Continous Assessment -3

EDA for Urban traffic density in cities

Analysing data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv("/content/futuristic_city_traffic.csv")
df
```

	City	Vehicle Type	Weather	Economic Condition	Day Of Week	Hour Of Day	Speed	Is Peak Hour	Oc
0	SolarisVille	Drone	Snowy	Stable	Sunday	20.0	29.4268	0.0	
1	AquaCity	Flying Car	Solar Flare	Recession	Wednesday	2.0	118.8000	0.0	
2	Neuroburg	Autonomous Vehicle	Solar Flare	Recession	Wednesday	16.0	100.3904	0.0	
3	Ecoopolis	Drone	Clear	Booming	Thursday	8.0	76.8000	1.0	
4	AquaCity	Autonomous Vehicle	Solar Flare	Stable	Saturday	16.0	45.2176	0.0	
...	...	...	...	...	...	...	...	...	
25583	AquaCity	Drone	Snowy	Stable	Monday	9.0	17.9282	0.0	
25584	MetropolisX	Autonomous Vehicle	Snowy	Booming	Monday	20.0	71.1556	0.0	

```
df.info()
```

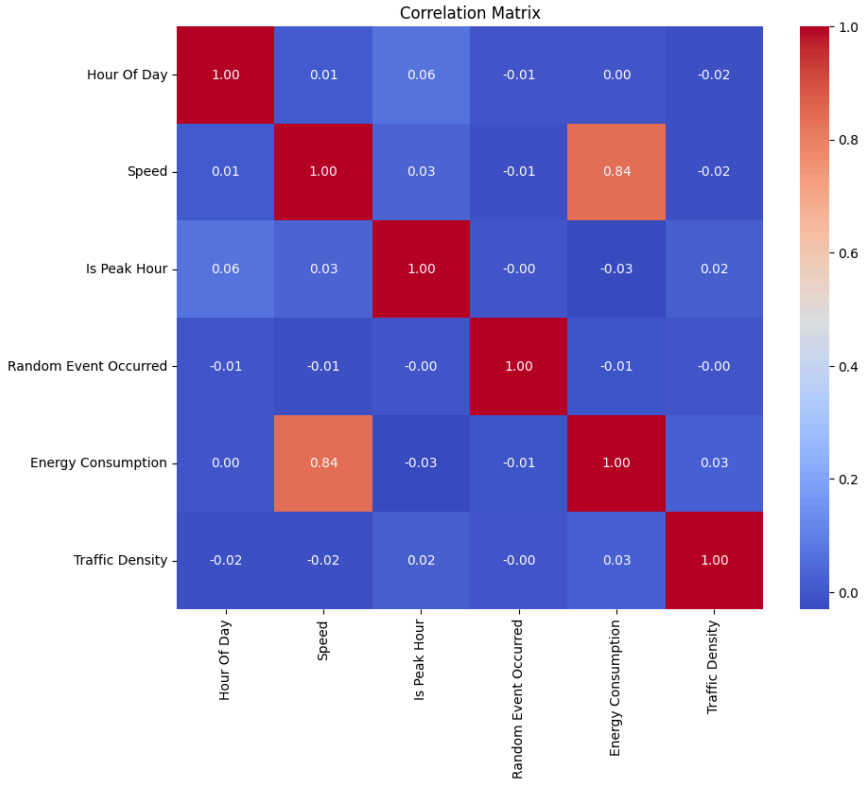
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25588 entries, 0 to 25587
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   City                   25588 non-null  object
 1   Vehicle Type           25587 non-null  object
 2   Weather                25587 non-null  object
 3   Economic Condition     25587 non-null  object
 4   Day Of Week            25587 non-null  object
 5   Hour Of Day            25587 non-null  float64
 6   Speed                  25587 non-null  float64
 7   Is Peak Hour           25587 non-null  float64
 8   Random Event Occurred  25587 non-null  float64
 9   Energy Consumption     25587 non-null  float64
 10  Traffic Density        25587 non-null  float64
dtypes: float64(6), object(5)
memory usage: 2.1+ MB
```

```
df["Day Of Week"].value_counts()
```

```
Wednesday    3744
Thursday     3676
Monday       3665
Tuesday      3664
Sunday       3635
Saturday     3603
Friday       3600
Name: Day Of Week, dtype: int64
```

```
# Explore correlation matrix
plt.figure(figsize=(10, 8))
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```

```
<ipython-input-5-a89d5688a269>:3: FutureWarning: The default value of numeric_only in
  corr_matrix = df.corr()
```



Gradient Boosting Model

What are the most influential factors affecting urban traffic density, as determined by the XGBoost model?

```
import xgboost as xgb
import plotly.graph_objs as go


# Separate features (X) and target variable (y)
X = df.drop(columns=["Traffic Density"])  # Assuming "Traffic Density" is the target variable
y = df["Traffic Density"]


# Convert categorical variables to one-hot encoding
X_encoded = pd.get_dummies(X)


# Combine X_encoded and y into a single DataFrame
data = pd.concat([X_encoded, y], axis=1)

# Remove rows with NaN values
data = data.dropna()

# Separate X and y after removing NaN values
X_cleaned = data.drop(columns=[y.name])
y_cleaned = data[y.name]
```

```
data.head()
```

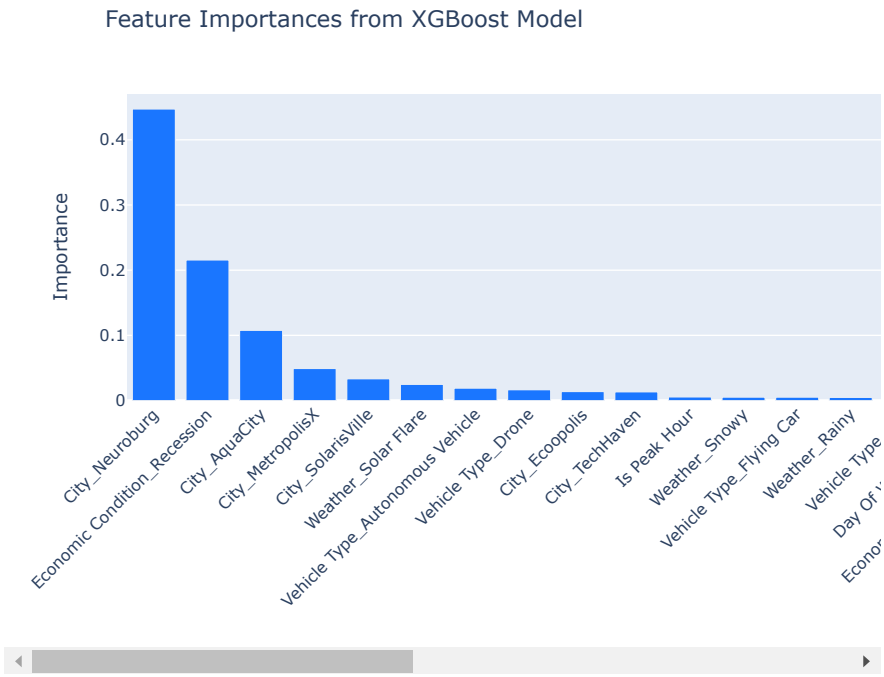
	Hour Of Day	Speed	Is Peak Hour	Random Event Occurred	Energy Consumption	City_AquaCity	City_Ecoopolis	City_Me
0	20.0	29.4268	0.0	0.0	14.7134	0	0	
1	2.0	118.8000	0.0	0.0	143.5682	1	0	
2	16.0	100.3904	0.0	0.0	91.2640	0	0	
3	8.0	76.8000	1.0	0.0	46.0753	0	1	
4	16.0	45.2176	0.0	0.0	40.1934	1	0	

5 rows × 32 columns

```
# Train an XGBoost model
model = xgb.XGBRegressor()

# Train XGBoost model on cleaned data
model.fit(X_cleaned, y_cleaned)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

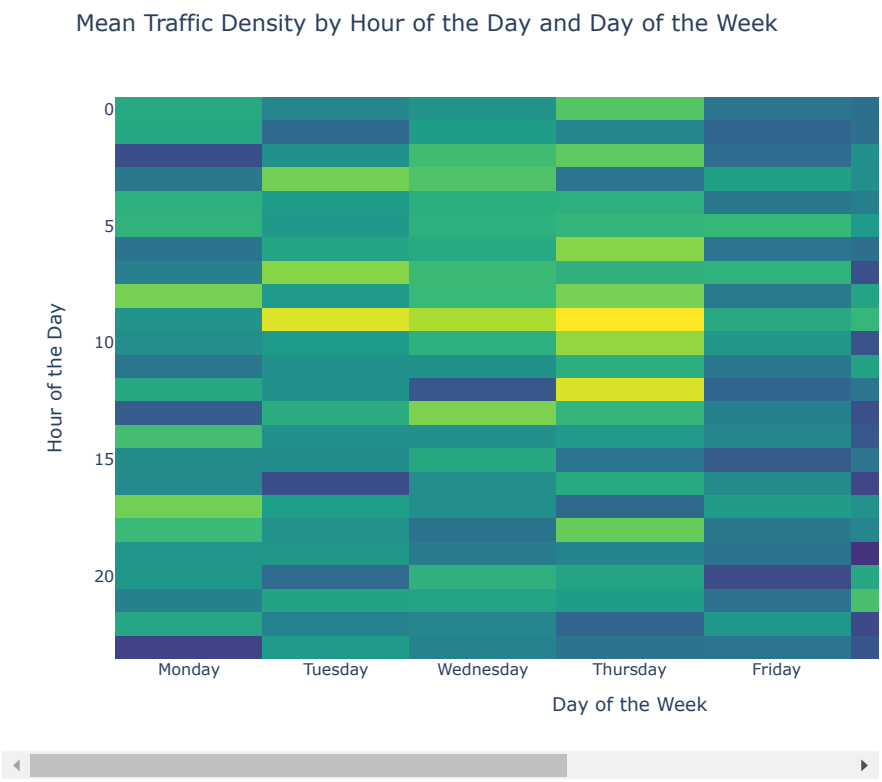
```
# Get feature importances
feature_importances = model.feature_importances_


# Get feature names (assuming X is a DataFrame)
feature_names = X_cleaned.columns


# Sort feature importances in descending order
sorted_indices = feature_importances.argsort()[::-1]
sorted_feature_importances = feature_importances[sorted_indices]
sorted_feature_names = feature_names[sorted_indices]


# Create a bar plot for feature importances using Plotly
fig = go.Figure(data=[go.Bar(
    x=sorted_feature_names,
    y=sorted_feature_importances,
    marker=dict(color='rgb(26, 118, 255)'),  # Change color if needed
)])

fig.update_layout(
    title="Feature Importances from XGBoost Model",
    xaxis_title="Features",
    yaxis_title="Importance",
    xaxis_tickangle=-45,  # Rotate x-axis labels for better readability
)

fig.show()
```



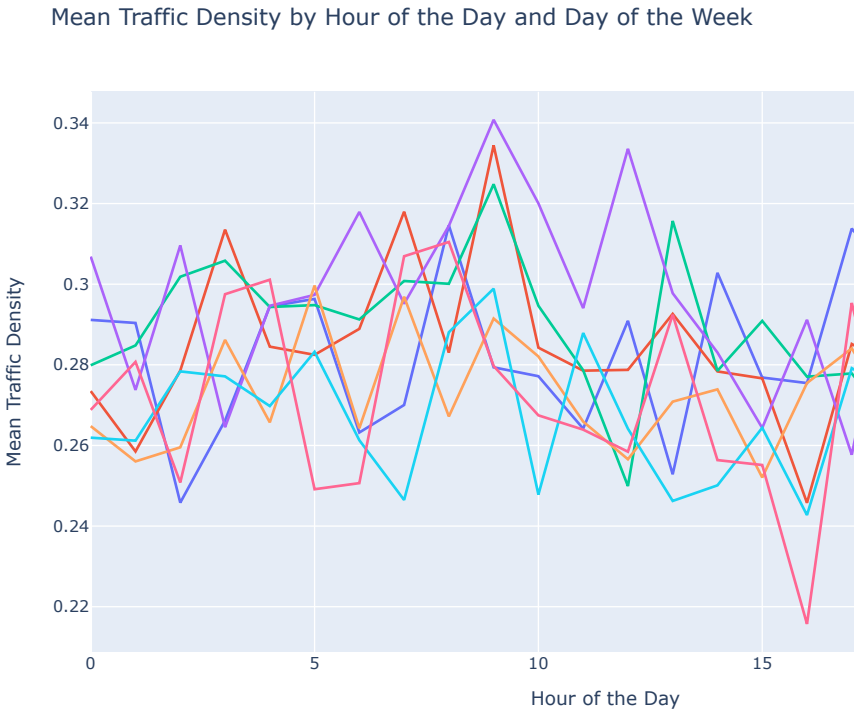
How traffic density varies over different hours of the day, days of the week, or months of the year?

```
import plotly.express as px

# Calculate mean traffic density by hour of the day and day of the week
mean_traffic_density = df.groupby(['Hour Of Day', 'Day Of Week'])['Traffic Density'].mean().reset_index()

# Create a heatmap
fig = px.imshow(mean_traffic_density.pivot('Hour Of Day', 'Day Of Week', 'Traffic Density'),
                labels=dict(x='Day of the Week', y='Hour of the Day', color='Traffic Density'),
                x=['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'],
                y=list(range(24)),
                color_continuous_scale='Viridis',
                title='Mean Traffic Density by Hour of the Day and Day of the Week')

# Update layout
fig.update_layout(
    xaxis_title='Day of the Week',
    yaxis_title='Hour of the Day',
    coloraxis_colorbar=dict(title='Traffic Density'),
    width=1000,
    height=600
)

# Show plot
fig.show()
```

```
<ipython-input-16-7eadca09567b>:10: FutureWarning:

In a future version of pandas all arguments of DataFrame.pivot will be keyword-only.
```




```
#Using line plot
import plotly.express as px

# Calculate mean traffic density by hour of the day and day of the week
mean_traffic_density = df.groupby(['Hour Of Day', 'Day Of Week'])['Traffic Density'].mean().reset_index()

# Create line plot
fig = px.line(mean_traffic_density, x='Hour Of Day', y='Traffic Density', color='Day Of Week',
              title='Mean Traffic Density by Hour of the Day and Day of the Week',
              labels={'Hour Of Day': 'Hour of the Day', 'Traffic Density': 'Mean Traffic Density'})

# Update layout
fig.update_layout(
    xaxis=dict(title='Hour of the Day'),
    yaxis=dict(title='Mean Traffic Density'),
    legend_title='Day of the Week',
    height=600,
    width=1000
)

# Show plot
fig.show()
```

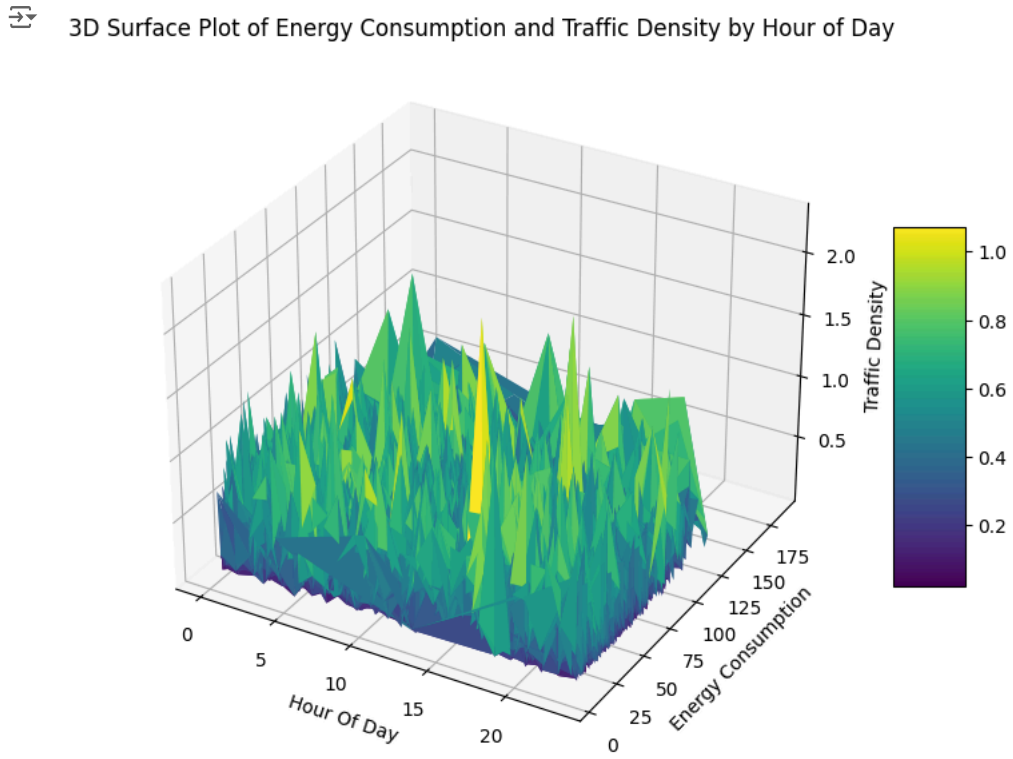


What is the impact of energy consumption on traffic density throughout the day?

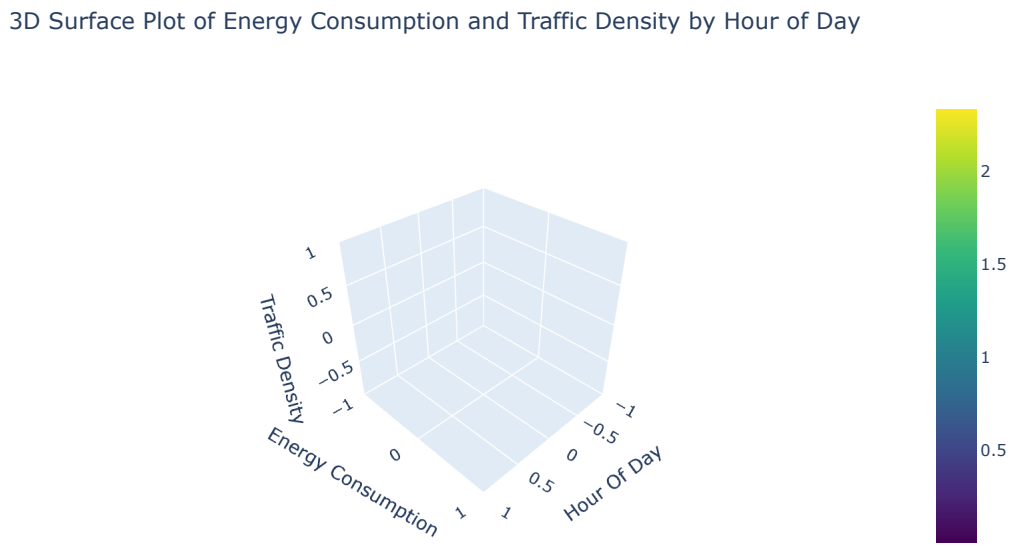
```
from mpl_toolkits.mplot3d import Axes3D


# Check for missing values and drop them
df = df.dropna()
df
```

	City	Vehicle Type	Weather	Economic Condition	Day Of Week	Hour Of Day	Speed	P H
0	SolarisVille	Drone	Snowy	Stable	Sunday	20.0	29.4268	
1	AquaCity	Flying Car	Solar Flare	Recession	Wednesday	2.0	118.8000	
2	Neuroburg	Autonomous Vehicle	Solar Flare	Recession	Wednesday	16.0	100.3904	
3	Ecoopolis	Drone	Clear	Booming	Thursday	8.0	76.8000	
4	AquaCity	Autonomous Vehicle	Solar Flare	Stable	Saturday	16.0	45.2176	
...	...	...	...	...	...	...	...	
25582	AquaCity	Car	Electromagnetic Storm	Recession	Friday	12.0	45.2848	
25583	AquaCity	Drone	Snowy	Stable	Monday	9.0	17.9282	
25584	MetropolisX	Autonomous Vehicle	Snowy	Booming	Monday	20.0	71.1556	
25585	Neuroburg	Autonomous Vehicle	Clear	Booming	Saturday	5.0	62.2353	
25586	AquaCity	Autonomous Vehicle	Snowy	Recession	Friday	23.0	62.6450	

```
# Create a 3D plot
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')

# Extracting variables
hour_of_day = df['Hour Of Day']
energy_consumption = df['Energy Consumption']
traffic_density = df['Traffic Density']

# Plotting
surf = ax.plot_trisurf(hour_of_day, energy_consumption, traffic_density, cmap='viridis')

# Add color bar
fig.colorbar(surf, ax=ax, shrink=0.5, aspect=5)

# Set labels and title
ax.set_xlabel('Hour Of Day')
ax.set_ylabel('Energy Consumption')
ax.set_zlabel('Traffic Density')
ax.set_title('3D Surface Plot of Energy Consumption and Traffic Density by Hour of Day')

# Show plot
plt.show()
```

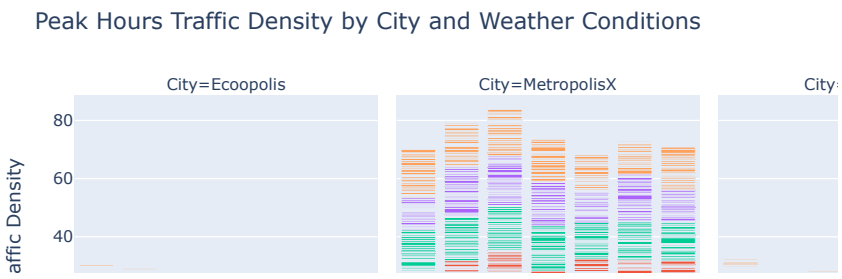



```
# Create the 3D surface plot
fig = go.Figure(data=[go.Surface(x=df['Hour Of Day'],
                                 y=df['Energy Consumption'],
                                 z=df['Traffic Density'],
                                 colorscale='Viridis')])

# Update layout
fig.update_layout(title='3D Surface Plot of Energy Consumption and Traffic Density by Hour of Day',
                  scene=dict(xaxis_title='Hour Of Day',
                             yaxis_title='Energy Consumption',
                             zaxis_title='Traffic Density'))

# Show plot
fig.show()
```



What are the peak hours of traffic density for each city, considering the day of the week and weather conditions?

```
import plotly.express as px

# Assuming you have a DataFrame named 'df' containing your dataset
# Replace 'df' with the name of your DataFrame and adjust the column names as needed

# Filter data for peak hours (e.g., 7 AM - 9 AM and 5 PM - 7 PM)
peak_hours = df[(df['Hour Of Day'].between(7, 9) | df['Hour Of Day'].between(17, 19))]

# Create a grouped bar chart
fig = px.bar(peak_hours, x='Day Of Week', y='Traffic Density', color='Weather',
             facet_col='City', facet_col_wrap=3,
             labels={'Day Of Week': 'Day of the Week', 'Traffic Density': 'Traffic Density'},
             title='Peak Hours Traffic Density by City and Weather Conditions')

# Update layout
fig.update_layout(
    xaxis_title='Day of the Week',
    yaxis_title='Traffic Density',
    showlegend=True,
    legend_title='Weather',
    height=600,
    width=1000
)

# Show plot
fig.show()
```



How does the traffic density vary with speed and energy consumption of the vehicle in urban areas?