TRAFFIC MANAGEMENT SYSTEM

# TEAM **MEMBER**

**953121106042 : Ramalakshmi.A**

**Phase 2 Submission document**



**Introduction:**

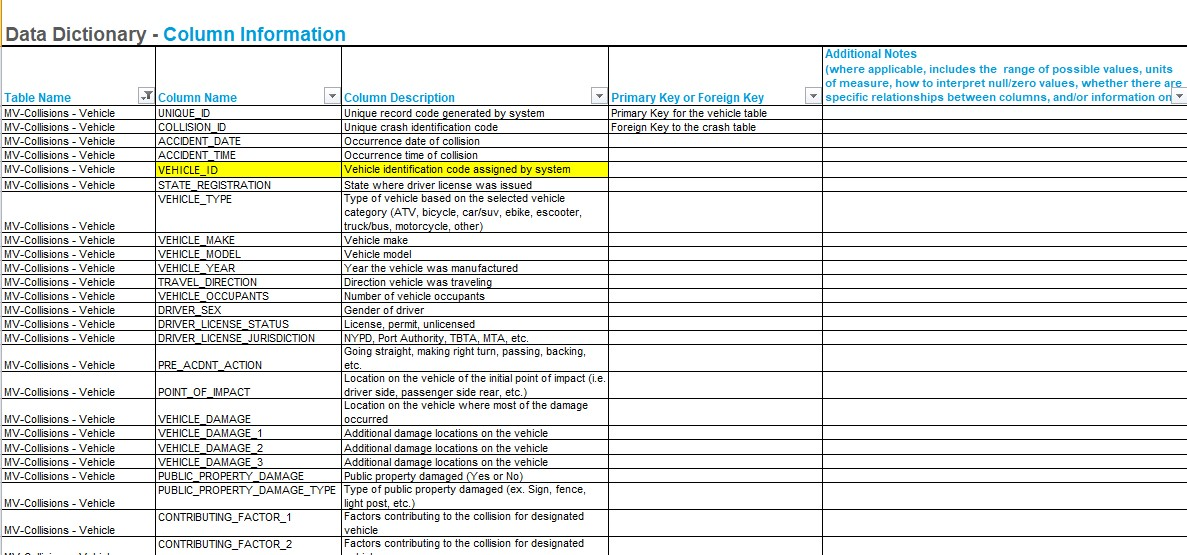
* **In the rapidly evolving landscape of urbanization, managing traffic congestion has become a critical challenge faced by cities worldwide. With the rise of smart technologies, the integration of Internet of Things (IoT) in urban infrastructure has opened up new avenues for addressing this issue effectively. The project, titled "Traffic Management System," harnesses the power of IoT to revolutionize the way traffic is monitored, analyzed, and controlled in urban areas.**
* **This innovative project aims to create a comprehensive and intelligent traffic management solution that not only enhances the overall traffic flow but also ensures the safety and convenience of commuters. By leveraging IoT devices and sensors placed strategically throughout the city, real-time data can be collected, processed, and analyzed. This data-driven approach enables authorities to make informed decisions, optimize traffic signals, and implement dynamic routing strategies, leading to reduced congestion and minimized travel time.**
* **The Traffic Management System is not just about managing vehicular movement; it's a testament to the synergy between technology and urban planning. By utilizing IoT devices such as smart cameras, sensors, and communication networks, the system empowers cities to move towards a sustainable and efficient transportation ecosystem. This project envisions a future where traffic congestion is mitigated, environmental impact is minimized, and citizens experience seamless mobility.**
* **Through this project, we embark on a journey to transform the way cities handle traffic challenges. By embracing IoT innovations, the Traffic Management System aims to create smarter, safer, and more livable urban environments, setting a new standard for intelligent transportation systems in the digital age.**

**Content for Project Phase : 2**

**In today's rapidly growing urban centers, managing traffic flow efficiently and ensuring the safety of commuters has become paramount. With the rise in population and vehicles on the roads, the need for an advanced Traffic Management System (TMS) is more crucial than ever before.**

**Data Source :**

**A good data source for traffic management system using machine learning should be Accurate,Complete,Covering the geographic area of Interest,Accessible.**



**Data Collection and Preprocessing:**

* **Ensure consistency in data format. For example, standardize date formats, units of measurement, and other categorical variables.**
* **Ensure consistency in data format. For example, standardize date formats, units of measurement, and other categorical variables.**

**Exploratory Data Analysis(EDA):**

* **Begin by loading the dataset and obtaining a summary. Understand the number of records, columns, and data types.**
* **Utilize functions like head(), info(), and describe() to get an overview of the data.**

**Feature Engineering:**

* **Calculate traffic density by dividing the number of vehicles by the road area or length. High traffic density can indicate congestion.**
* **Extract time-related features from timestamps, such as hour of the day, day of the week, or even specific time slots (e.g., morning rush hour).**
* **Create binary features indicating peak traffic hours.**
* **Include weather data like temperature, precipitation, and visibility, which can impact traffic flow.**
* **Create categorical features for weather conditions (e.g., sunny, rainy, snowy).**

**Advanced Regression Techniques:**

**1. Ridge Regression:**

* **Ridge regression is a regularization technique that prevents overfitting by adding a penalty term to the loss function.**
* **It is useful when there are many correlated features. The penalty term helps in reducing the impact of irrelevant features.**

**2. Lasso Regression:**

* **Lasso regression is another regularization technique that can shrink some coefficients to exactly zero.**
* **It is beneficial when you suspect that many features are irrelevant or when you want to perform feature selection automatically.**

**3. Elastic Net Regression:**

* **Elastic Net combines both L1 (Lasso) and L2 (Ridge) regularization techniques.**
* **It is useful when there are many features and some of them are correlated. Elastic Net addresses the limitations of Ridge and Lasso by balancing their penalties.**

**4. Support Vector Regression (SVR):**

* **SVR is effective for datasets with complex relationships between features and the target variable.**
* **It uses support vector machines to find the optimal hyperplane that best fits the data while allowing for a margin of error.**

**5. Gradient Boosting Regression:**

* **Gradient Boosting Regression builds multiple decision trees sequentially, each correcting the errors of its predecessor.**

**Model Evalution and Selection:**

* **Determine the appropriate evaluation metrics for your Traffic Management System. Common metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or specific metrics tailored to traffic-related objectives like traffic congestion prediction accuracy.**
* **Divide your dataset into training and testing subsets. A typical split might be 80% of the data for training and 20% for testing. Alternatively, consider using techniques like cross-validation for more robust evaluation, especially if the dataset is limited.**

**Model Interpretability:**

* Utilize techniques like feature importance scores from tree-based models (e.g., Random Forest, XGBoost) to identify which features have the most significant impact on traffic predictions.
* Visualize feature importance using bar charts or heatmaps to communicate the relevance of different features.
* PDPs show the relationship between a feature and the predicted outcome while keeping other features constant.

Deployement and Prediction:

* Serialize the trained machine learning model into a format that can be easily stored and loaded, such as a pickle file in Python.
* This step ensures that the model can be transported and used in different environments.

Program:

# Traffic Management System

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import matplotlib.pyplot as plt

import seaborn as sns

plt.style.use('fivethirtyeight')

sns.set\_style('whitegrid')

#plotly

import plotly.express as px

import plotly.graph\_objects as go

from plotly.subplots import make\_subplots

import folium

import datetime

import calendar

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

for filename in filenames:

print(os.path.join(dirname, filename))

Model 1- Linear Regression

In [1] :

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import matplotlib.pyplot as plt

import seaborn as sns

plt.style.use('fivethirtyeight')

sns.set\_style('whitegrid')

#plotly

import plotly.express as px

import plotly.graph\_objects as go

from plotly.subplots import make\_subplots

import folium

import datetime

import calendar

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

for filename in filenames:

print(os.path.join(dirname, filename))

In [2] :

df = pd.read\_csv('../input/nypd-motor-vehicle-collisions/nypd-motor-vehicle-collisions.csv', dtype=str)

In [3] :

df.head(3)

Out [3] :

ACCIDENT DATE ACCIDENT TIME BOROUGH ZIP CODE LATITUDE LONGITUDE LOCATION ON STREET NAME CROSS STREET NAME OFF STREET NAME ... CONTRIBUTING FACTOR VEHICLE 2 CONTRIBUTING FACTOR VEHICLE 3 CONTRIBUTING FACTOR VEHICLE 4 CONTRIBUTING FACTOR VEHICLE 5 COLLISION\_ID VEHICLE TYPE CODE 1 VEHICLE TYPE CODE 2 VEHICLE TYPE CODE 3 VEHICLE TYPE CODE 4 VEHICLE TYPE CODE 5

0 2019-08-05T00:00:00.000 16:30 QUEENS 11434 40.6760520 -73.7901840 {'type': 'Point', 'coordinates': [-73.790184, ... NaN NaN 150-08 123 AVENUE ... Unspecified NaN NaN NaN 4184637 Sedan Pick-up Truck NaN NaN NaN

1 2019-08-27T00:00:00.000 16:02 BROOKLYN 11225 40.6577800 -73.9510960 {'type': 'Point', 'coordinates': [-73.951096, ... NaN NaN 288 HAWTHORNE STREET ... Unspecified NaN NaN NaN 4195773 Station Wagon/Sport Utility Vehicle Station Wagon/Sport Utility Vehicle NaN NaN NaN

2 2019-08-15T00:00:00.000 17:57 MANHATTAN 10002 40.7181430 -73.9938350 {'type': 'Point', 'coordinates': [-73.993835, ... CHRYSTIE STREET GRAND STREET NaN ... NaN NaN NaN NaN 4202457 Sedan NaN NaN NaN NaN

In [4] :

print('DATASET SHAPE: ', df.shape)

DATASET SHAPE: (1612178, 29)

In [5] :

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1612178 entries, 0 to 1612177

Data columns (total 29 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 ACCIDENT DATE 1612178 non-null object

1 ACCIDENT TIME 1612178 non-null object

2 BOROUGH 1127553 non-null object

3 ZIP CODE 1127376 non-null object

4 LATITUDE 1415893 non-null object

5 LONGITUDE 1415893 non-null object

6 LOCATION 1415893 non-null object

7 ON STREET NAME 1298002 non-null object

8 CROSS STREET NAME 1079193 non-null object

9 OFF STREET NAME 219732 non-null object

10 NUMBER OF PERSONS INJURED 1612161 non-null object

11 NUMBER OF PERSONS KILLED 1612145 non-null object

12 NUMBER OF PEDESTRIANS INJURED 1612178 non-null object

13 NUMBER OF PEDESTRIANS KILLED 1612178 non-null object

14 NUMBER OF CYCLIST INJURED 1612178 non-null object

15 NUMBER OF CYCLIST KILLED 1612178 non-null object

16 NUMBER OF MOTORIST INJURED 1612178 non-null object

17 NUMBER OF MOTORIST KILLED 1612178 non-null object

18 CONTRIBUTING FACTOR VEHICLE 1 1607854 non-null object

19 CONTRIBUTING FACTOR VEHICLE 2 1396127 non-null object

20 CONTRIBUTING FACTOR VEHICLE 3 104254 non-null object

21 CONTRIBUTING FACTOR VEHICLE 4 21789 non-null object

22 CONTRIBUTING FACTOR VEHICLE 5 5622 non-null object

23 COLLISION\_ID 1612178 non-null object

24 VEHICLE TYPE CODE 1 1606597 non-null object

25 VEHICLE TYPE CODE 2 1346858 non-null object

26 VEHICLE TYPE CODE 3 128742 non-null object

27 VEHICLE TYPE CODE 4 46009 non-null object

28 VEHICLE TYPE CODE 5 10150 non-null object

In [6] :

dtypes: object(29)

#replace capslock to lowercase

df.columns = [i.lower() for i in df.columns]

df['accident date'] = pd.to\_datetime(df['accident date'])

df['accident time'] = pd.to\_datetime(df['accident time']).dt.time

In [7] :

num\_feat = [i for i in df.columns if 'number' in i] + ['latitude', 'longitude']

df[num\_feat] = df[num\_feat].apply(pd.to\_numeric, errors='coerce')

In [8] :

pd.DataFrame(df.isnull().sum() / df.shape[0] \*100, columns=['Missing Value %'])

Out [8] :

Missing Value %

accident date 0.000000

accident time 0.000000

borough 30.060266

zip code 30.071245

latitude 12.175144

longitude 12.175144

location 12.175144

on street name 19.487674

cross street name 33.059935

off street name 86.370488

number of persons injured 0.001054

number of persons killed 0.002047

number of pedestrians injured 0.000000

number of pedestrians killed 0.000000

number of cyclist injured 0.000000

number of cyclist killed 0.000000

number of motorist injured 0.000000

number of motorist killed 0.000000

contributing factor vehicle 1 0.268209

contributing factor vehicle 2 13.401188

contributing factor vehicle 3 93.533344

contributing factor vehicle 4 98.648474

contributing factor vehicle 5 99.651279

collision\_id 0.000000

vehicle type code 1 0.346178

vehicle type code 2 16.457240

vehicle type code 3 92.014405

vehicle type code 4 97.146159

vehicle type code 5 99.370417

In [9] :

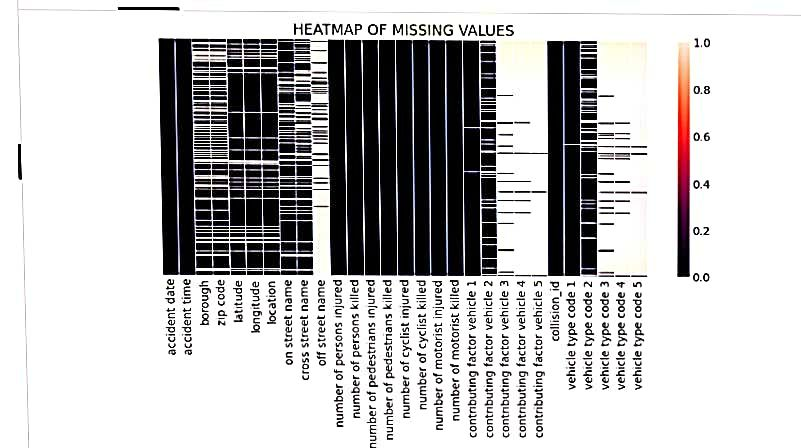
plt.figure(figsize=(12,5))

plt.title('HEATMAP OF MISSING VALUES', fontsize=18)

sns.heatmap(df.isnull(), yticklabels=False)

Out [9] :

<matplotlib.axes.\_subplots.AxesSubplot at 0x7b1df8dae710>



EDA

ANALYSIS BY BOROUGH

In [10] :

plt.figure(figsize=(10,5))

plt.title('ACCIDENTS COUNTPLOT PER BOROUGH')

sns.barplot(x=df.groupby('borough').size().index,

y=df.groupby('borough').size().values)

Out [10] :

<matplotlib.axes.\_subplots.AxesSubplot at 0x7b1df6b339d0>



In [11] :

accidents\_bor\_df = df.groupby('borough')[['number of persons injured', 'number of persons killed']].sum()

fig, ax = plt.subplots(1,2,figsize=(14,5))

plt.suptitle('DISTRIBUTION PER NUMBER OF INJURED AND KILLED')

ax[1].set\_xticklabels(labels=accidents\_bor\_df.index,rotation=30)

ax[0].set\_xticklabels(labels=accidents\_bor\_df.index,rotation=30)

sns.barplot(accidents\_bor\_df.index, accidents\_bor\_df['number of persons injured'], ax=ax[0])

sns.barplot(accidents\_bor\_df.index, accidents\_bor\_df['number of persons killed'], ax=ax[1], palette='deep')

Out [11] :

<matplotlib.axes.\_subplots.AxesSubplot at 0x7b1df5277610>

# 

# 

In [12] :

fig, ax = plt.subplots(1,2, figsize=(14,5))

plt.suptitle('DISTRIBUTION BY PERCENTAGE')

ax[1].set\_xticklabels(labels=accidents\_bor\_df.index,rotation=30)

ax[0].set\_xticklabels(labels=accidents\_bor\_df.index,rotation=30)

ax[0].set\_title('INJURED PERCENTAGE', fontsize=12)

ax[1].set\_title('KILLED PERCENTAGE', fontsize=12)

sns.barplot((accidents\_bor\_df['number of persons injured'] / df.groupby('borough').size() \*100).index,

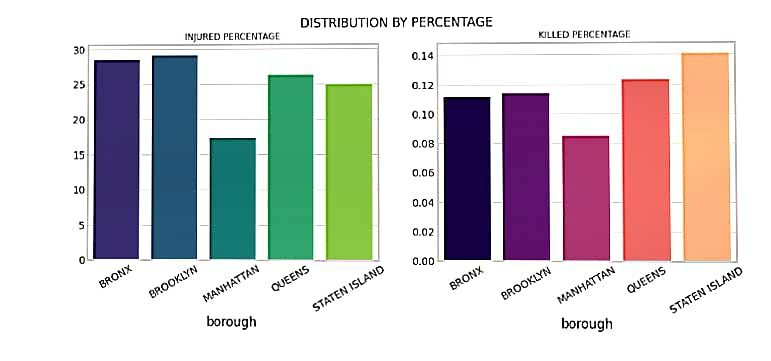
(accidents\_bor\_df['number of persons injured'] / df.groupby('borough').size() \*100).values, ax=ax[0], palette='viridis')

sns.barplot((accidents\_bor\_df['number of persons killed'] / df.groupby('borough').size() \*100).index,

(accidents\_bor\_df['number of persons killed'] / df.groupby('borough').size() \*100).values, ax=ax[1], palette='magma')

Out [12] :

<matplotlib.axes.\_subplots.AxesSubplot at 0x7b1df5179790>



**In [13] :**

**print('MEAN INJURED: ',(accidents\_bor\_df['number of persons injured'] / df.groupby('borough').size() \*100).values.mean())**

**print('MEAN KILLED: ',(accidents\_bor\_df['number of persons killed'] / df.groupby('borough').size() \*100).values.mean())**

**MEAN INJURED: 25.258721101545422**

**MEAN KILLED: 0.1151597454148819**

**In [14] :**

**datewise = df.groupby(['accident date', 'borough'])[[i for i in df.columns if 'number' in i]].sum()**

**In [15] :**

**fig = make\_subplots(rows=2,cols=1,**

**subplot\_titles=('NUMBER OF INJURED PER DAY', 'NUMBER OF KILLED PER DAY'))**

**cols = ['QUEENS', 'BROOKLYN', 'MANHATTAN', 'BRONX', 'STATEN ISLAND']**

**feat = [i for i in df.columns if 'number' in i] + ['accident date']**

**for i, bor in enumerate(cols):**

**data\_per\_bor = df[df['borough']== bor][feat]**

**data\_per\_bor = data\_per\_bor.groupby('accident date').sum()**

**fig.add\_trace(go.Scatter(x=data\_per\_bor.index, y=data\_per\_bor['number of persons injured'], name=bor), row=1,col=1)**

**fig.add\_trace(go.Scatter(x=data\_per\_bor.index, y=data\_per\_bor['number of persons killed'], name=bor), row=2, col=1)**

**fig.update\_layout(template='plotly\_dark', width=1000, height=800)**

**fig.show()**

**In [16] :**

**fig = make\_subplots(rows=2,cols=1,**

**subplot\_titles=('NUMBER OF INJURED PER DAY', 'NUMBER OF KILLED PER DAY'))**

**cols = ['QUEENS', 'BROOKLYN', 'MANHATTAN', 'BRONX', 'STATEN ISLAND']**

**feat = [i for i in df.columns if 'number' in i] + ['accident date']**

**for i, bor in enumerate(cols):**

**data\_per\_bor = df[df['borough']== bor][feat]**

**data\_per\_bor = data\_per\_bor.groupby('accident date').sum()[-365:]**

**fig.add\_trace(go.Scatter(x=data\_per\_bor.index, y=data\_per\_bor['number of persons injured'], name=bor), row=1,col=1)**

**fig.add\_trace(go.Scatter(x=data\_per\_bor.index, y=data\_per\_bor['number of persons killed'], name=bor), row=2, col=1)**

**fig.update\_layout(template='plotly\_dark', width=1000, height=800)**

**fig.show()**

**In [17] :**

**df.groupby('accident date').size().mean()**

**Out [17] :**

**596.660991857883**

**In [18] :**

**weekwise = df.copy()**

**def week\_of\_month(tgtdate):**

**days\_this\_month = calendar.mdays[tgtdate.month]**

**for i in range(1, days\_this\_month):**

**d = datetime.datetime(tgtdate.year, tgtdate.month, i)**

**if d.day - d.weekday() > 0:**

**startdate = d**

**break**

**# now we canuse the modulo 7 appraoch**

**return (tgtdate - startdate).days //7 + 1**

**weekwise['weekofmonth'] = weekwise['accident date'].apply(lambda d: (d.day-1) // 7 + 1)**

**weekwise['weekofyear'] = weekwise['accident date'].dt.weekofyear**

**weekwise['month'] = weekwise['accident date'].dt.month**

**weekwise['year'] = weekwise['accident date'].dt.year**

**In [19] :**

**weekwise\_month = weekwise.groupby('weekofmonth')[[i for i in weekwise.columns if 'number' in i]].sum()**

**fig,ax = plt.subplots(1,2,figsize=(14,5))**

**plt.suptitle('COUNTPLOT OF INJURED AND KILLED BY WEEK OF MONTH', x=0.5, y=1.02, fontsize=20)**

**ax[0].set\_title('INJURED', fontsize=14)**

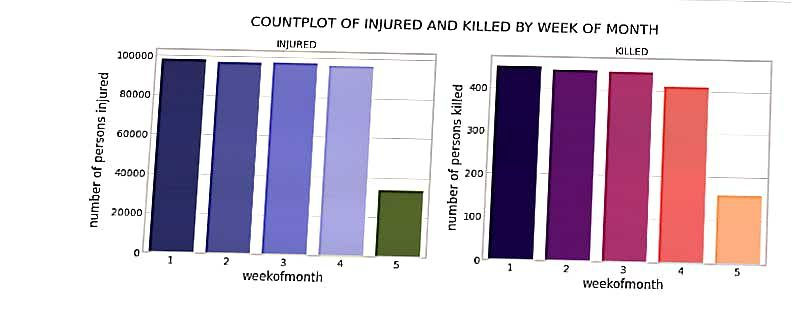
**ax[1].set\_title('KILLED', fontsize=14)**

**sns.barplot(x=weekwise\_month['number of persons injured'].index ,y=weekwise\_month['number of persons injured'], ax=ax[0], palette='tab20b')**

**sns.barplot(x=weekwise\_month['number of persons killed'].index ,y=weekwise\_month['number of persons killed'], ax=ax[1], palette='magma')**

**Out [19] :**

**<matplotlib.axes.\_subplots.AxesSubplot at 0x7b1df45c9990>**



**In [20] :**

**weekwise\_year = weekwise.groupby('weekofyear')[[i for i in weekwise.columns if 'number' in i]].sum()**

**fig,ax = plt.subplots(1,2,figsize=(14,5))**

**ax[1].set\_xticklabels(labels=accidents\_bor\_df.index,rotation=90)**

**ax[0].set\_xticklabels(labels=accidents\_bor\_df.index,rotation=90)**

**plt.suptitle('COUNTPLOT OF INJURED AND KILLED BY WEEK OF MONTH', x=0.5, y=1.02, fontsize=20)**

**ax[0].set\_title('INJURED', fontsize=14)**

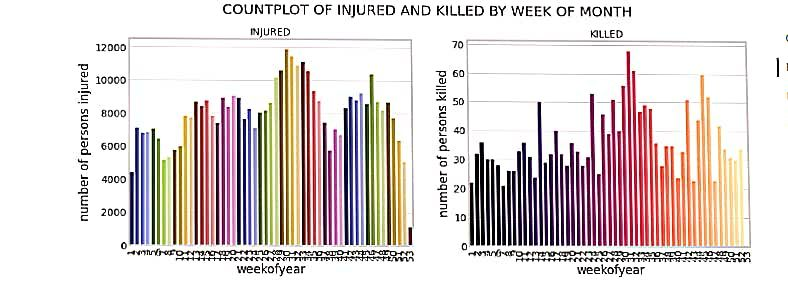
**ax[1].set\_title('KILLED', fontsize=14)**

**sns.barplot(x=weekwise\_year['number of persons injured'].index ,y=weekwise\_year['number of persons injured'], ax=ax[0], palette='tab20b')**

**sns.barplot(x=weekwise\_year['number of persons killed'].index ,y=weekwise\_year['number of persons killed'], ax=ax[1], palette='magma')**

**Out [20] :**

**<matplotlib.axes.\_subplots.AxesSubplot at 0x7b1df4efae10>**



**In [21] :**

**by\_month = weekwise.groupby('month')[[i for i in weekwise.columns if 'number' in i]].sum()**

**fig,ax = plt.subplots(1,2,figsize=(14,5))**

**plt.suptitle('COUNTPLOT OF INJURED AND KILLED BY MONTH', x=0.5, y=1.02, fontsize=20)**

**ax[0].set\_title('INJURED', fontsize=14)**

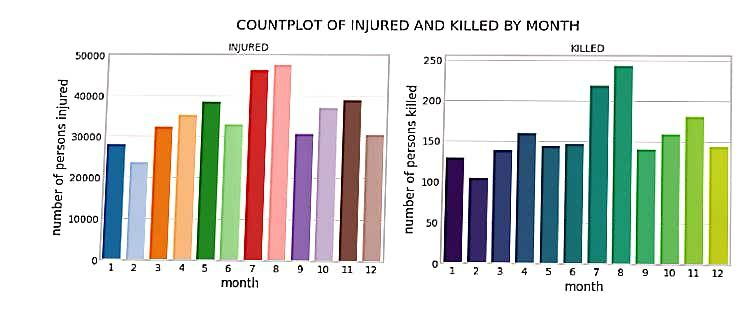
**ax[1].set\_title('KILLED', fontsize=14)**

**sns.barplot(x=by\_month['number of persons injured'].index ,y=by\_month['number of persons injured'], ax=ax[0], palette='tab20')**

**sns.barplot(x=by\_month['number of persons killed'].index ,y=by\_month['number of persons killed'], ax=ax[1], palette='viridis')**

**Out [21] :**

**<matplotlib.axes.\_subplots.AxesSubplot at 0x7b1dd8f44ed0>**



**In [22] :**

**by\_year = weekwise.groupby('year')[[i for i in weekwise.columns if 'number' in i]].sum()**

**fig,ax = plt.subplots(1,2,figsize=(14,5))**

**plt.suptitle('COUNTPLOT OF INJURED AND KILLED BY YEAR', x=0.5, y=1.02, fontsize=20)**

**ax[0].set\_title('INJURED', fontsize=14)**

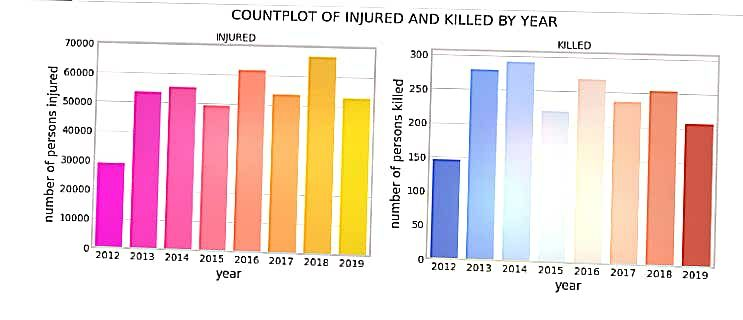
**ax[1].set\_title('KILLED', fontsize=14)**

**sns.barplot(x=by\_year['number of persons injured'].index ,y=by\_year['number of persons injured'], ax=ax[0], palette='spring')**

**sns.barplot(x=by\_year['number of persons killed'].index ,y=by\_year['number of persons killed'], ax=ax[1], palette='coolwarm')**

**Out [22] :**

**<matplotlib.axes.\_subplots.AxesSubplot at 0x7b1dd8d17d90>**



**In [23] :**

**per\_day\_val = round(df.shape[0]/df.groupby('accident date')['number of persons injured'].count().shape[0],2)**

**per\_week\_val = round(per\_day\_val \* 7, 2)**

**per\_month\_val = round(per\_day\_val \* 30, 2)**

**per\_year\_val = per\_month\_val \* 12**

**per\_hour\_val = (per\_day\_val / 24)**

**per\_5mins\_val = (per\_day\_val / 24) /60 \* 5**

**index = ['5mins', 'Hour', 'Day', 'Week', 'Month', 'Year']**

**data = [per\_5mins\_val, per\_hour\_val, per\_day\_val, per\_week\_val, per\_month\_val, per\_year\_val]**

**pd.DataFrame(index=index, data=data, columns=['Value']).T**

**Out [23] :**

**5mins Hour Day Week Month Year**

**Value 2.071736 24.860833 596.66 4176.62 17899.8 214797.6**

**In [24] :**

**gr\_injured = df[[i for i in df.columns for c in ['pedestrians injured', 'cyclist injured', 'motorist injured'] if c in i]].sum()**

**gr\_killed = df[[i for i in df.columns for c in ['pedestrians killed', 'cyclist killed', 'motorist killed'] if c in i]].sum()**

**gr\_injured.index = ['Pedestrian', 'Cyclist', 'Motorist']**

**gr\_killed.index = ['Pedestrian', 'Cyclist', 'Motorist']**

**fig, ax = plt.subplots(1,2,figsize=(14,5))**

**plt.suptitle('COUNTPLOT OF KILLED AND INJURED PER ACCIDENT TYPE', fontsize=20, x=0.5,y=1.02)**

**ax[0].set\_title('INJURED', fontsize=14)**

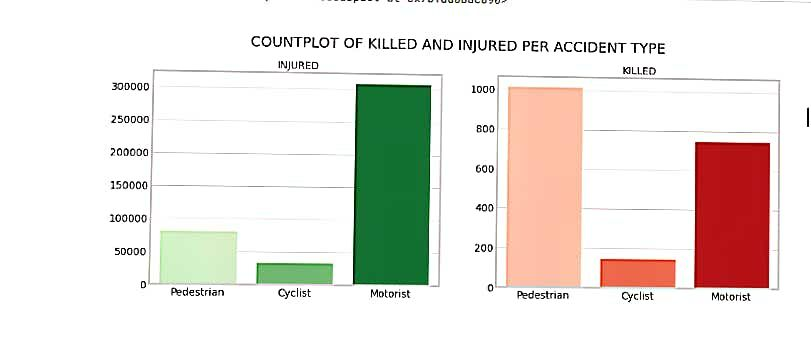
**ax[1].set\_title('KILLED', fontsize=14)**

**sns.barplot(gr\_injured.index, gr\_injured.values, ax=ax[0], palette='Greens')**

**sns.barplot(gr\_killed.index, gr\_killed.values, ax=ax[1], palette='Reds')**

**Out [24] :**

**<matplotlib.axes.\_subplots.AxesSubplot at 0x7b1dd8bdc890>**



**In [25] :**

**fig = make\_subplots(rows=3,cols=1,**

**subplot\_titles=('PEDESTRIAN', 'CYCLIST', 'MOTORIST'))**

**feat\_in = ['number of pedestrians injured',**

**'number of cyclist injured',**

**'number of motorist injured']**

**feat\_killed = ['number of pedestrians killed',**

**'number of cyclist killed',**

**'number of motorist killed']**

**for i, atype in enumerate(feat\_in):**

**data\_per\_acc = df.groupby('accident date')[atype].sum()**

**data\_per\_acc1 = df.groupby('accident date')[feat\_killed[i]].sum()**

**fig.add\_trace(go.Scatter(x=data\_per\_acc.index, y=data\_per\_acc.values, name='Injured'), row=i+1,col=1)**

**fig.add\_trace(go.Scatter(x=data\_per\_acc1.index, y=data\_per\_acc1.values, name='Killed'), row=i+1, col=1)**

**fig.update\_layout(title='NUMBER OF KILLED AND INJURED PER ACCIDENT TYPE',template='plotly\_dark', width=1000, height=1100)**

**fig.show()**

**In [26] :**

**contri\_df = df.groupby('contributing factor vehicle 1').size().sort\_values(ascending=False)**

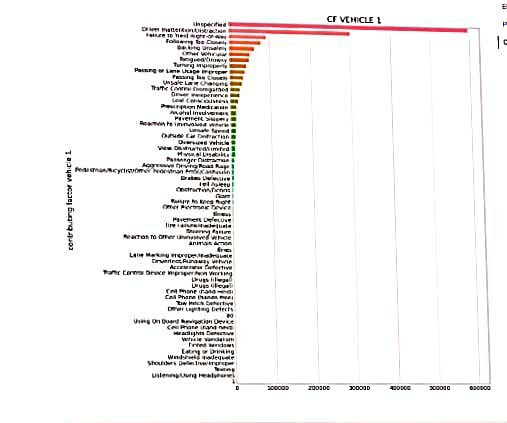
**plt.figure(figsize=(10,15))**

**plt.title('CF VEHICLE 1', fontsize=20)**

**sns.barplot(y = contri\_df.index, x = contri\_df.values)**

**Out [26] :**

**<matplotlib.axes.\_subplots.AxesSubplot at 0x7b1dd8c85950>**



**In [27] :**

**contri\_df = df.groupby('contributing factor vehicle 2').size().sort\_values(ascending=False)**

**plt.figure(figsize=(10,15))**

**plt.title('CF VEHICLE 2', fontsize=20)**

**sns.barplot(y = contri\_df.index, x = contri\_df.values)**

**Out [27] :**

**<matplotlib.axes.\_subplots.AxesSubplot at 0x7b1dd8c97bd0>**

