtypes(include=[np.number])
corr = numeric_df.corr()
corr
```

	temp	rain	snow	traffic_volume
temp	1.000000	0.009069	-0.019760	0.130051
rain	0.009069	1.000000	-0.000090	0.004714
snow	-0.019760	-0.000090	1.000000	0.000733
traffic_volume	0.130051	0.004714	0.000733	1.000000

```
plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, fmt=".2f", square=True, cbar_kws={"shrink": .8})
plt.title('Correlation Matrix Heatmap')
plt.show()
```



Outliers and Anomalies

There are no outliers or anomalies in the dataset.

Data Preprocessing Code Screenshots

Loading Data

```
data = pd.read_csv('traffic-volume.csv')
✓ 0.0s
```

```
data
✓ 0.0s
```

	holiday	temp	rain	snow	weather	date	Time	traffic_volume
0	0	288.28	0.0	0.0	1.0	02-10-2012	09:00:00	5545
1	0	289.36	0.0	0.0	1.0	02-10-2012	10:00:00	4516
2	0	289.58	0.0	0.0	1.0	02-10-2012	11:00:00	4767
3	0	290.13	0.0	0.0	1.0	02-10-2012	12:00:00	5026
4	0	291.14	0.0	0.0	1.0	02-10-2012	13:00:00	4918
...
48199	0	283.45	0.0	0.0	1.0	30-09-2018	19:00:00	3543
48200	0	282.76	0.0	0.0	1.0	30-09-2018	20:00:00	2781
48201	0	282.73	0.0	0.0	Thunderstorm	30-09-2018	21:00:00	2159
48202	0	282.09	0.0	0.0	1.0	30-09-2018	22:00:00	1450
48203	0	282.12	0.0	0.0	1.0	30-09-2018	23:00:00	954

48204 rows x 8 columns

Handling Missing Data

```
data['temp'] = data['temp'].fillna(data['temp'].median())
data['rain'] = data['rain'].fillna(data['rain'].median())
data['snow'] = data['snow'].fillna(data['snow'].median())
✓ 0.0s
```

```
print(data['weather'].value_counts())
✓ 0.0s
```

```
weather
Clouds      15444
Clear       13383
Part        9662
Rain       5605
Snow       2875
Drizzle     1818
Fog         1209
Thunderstorm 1033
Fog         912
Smoke       78
Squall       4
Name: count, dtype: int64
```

```
data['weather'] = data['weather'].fillna('Clouds', inplace=True)
✓ 0.0s
```

C:\Users\Anish\AnacondaLocal\envs\internz\15312\171091201.ps1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy. For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method(col: value, inplace=True)' or 'df[col] = df[col].method(value)' instead, to perform the operation inplace on the original object.

```
data['weather'] = data['weather'].fillna('Clouds', inplace=True)
```

```
# prompt: print NaN in holiday column to 0
data['holiday'] = data['holiday'].fillna(0, inplace=True)
✓ 0.0s
```

C:\Users\Anish\AnacondaLocal\envs\internz\15312\171091201.ps1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy. For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method(col: value, inplace=True)' or 'df[col] = df[col].method(value)' instead, to perform the operation inplace on the original object.

```
data['holiday'] = data['holiday'].fillna(0, inplace=True)
```

Data Transformation

```
data[["day","month","year"]] = data["date"].str.split("-", expand=True)
data[["hours","minutes","seconds"]] = data["Time"].str.split(":", expand=True)
data.drop(columns=['date','Time'],axis=1,inplace=True)
```

```
print(data.head())
```

	holiday	temp	rain	snow	weather	traffic_volume	day	month	year	hours	\
0	0	288.28	0.0	0.0	1.0	5545	02	10	2012	09	
1	0	289.36	0.0	0.0	1.0	4516	02	10	2012	10	
2	0	289.58	0.0	0.0	1.0	4767	02	10	2012	11	
3	0	290.13	0.0	0.0	1.0	5026	02	10	2012	12	
4	0	291.14	0.0	0.0	1.0	4918	02	10	2012	13	

	minutes	seconds
0	00	00
1	00	00
2	00	00
3	00	00
4	00	00

Feature Engineering

```
le = LabelEncoder()
```

✓ 0.0s

```
le.fit(data['weather'])
```

✓ 0.0s

LabelEncoder

```
#splitting into independant and dependant variables
y=data['traffic_volume']
x=data.drop(columns=['traffic_volume'],axis=1)
x['holiday'] = le.fit_transform(x['holiday'].astype(str))
x['weather'] = le.fit_transform(x['weather'].astype(str))
```

Save Processed Data

```
data.head()
```

✓ 0.0s

	holiday	temp	rain	snow	weather	traffic_volume	day	month	year	hours	minutes	seconds
0	0	288.28	0.0	0.0	1.0	5545	02	10	2012	09	00	00
1	0	289.36	0.0	0.0	1.0	4516	02	10	2012	10	00	00
2	0	289.58	0.0	0.0	1.0	4767	02	10	2012	11	00	00
3	0	290.13	0.0	0.0	1.0	5026	02	10	2012	12	00	00
4	0	291.14	0.0	0.0	1.0	4918	02	10	2012	13	00	00